

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

Mark-Matthias Zymla

University of Konstanz

Gloria Sigwarth

University of Konstanz

Proceedings of the LFG'19 Conference

Australian National University

Miriam Butt, Tracy Holloway King, Ida Toivonen (Editors)


2019

CSLI Publications

pages 374–392

<http://csli-publications.stanford.edu/LFG/2019>

Keywords: glue semantics, workbench, sorted type theory

Zymla, Mark-Matthias, & Sigwarth, Gloria. 2019. On the Syntax/Semantics Interface in Computational Glue Semantics: A Case Study. In Butt, Miriam, King, Tracy Holloway, & Toivonen, Ida (Eds.), *Proceedings of the LFG'19 Conference, Australian National University*, 374–392. Stanford, CA: CSLI Publications. 

Abstract

This paper describes an extension of the Glue semantics workbench by Meßmer & Zymła (2018). In particular, we present a version of the workbench that can deal with (at least) semantic formalisms based on (Two-)Sorted Type Theory. We illustrate this by providing a semantic analysis of different constructions that involve quantification over the verb: adverbs, raising verbs, control verbs and attitude verbs. Furthermore, we describe some additional features of the workbench that aim at improving the workflow with the system.

1 Introduction

This paper expands on the effort initiated by Meßmer & Zymła (2018) to provide a (new) computational implementation of Glue semantics (Dalrymple 2001). This implementation is called Glue semantics workbench (GSWB) and provides a linear logic prover based on an algorithm by Hepple (1996) and an implementation of lambda calculus within a Java program.

We present a user study for the Glue semantics workbench and report on various issues that have been detected and fixed. This study is based on a situation semantics implementation as presented in Kallmeyer & Romero (2008) for LTAG. Their paper discusses several constructions that have been explored in the LFG literature from a resource logic perspective: adverbs, raising verbs, control verbs and attitude verbs etc. Example (1) illustrates some expressions we use as working examples for this paper. We follow the line of work presented in Asudeh (2005, 2000, 2002) and reconcile it with the aforementioned situation semantics approach. The goal thereby is to achieve an appropriate treatment of the scope interactions shown to the right in (1).

- (1)
- a. John sometimes laughs.
 - b. Every girl sometimes laughs. $\forall > \exists, \exists > \forall$
 - c. John sometimes kisses every girl. $\forall > \exists, \exists > \forall$
 - d. Paul claims Mary apparently loves John. $claims > apparently$

The case study presented here illustrates how the GSWB works for different typed semantic systems. The insights described in this paper thus provide a broader perspective on the use of the GSWB in general.

The paper is structured as follows: in Section 2 we discuss the theoretical background of situation semantics and the constructions presented in (1). In Section 3.2 we illustrate how we provide derivations for these expressions with the help of the GSWB. In Section 3.3 we present further smaller additions to the workbench the need for which arose during this case study. Section 4 concludes this paper.

[†] We thank the *VALIDA* project for funding. Furthermore, we thank Maribel Romero and Miriam Butt for helpful discussion. Finally, we thank the anonymous reviewers as well as the audience of the LFG2019 conference.

2 Theoretical background

The original GSWB by Meßmer & Zymla (2018) provided a Glue semantics prover and a simple syntax/semantics interface with the help of which basic compositional issues such as the quantifier interactions in (2) and adjectival modification could be illustrated.

- (2) Every man loves a woman.
- a. $\forall x[man(x) \rightarrow \exists y[woman(y) \wedge love(x, y)]]$
 - b. $\exists x[woman(x) \wedge \forall y[man(y) \rightarrow love(x, y)]]$

In this paper, we presuppose a basic understanding of Glue semantics that underlies these kinds of constructions. For reference we refer to Meßmer & Zymla (2018), Dalrymple et al. (1999) and Crouch & van Genabith (2000). Our main goal is to move from quantification and modification in the nominal domain to quantification in the domain of verbs. The primary observation, in this domain, is that quantification scope is much more restricted. More specifically, the semantics of verbal quantifiers are more strictly intertwined with the verb element they scope over (Cinque 1999). This is dubbed by Kallmeyer & Romero (2008) the issue of *Quantification at the verbal spine*. Their leading point of discussion is provided by the examples below in (3) and (4). The former illustrates the “rigid scope of (ad)verbal attachments” (Kallmeyer & Romero 2008). (4) on the other hand illustrates the flexible scope of NP quantifiers in relation to verb quantification.

- (3) John seems to sometimes laugh.
- a. seem(sometimes(laugh(j)))
 - b. *sometimes(seem(laugh(j)))
- (4) John seems to have visited everybody *seem > \forall , \forall > seem*

Kallmeyer & Romero (2008) proceed to discuss the scope restrictions for both (ad)verbal elements as well as NP quantifiers. Examples (5) and (6) (still by Kallmeyer & Romero (2008)) highlight the role of finite clause boundaries. They state the assumption that NP quantifiers are limited to the first finite clause containing the NP.

- (5) A student wants to meet every professor $\exists > \forall, \forall > \exists$
- (6) A student said that you met every professor $\exists > \forall, * \forall > \exists$

This paper tests some hypotheses made by Kallmeyer & Romero (2008) as well as researchers within the glue community (e.g. Asudeh (2005), Dalrymple et al. (1999)). However, we deviate slightly from the typical practice of assuming an underlying event semantic framework. This is explained in the next section.

2.1 Situation semantics

In this paper we opt for an approach that uses situation variables of type s in the object language (i.e. the meaning side in linear logic), next to individuals of type e . According to Kratzer (2019), situations in natural language semantics date back to Barwise (1981), where they were used for the semantics of perception reports. These days, situation semantics is an alternative, or sometimes an extension, to possible world semantics. Intuitively, linguistic expressions are evaluated with regard to partial worlds and not to complete worlds.

