An instructive implementation of semantic parsing and reasoning using Lexical Functional Grammar

Mark-Matthias Zymla, Kascha Kruschwitz, Paul Zodl
Department of Linguistics
University of Konstanz
Konstanz, Germany
{mark-matthias.zymla | kascha.kruschwitz | paul.zodl}@uni-konstanz.de

Abstract

This paper presents a computational resource for exploring semantic parsing and reasoning through a strictly formal lense. Inspired by the framework of Lexical Functional Grammar, our system allows for modular exploration of different aspects of semantic parsing. It consists of a hand-coded formal grammar combining syntactic and semantic annotations, producing basic semantic representations. The system provides the option to extend these basic semantics via rewrite rules in a principled fashion to explore more complex reasoning. The result is a layered system enabling an incremental approach to semantic parsing. We illustrate this approach with examples from the Fracas testsuite, demonstrating its overall functionality and viability.

1 Introduction

Formal approaches to computational linguistics have been surpassed by quantitative methods in the fast-paced task-driven field of NLP. However, modern NLP approaches trade explainability and interpretability for performance gains. This puts a larger burden on researchers who need to evaluate whether a system captures the expected linguistic generalizations, and limits the possibility to test the effect of small tweaks to a system. Thus, understanding and exploring patterns in syntax and semantics is challenging and potentially affected by confounding factors (e.g., McCoy et al. 2019).

Formal approaches inherently require accurate descriptions of patterns. Computational approaches, in particular, often highlight wanted and unwanted interactions between linguistic descriptions. Detecting these is an essential skill of (computational) linguists and, thus, we deem it a worthwhile goal to make corresponding resources accessible. Concretely, we present a system for semantic parsing and reasoning based on Lexical Functional Grammar (LFG; Kaplan and Bresnan 1982).[1] LFG is characterized by modular but interconnected linguistic descriptions, allowing researchers to make comparatively simple statements about particular domains of language. We build on XLE+Glue (Dalrymple et al., 2020) designed for exploring the syntax/semantics interface in LFG. However, while XLE+Glue has been used in formal semantic research to verify analyses, it has not been used in a task oriented setting as is typical in computational linguistics and NLP. This is the main drawback that this paper attempts to address by extending the XLE+Glue[2] pipeline to also incorporate reasoning tools, particularly, the Vampire theorem prover (Kovács and Voronkov, 2013). This allows us to lay the foundation for task-oriented semantic parsing (i.e., for NLI). We take inspiration from the seminal work on semantic parsing by Blackburn and Bos (2005) but also consider more recent proposals, particularly Haruta et al. (2020, 2022).

The contribution of this paper is a comprehensive implementation of semantic parsing that is grounded in a rigorous formal framework. The system is designed to be accessible and extensible, building on the discipline of grammar engineering. It enforces incremental development of linguistic analyses and enables testing the interplay of these analyses in a task-oriented fashion. The paper is structured as follows: section 2 presents LFG as the formal foundation of our system. Section 3 describes the full system, focusing on the novel interface between XLE+Glue and Vampire. Section 4 presents a qualitative evaluation of the system based on examples from the Fracas testsuite. Section 5 discusses some limitations and, finally, section 6 concludes.

[1] For recent introductions see Börjars (2020); Asudeh (2022), or Dalrymple (2023).

[2] https://github.com/Mmaz1988/xleplusglue/tree/2024_inference

Proceedings of the Second Workshop on the Bridges and Gaps between Formal and Computational Linguistics (BriGap-2), pages 35–51

2 Formal background

Lexical Functional Grammar (LFG) is a grammar formalism that is well-known for its formal rigor, allowing for a faithful computational implementation in the form of the Xerox Linguistics Environment (XLE; Crouch et al. 2017). From its beginnings, it has established itself as a crosslinguistically viable tool for describing language. This is particularly highlighted by the ParGram project (Butt et al., 2002) which produced computational LFG grammars illustrated by virtue of the ParGramBank (Sulger et al., 2013).[3]

2.1 Lexical Functional Grammar

The main appeal of LFG lies in its modular architecture that allows researchers and grammar engineers to make comparatively simple statements about linguistic facts in one domain of analysis (e.g., syntax), while maintaining a clearly defined mapping to other aspects of grammar (e.g., prosody, semantics). This design is sometimes called *parallel projection architecture*. To make this intuition more clear, let us first look at the syntactic component of LFG which consists of two individual projections: c(onstituent)- and f(unctional)-structure. C-structure is stated in terms of phrase structure rules and captures information about constituency and linear order. F-structure captures information about grammatical functions, such as SUBJ, OBJ (i.e., dependencies; Meurer 2017), as well as functional features such as `number` and `tense`. It is stated in terms of the quantifier-free logic of equality (Kaplan, 1989). More concretely, equality terms are *co-descriptively* added to phrase structure rules and lexical entries using the meta variables ↑ and ↓. ↑ points at the c-structure node of the mother and ↓ at the current node. Thus, The NP rule in (1), for example, states that the determiner and the noun equally contribute to the f-structure of the NP. There, the lexical entry of the determiner in (2) specifies a substructure, SPEC, subordinating the determiner to the content word.

(1)

S	→	NP (↑ SUBJ) = ↓	VP ↑=↓
NP	→	(D) ↑ = ↓	N ↑=↓
VP	→	V ↑ = ↓	AP (↑ PREDLINK) =↓

[3]Hosted at https://clarino.uib.no/iness/page (Rosén et al., 2012).

(2)

the	D	(↑ SPEC PRED) = 'the' %f_t = (GF+ ↑) $\lambda P.\lambda Q.\exists x[P(x) \wedge Q(x)]$: $(\uparrow_e \multimap \uparrow_t) \multimap (\uparrow_e \multimap$ %$f_t) \multimap$ %f_t
cat	N	(↑ PRED) = 'cat' $\lambda x.cat(x)$: $\uparrow_e \multimap \uparrow_t$
is	V	(↑ PRED) = 'be<(↑ PREDLINK)>(↑ SUBJ)' (↑ PREDLINK SUBJ) = (↑ SUBJ)
fast	A	(↑ PRED) = 'fast<(↑ SUBJ)>' $\lambda x.fast(x)$: $(\uparrow \text{SUBJ})_e \multimap \uparrow_t$

Generally, by resolving equalities, meta variables pointing at individual c-structure nodes are resolved to f-structure indices (many-to-one mapping). This process is visualized in Figure 2. As the figure indicates, the dependencies that LFG's f-structure captures are more articulated than classic dependencies as they can share structures across different PREDs, as annotated in the lexical entry for *is*. For a more comprehensive comparison involving further differences between f-structure and dependencies see Haug (2023).

2.2 Glue semantics

LFG's Glue semantics (Asudeh, 2023) specifies meaning representations, called meaning constructors (MCs).[4] They can be defined in two ways: co-descriptively, i.e., in parallel to c- and f-structure information in the lexicon (highlighted in blue in (2)) and phrase structure, or via description-by-analysis, which takes an assembled f-structure as input and rewrites it into a semantic representation. As we propose a hybrid approach (Wedekind and Kaplan, 1993), this warrants further explanation.

Description-by-analyis (DBA) rules provide independent way of introducing meaning constructors to syntactic representations, here f-structures (e.g, Andrews 2010).[5] Compared to co-descriptive semantics, they are more suitable to capture variation in the immediate syntactic and semantic context that affects semantic interpretation (Zymla, 2017). For example, the comparative complementizer *than* in Figure 2 is interpreted differently de-

[4]A more comprehensive introduction can be found in Dalrymple (1999). See also Dalrymple et al. (1993).

