

The GLIF System: A Framework for Inference-Based Natural-Language Understanding

Jan Frederik Schaefer and Michael Kohlhase^[0000–0002–9859–6337]

Computer Science, FAU Erlangen-Nürnberg

Abstract. With the Grammatical Logical Inference Framework (GLIF), a user can implement the core of symbolic language understanding systems by describing three components, each of which is based on a declarative framework: parsing (with the Grammatical Framework GF), semantics construction (with MMT), and inference (with ELPI). The logical frameworks underlying these tools are all based on LF, which makes the connection very natural. Example applications are the prototyping of controlled natural languages for mathematics or experiments with new approaches to natural-language semantics. We use Jupyter notebooks for a unified interface that allows quick development of small ideas as well as testing on example sentences.

1 Introduction

Mathematical research and application is increasingly supported by software – e.g. computer algebra systems help with computational tasks and proof checkers can be used to formally verify results. However, while mathematicians are used to natural language (like English + formulas), such software systems typically require a formal input language. This raises the entry barrier not just for using mathematical software, but also for understanding the results: The formal verification of a proposition is of little use if the proposition or its axioms are unintelligible. Ideally, the software would accept natural language as input, but the formalization of general natural language is a very hard problem. An easier approach is to design formal input languages that imitate natural mathematical language, which we will call **controlled mathematical languages**.

To support the design of such languages, we introduce GLIF, the Grammatical Logical Inference Framework. GLIF allows users to implement natural-language understanding systems, by describing a pipeline consisting of three steps: **parsing**, **semantics construction** (mapping parse trees to logical expressions), and **inference**. In the context of controlled mathematical languages, the logical expressions may simply be expressions in the (formal) input language of some mathematical software. Each step in the pipeline is based on a different framework: Parsing and grammar development are based on the Grammatical Framework (GF) [Ran11], semantics construction and logic development are based on MMT [MMT], and inference is based on ELPI [SCT15], an extension of λ Prolog.

GLIF is an extension of the Grammatical Logical Framework (GLF) [KS], which doesn't have an inference component.

The inference step is essentially the “understanding part” in the pipeline. Depending on the application, it can have a variety of functions. It may simply modify the results of the semantics construction, which is, by design, bound to be compositional, with more complex operations, such as simplification or semantic pruning. The inference step can also be used for ambiguity resolution (e.g. by discarding contradictory readings) or even for theorem proving.

We have successfully used GLIF (and GLF) for various smaller experiments in the area of natural-language semantics as well as a one-semester lecture in logic-based natural language processing. More recently, we started reimplementing a variant of ForTheL [Pas07], the language of the System of Automated Deduction. As a running example, we will use a made-up language for specifying physical properties of different objects with the example sentence

“the ball has a mass of 5 kg and a kinetic energy of 12 mN”,

where we use the inference step to disambiguate whether “12 mN” stands for “12 meter Newton” or “12 milli Newton”.

2 Preliminaries: MMT

Before diving into details of the GLIF pipeline, we need to briefly introduce MMT, the center-piece of GLIF. MMT is a modular, foundation-independent knowledge representation framework. Knowledge is represented in the form of **theories**, which contain a sequence of symbol declarations. Theories can be linked via **theory morphisms**, which map symbols in the source theory to symbols in the target theory. While MMT aims at foundation-independence, in practice most theories are based on the Edinburgh Logical Framework LF or extensions of it – e.g. we need integer literals for our running example. GLIF exploits the similarity of LF with the logical frameworks underlying GF and ELPI, which results in very intuitive transitions between the systems.

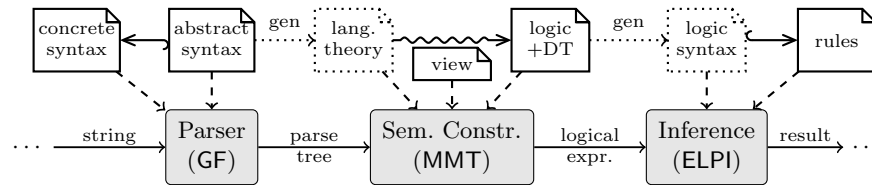




Fig. 1. The GLIF Pipeline:  indicates elements that have to be specified and  indicates elements that can be generated automatically.

3 The GLIF System

Figure 1 illustrates the GLIF pipeline. For the first step (parsing), we use the Grammatical Framework (GF), which provides powerful mechanisms for the development of natural language grammars and comes with a library that implements the basic morphology and syntax of various languages. GF grammars come in two parts: abstract syntax and concrete syntax. The **abstract syntax** specifies the parse trees supported by the grammar in a type-theoretical fashion, while the **concrete syntax** describes how these parse trees correspond to strings in a language. If a sentence is ambiguous according to the grammar, GF generates multiple parse trees. For our example sentence (“*the ball has a mass of 5 kg and a kinetic energy of 12 mN*”), the two parse trees shown in Figure 2.

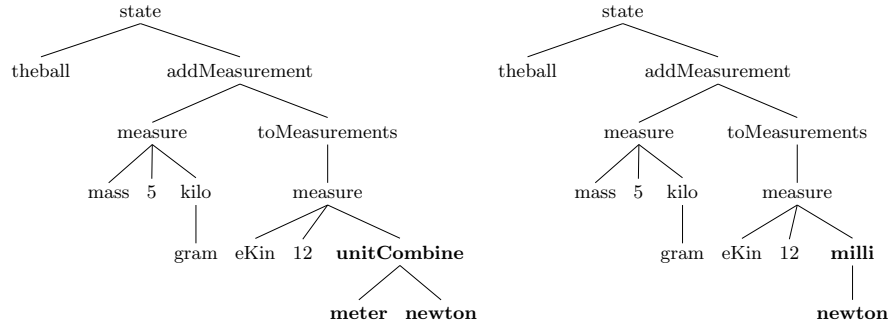


Fig. 2. The ambiguity of mN results in two different parse trees.