The use of situation semantics departs from the approaches presented in much previous work on the kind of constructions discussed in this paper. See, e.g., Asudeh (2000, 2002, 2005) for control and raising. On the other hand we take inspiration from works such as Haug (2008) and Lowe (2014) who deal with phenomena such as tense and aspect. Following Kallmeyer & Romero (2008), the chosen meaning language follows Two-Sorted Type Theory, meaning that each predicate has a situation argument in its semantic denotation (see (7-a)).

We believe that working with these more complex structures allows us to generalize to simpler structures.¹ In fact, many examples given in the works referred to here have been tested in the GSWB, although Haug (2008) has been slightly modified since the GSWB is not able to deal with compound types yet. This is due to the lack of multiplicative conjunction in the employed linear logic fragment.

In the present paper, the situation variable is assumed to be anchored to the TNS-ASP node and is, thus, taken to be part of the verb meaning constructor. This is illustrated in (7-a).²

- (7) a. $\llbracket \textit{laughs} \rrbracket = \lambda x_e. \lambda s_s. \textit{laughs}(x, s) : g \multimap (h \multimap f)$
 $\llbracket \textit{Tina} \rrbracket = t : g$
 $\llbracket \exists\text{-closure} \rrbracket = \lambda p_{st}. \exists s[p(s)] : (h \multimap f) \multimap f$
 b. $\llbracket \textit{Tina laughs} \rrbracket = \exists s[\textit{laughs}(t, s)] : f$

Especially, in the domain of control and raising verbs this opens up questions about the role of inflection on the embedded verb. This issue will be addressed briefly when reviewing Asudeh's proposals for these kinds of constructions in the light of the present paper in section 2.4.

2.2 Scope restrictions in Glue semantics

The advantage of using Glue semantics as a semantic formalism is, that many scope interactions fall out naturally as a consequence of the derivation system. This has long since been shown by Dalrymple et al. (1999) and was the first milestone of the GSWB. However, Gotham (2019) discusses flexible and rigid scope configurations of NPs and their treatment in LFG + Glue (i.e. Glue Semantics for Lexical

¹This is not meant as a critique of the more simple structures employed in some papers since they aim at resolving issues for which more complex semantics would needlessly complicate the analysis.

²In the derivations to come, we will usually omit the step of existential closure.

Functional Grammar). Furthermore, many more actual and apparent scope issues still remain unsolved, e.g. the role of definites (see Heim (2011) for an overview) as well as a comprehensive overview over the quantification of verbs, which takes center stage in this paper.

Similarly to Asudeh (2005) (in the domain of control and raising verbs), we aim at semantics of verb quantification, where the scope interactions are restricted naturally by the linear logic side of Glue semantics meaning constructors. This is mainly due to the fact, that we want to keep to the linear logic fragment employed in the GSWB as is.

To begin with, let us clarify the general semantics we envision for the types of verbs discussed in this paper. We base our work on Hintikka-style semantics (Hintikka 1969). An example of this is given in (8).³

$$(8) \quad \llbracket \textit{seem} \rrbracket = \lambda p_{\langle s,t \rangle} . \lambda s_0 . \forall s' [s' \in SEEM(s_0) \rightarrow p(s')]$$

Example (8) is a standard case of our treatment of adverbs, control and raising verbs, as well as attitude verbs, as universal quantifiers over situations in the semantics.⁴ The different kinds of quantifiers are distinguished by different restrictors over the situations they bind. Although we propose a fairly similar semantics for all the phenomena we discuss in this paper, as is to be expected, there will emerge subtle differences in the following sections of this paper. Some of these are influenced by the semantic side, e.g. the choice whether propositions or properties are embedded under a given kind of verb; others help to guarantee certain scope configurations. This will be done mainly by appropriate anchoring of situation variables to the f-structure and the use of linear logic variables. The next section on adverbs will illustrate this procedure.

2.3 Adverbs

In this section we discuss adverbs, in particular, those that attach directly to the verb. In computational LFG (i.e. in the English XLE grammar) adverbs are ambiguous between a sentence scope (s_{adv}) reading and a verb scope reading (v_{adv}). This ambiguity overlaps with the distinction between frequency adverbs and adverbs of quantification. In this paper, we focus on the latter. From an XLE perspective, we attribute quantificational adverbs or quantificational readings of adverbs to the category v_{adv} .⁵

We treat adverbs as simple modifiers. Modifier premises have a particular structure on their glue side: A linear implication with an equivalent antecedent and consequent. In the case we discuss below, they take propositions as their ar-

³The notation $SEEM(s_0)$ stands for the set $\{s' : s' \text{ conforms to what appears to be the case in } s_0\}$

⁴Not all quantifiers receive a Hintikka-style modal semantics, but they still follow the general template of universal quantification.

⁵The treatment of sentence adverbs and frequency adverbials (Bennett & Partee 1978) and the exact distinction between frequency adverbs and quantificational adverbs is left for future work.

gument. This is illustrated in (9). Although the adverb is attached directly to the top-level verb by means of the reference to $\uparrow\text{TNS-ASP}$ as well as \uparrow , the sentence has two readings. This is, of course, due to fact that the quantificational NP can scope under (surface scope) or over the adverb (inverse scope). (10) presents the appropriate lexical entries. The Glue formulas are demonstrated first in their general form and secondly, the resources are instantiated to nodes in the f-structure in Figure 1.⁶

