Glue semantics for Universal Dependencies

Matthew Gotham
University of Oxford

Dag Trygve Truslew Haug
University of Oslo

Proceedings of the LFG'18 Conference

University of Vienna

Miriam Butt, Tracy Holloway King (Editors)

2018

CSLI Publications

pages 208–226

http://csli-publications.stanford.edu/LFG/2018

Keywords: universal dependencies, glue semantics

Gotham, Matthew, & Haug, Dag Trygve Truslew. 2018. Glue semantics for Universal Dependencies. In Butt, Miriam, & King, Tracy Holloway (Eds.), *Proceedings of the LFG'18 Conference, University of Vienna*, 208–226. Stanford, CA: CSLI Publications.

Abstract

Universal Dependencies (UD) is a very widely-used standard for cross-linguistic annotation of syntactic structure. There is, therefore, interest in deriving semantic representations from UD structures, ideally in a language-independent way. In this paper we report on an approach to deriving semantic representations from UD structures that relies on adapting and exploiting techniques from Glue semantics for LFG.

1 Introduction

In recent years, the Universal Dependencies initiative (de Marneffe et al., 2014) has established itself as something of a *de facto* annotation standard for cross-linguistic annotation of syntactic structure (treebank development) and subsequent statistical parsing with models trained on those treebanks. However, many downstream tasks require not dependency parses but rather more elaborate semantic structures that must be derived from those parses. The challenge in any attempt to derive such structures is is to do so while retaining the principal advantages of UD, which means relying as little as possible on language-specific, typically lexical, resources that are not available for many of the 60 languages for which there are UD treebanks.

In this paper we outline an approach to this problem that builds on techniques developed for LFG + Glue. There are several motivations for this. First, LFG's f-structures track the same aspect of syntactic structure as UD dependency trees. Second, the particular version of dependency grammar that UD embodies has inherited much from LFG via the Stanford Dependencies and the PARC dependencies. Third, unlike many other approaches, LFG + Glue does not assume a one-to-one mapping from syntactic to semantic structures, but instead develops a syntax-semantics interface that can map a single syntactic structure to several meaning representations (i.e. the syntax underspecifies the semantics). The latter point becomes especially important because UD—for all its LFG inheritance—is a compromise between theoretical concerns such as language-specific and typological adequacy on the one hand, and computational concerns such as efficient annotation and reliable statistical parsing on the other hand. A typical UD tree therefore contains much less information than the ideal syntactic representations assumed in theoretical formal semantics.

The paper is organized as follows. In Section 2 we describe some relevant properties of UD syntactic structures that illustrate the task we set ourselves. In Section 3 we describe the meaning representation language (version of typed lambda calculus) and meaning composition language (fragment of linear logic) that make up

[†]This work was conducted during the authors' fellowship at the Oslo Center for Advanced Study at the Norwegian Academy of Science and Letters. We gratefully acknowledge their support. We also thank the members of the CAS research group for valuable feedback, and Johan Bos for help with the Boxer software.

the Glue semantics aspect of our proposal. In Section 4 we describe specifically how we connect that Glue theory to UD trees of the form described in Section 2. In Section 5 we discuss the strong points and limitations of our proposal, and point the way to future work.

2 The challenge of UD syntax

In common with other dependency grammar formalisms, UD structures always form a rooted tree over the tokens of the sentence. This means that every node corresponds to an overt token of the sentence (the **overt token constraint**) and has exactly one mother (the **single head constraint**), unless it is the root, in which case it has no mother.[1]

Both the single-head constraint and the overt token constraint limit the expressivity of the syntactic formalism in a way that impairs meaning construction. We illustrate this point on the basis of three example sentences that we will keep returning to, (1)–(3). These are chosen to illustrate different challenges faced by our endeavour and exemplify, respectively, control, VP coordination and a bare relative clause.

(1) Abrams persuaded the dog to bark.

(2) He laughed and smiled.

(3) The dog they thought we admired barks.

Figure 1 shows the UD annotation of (1).[2] The single head constraint makes it impossible to express that *the dog* is simultaneously the object of *persuaded* and the subject of *to bark*, and the overt token constraint makes it impossible to insert any other subject for *bark*, e.g. a PRO that could be coindexed with *the dog*. Compare this with the richer f-structure formalism of LFG, where the relevant information could be captured either through structure sharing (functional control) or a coindexed PRO (obligatory anaphoric control).

For similar reasons, there is no way to indicate in the UD annotation of (2), shown in Figure 2, that *he* is the subject of *smiled*. Nor is there any way to indicate the position of the gap in the UD annotation of a relative clause structure without a relativizer (relative pronoun or complementizer), such as Figure 3, which is the annotation of (3).

[1] These constraints do not apply to *enhanced* UD (http://universaldependencies.org/u/overview/enhanced-syntax.html), which is also part of the Universal Dependencies initiative. However, we restrict ourselves to basic (non-enhanced) UD because (i) enhanced dependency annotations are only available for a very small proportion of the UD treebanks, and (ii) state of the art parsing speed and accuracy is significantly worse for enhanced UD than for basic UD. As such, as things currently stand enhanced UD lacks two of the major attractions of UD.

