Regression-Tested Compositional Semantics: A Graphical Development Environment for Glue and Description-by-Analysis

Mark-Matthias Zymla, Kascha Kruschwitz
University of Konstanz
{mark-matthias.zymla|kascha.kruschwitz}@uni-konstanz.de

Abstract

We present a platform for developing compositional semantic annotations for formal syntactic representations that allows users to interact with and explore annotations and to track their progress and quality. For this, we provide several forms of visualizations and take inspiration from research in linguistic treebanking. Thus, we contribute to the development of formal semantic parsers and corresponding meaning banks. The system is designed with a regression testing paradigm in mind and provides support for NLI so that the created semantic resources can be developed and validated in a task-driven environment. We defend this paradigm in comparison to modern approaches to semantic parsing that are mainly evaluated on the basis of gold standard annotations.

1. Introduction

Formal semantic parsing has received little attention in the wake of machine-learning and deep learning driven approaches. This is despite the fact that such approaches suffer from various shortcomings. For example, they do not really solve the problem of the inability to properly generalize to out-of-domain data, sometimes even inventing completely nonsensical analyses (see, e.g., Haug et al. 2023; Zhang et al. 2022, 2025). However, their continuing popularity suggests that the benefits of these methods outweigh their flaws relative to formal approaches.

We believe that this is not entirely justified, as formal approaches provide a level of transparency and reliability that is not matched by quantitative methods.[1] Thus, we provide a system for deriving semantic parsers built around formal methods in computational linguistics.

Our system deviates from popular approaches in semantic parsing, concretely, the comparison of analyses against a gold standard (see Zhang et al. 2025 for an overview of methods). Rather, we follow a task-driven approach, evaluating semantic representations in the context of automated reasoning, in particular, the natural language inference (NLI) task (MacCartney and Manning, 2009), an ingredient of many NLP tasks (Bos, 2009).

According to Bos (2009), building a platform for developing and evaluating formal semantic parsers in this way is a major undertaking requiring a level of expertise in various disciplines, such as formal logic, computer science, linguistics, and NLP. This is certainly true. Thus, we believe it is crucial to focus on making computational linguistic resources accessible, such that linguistic analyses can be developed with less concern for the more technical aspects of semantic parsing.[2] We present an attempt at this by providing a browser-based application for developing compositional semantic analyses derived from syntactic parses, such as, for example, illustrated for the grammars developed in the framework of Lexical Functional Grammar in Zymla et al. (2025b) or for Universal Dependency analyses in (Findlay et al., 2023).[3]

Our application supports prototyping individual analyses and *regression testing* (Chatzichrisafis et al., 2007), enabling users to test analyses at a larger scale. However, we acknowledge the flaws of formal semantic parsing as they become more critical as coverage of a grammar is extended. We hypothesize that testing these boundaries can shed more light on the linguistic aspects of reasoning. Thus, we propose an incremental semantic parsing paradigm that matches particular semantic analyses with particular inference patterns, such that these analyses are covered by separately applicable rules. More concretely, we assume that building multiple interoperable medium-sized grammars, targeting specific inference patterns, is more beneficial than a broad coverage semantic parser built on a single set of grammatical and semantic rules. The tools presented here serve to foster such grammars and to generate corresponding meaning banks that may be beneficial in downstream tasks.

This paper is structured as follows: In section 2, we lay out the necessary background. We present our system in section 3, and in section 4 we describe incremental parsing as our ideal vision for using the system. Section 6 concludes.

[1] We do not mean reliability in the sense of robustness but rather in the sense that, e.g., interactions between semantic operators can be deterministically predicted.

[2] As is achieved in a number of linguistic annotation tools, e.g., (Stenetorp et al., 2012; Liu et al., 2017).

[3] The system is available at https://github.com/Mmaz1988/xleplusglue/tree/dmr2026_regression_testing. Using XLE grammars requires access to XLE binaries for Linux managed via the University of Konstanz. Please contact the authors for assistance.

Proceedings of The Seventh International Workshop on Designing Meaning Representations (DMR 2026) @ LREC 2026, pages 183–197

2. Background

The main goal of this paper is to present a tool that facilitates semantic parsing and the generation of corresponding tree-, or rather, meaning banks, based on formal linguistics. Zymla et al. (2025b) present a qualitatively tested approach to formal semantic parsing for NLI based on XLE+Glue, a system developed in the context of Lexical Functional Grammar (LFG; Bresnan et al. 2015; Dalrymple 2001, 2023). As described there and in Zymla et al. (2025a), XLE+Glue combines three resources: i) Linguistic Graph Expansion and Rewriting (LiGER), a resource for description-by-analysis, ii) the Glue semantics workbench (GSWB), a resource for calculating Glue derivations, and iii) an interface to the Vampire theorem prover.[4] Thus, they provide a workflow that allows for semantic parsing that is tested in the context of natural language inference (MacCartney and Manning, 2009). These tools are built around Glue semantics, a formalization of the syntax/semantics interface, which we introduce in the next section.

2.1. Glue and description-by-analysis

Glue semantics or Glue has recently received renewed interest (Meßmer and Zymla, 2018; Dalrymple et al., 2020) and the underlying tools continue to mature (Zymla et al., 2025a). They have been used, for example, to model aspects of the prosody-meaning mapping (Butt et al., 2024) or to verify complex formal analyses (Przepiórkowski and Patejuk, 2023).

The main idea of Glue is that compositionality is guided by the resource-sensitive linear logic (Dalrymple, 1999), such that the meaning of an expression and the way in which it interacts with other pieces of meaning are captured separately. The corresponding basic semantic representations are called meaning constructors.[5] Importantly, Glue is compatible with various syntactic and meaning representations, providing a very general view of the syntax/semantics interface that captures many of its fundamental aspects, while abstracting away from potential syntactic and semantic idiosyncrasies of specific formalisms.[6]

There exist at least two ways to introduce meaning constructors to a syntactic framework. The first is called *co-description*, and is somewhat LFG-specific in that semantic representations are stored lexically (or sometimes, as part of syntactic rules; Dalrymple 2001). The tools presented here mainly concern the second possibility: *description-by-analysis* (DBA), which models the syntax/semantics interface as rewrite rules applied to syntactic structures (e.g., Andrews 2008). Figure 1 provides an example of how to use DBA rules to add compositional semantic information to a UD analysis. As shown there, rules are applied sequentially to some input, adding additional structure and introducing meaning constructors. These can then be used for semantic composition.