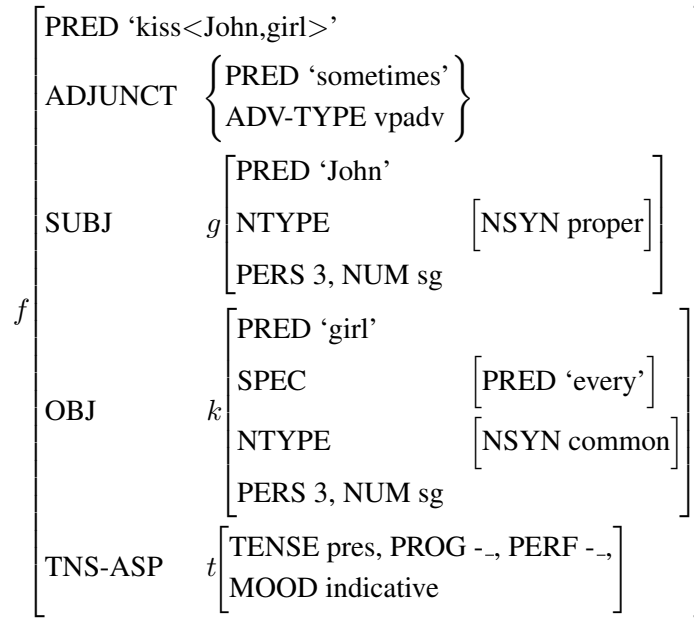


Figure 1: *John sometimes kisses every girl.*

- (9) John sometimes kisses every girl.
- (10) a. $\llbracket \text{John} \rrbracket = j : g$
b. $\llbracket \text{sometimes} \rrbracket = \lambda p_{\langle s,t \rangle} . \lambda s_0 . \exists s [s \leq s_0 \wedge p(s)] :$
 $(\uparrow \text{TNS-ASP} \multimap \uparrow) \multimap (\uparrow \text{TNS-ASP} \multimap \uparrow)$
 $= (t \multimap f) \multimap (t \multimap f)$
c. $\llbracket \text{kiss} \rrbracket = \lambda y_e . \lambda x_e . \lambda s . \text{kiss}(x, y, s) :$
 $\uparrow \text{OBJ} \multimap (\uparrow \text{SUBJ} \multimap (\uparrow \text{TNS-ASP} \multimap \uparrow))$
 $= k \multimap (g \multimap (t \multimap f))$
d. $\llbracket \text{every girl} \rrbracket = \lambda Q_{\langle e, \langle s,t \rangle \rangle} . \lambda s . \forall x [\text{girl}(x, s) \rightarrow Q(x)(s)] :$
 $\forall X, Y . ((\uparrow \text{OBJ} \multimap (Y \multimap X)) \multimap (Y \multimap X))$
 $= \forall X, Y . ((k \multimap (Y \multimap X)) \multimap (Y \multimap X))$

⁶Common nouns also have situation variables as can be inferred from the denotation of the quantifier *every girl*. It is less clear how to anchor this situation variable in the f-structure. If need be for anchoring, it would probably be somewhere in the `NTYPE` grammatical feature that specifies various semantic features in f-structure. `NTYPE` is used uniformly across the ParGram XLE grammars (Butt et al. 2002).

Note that the quantifier now maps onto elements of type $\langle s, t \rangle$ both of which are treated as variables in its denotation. This is how we model the more flexible nature of NP quantifiers. In contrast to the quantificational NP, there are no Glue variables needed in the meaning constructor for *sometimes*, because the adverb takes scope where it attaches.

Using the meaning constructors in (10), the Glue proofs for both readings of (9) can be constructed. The proofs are shown in Figures 2 and 3. For readability, the semantic side is omitted.

$$\begin{array}{c}
\frac{[k]^1 \quad k \multimap (g \multimap (t \multimap f))}{g \multimap (t \multimap f)} \multimap_E \quad g \multimap_E}{\frac{(k \multimap (Y \multimap X)) \multimap (Y \multimap X) \quad \frac{t \multimap f}{k \multimap (t \multimap f)} \multimap_{I,1}}{(t \multimap f) \multimap (t \multimap f)} \multimap_E \quad t \multimap f}{t \multimap f} \multimap_E} \multimap_E \\
\lambda s_0. \exists s [s \leq s_0 \wedge \forall x [girl(x, s) \rightarrow kiss(j, x, s)]]
\end{array}$$

Figure 2: *John sometimes kisses every girl* – Surface scope reading

$$\begin{array}{c}
\frac{[k]^1 \quad k \multimap (g \multimap (t \multimap f))}{g \multimap (t \multimap f)} \multimap_E \quad g \multimap_E}{\frac{(t \multimap f) \multimap (t \multimap f) \quad t \multimap f}{t \multimap f} \multimap_E} \multimap_E \\
\frac{(k \multimap (Y \multimap X)) \multimap (Y \multimap X) \quad \frac{t \multimap f}{k \multimap (t \multimap f)} \multimap_{I,1}}{t \multimap f} \multimap_E}{\lambda s_0. \forall x [girl(x, s_0) \rightarrow \exists s' [s' \leq s_0 \wedge kiss(j, x, s')]]} \multimap_E
\end{array}$$

Figure 3: *John sometimes kisses every girl* – Inverse scope reading

2.4 Control and Raising verbs

Asudeh (2005) building on Landau (2003) illustrates how Glue semantic accounts for the differences between control and raising verbs ((11) and (12)). He compellingly argues that an analysis for control and raising comes naturally in Glue semantics, generalizing over the specific properties of both kinds of verbs. These specific properties arise from the fact that both of these constructions invoke structure sharing. In particular, the subject is shared between the matrix verb and the embedded verb.

(11) Gonzo tried to leave. control

(12) Gonzo seemed to leave. raising

The only difference between the two kinds of verbs lies in their PRED structure. Raising verbs, as the name suggest, treat the SUBJ as a thematic subject. This is the case for expletive or raising subjects (Asudeh 2005). On the other hand, the subjects of control verbs are thematic subjects as shown in Figure 4 on the right. How does this affect their semantics? We begin with explaining the semantic composition of raising verbs in the present framework.