[2] In these annotations, the first line of text shows the tokens, the second line the lemmas, and the third line the parts of speech. Other information (for example, features) is included in full UD annotations.

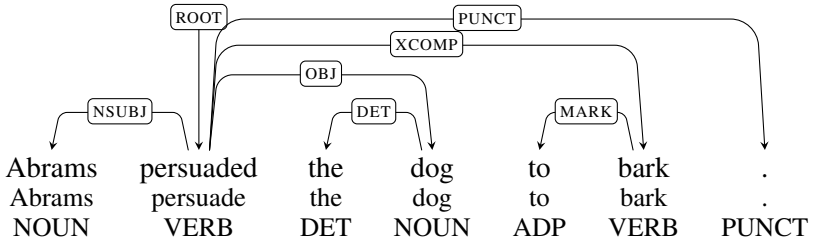

Figure 1: The UD annotation of (1)

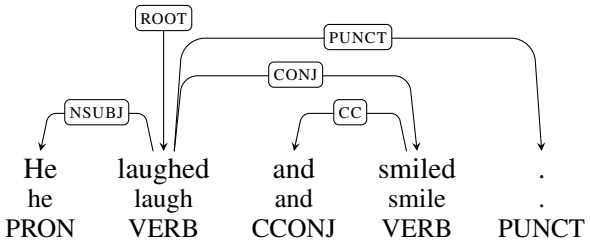

Figure 2: The UD annotation of (2)

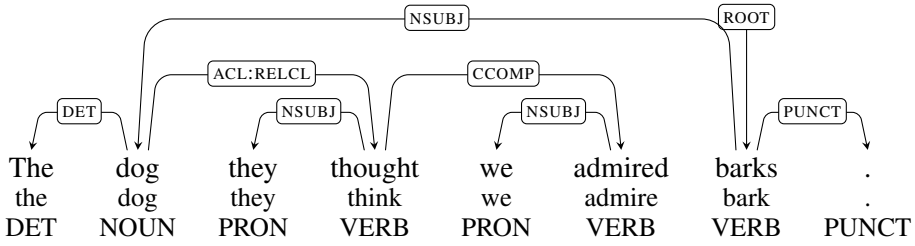

Figure 3: The UD annotation of (3)

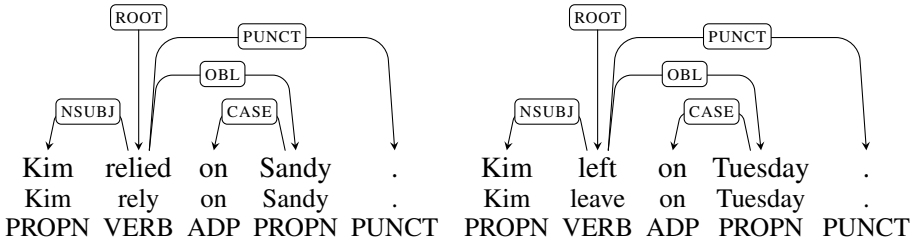

Figure 4: The UD annotations of (4)–(5)

We should also mention at this point that some UD design choices pose challenges from the perspective of semantics that are not shared with other dependency formalisms. For example, the UD annotation guidelines leave no room for an argument/adjunct distinction: both kinds of dependency can be annotated with the relation OBL(ique). An example is shown in Figure 4, which gives the UD annotations of (4)–(5). It can be seen that the two annotations have exactly the same edges and parts of speech, despite the fact that one '*on* phrase' is an argument of the main verb, while the other is an adjunct.

(4) Kim relied on Sandy.

(5) Kim left on Tuesday.

We will return to the discussion of (4)–(5) in Section 5.

3 Semantics

3.1 Meaning representation

Our target meaning representations are Discourse Representation Structures, the format of which is inspired by Boxer (Bos, 2008). The most obvious difference from the Boxer format is that we do not have separate DRSs for presupposed and asserted content. Instead, presupposed conditions are marked with the connective ∂, which is the propositional operator that maps TRUE to TRUE and maps FALSE or # (undefined) to # (we are working in a trivalent semantics). Presupposed discourse referents are given as arguments to the predicate ant; basically, this requires the discourse referent to have an antecedent. The predicates pron.he/they/we are sugaring for ant combined with the appropriate gender/number presuppositions.

With these considerations in mind, we can give the target DRSs for (1)–(3) in Figure 5. As in Boxer, we have three sorts for discourse referents: entities (x_n), eventualities (e_n) and propositions (p_n).

How are these DRSs put together compositionally? Conceptually, we are assuming an updated version of Partial Compositional Discourse Representation Theory (PCDRT, Haug 2014) in which, for example, the representation of (2) given in Figure 5 is an abbreviation of (6).