Bobrow et al. (2007); Crouch and King (2006); Crouch (2005) provide early approaches to semantic parsing with DBA based on LFG but do not employ Glue semantics per se. Lev (2007) more closely resembles the present approach in that it uses DBA to produce Glue meaning constructors. It serves as inspiration for XLE+Glue (Dalrymple et al., 2020; Zymla et al., 2025a), which we build upon in this paper. More recent approaches to DBA (Findlay et al., 2023; Zymla et al., 2025b) have also been built around (parts of) these tools. All these approaches to DBA use graph rewriting; thus, in principle, DBA can be applied to any linguistic representations that can be cast as directed graphs. Consequently, it synergizes well with the flexibility of Glue.

2.2. Semantic parsing for meaning banking

More generally, our tools contribute to grammar development (Butt et al., 1999) and treebanking (Marcus et al. 1993; or more specifically, meaning banking), which are essential aspects of traditional CL methods and have heavily influenced the design of linguistic annotations in the past.

Meaning banks exist for various formalisms, such as the DeepBank for HPSG (Flickinger et al., 2012) or the Groningen meaning bank for a version of DRT (Abzianidze et al., 2020). Further, the abstract meaning representation (AMR) effort provides annotations via the LDC (Knight et al., 2020) and unified meaning representations (UMR) built on the AMR formalism, providing corresponding data sets (Bonn et al., 2024). However, only a small portion of modern meaning banks still rely on formal parsers for data annotation, despite many meaning banks originating from these efforts (e.g., the predecessor of the PMB, the Groningen mean-

[4] The name XLE+Glue stems from the fact that these tools have been developed mainly in the context of providing semantic representations for the Xerox linguistics environment (XLE; Crouch et al. 2017). However, the system has been extended to work with UD parses by Stanza, and, more generally, other syntactic parsers may be substituted.

[5] We do not explain the details of Glue semantics here. A comprehensive overview can be found in Asudeh (2023), or the recent LFG handbook (Dalrymple, 2023). For a practical introduction, see Zymla et al. 2025b.

[6] For example, Glue semantics maintains the type-driven analysis of quantifiers, without requiring additional syntactic mechanisms, like quantifier raising, or storage-based methods (Dalrymple et al., 1999).

(1) Every dog sniffed **a tree**.

```
#n UPOS NOUN & #n LEMMA %l & #n SEM #s ==> #s GLUE [/x_e.%l(x)] : (#s_e -o #s_t).

#d DEFINITE Ind & #d ^(DET) #n SEM #s & #n ^(%) #p SEM #q  ==>
#d SEM #t & #t GLUE [/P_<e,t>.[/Q_<e,t>.Ex_e[P(x) \& Q(x)]]] :
                        ((#s_e -o #s_t) -o ((#s_e -o #q_t) -o #q_t)).
```

$$\frac{\begin{array}{cc}\lambda x.tree(x) : & \lambda P.\lambda Q.\exists x[P(x) \wedge Q(x)] : \\ 11_e \multimap 11_t & (11_e \multimap 11_t) \multimap (11_e \multimap 7_t) \multimap 7_t\end{array}}{\begin{array}{c}\lambda Q.\exists x[tree(x) \wedge Q(x)] : \\ (11_e \multimap 7_t) \multimap 7_t\end{array}}$$

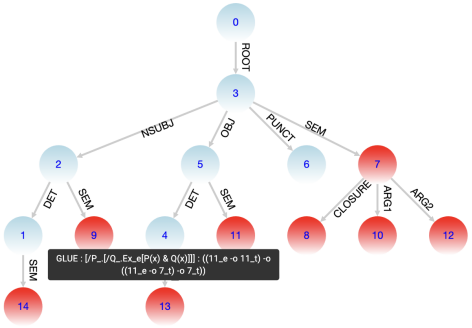

Figure 1: **Indefinite NP semantics example via LiGER:** DBA rules identify subgraphs in a UD parse to expand with semantic information. The first rule gives any matching dependent the semantics of a property (type $< et >$). The second rule identifies this property as the restrictor of the quantifier (identified by the variable #s = 11) and the semantics of the verb as the scope (identified by #q = 7). Together, they form the semantics for the quantifier attached to the corresponding node #t (= 13). They are combined to create a canonical quantifier of type $< et, t >$.

ing bank, relied on CCG parsers). Although LFG provides a variety of syntactic treebanks (e.g., Sulger et al. 2013), there do not yet exist correspondingly extensive meaning banks, making the present tools particularly interesting for this framework.

While we do not directly contribute to this research in the present paper, we assume that formal semantic parsing can lay the foundation for neuro-symbolic approaches which not only strive for best performance but also bridge the gap between formal and computational linguistics, by, for example, testing whether language models can efficiently learn from formal linguistic annotations (see, e.g., Lindemann et al. 2024; Li et al. 2021; Strubell et al. 2018). To achieve the necessary large-scale datasets, sufficiently powerful and robust parsers are needed. Importantly, these parsers should reliably capture complex syntactic and semantic interactions, particularly those that cannot be easily derived from surface patterns (e.g., scope ambiguities). However, permitting ambiguities can make formal parsing challenging (Bunt, 2008); thus, we borrow techniques from efforts to link grammar development and tree-banking to enable efficient management of ambiguities. Particularly, we build on the discriminant system presented in Rosén et al. (2006). Overall, we provide tools that allow for the generation of larger, more detailed datasets that can be used in the way envisioned above.

3. Supporting the development of compositional semantic representations

In this section, we present our application for supporting the development of compositional semantic representations via DBA using XLE+Glue resources. For this, our system provides possibilities to inspect and explore individual analyses and to perform regression testing, which allows for scaling up of the semantic parsers developed in this system. Before describing the system, we briefly discuss the underlying formal modeling.

3.1. Linguistic representations

XLE+Glue procedurally provides semantic parsing based on LFG. However, it also provides a formal representation of the form-to-meaning mapping, linking several linguistic annotation layers containing different levels of linguistic information, for example, c(onstituent) structure, f(unctional) structure, and s(emantic) structure. We uniformly store these different layers as interconnected or layered graphs (see Figure 2). Each graph is associated with a grammar and a sequence of rewrite (DBA) rules used to produce the graph. Thus, they are essentially a way of representing LFG's projection architecture (e.g., Asudeh 2006). Using a uniform graph format allows the representations to be easily extended with additional annotations (also inspired by, e.g., Ide and Bunt 2010). However, layered graphs can become very complex (see Figure 9). Thus, modularizing the ways of interacting with them is crucial and a goal of the present system.

To use our representations for NLI, we translate the semantic representation into the TPTP format for first-order logic (Sutcliffe et al., 2006), conjoining multiple premises to a single premise. Additional axioms and meaning postulates are added if necessary. We then use the Vampire theorem prover (Kovács and Voronkov, 2013) to derive *consistency* and *informativity* labels, which are heuristically mapped onto NLI labels (YES, NO, UNKNOWN). This is explained in detail in Zymla et al. (2025b).