[5]DBA is technically a framework agnostic way of introducing meaning constructors that can be applied to different types of syntactic representations. Notably, It has been applied to Universal Dependency parses Findlay et al. (2023); Zymla (2018).

pending on its complement (e.g., elided VP vs overt VP). A simplified rule that produces the MC used in Figure 2 is illustrated in Figure 1. There, `#f...#j` are variables over f-structure nodes which are related via the given relation labels. Thus, the left-hand side can be understood as a query searching for a matching graph structure. Given that it matches the f-structure in Figure 2, the MC on the right is added to the premise set, essentially enabling the interpretation of the comparative clause.[6]

```
#f PREDLINK #h SUBJ #g & #h DEGREE 'comparative'
& #h ADJUNCT #a in_set #o OBL-COMP #i OBJ #j ==>
    #i GLUE
```

$$\lambda R.\lambda x.\lambda y.\exists\delta[R(\delta)(x) \wedge \neg R(\delta)(y)] : (\#g_d \multimap \#g_e \multimap \#f_t) \multimap (\#g_e \multimap \#j_e \multimap \#f_t).$$

Figure 1: DBA-rule for the comparative construction

MCs separate the logic of composition (linear logic; Girard 1995) and meaning language, allowing for some flexibility in the choice of the latter. Composition is resource-sensitive and flexible.[7] The Curry-Howard isomorphism (Curry et al., 1958; Howard, 1980) postulates parallels between lambda calculus operations and deduction processes in linear logic proofs. Thus, example (3) draws the parallel between function application and implication ($\multimap$) elimination, and example (4) describes the parallels between lambda abstraction and implication introduction. Consequently, Glue semantics is compatible with any meaning language whose combinatorial possibilities can be stated in λ-terms. We use λ-FOL (first-order logic) and λ-DRT (discourse representation theory; Kamp and Reyle 1993) to illustrate this.

(3) $$\frac{f : A \multimap B \qquad a : A}{f(a) : B} \multimap_E$$

(4) $$\frac{\begin{array}{c}[x : A]^i \\ \vdots \\ f(x) : B\end{array}}{\lambda x.f(x) : A \multimap B} \multimap_{I,i}$$

[6] The first argument of the comparative semantics is the semantic contribution of the adjective. However, the rule requires a predicate with a degree variable which is not provided by the lexical entry in (2). Reconciling this mismatch is discussed in section 4.

[7] Linear logic is commutative and non-associative by default. According to Moot and Retoré (2012), this is too flexible. However, this issue has been at least partially addressed in, e.g., Lev (2007); Findlay and Haug (2022); Zymla (2024) from a computational perspective. We do not explore this point further in this paper.

2.3 Reasoning

Reasoning based on XLE's LFG grammars has been pursued, for example, by Bobrow et al. (2007) and Lev (2007). These two works represent two general approaches respectively: i) reasoning via rewriting with a focus on intensional semantics (see also, e.g., Condoravdi et al. 2001), and ii) reasoning with theorem provers. The work presented here aligns with Lev's (2007) approach. Furthermore, it is inspired by more recent work in formal computational semantics by Haruta et al. (2020, 2022), who compose meanings via categorical grammar parsers and use the Vampire theorem prover to prove inferences by refutation (Kovács and Voronkov, 2013). As the focus of this paper is the educational value of computational tools based on formal linguistics, we also draw parallels to Blackburn and Bos (2005), who developed a conversational agent, CURT (clever use of reasoning tools), and highlighted how reasoning may affect dialogue interactions.

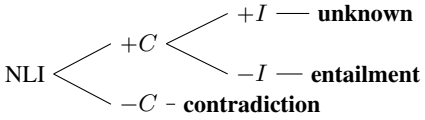
Consequently, two categories of reasoning can be considered: reasoning for natural language inference (NLI) and reasoning in dialogue. The first one is aptly exemplified by the Fracas testsuite (Cooper et al., 1996), which consists of examples like:

(5)
A Swede won a Nobel prize.
Every Swede is a Scandinavian.

Did a Scandinavian win a Nobel prize?
→ YES

As (5) shows, NLI examples consist of one or more premises, a conclusion, and a label corresponding to the entailment status of the conclusion (YES, NO, Don't know; MacCartney and Manning 2009). The goal of the task is to predict the correct label.

As part of their dialogue system, Blackburn and Bos (2005) establish informativity (+/-I) and consistency (+/-C) checks as essential tasks for reasoning in dialogue.[8] The NLI task can also be broken down to consist of these two tasks:

(6)


[8] While these checks are too strict to model the nuances of real world data, they provide useful insights into the incremental tracking of (shared) knowledge in dialogue settings.

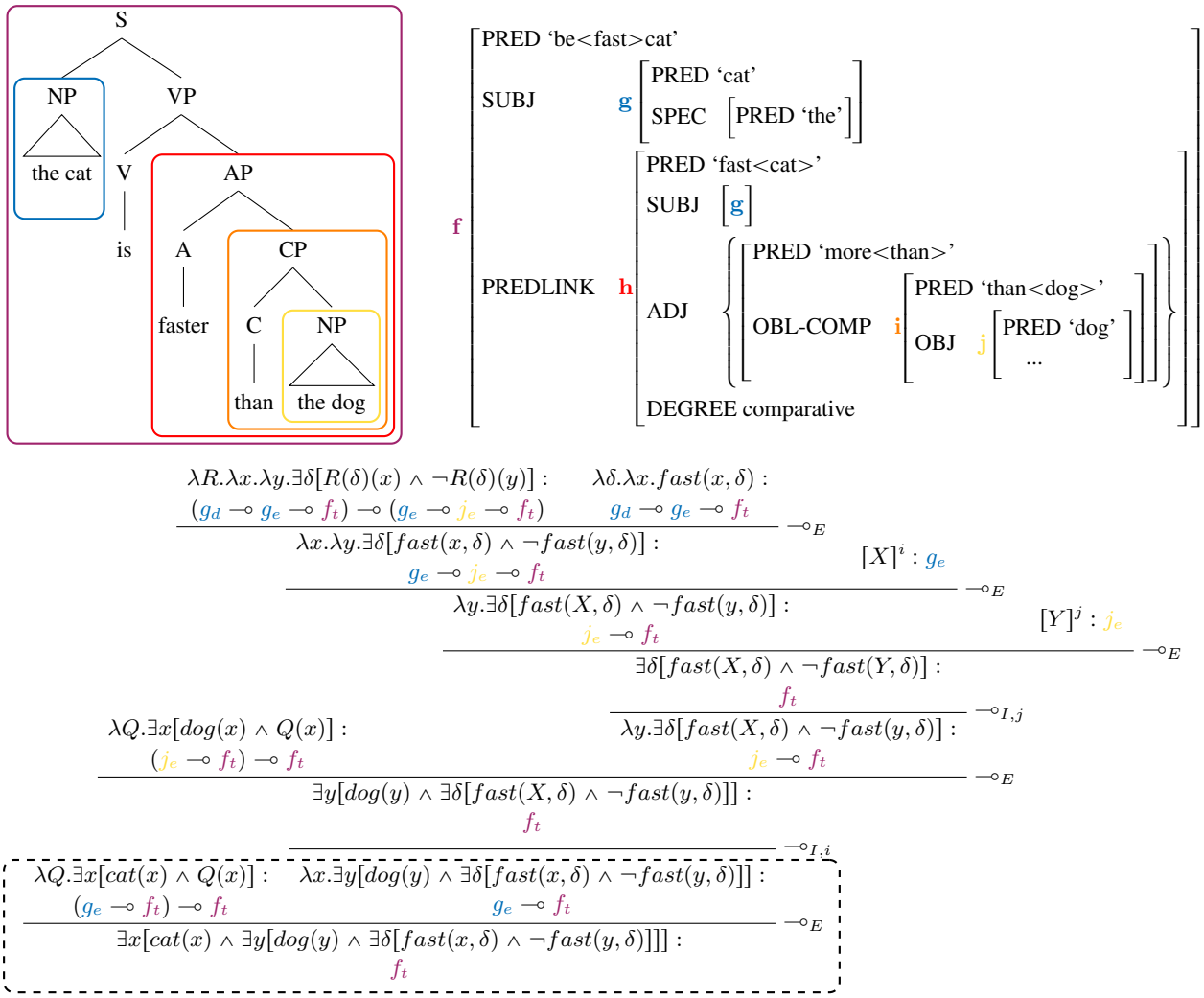


Figure 2: **LFG example derivation:** *The cat is faster than the dog.* This example illustrates the modular representation of c-structure, f-structure, and compositional semantics in LFG. C-structure captures linear order and constituency. F-structure abstracts away from surface form via a many-to-one mapping from c- to f-structure nodes. The semantics use f-structure indices as linear logic resources to encode combinatory possibilities which are stated in terms of a prooftree. Generally, hierarchical structures are broken down and re-assembled.