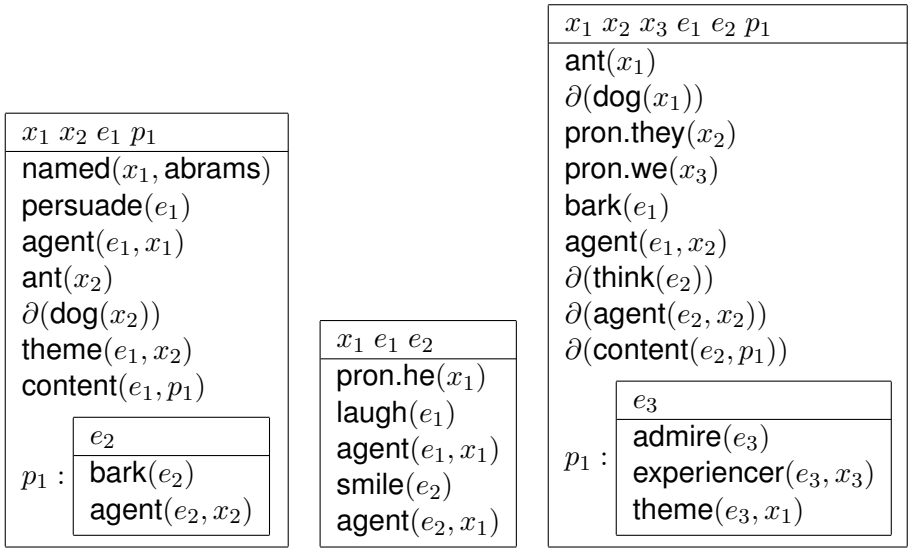

Figure 5: Target meaning representations for (1)–(3)

(6) $\lambda i.\lambda o.\partial(i[x_{i1}\ e_{i1}\ e_{i2}]o) \wedge \mathsf{ant}(o)(x_{i1}) \wedge \partial(\mathsf{male}(\nu(o)(x_{i1})))$
$\wedge\ \mathsf{laugh}(\nu(o)(e_{i1})) \wedge \mathsf{agent}(e_{i1}, x_{i1}) \wedge \mathsf{smile}(e_{i2}) \wedge \mathsf{agent}(e_{i2}, x_{i1})$

That is to say, (6) represents a relation between states i and o such that o extends i by a male individual (identical to one already defined in i) and two events, one of that individual laughing and one of him smiling. Unlike in other approaches, then, the lexical semantics of the word *he* introduces a new discourse referent, albeit one that must be identified with a contextually available discourse referent. This is shown in (7), which is an abbreviation of (8).

(7) $\lambda P.\ \boxed{\begin{array}{|c|}\hline x_1 \\ \hline \mathsf{pron.he}(x_1) \\ \hline \end{array}}\ ; P(x_1)$

(8) $\lambda P.\lambda i.\lambda o.\exists j.\partial(i[x_{i1}]j) \wedge \mathsf{ant}(j)(x_{i1}) \wedge \partial(\mathsf{male}(\nu(j)(x_{i1}))) \wedge P(j)(o)$

We assume PCDRT because:

1. It is defined in typed lambda calculus, and hence is straightforwardly compatible with Glue.

2. It has a treatment of unresolved anaphora, which is essential for an adequate meaning representation for many naturally-occurring examples such as are collected in treebanks.

3. It is representationally similar to standard DRT, allowing for comparison with computational linguistic resources prepared on the basis of DRT.

The assumption of PCDRT is certainly not crucial, however, any theory that meets conditions 1–3 would serve equally well. In our current implementation we use the beta reduction software for λ-DRT described by Blackburn & Bos (2006) and implemented in Boxer (Bos, 2008).

3.2 Meaning composition

On the meaning composition side, we assume a fragment of propositional linear logic that has $\multimap$ as the only connective and three undefined propositional function symbols: e, v and t—mnemonic for entities, eventualities (events and states) and truth values respectively. Following Andrews (2010), we express the fact that certain expressions can take scope at multiple locations by means of an inside-out functional uncertainty (over UD structures), and not in the linear logic fragment itself, which has no quantification. For example, the linear logic type of a quantifier is usually given in higher order glue as (9), where $\uparrow_\sigma$ is the semantic structure of the argument position in which the quantifier occurs, and H is any semantic structure, representing the fact that the quantifier can scope higher than the predicate of which it is an argument. By contrast, in a propositional glue setting, the quantification is replaced by a standard functional uncertainty as in (10).

(9) $\forall H.(\uparrow_\sigma \multimap H) \multimap H$

(10) $(\uparrow_e \multimap (\%H_t)) \multimap \%H_t, \text{GF*} \uparrow = \%H$

Using propositional glue makes it easier to exploit existing tools for linear logic.

In our lexicon, we will assign interpretations (and accompanying linear logic formulae) both to nodes and to edges of UD structures. The up and down arrows then should be read as shown in (11).[3]

(11)

	$\downarrow$	$\uparrow$
node	this node	this node's mother
edge	this edge's target	this edge's source

In our descriptions of linear logic formulae used in lexical entries we also make use of the Kleene star * and local names. One such local name, %R, is special in that it always picks out the root node in a dependency structure—this is how we replicate the treatment of proper names given by Kamp & Reyle (1993) insofar as they always take widest scope and hence remain as accessible antecedents for pronouns.