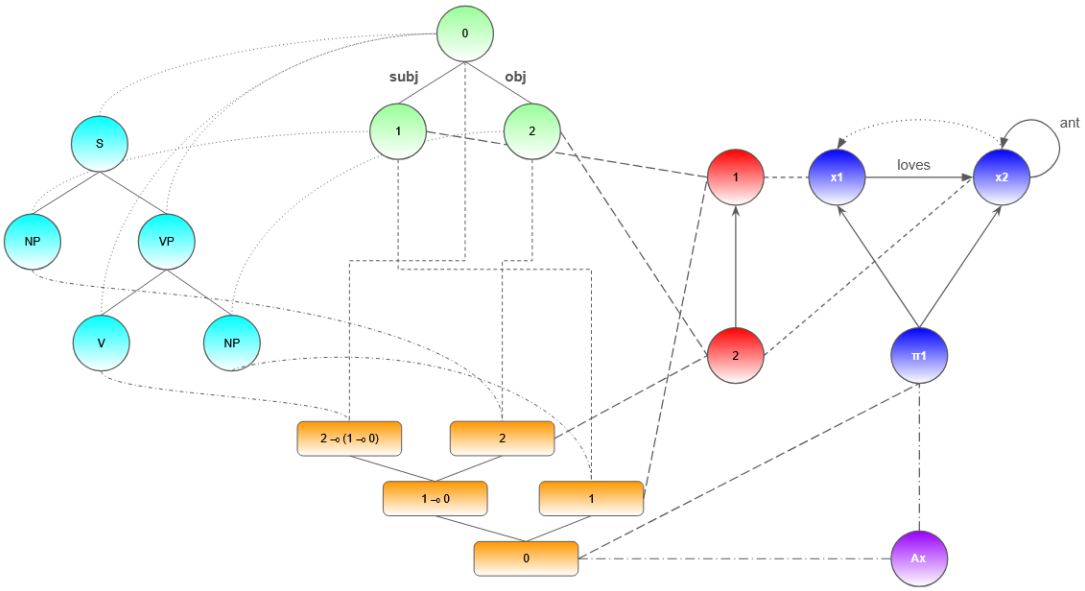

Figure 2: **A schematic layered XLE+Glue graph:** The graph illustrates an XLE+Glue analysis of the sentence *Sam loves herself*. Teal and green correspond to LFG's c(onstituent)- and f(unctional)-structure. Orange is the semantic composition graph, and red corresponds to s(emantic)-structure. The DRT representation is given in blue, and additional axioms are stored in a set associated with the purple node.

3.2. Individual analysis interface

Our application provides three different views: a view for the individual analysis of the mapping from syntactic input to semantics, a view for exploring (NLI) testsuites, and an explorative chat view for testing the NLI component qualitatively (see Zymla et al. 2025b). Here, we focus only on the first two as novel contributions we make to the system. Each view consists of multiple components that can be independently hidden, allowing users to focus only on particular tasks (e.g., designing rules, disambiguation of semantic representations, or debugging).[7] The individual view consists of two major components: a view for writing DBA rules and a view for inspecting Glue proofs, which act as user interfaces for LiGER and for the GSWB, respectively (see Figures 9 and 10 in the appendix).

The *DBA view* provides an editor window with syntax highlighting and visualizes the result of an annotation in an interactive graph window (a sample graph is displayed in Figure 1). During the derivation, all applied rules are collected in a list (i.e., a rule history), which is then displayed in the applied rules view (see Figure 3). Here, we provide the functionality to reapply rules incrementally to see how they affect the derivation. This serves the design and debugging of DBA rules in context. A common use of this view is to ascertain that rules are properly contextualized, meaning that rules

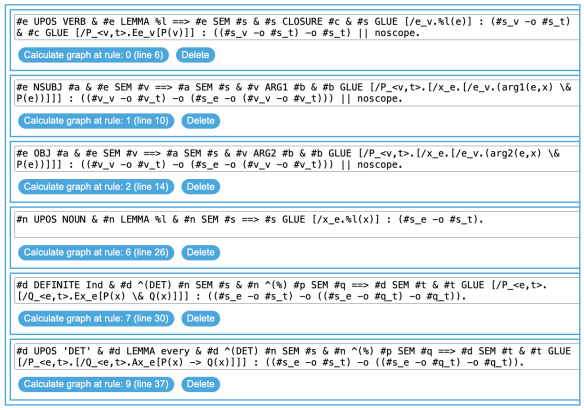

Figure 3: **Applied rules view (individual):** This view lists rules in the order they have been applied. It allows users to inspect the behavior of individual rules by calculating intermediate annotations.

interact with each other as anticipated. As DBA rules are processed sequentially, they (sometimes) form application chains, i.e., a sequence of rules where the output of a previous rule is required for the input of a subsequent rule. Wrongly modifying a rule in such a chain can easily break the whole analysis. Thus, the rule history helps track application chains, but also indicates whether individual rules have been applied as intended.

The *Glue view* presents the results of the DBA annotation in terms of added meaning constructors. It is also the main interface to the GSWB, which is used for calculating the semantic derivations. Cor-

[7] Comprehensive figures are provided in the appendix. In the paper, we only present aspects that have not been introduced elsewhere.

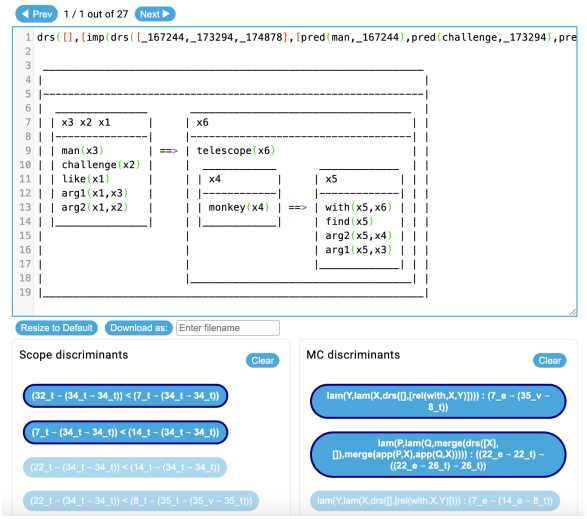

Figure 4: **Discriminant view:** Displays the results of a semantic composition, allowing users to disambiguate its results using two types of discriminants to filter out solutions that do not match them.

respondingly, it visualizes these derivations, such that they are represented as a graph or in a natural deduction format (Zymla et al., 2025a). For this paper, we added a discriminant component to this view that allows users to systematically disambiguate analyses. Figure 4 presents an example based on the following sentence:

(2) Every man who likes a challenge found every monkey with a telescope.