We connect GF to MMT by reinterpreting the abstract syntax as an MMT theory (the **language theory**). This lets us interpret the parse trees as terms in that theory. The target of the semantics construction is an MMT theory that describes the logic syntax and a domain theory. At the heart of the semantics construction is now a **view** – a particular type of theory morphism – that maps every symbol in the language theory to an object in the target logic/domain theory. The translation of parse trees to logical expressions thus boils down to applying a view to an MMT term. The compositionality of this process typically means that some subtrees have to be translated to λ -functions. In our case, for example, “*a mass of 5 kg*” gets translated to $\lambda x.mass\ x\ (quant\ 5\ kilo\ gram)$. The `addMeasurement` node, which combines measurements M and N , becomes $\lambda x.Mx \wedge Nx$. After the semantics construction, the λ -functions are eliminated through β -reduction, which gives us the following two logical expressions:

$$\begin{aligned} & (\text{mass theball (quant 5 kilo gram)}) \wedge (\text{ekin theball (quant 12 milli Newton)}) \\ & (\text{mass theball (quant 5 kilo gram)}) \wedge (\text{ekin theball (quant 12 meter \cdot Newton)}) \end{aligned}$$

For the inference step, we use ELPI, an extension of λ Prolog. The advantage of choosing λ Prolog over classical prolog variants is that variable binding can be naturally represented through λ -expressions, which is needed for many logics, including first-order logic. MMT supports the transition to ELPI by generating the signature of the logic and domain theory and by exporting the generated logical expressions in ELPI syntax. MMT can also generate ELPI provers from calculi specified in MMT [Koh+20].

For our example, we use the inference step to perform a dimensional analysis, which rejects the wrong reading.

```
In [11]: parse "the ball has a mass of 5 k g"
state theball (toMeasurements (measure mass 5 (kilo gram)))

In [12]: p "the ball has a mass of 5 k g and a kinetic energy of 12 m N" | construct -v SemConstr
(mass theball (quant 5 kilo gram))^(ekin theball (quant 12 milli Newton))
(mass theball (quant 5 kilo gram))^(ekin theball (quant 12 meter·Newton))

In [13]: p "the ball has a kinetic energy of 12 m N" | construct -elpi | elpi dimAnalysis check
(ekin ball (quant 12 (milli newton)))
REJECTED: milli newton has dimension mass*length/(time*time) but expected
length*length*mass/(time*time)

(ekin ball (quant 12 (mult meter newton)))
ACCEPTED
```

Fig. 3. The results of parsing, semantics construction and inference in Jupyter

To improve accessibility GLIF can be used through Jupyter notebooks via a custom kernel (Figure 3). The notebooks can be used to test the entire GLIF pipeline. For smaller projects, grammars and MMT content can be implemented directly in the notebook. Other features include the (visual) display of parse trees and stub generation e.g. for the semantics construction.

4 Conclusion

We have presented GLIF, a declarative framework in which natural-language understanding systems can be prototyped by specifying *i*) a grammar, *ii*) a target logic and domain theory, *iii*) the semantics construction, *iv*) and inference rules, which can be generated for some applications, as described in [Koh+20]. We have tested the GLIF pipeline in different case studies and used it in a one-semester course on logic-based natural-language semantics. One of the larger case studies is our on-going effort to re-implement a variant of ForTheL, the controlled mathematical language of the System for Automated Deduction (SAD). The grammar currently has 38 different node types and 52 different production rules. Like in SAD, the target of our semantics construction is first-order logic. It can parse e.g. the definition “a subset of S is a set T such that every element of

T belongs to S ”, which results (after some α -renaming for readability) in the logical expression

$$\forall T.(\text{subsetof } T S) \Leftrightarrow (\text{set } T) \wedge \forall x.(\text{elementof } x T) \wedge \top \Rightarrow (\text{belongto } x S) \wedge \top$$

GLIF can be used through Jupyter notebooks, which increases the accessibility significantly. The Jupyter kernel can be found at [GLIF]. The code and notebook for the running example can be found at [Dim].

References

- [Dim] *Dimensional Analysis in GLIF*. URL: <https://gl.mathhub.info/commma/glf/-/tree/master/source/%2Fcicm2020> (visited on 03/22/2020).
- [GLIF] *GLIF*. URL: <https://github.com/KWARC/GLIF> (visited on 03/22/2020).
- [Koh+20] Michael Kohlhase et al. “Logic-Independent Proof Search in Logical Frameworks (extended report)”. extended report of conference submission. 2020. URL: <https://kwarc.info/kohlhase/submit/mmtelpi.pdf>.
- [KS] Michael Kohlhase and Jan Frederik Schaefer. “GF + MMT = GLF – From Language to Semantics through LF”. In: *LFMTP 2019, Proceedings*. Electronic Proceedings in Theoretical Computer Science (EPTCS). URL: <https://kwarc.info/kohlhase/submit/lfmtp-19.pdf>.
- [MMT] *MMT – Language and System for the Uniform Representation of Knowledge*. URL: <https://uniformal.github.io/>.
- [Pas07] Andrei Paskevich. *The syntax and semantics of the ForTheL language*. 2007.
- [Ran11] Aarne Ranta. *Grammatical Framework: Programming with Multilingual Grammars*. ISBN-10: 1-57586-626-9 (Paper), 1-57586-627-7 (Cloth). Stanford: CSLI Publications, 2011.
- [SCT15] Claudio Sacerdoti Coen and Enrico Tassi. *The ELPI system*. 2015. URL: <https://github.com/LPCIC/elpi>.