3 Computational implementation

This section presents our system that computationally implements the pipeline described in Figure (3). The main innovation presented in this paper is the use of LiGER (Linguistic graph expansion and rewriting) to mediate between syntax, compositional semantics and reasoning. Furthermore, we put some focus on the Blackburn and Bos (2005)-style interface to the Vampire theorem prover. However, we also briefly discuss the contribution of the other components.

3.1 Parsing via XLE+Glue

We build on computational Glue resources that have been developed in the past few years, primarily, the Glue semantics workbench (GSWB; Meßmer and Zymla 2018), a Glue prover heavily inspired by Lev's (2007) work, building on Hepple (1996), and an interface to XLE called XLE+Glue (Dalrymple et al., 2020).[9] Generally, the current main use of these tools is the verification of analyses with a focus on semantics and its interfaces, e.g., Przepiórkowski and Patejuk (2023). Recently, Butt et al. (2024) have presented a system that allows for the incorporation of prosodic information to disambiguate semantic analyses of questions in Urdu, thus, covering the full pipeline from speech signal to semantic parsing.

[9]While XLE itself is available only under a restrictive license, the semantic resources developed for it are all open source, including the new tools presented in this paper.

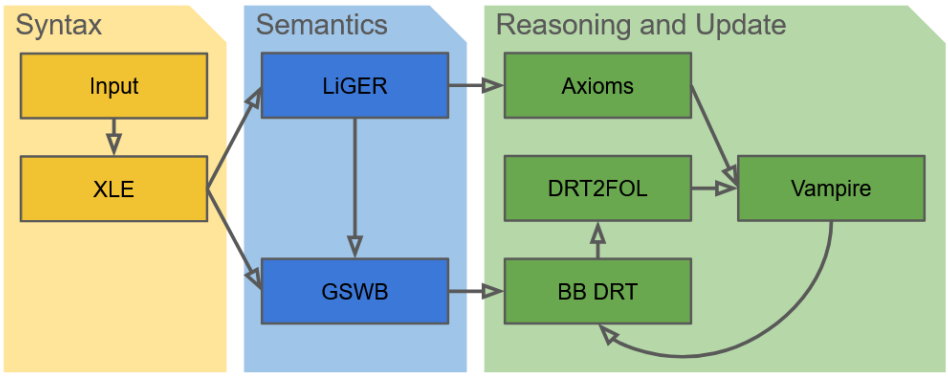


Figure 3: **The XLE+Glue pipeline:** three modular systems cover syntax, semantics, and reasoning. The syntax is specified in terms of LFG grammars written in the XLE. The XLE also specifies a core semantics that can be enriched and contextualized via LiGER. The GSWB calculates DRT-style meaning representations following Glue semantics principles. The DRSs are resolved in the reasoning module and translated into FOL for reasoning. LiGER optionally contributes additional axioms which are triggered by specific syntactic and semantic configurations to ensure correct reasoning.

.

The work presented in this paper is based on a newly developed Grammar for English that covers part of the Fracas testsuite, particularly, the section on quantifiers and the section on adjectives. Without going into detail, this grammar makes use of the various tools that XLE provides to develop larger-scale grammars, particularly, morphological analyzers, templates, parameterized rules, and more. To constrain ambiguities it makes use of OT-marks (loosely based on optimality theory). In terms of the presented syntactic analyses, it closely follows the large English ParGram grammar and Butt et al. (1999), but is extended with a co-descriptive semantics (cf. examples (1) and (2)). The semantics are resolved by the GSWB which generates Boxer-style DRT representations via a simple interface to Blackburn and Bos's Prolog code.[10]

3.2 Simple reasoning

For reasoning, DRSs are translated into first-order logic and then to the TPTP format (Sutcliffe et al., 2006). We built a Python interface that queries the Vampire theorem prover (Kovács and Voronkov, 2013) with positive and negative consistency and informativity checks. The positive checks rely on model building rather than satisfiability checking. Vampire supports this in addition to several other proof search strategies, but is mainly geared towards finding proofs via refutation.

(7) For some (set of conjoined) premise(s) p and a hypothesis q:

a.	$\neg(p \rightarrow q)$	+informative
b.	$p \rightarrow q$	-informative
c.	$p \wedge q$	+consistent
d.	$p \rightarrow \neg q$	-consistent

We extract meaningful labels from the Vampire output to present to users, concretely: the termination reason, whether a finite model was found, and the *SZS status* (Sutcliffe, 2008).[11] From these labels, we heuristically determine the success of the individual checks, and, consequently, the status with respect to the NLI task (cf. example (6)).

3.3 Extended reasoning mediated by LiGER

The system so far is essentially a re-implementation of Blackburn and Bos (2005) modeling the syntax/semantics interface in a different manner. While we believe that this has merits in its own right (particularly, the modularization of syntax and semantics), we extend its coverage with a principled approach to tackling more complex reasoning problems, such as those presented in Haruta et al. (2022). The key tool for this is LiGER which allows for the specification of rewrite rules to apply to f-structures. It plays two roles: i) non-invasively extending the semantics, and ii) determining relevant axioms needed for correct reasoning.

[10] Various aspects of the code have been adapted in accordance with the GNU license. The original files are available at https://www.let.rug.nl/bos/comsem/software2.html.

[11] Vampire's termination reason describes its result which also includes technical reasons, e.g., timeouts, whereas the SZS status focuses on the outcome of the reasoning process. Although there generally is a clear mapping from termination reason to SZS status, we use both for maximal informativity.

The first role is simply a rendering of the description-by-analysis idea presented in section 2.2. The important conceptual idea here is that the base grammar is self-sufficient, i.e., it produces semantic representations which can be optionally extended via DBA (see section 4).

The second role is inspired by Bobrow et al. (2007), who use DBA as an interface to external resources to enrich semantic representations. Concretely, we re-model Haruta et al.'s (2022) system for interpreting gradable adjectives and generalized quantifiers (a Python interface between a CCG parser and Vampire) into a system that is extensible and variable without the need to understand a complex code structure. To illustrate this, let us look at a core component of Haruta et al.'s (2022) analysis of gradable adjectives, particularly, their comparative use: the consistency postulate.

(8) (CP) $\forall x \forall y[\exists \delta[A(x,\delta) \wedge \neg A(y,\delta)] \rightarrow \forall \delta[A(y,\delta) \rightarrow A(x,\delta)]]$,
where A is an arbitrary gradable adjective.
(Haruta et al., 2022, p. 148)

The axiom in (8) is not intuitively part of the compositional semantics of an utterance, but rather is required for the semantics of gradable adjectives to fall out correctly. However, it essentially requires quantification over properties (indicated by the use of the variable A). This is accounted for in terms of a DBA rule of the following kind:

```
#a PRED %adj #a DEGREE 'comparative' ==>
  #a AXIOM   ∀x∀y[∃δ[%adj(x,δ) ∧ ¬%adj(y,δ)]
             → ∀δ[%adj(y,δ) → %adj(x,δ)]].
```

Figure 4: DBA-rule for extracting axioms

In essence, this (simplified) rule introduces an axiom based on the presence of an adjective with a comparative form and generates a CP axiom for that adjective. This information is integrated with the call to Vampire, as it affects how the prover is called. Concretely, reasoning about degrees following Haruta et al. (2022) requires arithmetic reasoning, a non-finite domain that eliminates model building as a proof search strategy.