As indicated in the Figure, our parser produces twenty-seven solutions for this sentence, permitting all possible scopings between the different quantifiers as well as different interpretations of the prepositional phrase. The sentence is disambiguated to a reading where each man uses a possibly different telescope to find all the monkeys. First, we use the *MC discriminants* that indicate differences in the used meaning constructors for different solutions to only pick solutions where the telescope is used as an instrument, i.e., is associated with an event variable. Furthermore, we ensure that the relative clause is interpreted locally as its scope is fixed (partially) syntactically. Then, we can use *scope discriminants* to fine-tune the scopings between the individual quantifiers that are determined semantically to arrive at the desired semantic representation.

As the name suggests, MC discriminants are calculated based on different sets of meaning constructors that can be produced by a grammar for a given sentence. However, each set may still correspond to multiple solutions. Thus, these discriminants provide a less granular separation of results, whereas scope discriminants are calculated during the derivation for each individual solution, making them more fine-grained. After the derivation, equivalence classes for all potential discriminants are calculated to reduce the total number of discriminants. Sometimes, the same scope-taking operator may be instantiated differently in a different meaning constructor set, which can lead to malformed discriminants. Thus, we impose a disjointness condition on discriminants, such that each discriminant is uniquely applicable.

The discriminants behave commutatively, such that individual decisions are not dependent on each other. However, we keep track of choices to make uninformative discriminants unavailable during the selection process, allowing users to quickly reduce the number of readings. Importantly, this system not only allows for the reduction to a single solution but also provides finer control over which ambiguities should potentially be retained for a given derivation.

3.3. Resolving discriminants for NLI

We provide two different inputs for regression testing: i) sentence-based items and inference-based items, where inference-based items consist of a list of premises, a conclusion, and a gold label (illustrated in Figure 6). For both, we provide a list view for inspecting individual results, highlighting failed parses and NLI label mismatches. Further, sentences can be disambiguated using the discriminant view. This can be essential for eliminating uncertainty in NLI testing and is accordingly implemented as a human-in-the-loop process (similar, in spirit, to He et al. 2016). Consequently, the semantic parsing process can be paused after semantic composition to select discriminants. The following paragraph explains the need for this.

Figure 5 shows an inference result based on ambiguous inputs. It is based on example (3). There, multiple quantifiers and negation lead to a number of different possible interpretations.

(3)
Every boxer with an injury lost a fight.

A boxer with an injury did not lose a fight.
→ NO

Without any constraints on quantifier scope, the first sentence has five solutions and the second sentence has eighteen solutions, leading to ninety different possible inferences. As can be deduced from Figure 5, many of these boil down to spurious ambiguity, but even categorically, the correct prediction of a contradiction represents a minority across the potentially resulting inferences. Thus, the correct prediction crucially requires human intervention for the given parser.[8]

[8]This also hinges on pragmatic factors that are difficult to completely capture in a purely formal setting. This

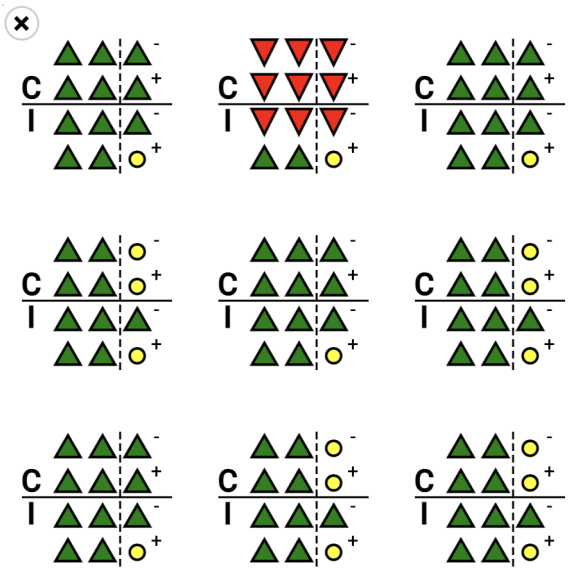

Figure 5: Inference output for ambiguous analyses based on example (3). The glyphs encode the results of a single potential inference and are based on *consistency* (C) and informativity (I) checking. Out of the nine shown possible results, only one (top center) correctly predicts the contradiction.

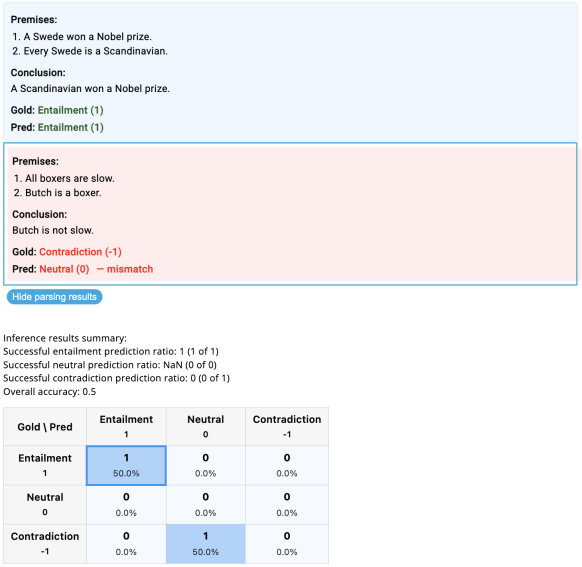

Figure 6: **Interactive confusion matrix:** Individual items are presented at the top. Mismatches in the NLI labeling are highlighted in red. In the confusion heatmap, classes can be selected to highlight individual items in blue accordingly.

3.4. Regression testing output

Parsing and inference results are summarized in overview reports. These reports contain information about the number of successful and failed parses and an accuracy overview regarding the NLI labeling (i.e., entailment, neutral, contradiction). This is visualized in an interactive confusion heatmap, which allows users to select classes, revealing the items associated with the corresponding predictions in blue (see Figure 6), facilitating error analysis.

In addition to inspecting and disambiguating items, we also extend this view with a visualization for DBA behavior across a testsuite. For each sentence, we calculate the application path (regardless of whether rules are dependent on each other or not). There, each rule corresponds to a node, and two nodes are connected if they are applied directly after one another. The global *applied rules graph* aggregates those paths into a single directed graph that highlights the complexity of the underlying annotation rules by highlighting disjunctions in the rule application. For example, in Figure 7, which is based on tense aspect rules, the complexity is fairly limited, mainly marking distinctions between past, present, and future tense at the top of the graph and aspectual distinctions between perfective and imperfective in the middle of the graph. The lower part of the graph is a little more complex as it further contextualizes some of those features (Zymla, 2024). Edges in the graph are associated with the sentences in the corresponding testsuite that pass through these paths. This is particularly useful when a testsuite is extended without prior individual testing. Furthermore, this view visualizes whether the rule set matches the syntactic input, as dangling nodes indicate rules that have not been applied, and which are, thus, not useful to the current testsuite.