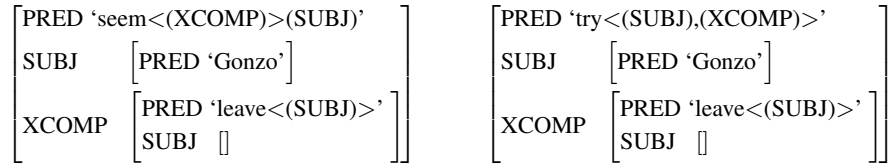


Figure 4: Structure sharing in control and raising (Asudeh 2005)

The meaning constructor for our treatment of raising verbs, inspired by Asudeh (2005, 2002, 2000), is shown in (13). The crucial component is the linear logic side, which almost looks like a modifier, but it is anchored to two different TNS-ASP nodes. Note that the quantifier denotation proposed in the proof for adverbs now allows us to derive the scope ambiguity shown in (14). In fact, by comparison to control verbs, we will see that the meaning constructor presented above in (13) makes explicit the distinction between de dicto and de re interpretations in the semantics.

$$\begin{aligned}
 (13) \quad \llbracket \textit{seem} \rrbracket &= \lambda p_{<s,t>}. \lambda s_0. \forall s' [s' \in SEEM(s_0) \rightarrow p(s')] : \\
 &(\uparrow \text{TNS-ASP}_{\text{XCOMP}} \multimap \uparrow \text{XCOMP}) \multimap (\uparrow \text{TNS-ASP}_f \multimap f) \\
 &= (i \multimap h) \multimap (t \multimap f)
 \end{aligned}$$

- (14) Every girl seems to laugh.
- a. every(girl,seem(laugh))
 - b. seem(every(girl(laugh)))

The proofs for the two readings are given in Figure 5 and 6. As illustrated above, the verb *seem* takes a proposition as its argument which, in combination with the NP quantifier, makes two readings available. Control verbs on the other hand do not allow for scope ambiguity with subject NP quantifiers. We contrast raising and control verbs by making raising verbs take propositions as their argument and control verbs verbs take properties as their argument. This is discussed next.

$$\frac{\frac{[g]^1 \quad g \multimap (i \multimap h)}{i \multimap h} \multimap_E \quad (i \multimap h) \multimap (t \multimap f)}{\frac{t \multimap f}{g \multimap (t \multimap f)} \multimap_{I,1}} \multimap_E \quad \frac{(g \multimap (Y \multimap X)) \multimap (Y \multimap X)}{t \multimap f} \multimap_E} \multimap_E$$

$$\lambda s_0. \forall x [girl(x, s_0) \rightarrow \forall s' [s' \in SEEM(s_0) \rightarrow laugh(x, s')]]$$

Figure 5: *Every girl seems to laugh* – Surface scope reading

$$\frac{(i \multimap h) \multimap (t \multimap f)}{t \multimap f} \multimap_E \quad \frac{(g \multimap (Y \multimap X)) \multimap (Y \multimap X) \quad g \multimap (i \multimap h)}{i \multimap h} \multimap_E} \multimap_E$$

$$\lambda s_0. \forall s' [s' \in SEEM(s_0) \rightarrow \forall x [girl(x, s') \rightarrow laugh(x, s')]]$$

Figure 6: *Every girl seems to laugh* – Inverse scope reading

Comparing control and raising verbs in a situation semantics account has lead us an interesting puzzle with respect to the difference between the two. To understand this, let us look at our proposed denotation for control verbs:

$$(15) \quad \llbracket try \rrbracket = \lambda P_{\langle e, \langle s, t \rangle \rangle}. \lambda x_e. \lambda s_0. \forall s' [s' \in INT_x(s_0) \rightarrow P(x)(s')] : \\
\begin{aligned}
& (\uparrow \text{XCOMP SUBJ} \multimap (\uparrow \text{TNS-ASP}_{\text{XCOMP}} \multimap \uparrow \text{XCOMP})) \\
& \multimap (\uparrow \text{SUBJ} \multimap (\uparrow \text{TNS-ASP}_f \multimap \uparrow)) \\
& = (g \multimap (i \multimap h)) \multimap (g \multimap (t \multimap f))
\end{aligned}$$

Following Asudeh (2000, 2002, 2005), the control verb *try* is treated as taking a property of type $\langle e, \langle s, t \rangle \rangle$ as argument. The puzzle lies in the relation between the situation variable and the `SUBJ` that the antecedent and the consequent of the linear logic formula are bound to. This remains implicit in the work by Asudeh (2000, 2002, 2005). Concretely, in the previously cited work, properties could be understood as having a compound type in the consequent of their linear logic representation: $g_e \multimap (h_s \otimes f_t)$. Our own proposal, on the other hand, can be understood as strengthening the relation between the `SUBJ` and the situation variable, by illustrating how each element is bound in the f-structure.

However, note that we co-bind the individual type variable and the situation variable with the matrix verb. Thus, the result is a semantics of *try* whose complement is neither a property nor a proposition, but a truth value. While this point requires more research, we tentatively assume that this allows us to explain why raising complements allow for aspectual modification, while control verbs do not.⁷

⁷Following this idea, aspectual features would be modifiers from propositions to propositions. The following data can explained by this:

In this paper we pay particular attention to the scopal restrictions of control and raising verbs. As Asudeh (2005) notes “scopal elements can take both wide and narrow scope with respect to raising verbs but can only take wide scope with respect to control verbs”. This observation is in agreement with (4) from Kallmeyer & Romero (2008) and is captured in the approach presented above. A proof using the control verb *try* is shown in Figure 7. In contrast to the previously discussed raising verb *seem*, only one reading is available as predicted.

$$\frac{\frac{g \multimap (i \multimap h) \quad (g \multimap (i \multimap h)) \multimap (g \multimap (t \multimap f))}{(g \multimap (Y \multimap X)) \multimap (Y \multimap X) \quad g \multimap (t \multimap f)} \multimap_E}{t \multimap f} \multimap_E$$

$$\lambda s. \forall x [girl(x, s) \rightarrow \forall s' [s' \in INT_x(s) \rightarrow laugh(x, s')]]$$

Figure 7: Glue proof without meanings: *Every girl tries to laugh.*

2.5 Attitude verbs

In contrast to control and raising verbs (in English), attitude verbs embed a finite clause. We already established that NP quantifiers are restricted to the first finite clause they occur in. However, we did not provide a concrete implementation for this. The intuitive idea is, that in LFG, the boundaries of such a clause are provided by the governing f-structure. In the XLE grammars of the ParGram project, finite clauses are distinguished by having a `CLAUSE-TYPE` feature. This does not exist in f-structures embedded under e.g., control and raising verbs, which, in English, describe non-finite clauses. Consequently, a quantificational NP can scope over a control verb like *try* and over a raising verb like *seem*, but not over an attitude verb like *think*. Thus, only the surface scope reading should be possible for (16).

