On the Syntax/Semantics Interface in Computational Glue Semantics: A Case Study

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

Gloria Mirjam Sigwarth
University of Konstanz
Gloria.Sigwarth@uni-konstanz.de

Introduction: This paper presents ongoing work in the domain of computational Glue semantics, which has experienced somewhat of a resurgence recently; see for example Gotham and Haug (2018) or the Glue Logic Theorem Prover that was written by Gianluca Giorgolo and the freely available Glue semantics workbench (Meßmer and Zymla, 2018). We focus in particular on this last contribution and extend it to handle quantification at the verbal spine (Kallmeyer and Romero, 2008). We pay particular attention to the issue of quantifier scope interactions (Cinque, 1999; Joshi et al., 2008). A subgoal of this endeavour is to introduce situation semantics as formalism of choice. Our study represents a further exploration of the syntax/semantics interface and provides us with some insights as to how it should be modeled when working with the Glue semantics workbench. Our approach also questions the compilation process presented in Meßmer and Zymla (2018) and we outline an improved compilation process. Furthermore, we present some further improvements of the Glue semantics workbench. These include, but are not limited to: A debugging mode, the possibility of computing partial results, and parsing of semantic expressions.

Theoretical background: Modeling quantification at the verbal spine ($\forall/\exists_V$) is an essential step for the analysis of several core elements of linguistic expressions, e.g., adverbs, raising verbs, control verbs and attitude verbs. Some relevant data are shown below:

(1)
a. John sometimes laughs. $\exists_V s[...]$
b. Every girl sometimes laughs. $\forall_{NP} x[...\exists_V s[...]]$
c. John sometimes kisses every girl. $\exists_V s[...\forall_{NP} x[...]]$
d. Paul claims Mary apparently loves John. $\forall_{claim} s[...\forall_{apparently} s[...]]$
e. John wants Mary to sometimes laugh. $\forall_{wants} s[...\exists_V s[...]]$
f. Intentionally, John often meets Mary. $\forall_{intentionally} s[...\exists_V s[...]]$

According to Kallmeyer and Romero (2008), the scope of the elements that are attached to the verbal spine is determined by their surface position. In (1a) there is no problem of quantifier interaction. In contrast, the subject quantifier in (1b) interacts with the adverb. It can scope under or over the adverb which is naturally expected considering the nature of quantifiers. In the third example, the quantificational elements are syntactically restrained. Thus, their order is described as fixed in the literature. This means that we have *sometimes* > *every* scope order in (1c). (1d) and (1e) represent examples where attitude verbs interact with adverbs. Here, only the surface readings are possible. This shows the rigid scope of quantificational elements attached to the verbal spine. The last sentence (1f) makes clear how adverbs interact with each other. According to the LTAG representation of Kallmeyer and Romero (2008), the first modifier adjoins at the sentence level and the second one at the verb level. So, *intentionally* scopes over *often*.

Quantification at the verbal spine requires a suitable particular to quantify over. Intuitively, verbal quantification quantifies over instances of the respective predicate. In this paper we model this in terms of situations, following Kallmeyer and Romero (2008). Thus (1a) is represented as:

(2)
a. $[\![\ \textit{John sometimes laughs}\ ]\!] = \lambda s_0.\exists s[s \leq s_0 \wedge laugh(j,s)]$
b. $[\![\ \textit{sometimes}\ ]\!] = \lambda p_{<s,t>}.\lambda s_0.\exists s[s \leq s_0 \wedge p(s)]$ $\quad \forall X,Y.((i \multimap X) \multimap (Y \multimap X))$
c. $[\![\ \textit{laugh}\ ]\!] = \lambda x.\lambda s.laugh(x,s)$ $\quad (g \multimap (i \multimap f))$
d. $[\![\ \textit{John}\ ]\!] = j$ $\quad g$

This simple example of quantification of the verbal spine neither raises any theoretic issues, nor does it challenge the capabilities of the Glue semantics workbench. Even more complex interactions as in (1b) whose meaning is shown in (3) benefit from a Glue based computation. In what follows, we first show how the existing capabilities can deal with the simple examples and then use that discussion to explicate the challenges presented by the more complex examples.