3.5. Implementation

The present architecture was first described in Zymla et al. (2025b). As illustrated in Figure 8, the system spans two Java modules (LiGER and the GSWB) and a Python module that serves as an interface to the Vampire theorem prover. For the present paper, LiGER is extended to keep track of rule application paths during the annotation process and to calculate the corresponding histories and graphs. The GSWB is extended to calculate the different kinds of discriminants, described in the previous section. Furthermore, the Vampire interface is extended to permit batch processing. Finally, the frontend, which visualizes these components, is written in Angular and extended to present the newly added data structures.

All components are deployed as modular Docker containers. To use the system, a syntactic parser must be interfaced. We provide such an interface for the XLE, which is integrated natively as part of LiGER. Furthermore, we provide an external interface to UD by virtue of an additional Docker

is discussed in more detail in section 4.

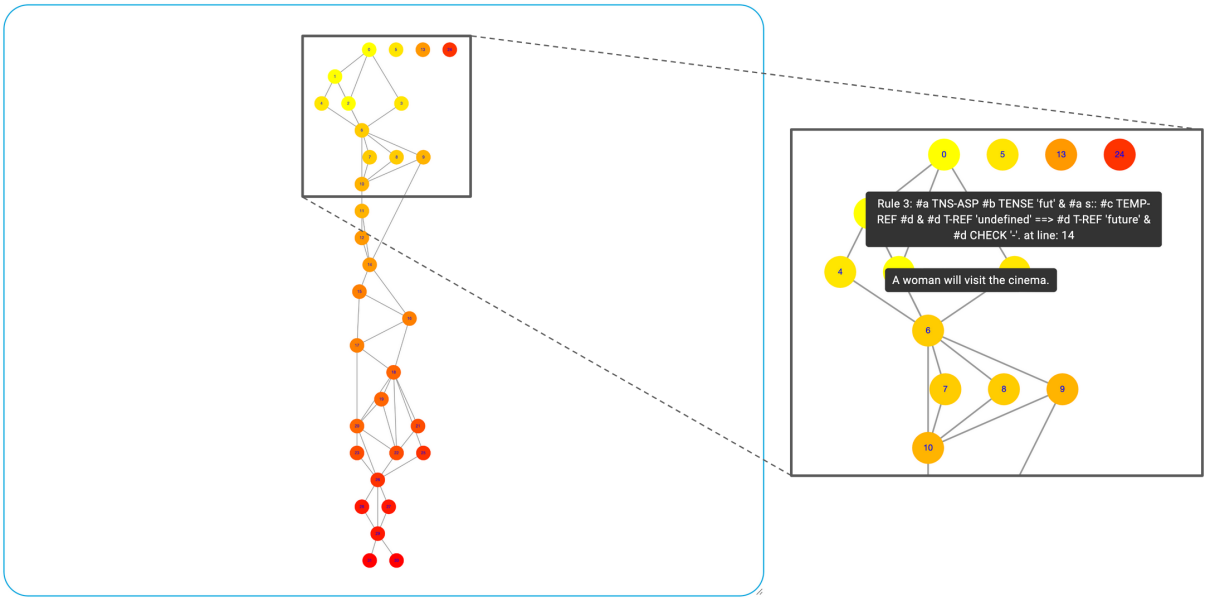

Figure 7: **Rule application view (testsuite):** This view captures the behavior of DBA rules across a testsuite. In the graph, each node corresponds to a rule, and each edge indicates which rules feed into each other. Hovering over nodes reveals the corresponding rules, and hovering over edges reveals which sentences follow the corresponding rule application path. The coloring of the nodes indicates the position of a rule in the sequence of all rules.

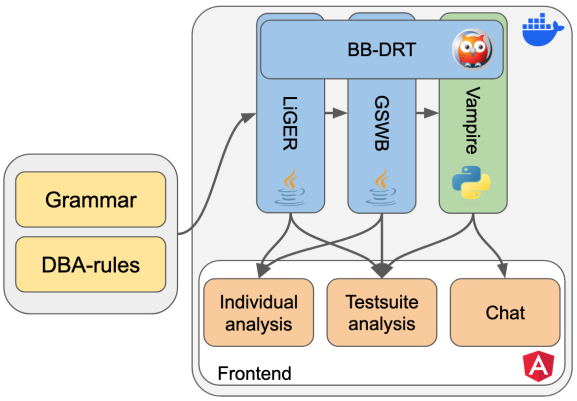

Figure 8: **System architecture:** LiGER is used to apply DBA-rules to a grammar. The resulting semantic annotations are passed on to the GSWB to resolve the compositional semantics. Both of these steps are visualized as part of the *individual analysis*, as well as the *testsuite analysis*, which optionally also takes reasoning with Vampire into account. The *chat* provides an explorative testing environment for the reasoning component.

container hosting an instance of the Stanza dependency parser (Qi et al., 2020).

4. Task-driven semantic parsing

There are good reasons to eschew formal semantic parsing in NLP applications, as maintenance is tedious and the systems are often still brittle. Furthermore, a one-fits-all broad coverage semantic parser is unlikely to be feasible, as we explain in more detail below. This may also have been a factor that led to the development of modular annotations such as those by the interoperable semantic annotation effort (ISA; Bunt 2015). In this section, we present some considerations regarding the question of how to approach formal semantic parsing from a research-driven computational linguistics perspective.

First, we propose to adopt ISA's "crystal growth" strategy to semantic annotation as the basis for developing formal semantic parsers. This is also more in line with formal semantic theory, which provides highly consistent analyses locally but often relies on approximate consistency in a broader context, such that interactions between unrelated phenomena are not always explicitly tested (but are testable). As an example, consider the classic *modus ponens* inference in (4).

(4)
A Swede won a Nobel Prize.
$\exists x, y[sw(x) \wedge prize(y) \wedge win(x, y)]$
Every Swede is a Scandinavian.
$\forall x[sw(x) \rightarrow \exists y[sc(y) \wedge x = y]]$

A Scandinavian won a Nobel Prize.
$\exists x, y[sc(x) \wedge prize(y) \wedge win(x, y)]$
→ YES

This inference falls out naturally from the interaction of the quantifiers and the identity predicate

be. However, adding, for example, semantic information about tense and aspect makes a formal analysis of the inference disproportionately more complex, as it would require an appropriate analysis of the persistence of states compared to events and how this interacts with tense and aspect. This would further require the design of additional axioms about temporal logic, also changing the inference process. A similar observation is made in Zymla et al. (2025b), who introduce a degree semantics inspired by Haruta et al. (2022) to deal with comparatives. This requires concessions during automated inference, as different proof search strategies are required compared to an example like (4) (without tense and aspect).