(16) Mary thinks John loves every girl.

To model this difference between the verbs involving functional control and attitude verbs in the Glue semantics analysis of this paper, the lexical entry for the quantificational NP has to be modified in the scope of a `COMP`, if we want to preserve the general idea pursued in this paper. Thus, we require a mechanism that anchors the variables scoping over situations in the denotation of quantifiers to specific Glue constants, i.e. `TNS-ASP` nodes. This makes the situation variable the pivotal element in scope restrictions (section 3.1).

- | | | |
|------|--|---------------------------|
| (i) | a. John seems to have visited everybody. | Kallmeyer & Romero (2008) |
| | b. Mary seems to be going to the park. | |
| (ii) | a. ??John tries to have visited everybody. | |
| | b. #John tries to be going to the park. | |

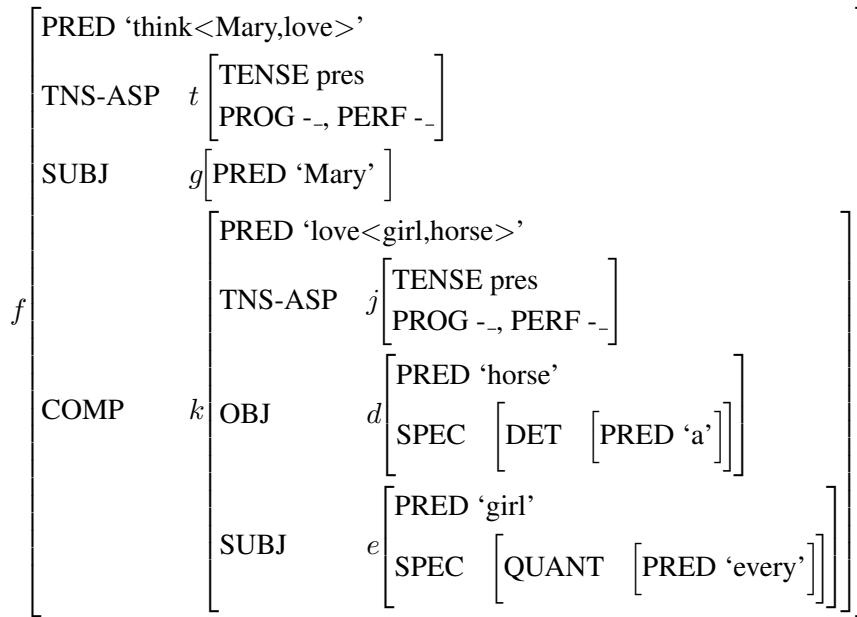


Figure 8: *Mary thinks every girl loves a horse.*

We illustrate the general idea in terms of example (17). It has two readings, as *a* can scope over *every* within the embedded finite clause. This means, within the scope boundary provided by the COMP quantifiers are to work as expected.

(17) Mary thinks every girl loves a horse.

- (18) a. $\llbracket \textit{think} \rrbracket = \lambda p_{\langle s,t \rangle} . \lambda x_e . \lambda s_0 . \forall s' [s' \in \textit{DOX}_x(s_0) \rightarrow p(s')]$:
 $(\uparrow \textit{TNS-ASP}_{\textit{COMP}} \multimap \uparrow \textit{COMP})$
 $\multimap (\uparrow \textit{SUBJ} \multimap (\uparrow \textit{TNS-ASP}_f \multimap \uparrow))$
 $= (j \multimap k) \multimap (g \multimap (t \multimap f))$
- b. $\llbracket \textit{every girl} \rrbracket = \lambda Q_{\langle e, \langle s,t \rangle \rangle} . \lambda s . \forall x [\textit{girl}(x, s) \rightarrow Q(x)(s)]$:
 $\forall X . ((\uparrow \textit{COMP SUBJ} \multimap (\uparrow \textit{TNS-ASP}_{\textit{COMP}} \multimap X))$
 $\multimap (\uparrow \textit{TNS-ASP}_{\textit{COMP}} \multimap X))$
 $= \forall X . ((e \multimap (j \multimap X)) \multimap (j \multimap X))$
- c. $\llbracket \textit{a horse} \rrbracket = \lambda Q_{\langle e, \langle s,t \rangle \rangle} . \lambda s . \exists x [\textit{horse}(x, s) \wedge Q(x)(s)]$:
 $\forall X . ((\uparrow \textit{COMP OBJ} \multimap (\uparrow \textit{TNS-ASP}_{\textit{COMP}} \multimap X))$
 $\multimap (\uparrow \textit{TNS-ASP}_{\textit{COMP}} \multimap X))$
 $= \forall X . ((d \multimap (j \multimap X)) \multimap (j \multimap X))$

The combination steps are those for the typically expected quantifier ambiguity within the complement (see Figure 9 and 10). The result is then simply combined with the attitude verb, which in turn is combined with the subject of the matrix sentence to yield the final result.