In summary, LiGER is used on two fronts to extend the expressiveness of the compositional semantics and to trigger the axioms required to maintain correct results during inference. The LiGER output also affects the interface to Vampire directly to account for different input requirements and outputs for different kinds proofs.

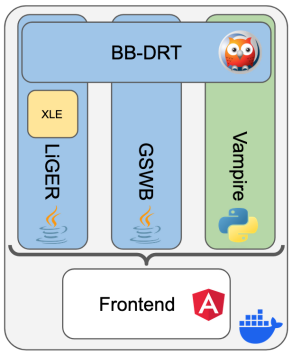


Figure 5: **System architecture:** a modularized service architecture that is accessed via a browser-based user interface.

3.4 Technical details

The whole system is couched in a modular service architecture, where individual modules are deployed in Docker containers. A browser-based application allows users to access the containers and links their functionalities (see Figure 5). The LiGER container also provides a lightweight interface to XLE.[12] This architecture enables cross-platform use of the system and minimizes the need for technical know-how. The Prolog code for BB-DRT is not deployed in a separate container but is copied across containers as it is relatively lightweight, reducing traffic across containers.

3.5 User interface and visualization

The system provides seperate interfaces to i) *parsing*, ii) *regression testing*, and iii) *inference*. Number i) and iii) are (partially) illustrated in Figure 7. The inference interface is inspired by Blackburn and Bos's (2005) conversational agent CURT. It provides access to the conversation history with the possibility to prune it. Furthermore, it allows the manual specification of axioms to test their effect on reasoning (not in the picture).

We use a glyph to optionally relay the detailed results of the inference checks to users. This is illustrated in Figure 6. The example there is an instance of a contradiction, as indicated by the refuted positive consistency check and successful negative consistency check. Although this makes informativity checks obsolete according to example (6), the glyph always displays all checks, highlighting the interplay between consistency and informativity.

[12] For licensing reasons, the XLE is not packed with the system but needs to be acquired independently and added to the repository.

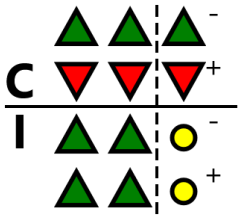


Figure 6: **Reasoning result glyph:** Green triangles indicate positive results (satisfiability), red triangles indicate negative results (refutation), and yellow symbol indicate unknown results (timeout/saturation). The glyph is horizontally separated into consistency (C) checks and informativity (I) checks with negative and positive polarity respectively. The vertical line separates satisfiability checks (left) and model building attempts (right).

Generally, the various interfaces are designed with explorative use in mind but they also enable incremental development of analyses with the regression testing interface, which is tailored towards developing new description-by-analysis rules. This supports the development of larger grammars.

4 Worked examples

We will now elaborate on the semantics we assume. This includes simpler cases including reasoning about properties and relations, but also more complex cases for which we employ a version of degree semantics following Haruta et al. (2022).

4.1 Basic semantic assumptions

Our semantics are based on a Neo-Davidsonian event semantics rendered in DRT. The first worked example is shown in Figure 7, demonstrating the correct reasoning for the problem in example (5).[13]

Due to quantifier ambiguity, the second hypothesis of (5) has two parses, presented in their equivalent FOL form in (9) and (10).[14] Here, the representation of *be* does not express anything meaningful, just that there is a *be*ing-eventuality (in the sense of Bach 1986) with two arguments.

(9) $\forall y[swede(y) \rightarrow \exists x[scandinavian(x) \wedge \exists e[be(e) \wedge arg1(e) = y \wedge arg2(e) = x]]]$

(10) $\exists x[scandinavian(x) \wedge \forall y[swede(y) \rightarrow \exists e[be(e) \wedge arg1(e) = x \wedge arg2(e) = y]]]$

[13] All analyses are laid out in detail in the appendix.

[14] Although the reading in (10) is not intuitively sensible, it does allow for the same inference. Nonetheless, this indicates the need for more fine-grained management of quantifier ambiguities, wich we leave for future work (first steps are taken in Zymla 2024).

For the inference in (5) to come out correctly, we need to add a meaning postulate (Zimmermann, 1999), as in (11).[15]

(11) $\forall x, y, e[be(e) \wedge arg1(e) = x \wedge arg2(e) = y \rightarrow x = y]$

This is to show that there is a wide range of axioms that one can consider adding to an analysis. Introducing meaning postulates via DBA-rules allows for the exploration of their impact on reasoning before integrating them into the grammar proper.

4.2 Layered analysis

In this section, we finalize the analysis of gradable adjectives following Haruta et al. (2022). However, note first that the Fracas testsuite contains several examples containing gradable adjectives that can be analyzed as simple properties, such as (12). Here, the challenge rather lies in modeling the syntax/semantics interface correctly to capture the modifying nature of the relative clause (e.g., Heim and Kratzer 1998).

(12) Some great tenors are Swedish.
Are there great tenors who are Swedish?

This extends to examples with more complicated constructions like the superlative:

(13) An Italian became the world's greatest tenor.
Was there an Italian who became the world's greatest tenor?

Thus, in many cases, reasoning via pattern matching is sufficient: as the noun phrases perfectly match, their exact semantics become less relevant.

However, often, gradable adjectives are challenging for automated reasoning because they are highly context sensitive. To make this intuition clear, first consider example (14), which illustrates the context sensitivity of the adjective *large*, whose meaning is mediated by the immediate context, here the modified noun. Generally, positive instances of gradable adjectives are evaluated based on contextually determined *comparison classes*.[16]

[15] We use a classic analysis of *be* also used in Blackburn and Bos (2005), which could also be stated directly in the semantics.

[16] The examples in the Fracas testsuite determine comparison classes in the immediate linguistic context, but comparison classes may be determined by the wider context, including extralinguistic cues (e.g., Kennedy and McNally 2005).