4.1. Building a foundation through interoperable semantic parsers

An interoperable approach advocates for first developing rules and testsuites, covering diverse phenomena (such as the ones mentioned above) in isolation but also in detail; however, with a shared set of assumptions about how to design annotations, allowing annotations of different phenomena to interact. For us, the common core is provided by Glue semantics and description-by-analysis, building on more general aspects of formal semantics.[9]

An interoperable semantic parser then consists of a (syntactic) grammar, a set of DBA rules providing semantic annotations, and a domain of testing. Given this approach, we can treat each pair of rules and test cases as building blocks for more comprehensive rule sets. The current system allows users to easily put building blocks together to investigate whether they provide a stable foundation for exploring more complex cases. Consider, for example, (5) which combines temporal and comparative semantics. If the individual building blocks are engineered properly, the analysis is expected to fall out compositionally when they are combined; however, if not, we can use the tools presented here to revise the individual building blocks and their interactions by investigating which rules break or stand in conflict with one another. We can also explore the emergence of unwanted ambiguities or other interactions that may obfuscate the reasoning process. Regression testing ensures that the arising more complex rule sets remain stable when introducing new phenomena and allows us to keep track of where the foundation may be weak and in need of revision.

[9]Furthermore, we assume a shared core grammar for syntactic analysis (e.g., an XLE grammar or a UD parser), but this is a design decision rather than a requirement of our approach.

(5)
Yesterday, Sam was faster than Jordan.
Today, Jordan is faster than Sam.

Sometimes, Sam is faster than Jordan.
→ YES

4.2. Exploring inference

While this remains to be tested, we hypothesize that stacking more and more rule sets on top of each other leads to less and less precise inference. This matches what we observe during development as grammars become more complex: i) more ambiguities are introduced, and ii) the proof search is more likely to time out (this is already observable for "simple" extensions like the one presented in Zymla et al. 2025b). Thus, an interoperable approach may give us further insights into how to best constrain semantic parsing to deal with particular inferences.

4.3. The trade-off between complexity and informativity

Formal semantic representations present a trade-off between complexity and informativity. More precisely, they focus on encoding certain entailments predicted by a given analysis, while leaving others unspecified. Thus, formal representations are, in a sense, fragmentary, only providing a partial picture; however, they still provide a precise view of certain reasoning patterns. They are distillates of aspects of natural language semantics.

On the one hand, this view seems sensible enough; however, on the other hand, in this view, formal semantic representations are prescriptive, capturing exactly those meanings we allow them to. This is why a task-driven approach to semantic parsing is so essential: only if we compare the output of our parsers to empirically grounded inference judgments can we ensure that semantic representations serve as falsifiable instantiations of linguistic theory. (We are, of course, not alone in this, see, e.g.: Haruta et al. 2022; Pulman 2018; Funakura et al. 2025).

Another point where we depart from currently popular methodologies in semantic parsing is that not all analyses should be fully disambiguated (cf. Abzianidze and Bos 2019). This is due to the fact that ambiguity may inform the annotation of NLI labels, particularly the neutral label, which can be most appropriate if a given sentence has two interpretations that lead to conflicting NLI labels during inference. This is another point where ambiguity management serves as a way of calibrating the boundary between logically and pragmatically driven inference.

If the output of a semantic parser is to be compared to empirically tested data, inferences should be tested under similar conditions, which also

paves the way for testing novel hypotheses such as the interaction between compositional ambiguity and inference. There, discriminant-based disambiguation enables us to separate spurious ambiguities from meaningful ambiguities, and it allows us to constrain meaningful ambiguities further. Essentially, we are not arguing for fully automated semantic parsing but rather for linguistically motivated and testable semantic parsing that bridges computational and formal methods. Given the overall approach to semantic parsing based on empirical adequacy that we advocate for, we, thus, also provide the foundation for exploring more fine-grained linguistic predictions computationally.

5. Limitations

We already mentioned some of the well-known limitations of formal approaches to semantic parsing. Furthermore, we have explained that NLI based on theorem proving can become computationally expensive as ambiguities grow and inferences become more reliant on more complex logics (e.g., temporal logic, degree logic). This can also result in long processing times when using the system.

Working with formal semantic parsers and, particularly, the process of disambiguation requires a certain level of competence in formal semantics. This alone can be seen as problematic, as there is a large strain of NLI work that argues that humans do not always reason logically, and that NLI labels should originate not from experts but from an empirically representative sample of annotators (see Manning 2006 for discussion). However, as addressed in the paper, we share the thought that NLI labels should be empirically motivated, and it stands to reason that at least some reasoning processes are best approximated by a logical system. However, this requires extensive testing of the kind of semantic parsers for which the tools here are envisioned.

The tools have been tested using XLE+Glue (Zymla et al., 2025a) during development. The corresponding GitHub page (see fn. 3) provides corresponding test sets based on Zymla et al. (2025b); however, they are limited in scope. Thus, while the tools have been tested for overall soundness, more detailed empirical testing remains for future work.

6. Conclusion

In this paper, we have presented a system that supports the development of task-driven formal semantic parsing at a larger scale by allowing for individual and regression testing. The focus of this system lies on LFG's description-by-analysis Glue approach, which provides a flexible and portable way to design meaning representations.

Furthermore, we discuss the role of such meaning representations and come to the conclusion that they are best observed in the context of automated reasoning, or NLU tasks more generally, because then they establish a firm connection between formal and computational linguistics and allow for the exploration of testable predictions.

We highlight a number of potential future directions with respect to this connection. Importantly, we suggest that an interoperable approach to semantic parsing is most likely to be promising. However, this comes with the caveat that a computational system must also be able to decide how to analyze a given problem. Thus, the present tools work best as a human-in-the-loop system, such that human experts produce meaning representations interactively.

Acknowledgments

7. Bibliographical References

Lasha Abzianidze and Johan Bos. 2019. Thirty musts for meaning banking. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 15–27. Association for Computational Linguistics.

Lasha Abzianidze, Rik Van Noord, Chunliu Wang, and Johan Bos. 2020. The parallel meaning bank: A framework for semantically annotating multiple languages. *Applied Mathematics and Informatics*, 25(2):45–60.