$$\begin{array}{c}
\frac{[d]^1 \quad d \multimap (e \multimap (j \multimap k))}{e \multimap (j \multimap k)} \multimap_E \quad \frac{[e]^2}{j \multimap k} \multimap_E}{\frac{(d \multimap (j \multimap X)) \multimap (j \multimap X) \quad d \multimap (j \multimap k)}{j \multimap k} \multimap_E} \multimap_E \\
\frac{(e \multimap (j \multimap k)) \multimap (j \multimap k) \quad \frac{j \multimap k}{e \multimap (j \multimap k)} \multimap_{I,2}}{j \multimap k} \multimap_E \quad \frac{(j \multimap k) \multimap (g \multimap (t \multimap f))}{g \multimap (t \multimap f)} \multimap_E}{\frac{g \multimap (t \multimap f) \quad g}{t \multimap f} \multimap_E} \multimap_E
\end{array}$$

every > a:

$$\lambda s_0. \forall s' [s' \in DOX_m(s_0) \rightarrow \forall x [girl(x, s') \rightarrow \exists y [horse(y, s') \wedge love(x, y, s')]]]$$

Figure 9: Glue proof without meanings:

Mary thinks every girl loves a horse – Reading: *every > a*

$$\begin{array}{c}
\frac{[d]^1 \quad d \multimap (e \multimap (j \multimap k))}{e \multimap (j \multimap k)} \multimap_E \quad \frac{(e \multimap (j \multimap X)) \multimap (j \multimap X)}{j \multimap k} \multimap_E}{\frac{(d \multimap (j \multimap k)) \multimap (j \multimap k) \quad \frac{j \multimap k}{d \multimap (j \multimap k)} \multimap_{I,1}}{j \multimap k} \multimap_E} \multimap_E \\
\frac{j \multimap k \quad \frac{(j \multimap k) \multimap (g \multimap (t \multimap f))}{g \multimap (t \multimap f)} \multimap_E}{g \multimap (t \multimap f)} \multimap_E \quad \frac{g}{g} \multimap_E}{t \multimap f} \multimap_E
\end{array}$$

a > every:

$$\lambda s_0. \forall s' [s' \in DOX_m(s_0) \rightarrow \exists y [horse(y, s') \wedge \forall x [girl(x, s') \rightarrow love(x, y, s')]]]$$

Figure 10: Glue proof without meanings:

Mary thinks every girl loves a horse – Reading: *a > every*

3 GSWB Case Study

In this section we test the analysis presented above within the GSWB. We discuss the challenges that arose during this case study and present a solution based on Lev (2007). The case study follows a simple procedure. We designed a small Treebank and produced a list of premise sets for each sentence manually. Testing them in the workbench revealed a number of issues, the most crucial of which and their solution we discuss in the next section.

3.1 Scope restrictions

As discussed in the previous section, most of the scope predictions discussed in the previous section fall out naturally due to the chosen Glue semantics representation. However, the final example of attitude verbs does not fit into the general picture of this paper in the sense that we modified the involved NP quantifiers. In particular, we anchored the situation variable used in NP quantifiers to constants corresponding to certain *TNS-ASP* nodes.

We assume that this operation is done by means of some rewriting system (in particular, the rewrite system in XLE). As Gotham (2019) points out, this is not desirable from a theoretical point of view, but we want to avoid changing the linear logic fragment of the GSWB. This rules out the Gotham (2019) approach and leaves us with constraining either proofs or meaning constructors. We chose to do the latter. Concretely, quantifiers that occur within a specific *COMP* are rewritten such that they are anchored to the specific *COMP*'s *TNS-ASP* node.

3.2 Typed semantics in the GSWB

This section describes the main results of our exploration of the capabilities of the workbench. The primary issue was that the GSWB could not deal properly with typed semantic representations. To understand this, let us first describe the implementation of the Hepple (1996) algorithm in the Glue semantics workbench Meßmer & Zymla (2018).

The algorithm presented in Meßmer & Zymla (2018) is a recursive algorithm following the proposal made by Hepple (1996) quite strictly. Higher-order premises (i.e. premises which have a linear implication as antecedent) are compiled. The compilation process is a simplification of complex linear logic formulas with the result of only having to deal with simple combination steps between atoms and corresponding linear implications. The main issue with the algorithm used in Meßmer & Zymla (2018) lies within this compilation process.

The process works as follows: the antecedent of a given complex premise is divided into antecedent and conclusion. This antecedent is cut off from the original formula. The result is two formulas: the remains of the original formula and a compiled-out assumption. The process is shown in (19).⁸ The new assumption

⁸The indices in brackets are used to track premises. Introducing a new assumption also introduces

receives an unused variable as meaning representation. This variable is indirectly anchored to the original meaning representation. First, the whole representation is wrapped by a lambda binder which binds another unused variable. Second, the original meaning representation receives as argument a lambda function which binds the variable of the compiled out assumption and scopes over the previously introduced fresh variable.

$$(19) \quad \alpha : (a \multimap b) \multimap c [0] \Rightarrow_{\text{compile}} \lambda u. \alpha(\lambda v. u) : b[a] \multimap c [0]; \\ v : \{a\} [1]$$

A formally undesirable consequence of Hepple’s (1996) algorithm is, that it relies on accidental binding. The newly introduced lambda binder in the argument of the meaning representation α needs to bind the compiled-out variable v .⁹ A concrete example is given in figure 11.

$$\frac{\frac{H[g_2] \multimap H : \lambda u. \lambda P. \forall x[\text{person}(x) \wedge P(x)](\lambda v. u) \quad \frac{g_1 \multimap f : \lambda y. \text{sleep}(y) \quad \{g_2\} : v}{f\{g_2\} : \text{sleep}(v)} [\text{H/f}]}{f : \lambda P. \forall x[\text{person}(x) \wedge P(x)](\lambda v. \text{sleep}(v))} \quad \beta\text{-conversion}}{f : \forall x[\text{person}(x) \wedge \text{sleep}(x)]}$$