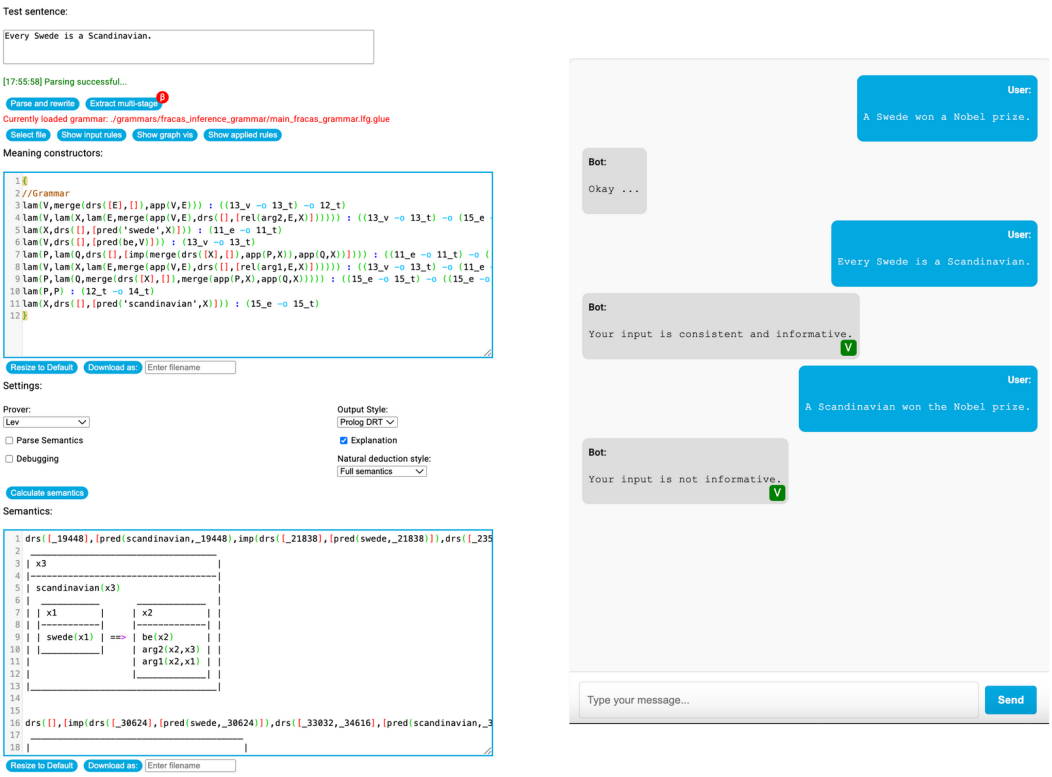


Figure 7: **XLE+Glue browser-based user interface:** On the left, the parsing interface is highly customizable and allows for the exploration of the compositional semantics underlying a parse. On the right, the chat interface allows for testing of the inference capabilities.

(14)
All mice are small animals.
Mickey is a large mouse.

Is Mickey a large animal? → NO

By making adjectives sensitive to a comparison class, the two different meanings of *large* in (14) can be explained. Concretely, we want to express that Mickey surpasses the threshold for a large mouse but not the threshold for a large animal. However, first we need to type-raise *large* to become a degree predicate.[17] The two steps can be encoded in terms of DBA-rules that extend the compositional semantics, as shown in Figure 8.

```
#a PRED %adj & #a DEGREE %d ==>
#a GLUE   λP.λδ.λx.%adj(x, δ) :
          (#a_e ⊸ #a_t) ⊸ (#a_d ⊸ #a_e ⊸ #a_t).

#n PRED %pred & #n ADJ #a PRED %adj &
#a DEGREE 'positive' ==>
#a GLUE   λP.λx.∃δ[δ > θ_%adj(%pred) ∧ P(δ)(x)]
          (#a_d ⊸ #a_e ⊸ #a_t) ⊸ #a_e ⊸ #a_t .
```

Figure 8: DBA-rule for positive gradable adjectives

In addition to the CP (see example (8)), we need to further specify the meaning of *large* a *positive* adjective, and *small*, a *negative* adjective:

(15)
(up) $\forall x, \delta'[large(x, \delta') \rightarrow \forall \delta''[\delta'' \leqslant \delta' \rightarrow large(x, \delta'')]]$
(down) $\forall x, \delta'[small(x, \delta') \rightarrow \forall \delta''[\delta' \leqslant \delta'' \rightarrow small(x, \delta'')]]$

These axioms say that if something is *large* to some degree δ it is also large to any degree smaller than that. Conversely, if something is small to a degree δ, it is also small to any larger degree.[18] Together with axioms for the antonym relation between *large* and *small*, the inference in (14) succeeds. An appropriate DBA-rule generalizes over positive and negative adjectives accordingly.

The type-raising rule also resolves the mismatch mentioned in footnote 6, allowing us to deal with comparatives as in (16) (see Appendix A.5).

[17] Accordingly, we have to slightly change the rule in Figure 1, which we simplified for sake of exposition. We accept this extra step to preserve the integrity of the core grammar.

[18] A reviewer points out that Haruta et al. (2022) show that the CP, (8), follows from *up* and *down* questioning the necessity of these axioms. However, the inverse is not true. Thus, they are required for cases like (14) which do not contain explicit comparatives (see Appendix A.4).

(16)
The PC-6082 is faster than every ITEL computer.
The ITEL-ZX is an ITEL computer.

Is the PC-6082 faster than the ITEL-ZX? → YES

5 Limitations

While the present system is developed in a task-oriented fashion, it suffers from the usual drawbacks of formal computational linguistics, such as a lack of robustness (particularly, with respect to unseen data), and tedious ambiguity management (Bunt, 2008), particularly as one attempts to scale up grammars (Flickinger et al., 2017). Thus, the present system should not be seen as ready for real NLP applications (yet). Nonetheless, it contributes to closing the gap between formal and computational linguistics, by making the latter more accessible to practitioners of the former, which should be mutually beneficial for both disciplines (e.g., Bender 2008; King 2016). Furthermore, through regression testing (Chatzichrisafis et al., 2007), the grammar presented here, as well as the system as a whole, are continuously expanded.

Although we see the modular architecture of LFG as a benefit regarding the explainability of different aspects of language that affect semantic interpretation, the reliance on XLE can be a drawback. To address this, we also provide an integration of the semantics tool with Stanza's dependency parser (Qi et al., 2020).[19] However, we do not yet provide a reasonably-sized set of semantic rules ready for inference testing. This is an avenue for future work.

6 Summary

This paper presents an open source computational resource that enables the exploration of computational semantics and reasoning through the lense of LFG's Glue semantics. Its hallmarks are i) an interface to the Vampire theorem prover, ii) a principled system for exploring formal semantics and their use in automated reasoning at various levels of complexity, and iii) a grounding in the seminal work on formal approaches to natural language inference by Blackburn and Bos (2005). These hallmarks come with various avenues for future work. On the LFG-side, Butt et al. (2024) integrate prosodic information into a fully formal system for semantic parsing, thus, enabling a comprehensive exploration of formal linguistic insights from the speech signal to reasoning. Linking their work with present resources would grant an even deeper understanding of the interplay between syntax, prosody, and semantic interpretation.

On the reasoning side, the present work not only allows users to explore Blackburn and Bos (2005), but also extends to their (unpublished) material on discourse representation theory, enabling, for example, the exploration of anaphora and presuppositions. Furthermore, this paves the way for the exploration of discourse relations (Asher and Lascarides, 2003) further down the line.

More speculatively, we believe that aiming for a hybrid system, where machine-learning methods are used to intervene at various levels of linguistic analysis (syntax, semantics, pragmatics) could be mutually beneficial, potentially increasing the explainability of large language models (as a representative of the most prevalent machine learning methods in current NLP), but, importantly also improving the robustness of the present system, e.g., by helping with disambiguation and possibly by modulating the reasoning process.

All in all, we present a principled and thus instructive way to explore formal semantics and reasoning. The system can be locally deployed as a browser app with an accessible user interface, making it interesting for a broad audience within computational linguistics and adjacent fields.

[19] https://github.com/Mmaz1988/xleplusglue/tree/2025_xleplusud

Acknowledgments

We thank the German Research Foundation (DFG) for financial support within the project D02 of the SFB/Transregio 161.

References

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

Nicholas Asher and Alex Lascarides. 2003. *Logics of Conversation*. Cambridge University Press.

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

Ash Asudeh. 2023. Glue semantics. In *Handbook of Lexical Functional Grammar*, pages 651–697. Language Science Press, Berlin.

Emmon Bach. 1986. The algebra of events. *Linguistics and Philosophy*, 9(1):5–16.

Emily M. Bender. 2008. Grammar engineering for linguistic hypothesis testing. In *Proceedings of the Texas Linguistics Society X Conference: Computational Linguistics for Less-Studied Languages*, pages 16–36, Stanford, CA. CSLI Publications.

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, Stanford, CA.

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.

Kersti Börjars. 2020. Lexical-functional grammar: An Overview. *Annual Review of Linguistics*, 6:155–172.

Harry Bunt. 2008. Semantic underspecification: Which technique for what purpose? In *Computing Meaning*, pages 55–85. Springer.

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.

