A new System for Linguistic Rewriting and Formal Semantics

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

Universität Konstanz

About

We present a system for **rewriting** and **expansion** of linguistic annotations. It is illustrated by virtue of syntax/semantics rules producing semantic representations based on Glue semantics.

- Interfaced with the Stanford CoreNLP and the XLE for **syntactic parsing**
- Interfaced with the Glue Semantics Workbench for **semantic analysis**
- **Micro-service** architecture
- Developed in **Java** and licensed under **GPL 3.0**
- Available at: `https://github.com/Mmaz1988/abstract-syntax-annotator-web`
- Feedback and feature requests welcome!

Simple Graph matching techniques are combined with constraint and equation checking techniques inspired by Lexical Functional Grammar (LFG) to define expansion and rewrite rules.

- Inspired by the Packed Rewrite System in XLE (Crouch, 2005)
- Translation of graphs into fact notation → useful for ambiguity management via packing (Maxwell III and Kaplan, 1989)

(1) a. Fact notation:
```
#[a-z]+ REL {#[a-z]+|VALUE}
```
b.
```
query ==> expansion.
```

- Choice-packing currently in development

→ Ambiguous rules don't scale well yet

The system makes use of the **Glue Semantics Workbench**(GSWB; Meßmer and Zymla (2018)) for semantic analysis.

- Glue semantics is a **resource conscious** approach to formal semantics (Dalrymple et al., 1999)
- Composition is guided by **linear logic**
- Compatible with **various semantic representations** that preserve the Curry-Howard-isomorphism, e.g. λ-DRT, FOL, and other semantics based on lambda calculus

$$\dfrac{\dfrac{\lambda x.\lambda y.loves(x,y) : 1 \multimap (3 \multimap 2) \quad john : 1}{\lambda y.kiss(john,y) : 3 \multimap 2} \quad mary : 3}{loves(john, mary) : 2}$$

Figure 1: Glue derivation of *John loves Mary.*

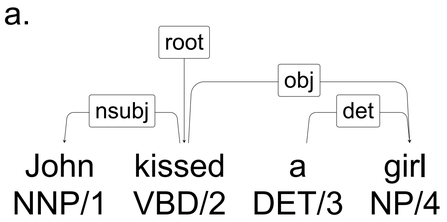
Semantic representations via linguistic rewriting

1. Normalizing linguistic annotations

- Linguistic annotations are normalized into graph structures (inspired by Ide and Bunt (2010))

(2) John kissed a girl.

a.

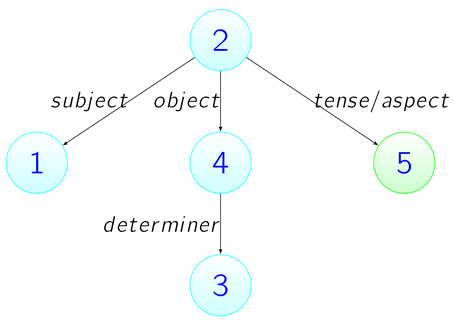


b.

$$2\begin{bmatrix} \text{PRED} & \text{`kiss}\langle \text{SUBJ,OBJ} \rangle\text{'} \\ \text{SUBJ} & 1\begin{bmatrix}\text{PRED} & \text{`John'}\end{bmatrix} \\ \text{OBJ} & 4\begin{bmatrix}\text{PRED} & \text{`girl'} \\ \text{DET} & 3\begin{bmatrix}\text{PRED} & \text{`a'}\end{bmatrix}\end{bmatrix} \\ \text{TNS-ASP} & 5\begin{bmatrix}\text{TENSE} & \text{past}\end{bmatrix}\end{bmatrix}$$

Figure 2: Enumerated syntax for: *John kissed a girl*

Relational information:



Node representation:

$$n\begin{bmatrix}\text{Attr1} & \text{Value1} \\ \text{Attr2} & \text{Value2} \\ \dots & \end{bmatrix}$$

Dependency example:

$$2\begin{bmatrix}\text{TOKEN} & \text{kissed} \\ \text{TAG} & \text{VBD} \\ \text{POS} & 2\end{bmatrix}$$

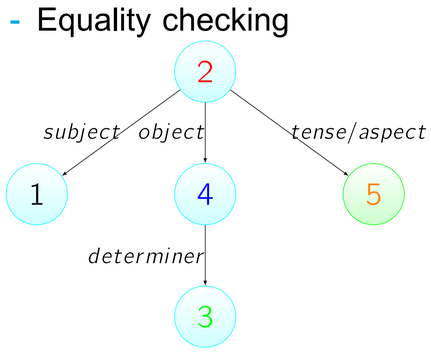
f-structure example:

$$2\begin{bmatrix}\text{PRED} & \text{`kiss<1,4>'}\end{bmatrix}$$

Figure 3: Abstract syntactic graph for *John kissed a girl.*

2. Designing rewrite rules

- Query language for Graph-matching
  - Variables over nodes (`#a,#b,#c,...`)
  - Variables over values (`%a,%b,%c,...`)
- LFG-style constraint checking (for directed graphs)
  - Functional application, functional uncertainty
- Dictionary specification and look-up
- Equality checking