3.3 The form of lexical knowledge assumed

As alluded to above, we are attempting to give our semantics for UD in such a way as to postpone as much as possible the need for language-specific lexical knowl-

[3] In the actual UD encoding, edges are treated as features of their target nodes, and so the bifurcation given in (11) is strictly speaking unnecessary.

edge. This means that our lexical entries are underspecified in various respects. The principle can best be illustrated by way of an example such as (12),[4] which is what 'admired' retrieves from the lexicon when instantiated as in Figure 3 (there will be more on how this works in the Section 4).

(12) $\lambda x.\lambda F.\left[\begin{array}{|l|}\hline e_1 \\ \hline \mathsf{admire}(e_1) \\ \mathsf{nsubj}(e_1, x) \\ \hline\end{array}\right]; F(e_1) : e_{\downarrow \text{NSUBJ}} \multimap (v_\downarrow \multimap t_\downarrow) \multimap t_\downarrow$

The F argument leaves the event variable open for further modification; we will explain how existential closure happens in Section 4. There are two main points to note about how (12) is underspecified. Firstly, we are not assuming that we have subcategorization information available for individual verbs. Consequently, (12) involves abstraction over one argument of type e just because the node in Figure 3 has one dependent; we don't have the information that there's a 'missing' object in that structure. Secondly, we are not assuming that we have thematic information available either. The thematic relation name nsubj shown in the DRS is lifted from the label of the arc going from *admired* to *we*. In the same way, the name of the event predicate admire is taken from the lemma of the linguistic token. For many purposes these underspecified representations will have to be more fully fleshed out; we will discuss how we anticipate this working in Section 5.

On the other hand, for our approach to produce anything usable we are going to have to assume some lexical knowledge, specifically that associated with 'logic words' (determiners and conjunctions). For example, the treatment of conjunction as exemplified in the case of (2) depends on each of the meaning constructors shown in (13)–(15) below. The meaning constructor in (13) is triggered by a CONJ edge with a verb target node, while (14) is triggered by a CC edge with a source node the mother of which is a verb—hence, in principle, these are independent of specific lexical knowledge. However, the whole analysis still depends on the meaning constructor given in (15), which is triggered by the lemma *and*.[5] The precise nature of how this triggering works will be described in Section 4.

(13) $\lambda P.\lambda S.\lambda C.\lambda E.C(P(E))(S(C)(E)) : ((v_\downarrow \multimap t_\downarrow) \multimap t_\downarrow) \multimap$
$((t_\uparrow \multimap t_\uparrow \multimap t_\uparrow) \multimap ((v_\uparrow \multimap t_\uparrow) \multimap t_\uparrow)) \multimap$
$(t_\uparrow \multimap t_\uparrow \multimap t_\uparrow) \multimap (v_\uparrow \multimap t_\uparrow) \multimap t_\uparrow$

(14) $\lambda P.\lambda \_.P :$
$((v_{\uparrow\uparrow} \multimap t_{\uparrow\uparrow}) \multimap t_{\uparrow\uparrow}) \multimap (t_{\uparrow\uparrow} \multimap t_{\uparrow\uparrow} \multimap t_{\uparrow\uparrow}) \multimap (v_{\uparrow\uparrow} \multimap t_{\uparrow\uparrow}) \multimap t_{\uparrow\uparrow}$

(15) $\lambda p.\lambda q.p; q : t_{\uparrow\uparrow} \multimap t_{\uparrow\uparrow} \multimap t_{\uparrow\uparrow}$

[4]To save space and improve readability, we write arguments to the propositional functions as subscripts rather than in brackets, e.g. we write '$t_\downarrow$' rather than '$t(\downarrow)$'.

[5]We are directly applying the analysis of coordination given by Asudeh & Crouch (2002), with CC fulfilling the role of the 'seed' conjunct and CONJ fulfilling the role of the non-'seed' conjunct(s).