Miriam Butt, Tracy Holloway King, María-Eugenia Niño, and Frédérique Segond. 1999. *A Grammar writer's cookbook*. CSLI Publications.

Nikos Chatzichrisafis, Dick Crouch, Tracy Holloway King, Rowan Nairn, Manny Rayner, and Marianne Santaholma. 2007. Regression testing for grammar-Based systems. In *Proceedings of the Grammar Engineering Across Frameworks (GEAF07) Workshop*, pages 128–143, Stanford, CA. CSLI Publications.

Cleo Condoravdi, Dick Crouch, John Everett, Valeria Paiva, Reinhard Stolle, Danny Bobrow, and Martin van den Berg. 2001. Preventing existence. In *Proceedings of the International Conference on Formal Ontology in Information Systems - Volume 2001*, FOIS '01, pages 162–173, New York, NY, USA. ACM.

Robin Cooper, Dick Crouch, Jan van Eijck, Chris Fox, Josef van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, and Steve Pulman. 1996. Using the framework. Technical Report LRE 62-051 D-16, The FraCaS Consortium.

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

Haskell Brooks Curry, Robert Feys, William Craig, J. Roger Hindley, and Jonathan P. Seldin. 1958. *Combinatory logic*, volume 1. North-Holland Amsterdam.

Mary Dalrymple. 1999. *Semantics and syntax in lexical functional grammar: The resource logic approach*. The MIT Press, Cambridge, MA.

Mary Dalrymple, editor. 2023. *Handbook of Lexical Functional Grammar*. Number 13 in Empirically Oriented Theoretical Morphology and Syntax. Language Science Press, Berlin.

Mary Dalrymple, John Lamping, and Vijay Saraswat. 1993. LFG semantics via constraints. In *Proceedings of the Sixth Conference on European Chapter of the Association for Computational Linguistics (EACL '93)*, page 97–105, USA. Association for Computational Linguistics.

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*, pages 89–108, Stanford, CA. CSLI Publications.

Jamie Y. Findlay and Dag T. T. Haug. 2022. Managing scope ambiguities in Glue via multistage proving. In *Proceedings of the LFG'22 Conference*, pages 144–163, Konstanz, Germany. PubliKon.

Jamie Y Findlay, Saeedeh Salimifar, Ahmet Yıldırım, and Dag T. T. 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 M. 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, Dordrecht.

Jean-Yves Girard. 1995. Linear logic: Its syntax and semantics. *London Mathematical Society Lecture Note Series*, pages 1–42.

Izumi Haruta, Koji Mineshima, and Daisuke Bekki. 2020. Logical inferences with comparatives and generalized quantifiers. In *Proceedings of the 58th Annual Meeting of the ACL: Student Research Workshop*, pages 263–270, Online. ACL.

Izumi Haruta, Koji Mineshima, and Daisuke Bekki. 2022. Implementing natural language inference for comparatives. *Journal of Language Modelling*, 10(1).

Dag T. T. Haug. 2023. LFG and dependency grammar. In Mary Dalrymple, editor, *Handbook of Lexical Functional Grammar*, chapter 43, pages 1829–1859. Language Science Press, Berlin.

Irene Heim and Angelika Kratzer. 1998. *Semantics in generative grammar*. Blackwell, Oxford, UK.

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.

William A. Howard. 1980. The formulae-as-types notion of construction. *To HB Curry: Essays on Combinatory Logic, Lambda calculus, and Formalism*, 44:479–490.

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 of *Studies in Linguistics and Philosophy*. Kluwer Academic Publishers, Dordrecht/Boston.

Ronald M. Kaplan. 1989. The formal architecture of Lexical-Functional Grammar. *Journal of Information Science and Engineering*, 5(4):305–322.

Ronald M. Kaplan and Joan Bresnan. 1982. Lexical-Functional Grammar: A formal system for grammatical representation. In Mary Dalrymple, Ronald M. Kaplan, John T. Maxwell III, and Annie Zaenen, editors, *Formal Issues in Lexical-Functional Grammar*, pages 1–102. Stanford University, Stanford, CA.

Christopher Kennedy and Louise McNally. 2005. Scale structure, degree modification, and the semantics of gradable predicates. *Language*, pages 345–381.

Tracy Holloway King. 2016. Theoretical linguistics and grammar engineering as mutually constraining disciplines. In *Proceedings of the Joint 2016 Conference on Head-driven Phrase Structure Grammar and Lexical Functional Grammar, Polish Academy of Sciences, Warsaw, Poland*, pages 339–359, Stanford, CA. CSLI Publications.

Laura Kovács and Andrei Voronkov. 2013. First-order theorem proving and vampire. In *International Conference on Computer Aided Verification*, pages 1–35. Springer.

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

Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *Proceedings of the eight International Conference on Computational Semantics*, pages 140–156.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

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*, pages 249–263, Stanford, CA. CSLI Publications.

Paul Meurer. 2017. From LFG structures to dependency relations. *Bergen Language and Linguistics Studies*, 8(1).

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.

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

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

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 Çetinoğlu, I Wayan Arka, and Meladel Mistica. 2013. ParGramBank: The ParGram parallel treebank. In *ACL*, pages 550–560.

Geoff Sutcliffe. 2008. The SZS ontologies for automated reasoning software. In *Proceedings of the LPAR Workshops: Knowledge Exchange: Automated Provers and Proof Assistants (KEAPA 2008), and the 7th International Workshop on the Implementation of Logics (IWIL-2008)*, volume 418 of *CEUR Workshop Proceedings*, pages 38–49. CEUR-WS.org.

Geoff Sutcliffe, Stephan Schulz, Koen Claessen, and Allen Van Gelder. 2006. Using the TPTP language for writing derivations and finite interpretations. In *Automated Reasoning – IJCAR 2006*, volume 4130 of *Lecture Notes in Computer Science*, pages 67–81, Seattle, WA, USA. Springer.

Jürgen Wedekind and Ronald M. Kaplan. 1993. Type-driven semantic interpretation of f-structures. In *Proceedings of the Sixth EACL*, pages 404–411.

Thomas Ede Zimmermann. 1999. Meaning postulates and the model-theoretic approach to natural language semantics. *Linguistics and Philosophy*, 22(5):529–561.

Mark-Matthias Zymla. 2017. Comprehensive annotation of cross-linguistic variation in tense and aspect categories. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS) — Long Papers*, Montpellier, France.

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

A Worked examples

A.1 Example (12)

(17) a. Some great tenors are Swedish.
$\exists x, e[tenor(x) \wedge great(x) \wedge swedish(e) \wedge be(e) \wedge arg1(e) = x]$

b. There are some great tenors who are Swedish.
$\exists x, y, e_1, e_2[tenor(x) \wedge great(x) \wedge swedish(e) \wedge be(e_1) \wedge arg1(e) = x \wedge be(e_2) \wedge arg1(e_2) = y \wedge arg2(e_2) = x]$

Generated semantics:

```
 _____________
| x2 x1       |
|-------------|
| great(x2)   |
| tenor(x2)   |
| swedish(x1) |
| be(x1)      |
| arg1(x1,x2) |
|_____________|
```

```
 _____________
| x2 x4 x3 x1 |
|-------------|
| great(x2)   |
| tenor(x2)   |
| swedish(x4) |
| be(x4)      |
| arg1(x4,x2) |
| be(x1)      |
| arg1(x1,x3) |
| arg2(x1,x2) |
|_____________|
```

Used axioms:

```
fof(be_axiom, axiom,
   ![X,Y,Z] : ((be(X) & arg1(X,Y) & arg2(X,Z)) => Y = Z)).
```

Inference output:

	+consistent	-consistent	+informative	-informative
Termination reason	+	+	-	+
SZS status	+	+	-	+
Model found	+	+	-	+

A.2 Example (5)