Figure 11: Every person sleeps. – Hepple style

This process works for compiling simple antecedents, however, the implementation in Meßmer & Zymla (2018) does not deal properly with the recursive nature of the algorithm. To illustrate this, consider the nominal quantifier *every girl* translated into a version that includes situations as semantic objects of type s in the ontology.

$$(20) \quad \begin{aligned} \text{a. } \llbracket \text{Every} \rrbracket &= \lambda P_{\langle e, \langle s, t \rangle \rangle}. \lambda Q_{\langle e, \langle s, t \rangle \rangle}. \lambda s_s. \forall x [P(x)(s) \rightarrow Q(x)(s)] \\ \text{b. } \llbracket \text{student} \rrbracket &= \lambda x_e. \lambda s_s. \text{student}(x, s) \\ \text{c. } \llbracket \text{Every student} \rrbracket &= \lambda Q_{\langle e, \langle s, t \rangle \rangle}. \lambda s_s. \forall x [\text{student}(x, s) \rightarrow Q(x)(s)] \end{aligned}$$

This corresponds to the following linear logic formulas:¹⁰

$$(21) \quad \begin{aligned} \text{a. } g \multimap (h \multimap i) \\ \text{b. } (g \multimap (h \multimap i)) \multimap \forall X, Y. (j \multimap (t \multimap X)) \multimap (t \multimap X) \end{aligned}$$

This requires a compilation step that is not anticipated in the algorithm that was previously implemented in the workbench. It is not difficult to imagine, how the compilation goes on the linear logic side: First the resource g would be compiled out. 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.

a new index. However, this is not relevant for discussing the flaw of the algorithm.

⁹Thus, the functional application used within the linear lambda calculus (the lambda calculus used for the compilation) is different from classical functional application in so far that it does allow for this accidental binding. As a result of this, Hepple’s (1996) original version is not readily compatible with out-of-the-box beta conversion tools. Another issue, that the present paper remedies.

¹⁰The issue occurs in particular in COMP embedded NP quantifiers, since our original quantifier denotation results in a modifier resource, which is not compiled (Meßmer & Zymla 2018).

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 (20-a) is bound by four lambdas after the compilation steps.

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

The issue here is the number of arguments that are created during the compilation process. At first glance, (22) suggests, that there are four arguments to the denotation of the quantifier. However, it is apparent that there should only be two arguments (corresponding to λP and λQ in (20-a)). In the original single-type version of the prover, saturating the outermost lambda results in a number of undesired lambda applications. The result of these applications is, that there is only one argument, instead of two, that has been passed along through all the lambda slots. There are several possible solutions to this problem. We opt for a solution along the lines of Lev (2007).

However, first some fundamental issues have to be fixed. The first step is to properly type the lambda binders as in (23) to achieve the desired solution given in the example. This approach runs into a number of problems still. While the formula given in (23) technically works, deriving it via the compilation process is not straightforward. To solve this issue, let us look at it from another perspective:

There is a second problem with the original Hepple (1996) approach, namely, that it requires accidental binding. Lev (2007) presents a way to circumvent the need for this semantically “unsound” approach. The gist of his improved algorithm is that the meaning side of the compiled resource is not modified, but as before, the compilation process creates a new meaning side variable of the type of the resource that has been compiled out on the meaning side. However, this step does not coincide with adding a new argument to the meaning representation as done in (22) (the four lambda binders that apply to the meaning representation MR are introduced by the compilation step).

The role of making sure that the compiled out variables are inserted into the proof appropriately is taken by the discharge system in Lev (2007). Discharges mark where a specific variable has been compiled out. Now, if a variable that has been compiled out (or any element that has combined with that variable) combines with the meaning representation that has discharged it, a modified functional application procedure is applied. Lev (2007) formalizes this as shown in Figure 12.

$$\frac{\phi : A : S_1 \quad \delta : A_L \rightarrow B : S_2 \quad \text{provided } S_1 \cap S_2 = \emptyset \text{ and } L \subset S_1}{\delta(\lambda v_{i_1}, \dots, \lambda v_{i_n}.\phi) : B : S_1 \cup S_2 \quad \text{and } L = [i_1, \dots, i_n]}$$

Figure 12: Functional Application in Lev (2007) style compilation

Given a function and an argument with compatible linear logic resources, the two

are combined only if the discharges of the function are a subset of the assumptions made for the argument. In that case, for all discharges a lambda introduction step is executed. This requires the discharges to be a list rather than set, to preserve appropriate types. Elements that are cut off (i.e. discharged) have to be reintroduced in the reverse order. The whole procedure is exemplified in (24) to (26).

$$(24) \quad \text{Premises: } \begin{array}{l} a_e \multimap b_t \multimap f_t : \alpha \\ a_e \multimap b_t : \beta \end{array}$$

$$(25) \quad \text{Compiled: } \begin{array}{l} b_t[a] \multimap f_t : \alpha \\ \{a_e\} : v_e \\ a_e \multimap b_t : \beta \end{array}$$

$$(26) \quad \frac{\frac{a_e \multimap b_t : \beta \quad \{a_e\} : v_e}{b_t\{a_e\} : \beta(v_e)} \quad b_t[a] \multimap f_t : \alpha}{f_t : \alpha(\lambda v_e. \beta(v_e))}$$

3.3 Further additions to the GSWB

This section very briefly describes some additional features of the workbench which were added when implementing the verbal scoping analysis described in this paper. These involve a debugging mode and improved file handling.