$$5\begin{bmatrix}\text{TENSE} & \text{past}\end{bmatrix}$$

(3) a. `#g OBJ #h DET #i & #g TNS-ASP #h TENSE past` → ✗

b. `#g OBJ #h DET #i & #g TNS-ASP #j TENSE past` → ✓

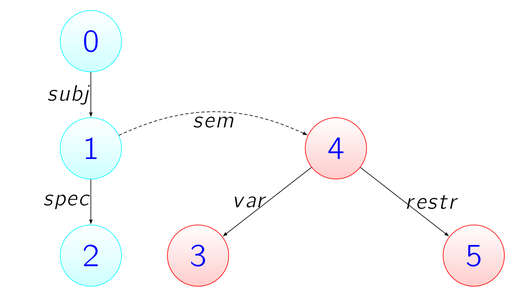
(4) a. `#g !(OBJ>DET) #h`

b. `#h ^(DET>OBJ) #g`

3. Formal semantics via rewriting

- Quantifier treatment according to Dalrymple et al. (1999)
- Syntax expanded with `SEM`antic structure (see (5))
- The glue representation is instantiated in rule (6)

$$\begin{bmatrix}\text{OBJ} & g\begin{bmatrix}\text{PRED} & \text{`girl'} \\ \text{SPEC} & \begin{bmatrix}\text{PRED} & \text{`a'}\end{bmatrix}\end{bmatrix}\end{bmatrix} \qquad g_\sigma\begin{bmatrix}\text{VAR} & v_1[] \\ \text{RESTR} & r_1[]\end{bmatrix}$$



(5) `#h ^(SPEC) #g ==> #g SEM #i VAR #j & #i RESTR #k.`

(6) a.
```
#g PRED `every' & #g ^(SPEC) #h SEM #i VAR #j &
  #i RESTR #k & #h ^(%%) #f ==>
  #i GLUE (#j -o #k) -o ((#i -o #f)-o #f)) : every
```
b. $(3 \multimap 5) \multimap ((4 \multimap 0) \multimap 0) : every$

- Description-by-analysis approach based on Andrews (2008)
- GSWB reads out `GLUE` values and calculates semantics

4. Demo

A demo illustrating the implementation of the system as a micro-service is available at:

`https://github.com/Mmaz1988/abstract-syntax-annotator-client`

It makes use of `cytoscape.js` to present the abstract syntactic graph in **cyan** and the added annotations in **red**. Furthermore, it presents the resulting semantic derivation, if available.

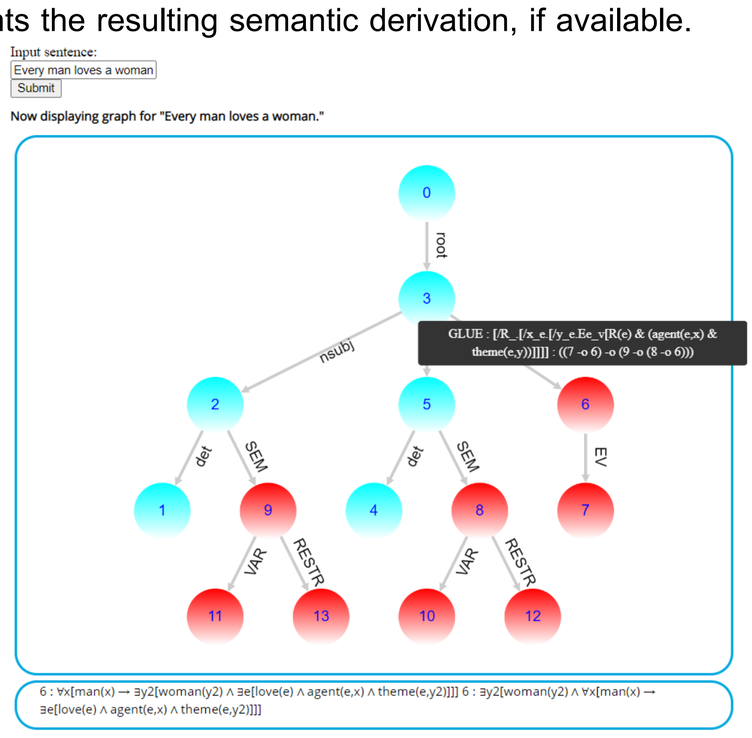


Figure 4: Semantic parsing visualized in the demo

Selected references: • Andrews, Avery D. 2008. The Role of PRED in LFG + Glue. In *Proceedings of the LFG08 Conference*, Pages 47–67. • Crouch, Richard. 2005. Packed Rewriting for Mapping Semantics to KR. In *Proceedings of the Sixth International Workshop on Computational Semantics (IWCS-6)*, Pages 103–114. Tilburg. • Dalrymple, Mary, John Lamping, Fernando Pereira, and Vijay Saraswat. 1999. Quantification, Anaphora, and Intensionality. In M. Dalrymple, editor, *Semantics and Syntax in Lexical Functional Grammar – The Resource Logic Approach*, Pages 39–89. • Ide, Nancy and Harry Bunt. 2010. Anatomy of Annotation Schemes: Mapping to GrAF. In *Proceedings of the Fourth Linguistic Annotation Workshop*, Pages 247–255. • Maxwell III, John T and Ronald M Kaplan. 1989. An Overview of Disjunctive Constraint Satisfaction. In *Proceedings of the First International Workshop on Parsing Technologies*, Pages 18–27. • 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.