Avery D Andrews. 2008. The role of PRED in LFG + Glue. In *Proceedings of the LFG'08 Conference*, pages 47–67, Stanford, CA. CSLI Publications.

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

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.

Julia Bonn, Matthew J. Buchholz, Jayeol Chun, Andrew Cowell, William Croft, Lukas Denk, Sijia Ge, Jan Hajič, Kenneth Lai, James H. Martin, Skatje Myers, Alexis Palmer, Martha Palmer,

Claire Benet Post, James Pustejovsky, Kristine Stenzel, Haibo Sun, Zdeňka Urešová, Rosa Vallejos, Jens E. L. Van Gysel, Meagan Vigus, Nianwen Xue, and Jin Zhao. 2024. Building a broad infrastructure for uniform meaning representations. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 2537–2547, Torino, Italia. ELRA and ICCL.

Johan Bos. 2009. Applying automated deduction to natural language understanding. *Journal of Applied Logic*, 7(1):100–112.

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

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

Harry Bunt. 2015. On the principles of semantic annotation. In *Proceedings of the 11th Joint ACL-ISO Workshop on Interoperable Semantic Annotation (ISA-11)*.

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, 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 GEAF07 Workshop*, pages 128–143, Stanford, CA. CSLI Publications.

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 LFG'06 Conference*, pages 145–165, Stanford, CA. CSLI Publications.

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

Mary Dalrymple. 2001. *Lexical Functional Grammar*. Number 34 in Syntax and Semantics. Academic Press, San Diego, CA.

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, 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, 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, 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, Yi Zhang, and Valia Kordoni. 2012. Deepbank. a dynamically annotated treebank of the wall street journal. In *Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories*, pages 85–96.

Hayate Funakura, Hyunsoo Kim, and Koji Mineshima. 2025. A theorem-proving-based evaluation of neural semantic parsing. In *Proceedings of the 8th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 295–306.

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

Dag Haug, Jamie Yates Findlay, and Ahmet Yıldırım. 2023. The long and the short of it: DRASTIC, a semantically annotated dataset containing sentences of more natural length. In *Proceedings of the Fourth International Workshop on Designing Meaning Representations*, pages 89–98.

Luheng He, Julian Michael, Mike Lewis, and Luke Zettlemoyer. 2016. Human-in-the-loop parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2337–2342, Austin, Texas. 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.

Kevin Knight, Bianca Badarau, Laura Baranescu, Claire Bonial, Madalina Bardocz, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, Tim O'Gorman, and Nathan Schneider. 2020. Abstract meaning representation (amr) annotation release 3.0. LDC2020T02, Web Download. DOI: 10.35111/44cy-bp51.

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.

Zhongli Li, Qingyu Zhou, Chao Li, Ke Xu, and Yunbo Cao. 2021. Improving BERT with syntax-aware local attention. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 645–653, Online. Association for Computational Linguistics.

Matthias Lindemann, Alexander Koller, and Ivan Titov. 2024. Strengthening structural inductive biases by pre-training to perform syntactic transformations. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 11558–11573, Miami, Florida, USA. Association for Computational Linguistics.

Shixia Liu, Xiting Wang, Mengchen Liu, and Jun Zhu. 2017. Towards Better Analysis of Machine Learning Models: A Visual Analytics Perspective. *Visual Informatics*, 1(1):48–56. Publisher: Elsevier.

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.

Christopher D. Manning. 2006. Local Textual Inference: It's Hard to Circumscribe, but You Know It When You See It — and NLP Needs It. Manuscript, Stanford University.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

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.

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

Stephen Guy Pulman. 2018. Second order inference in natural language semantics. *Journal of Language Modelling*, 6(1):1–40.

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.

Victoria Rosén, Koenraad De Smedt, and Paul Meurer. 2006. Towards a toolkit linking treebanking to grammar development. In *Proceedings of the Fifth Workshop on Treebanks and Linguistic Theories*, pages 55–66.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

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, 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.

Shuai Zhang, Lijie Wang, Xinyan Xiao, and Hua Wu. 2022. Syntax-guided contrastive learning for pre-trained language model. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2430–2440, Dublin, Ireland. Association for Computational Linguistics.

Xiao Zhang, Gosse Bouma, and Johan Bos. 2025. Neural semantic parsing with extremely rich symbolic meaning representations. *Computational Linguistics*, 51(1):235–274.

Mark-Matthias Zymla. 2024. *Tense and aspect in multilingual semantic construction*. Ph.D. thesis, University of Konstanz.

Mark-Matthias Zymla, Mary Dalrymple, and Agnieszka Patejuk. 2025a. Computational semantic tools for Glue semantics. In *Proceedings of the 16th International Conference on Computational Semantics (IWCS 2025)*, pages 189–207, Düsseldorf. Association for Computational Linguistics.

Mark-Matthias Zymla, Kascha Kruschwitz, and Paul Zodl. 2025b. An instructive implementation of semantic parsing and reasoning using lexical functional grammar. In *Proceedings of the 2nd Bridging the Gap between Human and Automated Reasoning Workshop (BriGap-2)*.