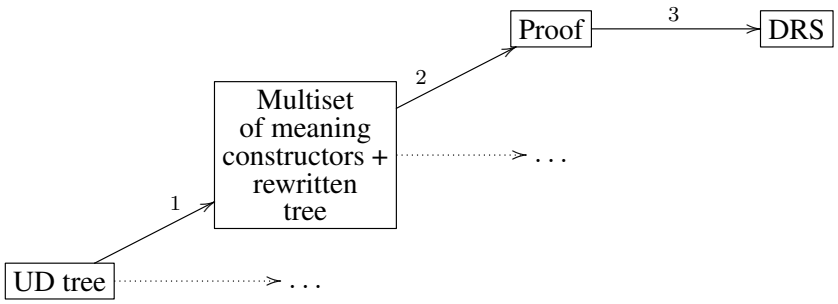

Figure 6: The pipeline

4 Our pipeline

4.1 Overview

The overall architecture of our system is shown in Figure 6. As can be seen, we proceed in three steps. In the first step, the UD tree is simultaneously being (possibly) enriched and rewritten as a multiset of glue-type meaning constructors in a non-deterministic manner. This yields a set of pairs $\langle T, M \rangle$ where T is a (possibly) enriched tree. One should not read too much into this tree enrichment, however: As we will see, the tree delivers the correct types for the meaning constructors and will in many cases show perspicuously which reading the meaning constructors capture, but it does not otherwise play a role in the further processing. The semantic derivation proceeds as in standard glue semantics, by combining the meaning constructors in one or more linear logic proofs (step 2) and then getting a meaning term (in our case, a DRS) via the Curry-Howard isomorphism (step 3). As in standard glue, step 2 is relational (i.e. there can be several different proofs from a single set of premises) but step 3 is functional (i.e. each proof corresponds to a single meaning). Steps 2 and 3 are implemented via Miltiadis Kokkonidis' Instant Glue linear logic prover[6] and Johan Bos's Boxer (Bos, 2008).

The basic idea behind our approach is to traverse the UD tree depth-first and create meaning constructors for each node. As meaning constructors are created at each visited node, the UD tree may be extended non-deterministically; and each of the extended trees are fed to the algorithm. That is, the function f that creates the meaning constructors is of type

(16) $\quad f :: \langle M, T* \rangle \rightarrow [\langle M, T* \rangle]$

where M is a multiset of meaning constructors and $T*$ is a UD tree with a pointer to the current node. The function is recursively applied using the bind operator of the List monad (Haskell's $>>=$).

The output of f is governed by a set of hand-written rules for creating meaning constructors. Each rule has two parts, a criterion, i.e. a tree description that must

[6] http://users.ox.ac.uk/~cpgl0036/prover/glue_prover.pl

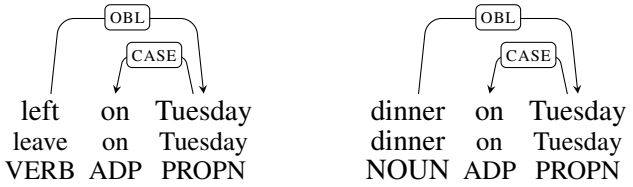

Figure 7: PP attachment to nouns and verbs

evaluate to true at the current node for the rule to apply, and a meaning constructor that will be created if the rule applies. Several meaning constructors can be created from a single node if it maches several criteria. There is a simple control structure in the rules file: rules are matched in the order that they are listed, and stop rules with an empty meaning constructor part will stop the algorithm from searching for more matching rules.

This control structure can be used to encode defaults. For example, in the UD annotation, prepositions are CASE dependents of what would be their complements in a phrase structure analysis. For example, as shown in Figure 7, *on* is a CASE dependent of *Tuesday*, rather than the head of a PP with *Tuesday* as its complement. Semantically, they denote a relation between their mother and their grandmother nodes. While the mother is always of type e, the grandmother can be of either type v or type e, so the type of the preposition is either $\langle e, \langle v, t\rangle\rangle$ or $\langle e, \langle e, t\rangle\rangle$. The following three rules assign the first type if the grandmother is a verb, otherwise the second type.

(17)
```
relation = case; ↑↑ pos = VERB ->
```
$\lambda y.\lambda x.$ [:LEMMA:(y, x)] : $e_{\uparrow} \multimap v_{\uparrow\uparrow} \multimap t_{\downarrow}$
```
relation = case; ↑↑ pos = VERB ->
relation = case ->
```
$\lambda y.\lambda y.$ [:LEMMA:(y, x)] : $e_{\uparrow} \multimap e_{\uparrow\uparrow} \multimap t_{\downarrow}$

The first rule matches any node whose relation is CASE and whose grandmother is a verb and assigns the appropriate semantics and type ($\langle e, \langle v, t\rangle\rangle$). The second rule stops further generation of meaning constructors from such nodes, and the final rule then assigns the default type $\langle e, \langle e, t\rangle\rangle$ to any (other) nodes bearing the CASE relation.[7]

Another use of the stop rules is to avoid giving any semantics at all for certain items. For example, there is a stop rule for elements bearing the `PType=Rel`

[7]This was a more or less random choice for our implementation. In a production system, the choice of default could of course affect performance.

feature, which applies *before* the rule that assigns semantics to pronouns, thereby ensuring the relative pronouns are treated as gaps with no semantics.

In the next step, the meaning constructors that come from the rules are *instantiated*. This means that we substitute actual node indices for the $\uparrow$ and $\downarrow$ metavariables. Furthermore, the function :LEMMA: extracts the lemma from the current node. For example, if we assume that the nodes in the left example in Figure 7 have indices 1, 2, 3, the meaning constructor produced by the first rule in (17) will be instantiated as in (18).

(18) $\lambda y.\lambda x.\left[\begin{array}{|c|}\hline \\ \hline \mathsf{on}(y,x) \\ \hline\end{array}\right] : e_2 \multimap v_1 \multimap t_3$

Instantiation can be more complex when we are dealing with verbal nodes. The verb rule looks like (19).