(3) $[\![\ \textit{Every girl sometimes laughs.}\ ]\!] = \lambda s_0.\forall x[girl(x,s_0) \rightarrow \exists s[s \leq s_0 \wedge laugh(x,s)]]$
$\lambda s_0.\exists s[s \leq s_0 \wedge \forall x[girl(x,s) \rightarrow laugh(x,s)]]$

Let us first consider the appropriate lexical entries. According to these, the universal quantifier representing *every girl* would be assumed to take a predicate of type $< e, < s, t >>$ as its argument. This is shown in (4). Thus, it can take the verb *laugh* (as in (2c)) as its argument. The resulting formula can be picked up by the existential quantifier over situations, (2b) and results in the second reading in (3).

(4) $[\![\ \textit{Every girl}\ ]\!] = \lambda Q_{<e,<s,t>>}.\lambda s_s.\forall x[girl(x,s) \rightarrow Q(x)(s)]$ $\quad \forall X,Y.((g \multimap (i \multimap X)) \multimap (Y \multimap X))$

The second reading of (3) can be reached via the assumption of a resource g, which can combine with the verb *laugh*, which in turn becomes an appropriate resource for the verbal quantifier. Reintroduction of the variable then creates a resource that may be fed into the universal quantifier resulting in the remaining first reading in (3). It is important to note that the final situation binder provided by the quantifiers is a Glue variable, since it depends on the order of the application of the respective quantifiers.

Implementing this solution within the compilation-style Glue derivation provided by the Glue semantics workbench is one of the challenges this paper addresses. It is relevant for all semantics that quantify over several types. This is described below as part of the **technical considerations.**

We are now ready to consider how to derive the Glue premises from the f-structures. Many approaches to Glue semantics use a description-by-analysis approach, however, originally it was envisioned as a co-descriptive approach describing syntax and semantics in parallel in the lexicon.

However, before we can address this, we first have to consider the restrictions that the syntax places on adverbial quantifiers. In the previous example, we do not actually require any specific mechanisms. We can simply map the quantifiers onto their semantic structure, which is populated by the verb, to derive appropriate semantic representations (Dalrymple, 1999). A similar approach is used in the toy grammar provided by Meßmer and Zymla (2018). Consequently, the lexical entries can be given in lexicon simply as in Figure 1. We assume for the sake of completeness that a situation binder may be introduced given the existence of a `TENSE` feature. As a result, verbal elements make reference to a tense feature. Thus, untensed verbs only appear in the scope of an appropriate binder.

$$\begin{bmatrix} \text{PRED ‘laugh<[girl]>’} \\ \text{ADJUNCT} \quad \left\{ \begin{bmatrix} \text{PRED ‘sometimes’} \\ \text{ADV-TYPE sadv/vpadv} \end{bmatrix} \right\} \\ \text{SUBJ} \quad \begin{bmatrix} \text{PRED ‘girl’} \\ \text{SPEC} \quad \begin{bmatrix} \text{QUANT} \quad \begin{bmatrix} \text{PRED ‘every’} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{TENSE} \quad [...] \end{bmatrix}$$

(5)
a. $[\![\ \textit{sometimes}\ ]\!] = \lambda p_{<s,t>}.\lambda s_0.\exists s[s \leq s_0 \wedge p(s)]$ $\quad \forall X, Y.((\uparrow TENSE \multimap X) \multimap (Y \multimap X))$
b. $[\![\ \textit{laugh}\ ]\!] = \lambda x.\lambda s.laugh(x, s)$ $\quad (\uparrow SUBJ \multimap (\uparrow TENSE \multimap f))$
c. $[\![\ \textit{Every girl}\ ]\!] = \lambda Q_{<e,<s,t>>}.\lambda s_s.\forall x[girl(x, s) \rightarrow Q(x)(s)]$ $\quad \forall X, Y.((\uparrow SUBJ \multimap (\uparrow TENSE \multimap X)) \multimap (Y \multimap X))$