A. Appendix: UI figures

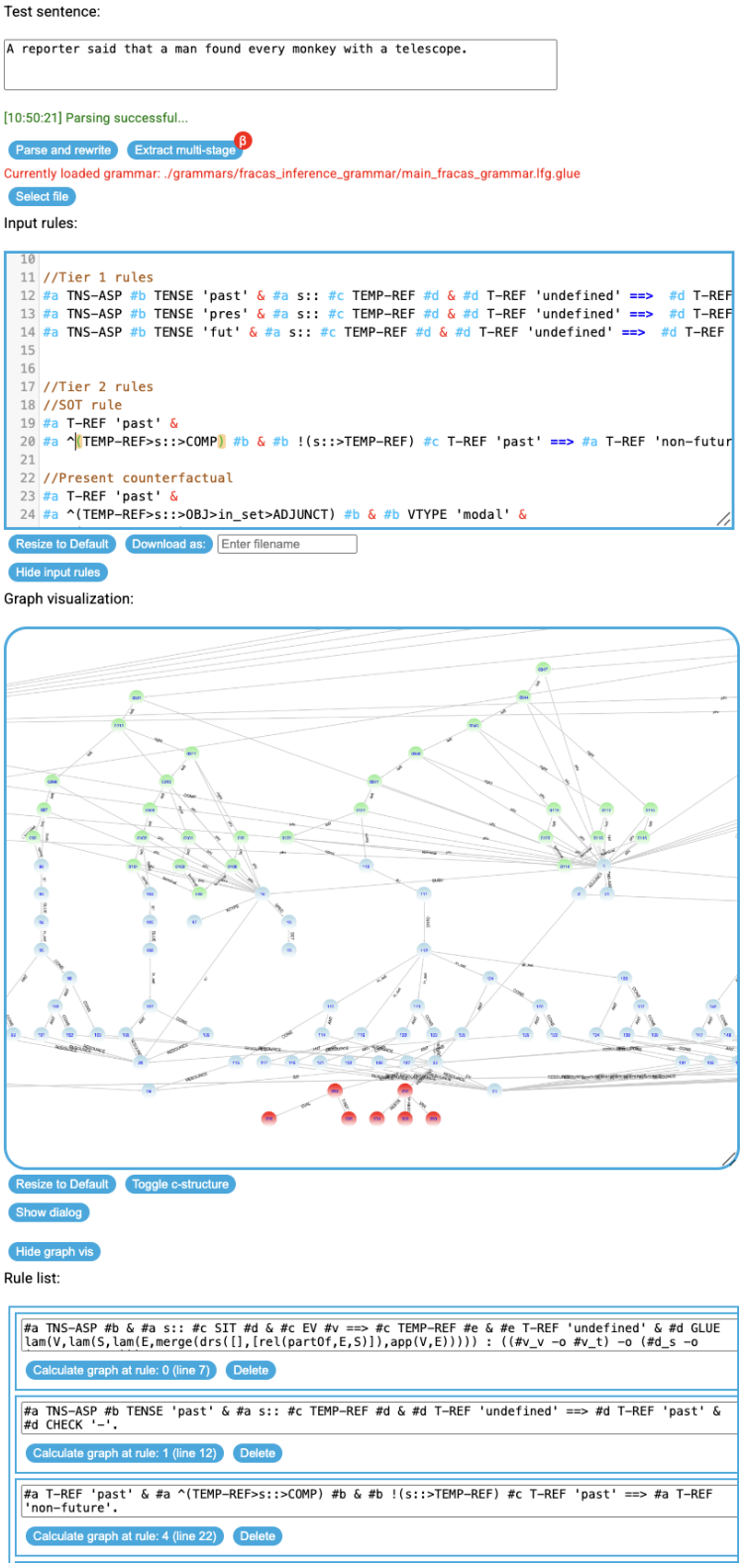

Figure 9: **Description-by-analysis view:** This view allows viewing of the specified DBA rules and the resulting annotated graph (here, the graph contains c-structure, f-structure, and semantic structure).Furthermore, it provides access to the history of applied rules.

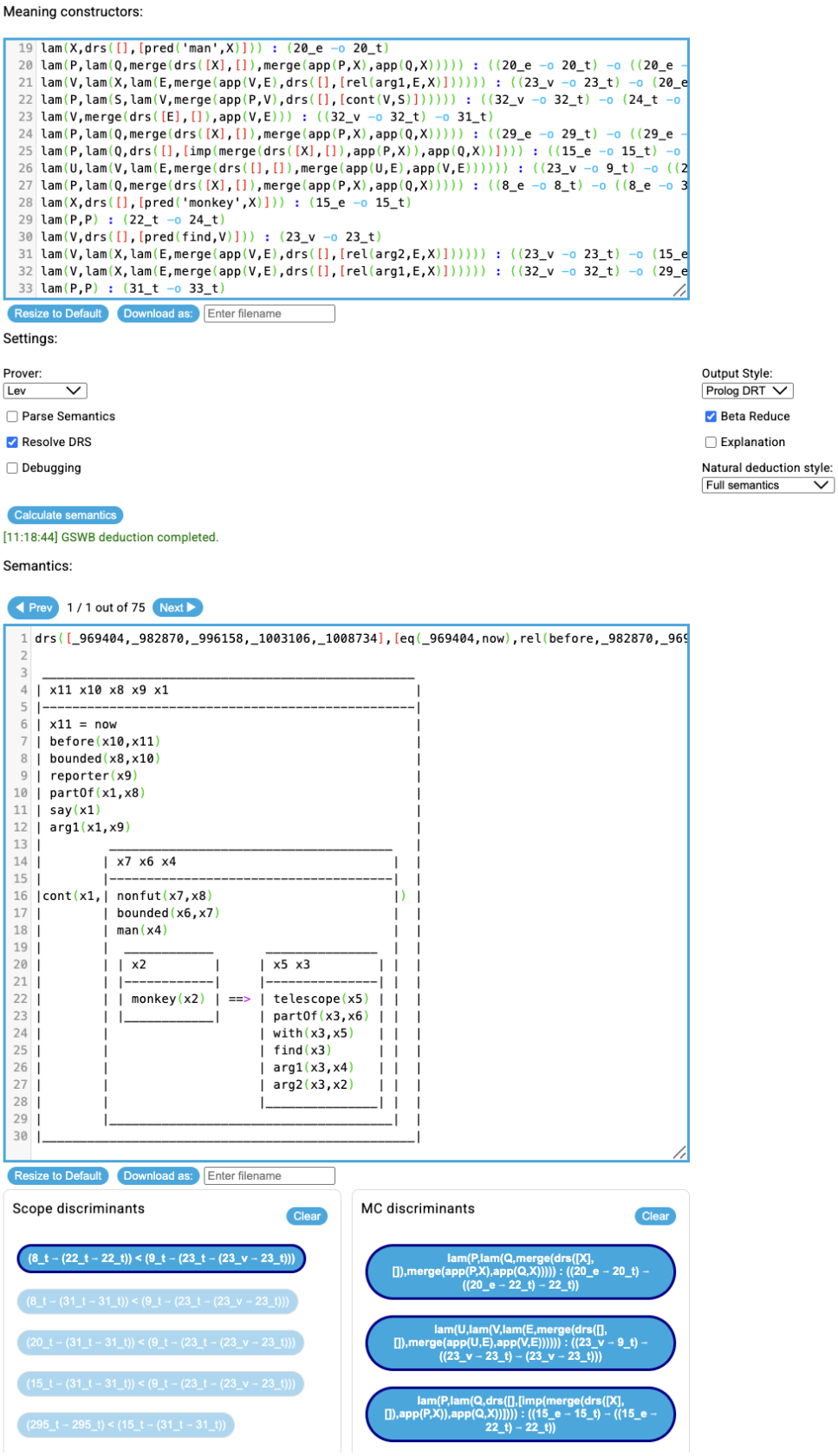

Figure 10: **Semantic composition view:** This view presents the meaning constructor sets available for a given sentence at the top and the resulting derivations at the bottom, which can be filtered using scope and MC discriminants. This view can be further extended to include information about the derivation.

Figure 11: **Full view:** This view combines the DBA view and the semantic composition view. The two major components can be scrolled individually. Most components can be hidden, including the two major ones.