(19) `pos = VERB ->`
$\lambda F.\left[\begin{array}{|c|}\hline e \\ \hline \mathsf{:LEMMA:}(e) \\ \hline\end{array}\right]$; `:DEP:`(e); $F(e) : (v_\downarrow \multimap t_\downarrow) \multimap t_\downarrow$

As mentioned in section 3.3, we do not assume that we have an external valency lexicon available. Instead, we construct the valency from the syntactic tree. The function :DEP: extracts the appropriate semantics from the dependents of the verb according to a separate rule file. For example, if there is an NSUBJ and an OBJ dependent, the rule in (19) will instantiate as in (20), assuming that the indices of the subject, verb and object are 1, 2 and 3 respectively and the verb has the lemma *kiss*.

(20) $\lambda x.\lambda y.\lambda F.\left[\begin{array}{|c|}\hline e \\ \hline \mathsf{kiss}(e) \\ \hline\end{array}\right]$; $\left[\begin{array}{|c|}\hline \\ \hline \mathsf{nsubj}(e,x) \\ \mathsf{obj}(e,y) \\ \hline\end{array}\right]$; $F(e) :$
$e_1 \multimap e_3 \multimap (v_2 \multimap t_2) \multimap t_2$

The F argument here serves as a "handle" for further modification of the event, without making it possible for such modifiers to scope under the event variable. We refer to Champollion (2015) for more details. The semantic composition then ends by saturating F with a property of all events, rather than existential closure of the event variable itself. The relevant meaning constructor is triggered by the ROOT relation and is as in (21), where $\downarrow$ will be instantiated to the index of the root verb, in our case 2.

(21) $\lambda\_.\left[\begin{array}{|c|}\hline \\ \hline \\ \hline\end{array}\right] : v_\downarrow \multimap t_\downarrow$

After the meaning constructors have been instantiated, composition can proceed as in ordinary Glue Semantics. We show this in more detail in the next section.

4.2 Worked examples

Let us now see in more detail how we derive the meaning for (1), which has the UD tree in Figure 1.

The first, and most interesting, step is the creation of a meaning for the root node *persuaded*. The relevant rule was given in (19). Instantiation of :DEP: gives (22).

(22) $\lambda P.\lambda y.\lambda x.\lambda F.$
$\left[\begin{array}{|l|}\hline e_1\ x_1\ p_1 \\ \hline \textsf{persuade}(e_1) \\ \textsf{controldep}(e_1, x_2) \\ \textsf{xcomp}(e_1, p_1) \\ \textsf{obj}(e_1, y) \\ \textsf{nsubj}(e_1, x) \\ p_1 : P(x_1)(\lambda\_.[\ |\ ]) \\ \hline \end{array}\right]$
$; F(e_1) :$
$(e_{\downarrow \textsc{xcomp nsubj}} \multimap (v_{\downarrow \textsc{xcomp}} \multimap t_{\downarrow \textsc{xcomp}}) \multimap t_{\downarrow \textsc{xcomp}}) \multimap (e_{\downarrow \textsc{nsubj}}) \multimap (e_{\downarrow \textsc{obj}}) \multimap (v_{\downarrow} \multimap t_{\downarrow}) \multimap t_{\downarrow}$

Several noteworthy things happen in (22). First, although we assume no lexical knowledge at this stage in the derivation, we know that *persuade* is a control verb, since it has an XCOMP dependent. But we do not know whether it is a subject or object control verb. Instead we introduce an individual x_1 which bears the relation CONTROLDEP to the matrix event, and is also fed to the embedded controlled predicate as its subject. Notice that CONTROLDEP is a purely semantic relation which does not correspond to anything in the syntactic tree. Furthermore, we introduce a propositional discourse referent p_1 for the proposition we get from feeding the downstairs verb with that subject and closing off the composition (in the way the ROOT relation would do for the matrix verb). This discourse referent bears the XCOMP relation to the matrix event.

Next, the meaning constructor in (22) must be instantiated. If we assume that the nodes in Figure 1 are indexed consecutively from 1, then $\downarrow$ instantiates to 2, $\downarrow$ NSUBJ to 1, $\downarrow$ OBJ to 4 and $\downarrow$ XCOMP to 6. But the linear logic type side of (22) also references the node $\downarrow$ XCOMP NSUBJ, which does not exist. When this happens, the tree is enriched with such a node, yielding the tree in Figure 8. Given this tree, we can instantiate (22) as (23), where 8 is the index of the newly created node.