Figure 1: *Every girl sometimes laughs.*

Moving on to the more challenging example, this is provided by the interaction between adverbial modifiers as illustrated in Figure 2. The analysis is as produced by the English grammar available via the INESS XLE-Web site (Rosén et al., 2012). Kallmeyer and Romero (2008) argue that only one reading should be available for this sentence, namely: $\forall_{intentionally} > \exists_{often}$. This is opposed to existing analyses within and outside Glue semantics (Kallmeyer and Romero, 2008). However, this is also what the f-structure would make us believe may happen. Note however, that the lower scoping adverbial *often* is a packed, i.e. ambiguous structure. At the level of f-structure both quantifiers attach to the verbal spine as `ADJUNCT`s which suggests a free order of application. However, a description-by-analysis approach could rely on the syntactic input given in Figure 2 and generate an intermediate structure for adverbial quantification based on rules that fill the sentence level adverbial by a explicit sentence adverb instead of an ambiguous one. Thus we could present a syntactic solution of the problem. Implementing phenomena such as these has proven to be a fruitful area for exploring and evaluating co-descriptive vs. description by analysis approaches in computational Glue.

$$\begin{bmatrix} \text{PRED ‘visit<[John],[Mary]>’} \\ \text{ADJUNCT} \quad \left\{ \begin{bmatrix} \text{PRED ‘often’} \\ \text{ADV-TYPE sadv/vpadv} \end{bmatrix}, \begin{bmatrix} \text{PRED ‘intentionally’} \\ \text{ADV-TYPE sadv} \end{bmatrix} \right\} \\ \text{OBJ} \quad \begin{bmatrix} \text{PRED ‘Mary’} \end{bmatrix} \\ \text{SUBJ} \quad \begin{bmatrix} \text{PRED ‘John’} \end{bmatrix} \\ \cdots \end{bmatrix}$$

Figure 2: *Intentionally, John often visits Mary.*

Technical considerations: As described above, for a Glue-based situation semantics to work, we require a refined compilation process. The compilation process is described as a process that decomposes higher order Glue formulas into first-order Glue formulas. Thereby, first order Glue formulas are defined as linear implication formulas, which do not have a linear implication as their antecedent. If we adopt this notion, then NP quantifiers are second order Glue formulas, since they have one implication in their antecedent. However, if we introduce a

new particular, i.e. times or situations we get representations beyond second order. NPs quantifiers would look something like:

(6) a. $[\![\textit{Every}]\!] = \lambda P_{<e,<s,t>>}.\lambda Q_{<e,<s,t>>}\lambda s_s.\forall x[P(x)(s) \rightarrow Q(x)(s)]$
b. $[\![\textit{student}]\!] = \lambda x_e.\lambda s_s.student(x,s)$
c. $[\![\textit{Every student}]\!] = \lambda Q_{<e,<s,t>>}.\lambda s_s.\forall x[student(x,s) \rightarrow Q(x)(s)]$

This corresponds to the following linear logic formulas:

(7) a. $g \multimap (h \multimap i)$
b. $(g \multimap (h \multimap i)) \multimap \forall X.(j \multimap (k \multimap X) \multimap X)$

This requires a compilation step that is not anticipated in the existing algorithm. It is not difficult to imagine, how the compilation goes on the linear logic side: First the resource g would be compiled out and in a next step the resource h would be compiled out. This would result in two accidental lambda bindings which are in fact desired (Hepple, 1996). However, the part of the linear logic formula that is quantified over also requires compilation. This adds up to a total of four lambda binders, introducing four lambda functions in its scope. This means the compiled meaning representation (MR) of the quantifier in (6a).