(18) a. A Swede won a Nobel prize
$\exists x, e[swede(x) \wedge prize(y) \wedge win(e) \wedge arg1(e) = x \wedge arg2(e) = y]$

b. Every Swede is a Scandinavian
$\forall x, e[swede(x) \rightarrow \exists e[scandinavian(e) \wedge be(e) \wedge arg1(e) = x]]$

c. A Scandinavian won the Nobel prize.
$\exists x, e[scandinavian(x) \wedge prize(y) \wedge win(e) \wedge arg1(e) = x \wedge arg2(e) = y]$

Generated semantics:

```
 _____________
| x3 x2 x1    |
|-------------|
| swede(x3)   |
| prize(x2)   |
| win(x1)     |
| arg1(x1,x3) |
| arg2(x1,x2) |
|_____________|
```

```
 ___________________________________________
|                                           |
|-------------------------------------------|
|  ___________       __________________     |
| | x1        |     | x3 x2            | |
| |-----------|     |------------------| |
| | swede(x1) | ==> | scandinavian(x3) | |
| |___________|     | be(x2)           | |
|                   | arg1(x2,x1)      | |
|                   | arg2(x2,x3)      | |
|                   |__________________| |
|___________________________________________|
```

```
 _____________
| x3 x2 x1    |
|-------------|
| scandi(x3)  |
| prize(x2)   |
| win(x1)     |
| arg1(x1,x3) |
| arg2(x1,x2) |
|_____________|
```

Used axioms:

```
fof(be_axiom, axiom,
    ![X,Y,Z] : ((be(X) & arg1(X,Y) & arg2(X,Z)) => Y = Z)).
```

Inference output:

	+consistent	-consistent	+informative	-informative
Termination reason	+	+	-	+
SZS status	+	+	-	+
Model found	+	+	-	+

A.3 Example (13)

(19) a. An Italian became the greatest tenor.
$\exists x, y, e[italian(x) \wedge great(y) \wedge tenor(y) \wedge become(e) \wedge arg2(e) = x \wedge arg1(e) = y]$

b. There was an Italian who became the greatest tenor.
$\exists x, y, e[italian(x) \wedge great(y) \wedge tenor(y) \wedge become(e) \wedge arg1(e) = x \wedge arg2(e) = y]$

```
 _____________
| x2 x3 x1    |
|-------------|
| italian(x2) |
| great(x3)   |
| tenor(x3)   |
| become(x1)  |
| arg2(x1,x3) |
| arg1(x1,x2) |
|_____________|
```

```
 ________________
| x3 x5 x4 x2 x1 |
|----------------|
| italian(x3)    |
| great(x5)      |
| tenor(x5)      |
| become(x4)     |
| arg2(x4,x5)    |
| arg1(x4,x3)    |
| be(x1)         |
| arg2(x1,x3)    |
| arg1(x1,x2)    |
|________________|
```

Used axioms:

```
fof(be_axiom, axiom,
    ![X,Y,Z] : ((be(X) & arg1(X,Y) & arg2(X,Z)) => Y = Z)).
```

Inference output:

	+consistent	-consistent	+informative	-informative
Termination reason	+	+	-	+
SZS status	+	+	-	+
Model found	+	+	-	+

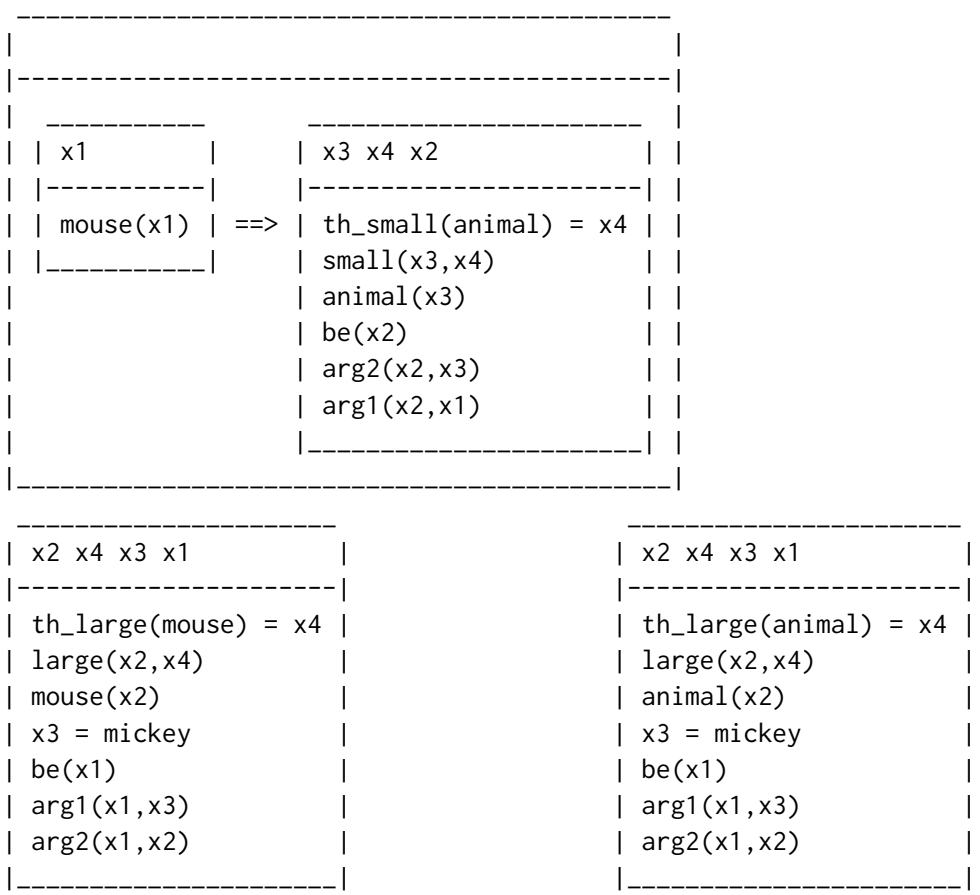
A.4 Example (14)

(20) a. All mice are small animals.
$\forall x[mouse(x) \rightarrow \exists y, \delta, e[animal(y) \wedge small(y, \delta) \wedge \theta_{\texttt{small}}(animal) = \delta \wedge be(e) \wedge arg1(e) = x \wedge arg2(e) = y]$

b. Mickey is a large mouse.
$\exists x, y, \delta, e[x = Mickey \wedge mouse(y) \wedge large(y, \delta) \wedge \theta_{\texttt{large}}(mouse) = \delta \wedge be(e) \wedge arg1(e) = x \wedge arg2(e) = y]$

c. Mickey is a large animal.
$\exists x, y, \delta, e[x = Mickey \wedge animal(y) \wedge large(y, \delta) \wedge \theta_{\texttt{large}}(animal) = \delta \wedge be(e) \wedge arg1(e) = x \wedge arg2(e) = y]$

Generated semantics:



Used axioms:

```
% adjectives
tff(large_type, type, large: ($i * $int) > $o).
tff(small_type, type, small: ($i * $int) > $o).

%comparison classes
tff(large_cc_type, type, th_large: $i > $int).
tff(small_cc_type, type, th_small: $i > $int).

%predicatives
tff(be_type, type, be: $i > $o).
tff(arg1_type, type, arg1: ($i * $i) > $o).
tff(arg2_type, type, arg2: ($i * $i) > $o).

%nouns
tff(mouse_type, type, mouse: $i > $o).
tff(animal_type, type, animal: $i > $o).

%names
tff(pn_type1, type, mickey: $i).
tff(pn_type2, type, minni: $i).
tff(pn_type3, type, animal: $i).
tff(pn_type4, type, mouse: $i).

%predicative meaning postulate
tff(axiom1,axiom,(![A : $i]: (![B : $i]: (![C : $i]: ((be(A) & (arg1(A,B) & arg2(A,C))) => (B = C))))
    )).

%from events to adjectives
tff(axiom2,axiom,(![A : $i]: (![B : $i]: (![C : $int]: ((arg1(A,B) & large(A,C)) => large(B,C)))))).
tff(axiom3,axiom,(![A : $i]: (![B : $i]: (![C : $int]: ((arg1(A,B) & small(A,C)) => small(B,C)))))).

tff(axiom4,axiom,(![A : $i]: (?[B : $int]: (large(A,B) & ~ (?[C : $int]: ($greater(C,B) & large(A,C))
```

```
    )))))).
tff(axiom5,axiom,(![A : $i]: (![B : $int]: (large(A,B) <=> (![C : $int]: ($lesseq(C,B) => large(A,C))
    )))))).

tff(axiom6,axiom,(![A : $i]: (?[B : $int]: (small(A,B) & ~ (?[C : $int]: ($greater(B,C) & small(A,C))
    )))))).
tff(axiom7,axiom,(![A : $i]: (![B : $int]: (small(A,B) <=> (![C : $int]: ($lesseq(B,C) => small(A,C))
    )))))).

%comparison class
tff(cclass,axiom, (![D: $int, D1: $int]:((th_large(animal) = D & th_small(animal) = D1) => $less(D1,D
    )))).

%antonym
tff(antonym1, axiom, ![X: $i, D: $int]: (large(X,D) <=> ~small(X,D))).
tff(antonym2, axiom, ?[X: $i, D: $int]: (large(X,D)) <=> ?[X1: $i, D1: $int]: (small(X1,D1))).
```

Inference output:

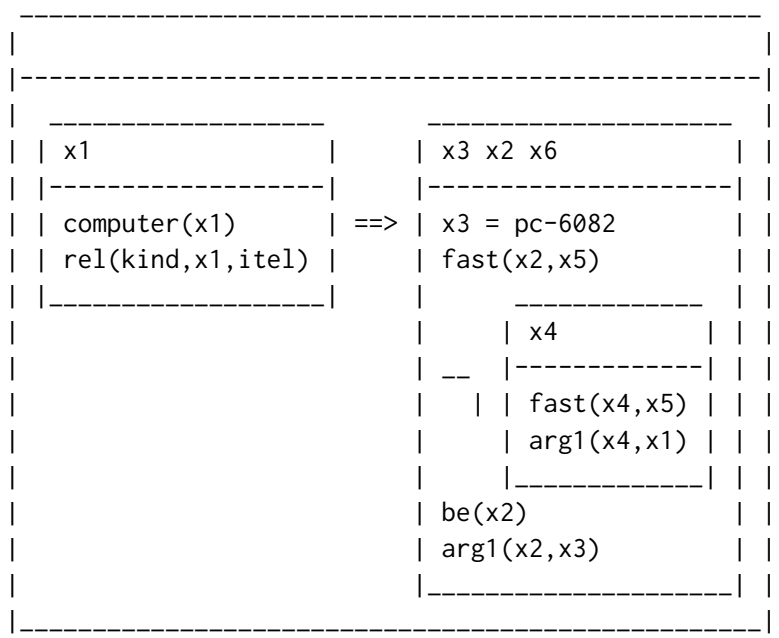
	+consistent	-consistent	+informative	-informative
Termination reason	-	?	?	?
SZS status	-	?	?	?

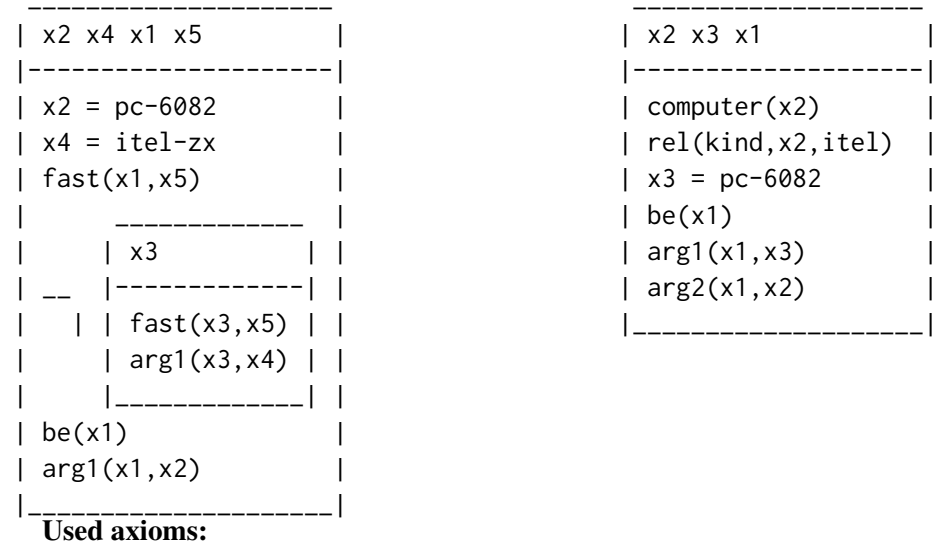
A.5 Example (16)

(21) a. The PC-6082 is faster than every ITEL computer.
$\forall y[computer(y) \wedge kind(y, itel) \rightarrow \exists x, \delta, e[fast(e, \delta) \wedge arg1(e) = x \wedge x =$ PC-6082 $\wedge$ $arg2(e) = y]]$

b. The ITEL-ZX is an ITEL computer.
$\exists x, y, e[x =$ ITEL-ZX $\wedge computer(y) \wedge kind(y, itel) \wedge be(e) \wedge arg1(e) = x \wedge arg2(e) = y]$

c. The PC-6082 is faster than the ITEL-ZX.
$\exists x, y, \delta, e[fast(e, \delta) \wedge x =$ PC-6082 $\wedge\ y =$ ITEL-ZX $\wedge\ arg1(e) = x \wedge arg2(e) = y \wedge \neg\exists e'[fast(e', \delta) \wedge arg1(e') = y]]$





Used axioms:

```
%adjectives
tff(fast_type, type, fast: ($i * $int) > $o).

%modifiers
tff(kind_type, type, kind: ($i * $i) > $o).

%predicatives
tff(be_type, type, be: $i > $o).
tff(arg1_type, type, arg1: ($i * $i) > $o).
tff(arg2_type, type, arg2: ($i * $i) > $o).

%nouns
tff(computer_type, type, computer: $i > $o).

%names
tff(pn_type1, type, 'pc-6082': $i).
tff(pn_type2, type, 'itel-zx': $i).

%predicative meaning postulate
tff(axiom1,axiom,(![A : $i]: (![B : $i]: (![C : $i]: ((be(A) & (arg1(A,B) & arg2(A,C))) => (B = C))))
    )).

%from events to adjectives
tff(axiom2, axiom, (![A: $i]: (![B: $i]: (![C: $int]: ((arg1(A,B) & fast(A,C)) => fast(B,C)))))).

tff(axiom3, axiom, (![A: $i]: (![B: $int]: (fast(A,B) <=> (![C: $int]: ($lesseq(C,B) => fast(A,C)))))
    )).
tff(axiom4, axiom, (![A: $i]: (![B: $i]: ((?[C: $int]: (fast(A,C) & ~fast(B,C))) => (![D: $int]: (
    fast(B,D) => fast(A,D)))))).
tff(axiom5, axiom, (![A: $i]: (?[B: $int]: (fast(A,B) & ~(?[C: $int]: ($greater(C,B) & fast(A,C)))))
    ).
```

Inference output:

	+consistent	-consistent	+informative	-informative
Termination reason	?	?	-	?
SZS status	?	?	-	?

- For comparatives, only a partial answer based on the refutation of the positive informativity check is given.
- ? refers to proof searches that have timed out.
- The search strategy differs in this examples as model building is not an option.