(23) **persuade:** $\lambda P.\lambda y.\lambda x.\lambda F.$
$\left[\begin{array}{|l|}\hline e_1\ x_1\ p_1 \\ \hline \textsf{persuade}(e_1) \\ \textsf{controldep}(e_1, x_2) \\ \textsf{xcomp}(e_1, p_1) \\ \textsf{obj}(e_1, y) \\ \textsf{nsubj}(e_1, x) \\ p_1 : P(x_1)(\lambda\_.[\ |\ ]) \\ \hline \end{array}\right]$
$; F(e_1) :$
$(e_8 \multimap (v_6 \multimap t_6) \multimap t_6) \multimap e_4 \multimap e_1 \multimap (v_2 \multimap t_2) \multimap t_2$

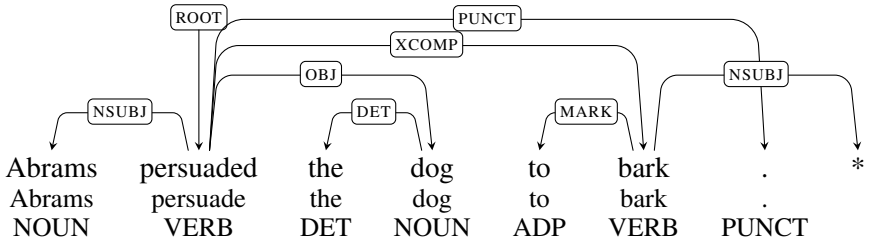

Figure 8: The enriched UD annotation of (1)

The creation and instantiation of meaning constructors for *Abrams*, *the* and *dog* is relatively trivial, keeping in mind that we do allow for use of lexical information about "logic words" such as *the*. The result is shown in (24).

(24) a. **Abrams:** $\lambda P.\boxed{\begin{array}{|l}\hline x_1 \\ \hline \mathsf{named}(x_1, \mathsf{abrams}) \\ \hline\end{array}}; P(x_1) : (e_1 \multimap t_2) \multimap t_2$

b. **the:** $\lambda P.\lambda Q.\boxed{\begin{array}{|l}\hline x_1\ p_1 \\ \hline \mathsf{ant}(x_1) \\ \partial(p_1) \\ p_1 : P(x_1) \\ \hline\end{array}}; Q(x_1) : (e_4 \multimap t_4) \multimap (e_4 \multimap t_2) \multimap t_2$

c. **dog:** $\lambda x.\boxed{\begin{array}{|l}\hline \\ \hline \mathsf{dog}(x_1) \\ \hline\end{array}} : e_4 \multimap t_4$

The meaning for the definite article requires some comment. It introduces a dref x_1. P is the restrictor argument, which in the case of a definite description is presupposed. To capture this, we introduce a propositional dref p_1 for the proposition $P(x_1)$ and put it in the scope of ∂.

The interesting part comes when we reach *bark*. The uninstantiated meaning constructor will be as in (19). And the NSUBJ dependent that was created during the processing of *persuaded* will now make sure that :DEP: triggers a dependency on the subject so that we get (25).

(25) **bark:** $\lambda x.\lambda F.\boxed{\begin{array}{|l}\hline e \\ \hline \mathsf{bark}(e) \\ \hline\end{array}}; \boxed{\begin{array}{|l}\hline \\ \hline \mathsf{nsubj}(e, x) \\ \hline\end{array}} : e_8 \multimap (v_6 \multimap t_6) \multimap t_6)$

Once we have the meaning constructors in (23), (24) and (25), we can assemble them in an ordinary glue proof, as shown in Figure 9. The lambda term corresponding to t_2 in that proof beta reduces to (26).

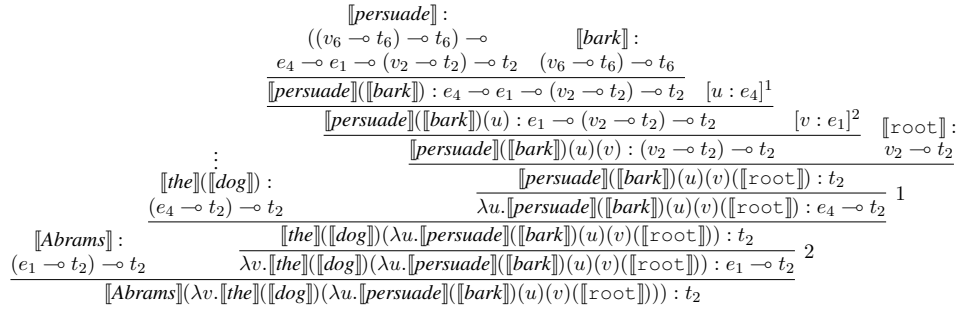

Figure 9: Linear logic proof for (1)

(26)

x_1 x_2 x_3 e_1 p_1
named(x_1, abrams), ant(x_2)
∂(dog(x_2)), persuade(e_1)
nsubj(e_1, x_1), obj(e_1, x_2)
controldep(e_1, x_3), xcomp(e_1, p_1)
p_1 : e_2 | bark(e_2), nsubj(e_2, x_3)

This, then, is the output we derive for Figure 9. In section 5 we will compare it to the target representation that we showed in Figure 5.

To illustrate another feature of our algorithm, we will now see how we derive the correct meaning constructor for bare relative clauses, as in (3), with the UD tree in Figure 3. The challenge in this case is that there is no indication of where the gap in the relative clause is. On the other hand, we do know (by virtue of the ACL:RELCL relation) that there is a gap somewhere. Drawing on the LFG apparatus, we model this using a functional uncertainty. The uninstantiated meaning constructor triggered by ACL:RELCL is as in (27).