(8) a. $\lambda m.\lambda p.\lambda r.\lambda u.MR(\lambda v.u)(\lambda s.r)(\lambda q.p)(\lambda n.m)$
b. $\lambda m_t.\lambda p_{s,t}.\lambda r_t.\lambda u_{s,t}.MR(\lambda v_e.u)(\lambda s_s.r)(\lambda q_e.p)(\lambda n_s.m)$

Based on the compilation process MR should have four different lambda bound arguments (ignoring the situation binder). However, a quantifier only takes two arguments: Its scope and its restrictor. One solution to this issue is to add appropriate typing to the lambda binders. This is shown in (8b). In this approach, the compilation process gives the formula initial lambda binder the type of the remaining linear logic formula. Thus, decompiling $g_e \multimap (h_s \multimap f_t)$ leads to an assumption $\{g_e\}$ and a formula initial lambda binder of type $< s,t >$. This typing is strictly necessary because else the lambda binders would simply take their next higher element as argument leading to only a single argument for the quantifier as opposed to the two that are required.

Additional features: In working on this issue and in cooperation with the Glue semantics community, the Glue semantics workbench is being extended as follows.

1. The possibility to return partial results: The analysis of partial results allows for better tracking of errors in lexical entries.

2. Debugging tool: The debugging tool primarily generates a log of ongoing processes that makes the tracking of errors easier. Thereby, processing parameters such as runtime can be globally and locally measured.

3. String-based internal semantic parsing: The Glue semantics workbench provides an implementation of Montague-Style semantics; however, it does not provide the capability of parsing strings as semantic objects in accordance with the implementation. Another sub-project is the possibility to parse strings to semantic objects. Furthermore, new semantic types can be introduced without in-depth knowledge of the code.

Summary: In this paper we present an extension of the semantics generally employed in Glue semantics to a situation based semantics. With this we can translate insights from Kallmeyer and Romero (2008) to Glue semantics. By this, we provide a resource sensitive situation semantics for scope effects between verbal quantification, NP quantification and further elements with quantificational properties such as raising verbs or propositional attitude verbs. We pay particular attention to the fact that verbal quantification is usually rigid while NP quantification allows for free scope variation. The work presented here is of interest for researchers that are concerned with the computational viability of linear logic as well as the exploration of different domains of semantics such as situation semantics or tense semantics.

References

Cinque, Guglielmo. 1999. *Adverbs and Functional Heads: A Cross-Linguistic Perspective*. Oxford University Press.

Dalrymple, Mary. 1999. *Semantics and Syntax in Lexical Functional Grammar: The Resource Logic Approach*. MIT Press.

Gotham, Matthew and Dag Haug. 2018. Glue semantics for Universal Dependencies. In M. Butt and T. H. King, editors., *Proceedings of the LFG'18 Conference, University of Vienna*, Pages 208–226. Stanford, CA: CSLI Publications.

Hepple, Mark. 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.

Joshi, Aravind K, Laura Kallmeyer, and Maribel Romero. 2008. Flexible composition in ltag: Quantifier scope and inverse linking. In *Computing meaning*, Pages 233–256. Springer.

Kallmeyer, Laura and Maribel Romero. 2008. Scope and Situation Binding in LTAG Using Semantic Unification. *Research on Language and Computation* 6(1):3–52.

Meßmer, Moritz and Mark-Matthias Zymla. 2018. The Glue Semantics Workbench: A modular toolkit for exploring Linear Logic and Glue Semantics. In M. Butt and T. H. King, editors., *Proceedings of the LFG'18 Conference, University of Vienna*, Pages 249–263. Stanford, CA: CSLI Publications.

Rosén, Victoria, Koenraad De Smedt, Paul Meurer, and Helge Dyvik. 2012. An Open Infrastructure for Advanced Treebanking. In J. Hajič, K. D. Smedt, M. Tadić, and A. Branco, editors., *META-RESEARCH Workshop on Advanced Treebanking at LREC2012*, Pages 22–29.