The debugging mode provides some basic information about performance of the system. This is illustrated in Figure 13. As shown there, the computation time is measured. Additionally, information about attempted inference (combination) steps is collected. This includes those leading up to the proof as well as those that are not used in the final derivation. Furthermore, the number of compilation steps is counted. This metric has been added to provide transparency about the conversion from higher-order linear logic formulas to first-order formulas.

```
Debugging report :
The following data was collected :
computationTime: 12ms
Number of iterations through Sequent: 18
Number of combination steps: 12
Number of proper compilation steps:2
```

Figure 13: Debugging mode sample output

In addition to the debug mode, several file handling features have been added. Primarily, the system now allows the user to specify input and output files, which makes it easier to use the workbench in a pipeline architecture. Furthermore, the workbench now also allows the user to process multiple proofs from a single file, another functionality that has been developed with the creation of pipelines in mind. In general, the modular nature of the GSWB was been improved.

4 Conclusion

In this paper we presented a case study of computational Glue semantics that serves as the first use case of the Glue semantics workbench. This study was based on two-type situation semantics and allowed us to cover phenomena in the domain of quantification over and within certain kinds of verbs, such as control and raising verbs. Our main goal was to explore scope interactions between quantificational NPs and kinds of verbs that are semantically quantifiers. We have shown that many scope interactions can be derived by anchoring quantifiers to specific TNS-ASP nodes, which are assumed to map to situation variables. We highlighted further possible roles of TNS-ASP nodes in the distinction of control vs. raising verbs. Furthermore, we established that quantifier scope cannot be simply restricted by TNS-ASP nodes when embedded under an attitude verbs. This is due to the fact that quantifiers do not distinguish between matrix clause NPs and COMP clause NPs. We have provided a tentative technical solution but more work needs to be done in this area.

By virtue of implementing a situation semantics approach for Glue, we could test the GSWB intensively. Although the case study has shown that there is a lot of work yet to be done with respect to the GSWB, it helped get a better understanding of what has to be done. The result is a more functionally robust and flexible system for working with Glue semantics. The next important step is to improve the syntax/semantics interface. Both co-descriptive and description-by-analysis approaches are currently being developed and help to push the improvement of the GSWB.

References

- Asudeh, Ash. 2000. Functional Identity and Resource-Sensitivity in Control. In *Proceedings of the LFG00 Conference*, 2–24.
- Asudeh, Ash. 2002. A Resource-Sensitive Semantics for Equi and Raising. In David Beaver, Stefan Kaufmann, Brady Clark & Luis D. Casillas Martínez (eds.), *The Construction of Meaning*, 1–21. Stanford, CA: CSLI Publications.
- Asudeh, Ash. 2005. Control and Semantic Resource Sensitivity. *Journal of Linguistics* 41(3). 465–511.
- Barwise, Jon. 1981. Scenes and Other Situations. *The Journal of Philosophy* 78(7). 369–397.
- Bennett, Michael & Barbara Hall Partee. 1978. Toward the Logic of Tense and Aspect in English. In Barbara H Partee (ed.), *Compositionality in Formal Semantics*, vol. 84, Blackwell Publishing Ltd.
- Butt, Miriam, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi & Christian Rohrer. 2002. The Parallel Grammar Project. In *Proceedings of the 2002 Work-*

- shop on Grammar Engineering and Evaluation*, vol. 15, 1–7. Association for Computational Linguistics.
- Cinque, Guglielmo. 1999. *Adverbs and Functional Heads: A Cross-Linguistic Perspective*. Oxford University Press on Demand.
- Crouch, Richard & Josef van Genabith. 2000. Linear Logic for Linguists. URL: <http://www2.parc.com/istl/members/crouch>.
- Dalrymple, Mary. 2001. *Lexical Functional Grammar*, vol. 34. New York: Academic Press.
- Dalrymple, Mary, John Lamping, Fernando Pereira & Vijay Saraswat. 1999. Quantification, Anaphora, and Intensionality. In Mary Dalrymple (ed.), *Semantics and Syntax in Lexical Functional Grammar – The Resource Logic Approach*, 39–89.
- Gotham, Matthew. 2019. Constraining Scope Ambiguity in LFG+Glue. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG’19 Conference*, CSLI Publications.
- Haug, Dag. 2008. Tense and Aspect for Glue Semantics: The Case of Participial XADJ’s. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG08 Conference*, 291–311. Cent. Study Lang. Inf. Stanford, CA.
- Heim, Irene. 2011. Definiteness and Indefiniteness. In Paul Portner, Klaus von Stechow & Claudia Maienborn (eds.), *Semantics – Noun Phrases and Verb Phrases*, .
- Hepplé, Mark. 1996. A Compilation-Chart Method for Linear Categorical Deduction. In *Proceedings of the 16th Conference on Computational Linguistics-Volume 1*, 537–542. Association for Computational Linguistics.
- Hintikka, Jaakko. 1969. Semantics for Propositional Attitudes. In Donald Davidson, Jaako Hintikka, G. Nuchelmas & Wesley C. Salmon (eds.), *Models for Modalities*, 87–111. Dordrecht: D. Reidel Publishing Company.
- Kallmeyer, Laura & Maribel Romero. 2008. Scope and Situation Binding in LTAG Using Semantic Unification. *Research on Language and Computation* 6(1). 3–52.
- Kratzer, Angelika. 2019. Situations in Natural Language Semantics. In Edward N. Zalta (ed.), *The Stanford Encyclopedia of Philosophy*, Summer 2019 edn. <https://plato.stanford.edu/archives/sum2019/entries/situations-semantics/>.
- Landau, Idan. 2003. Movement out of Control. *Linguistic Inquiry* 34(3). 471–498.
- Lev, Iddo. 2007. *Packed Computation of Exact Meaning Representations*: Stanford University dissertation.
- Lowe, John. 2014. Gluing Meanings and Semantic Structures. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG14 Conference*, CSLI Publications.

Meßmer, Moritz & Mark-Matthias Zymla. 2018. The Glue Semantics Workbench: A Modular Toolkit for Exploring Linear Logic and Glue Semantics. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG'18 Conference, University of Vienna*, 249–263. Stanford, CA: CSLI Publications. <http://cslipublications.stanford.edu/LFG/2018/lfg2018-messmer-zymla.pdf>.