(27) $\lambda P.\lambda V.\lambda x.P(x); V(x)(\lambda\_.[\,|\,]) :$
$(e_{\uparrow} \multimap t_{\uparrow}) \multimap (e_{\downarrow \text{DEP}^* \text{ DEP}\{PType=Rel\}} \multimap (v_{\downarrow} \multimap t_{\downarrow}) \multimap t_{\downarrow}) \multimap e_{\uparrow} \multimap t_{\uparrow}$

We see that this takes a predicate and a clause with a gap and turns it into a new predicate – in other words, it turns the clause with the gap into a modifier of predicates, as we expect for a relative clause. The type of the clause with the gap is $(e_{\downarrow \text{DEP}^* \text{ DEP}\{PType=Rel\}} \multimap (v_{\downarrow} \multimap t_{\downarrow}) \multimap t_{\downarrow})$, where $(v_{\downarrow} \multimap t_{\downarrow}) \multimap t_{\downarrow})$ is the by now familiar type of verbs. The gap itself is $e_{\downarrow \text{DEP}^* \text{ DEP}\{PType=Rel\}}$, that is, a type e resource associated with some downstairs node (at any level of embedding, hence the Kleene star) bearing the feature PType=Rel. If there is none – as in our case – that node will be created. Let us assume the new node bears the index 9. Then the instantiation of (27) will be as in (28).

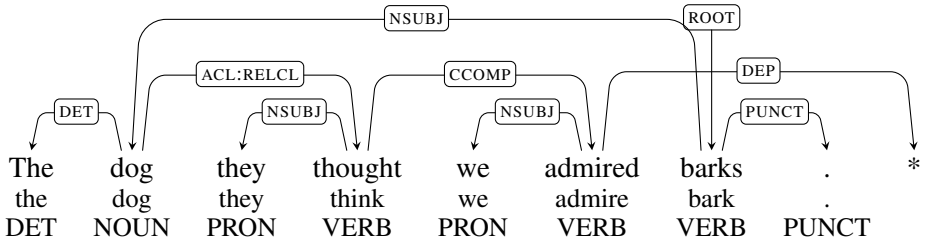

Figure 10: One enrichment of the UD annotation of (3)

(28) $(e_2 \multimap t_2) \multimap (e_9 \multimap (v_4 \multimap t_4) \multimap t_4) \multimap e_2 \multimap t_2$

However, there are many possibilities for situating the new node 9 in the tree, since all we know is that a path of DEP relations leads down to it from the verb of the relative clause. In Figure 10 we show the correct enrichment, with the new node attached under *admired*, but our system generates all four possible attachments (i.e. under *they, thought* and *we* as well as *admired*). Of these, the two readings that attach the gap to a pronoun will fail to produce a correct proof, since there is no interpretation of a gap under a pronoun. But both the attachment under *thought* and *admired* will produce possible meanings as shown in (29).

(29)

$x_1\ x_2\ x_3\ e_1\ e_2\ p_1$
$\mathsf{ant}(x_1)$
$\partial(\mathsf{dog}(x_1))$
$\mathsf{pron.they}(x_2)$
$\mathsf{pron.we}(x_3)$
$\mathsf{bark}(e_1)$
$\mathsf{nsubj}(e_1, x_2)$
$\partial(\mathsf{think}(e_2))$
$\partial(\mathsf{nsubj}(e_2, x_2))$
$\partial(\mathsf{ccomp}(e_2, p_1))$
p_1 : e_3 | $\mathsf{admire}(e_3)$, $\mathsf{nsubj}(e_3, x_3)$, $\mathsf{dep}(e_3, x_1)$

$x_1\ x_2\ x_3\ e_1\ e_2\ p_1$
$\mathsf{ant}(x_1)$
$\partial(\mathsf{dog}(x_1))$
$\mathsf{pron.they}(x_2)$
$\mathsf{pron.we}(x_3)$
$\mathsf{bark}(e_1)$
$\mathsf{nsubj}(e_1, x_2)$
$\partial(\mathsf{think}(e_2))$
$\partial(\mathsf{nsubj}(e_2, x_2))$
$\partial(\mathsf{ccomp}(e_2, p_1))$
$\partial(\mathsf{dep}(e_2, x_1))$
p_1 : e_3 | $\mathsf{admire}(e_3)$, $\mathsf{nsubj}(e_3, x_3)$

The choice between these two readings can in fact only be made once we use valency information to discard the reading where *thought* takes a subject, a complement clause and a third nominal argument. We return to this point in the next section.

Finally, let us have a look at the coordination example (2), with the UD annotation in Figure (2). The uninstantiated meaning constructors were shown in (13) for the CONJ relation, so ↓= 4 and ↑= 2, (14) for the CC relation (↑= 4, ↓= 3) and

(15) for the lemma *and* (↑= 4). Using these instantiations, we get the meaning constructors in (30)–(32).

(30) **conj:** $\lambda P.\lambda S.\lambda C.\lambda E.C(P(E))(S(C)(E)) : ((v_4 \multimap t_4) \multimap t_4) \multimap$
$((t_2 \multimap t_2 \multimap t_2) \multimap ((v_2 \multimap t_2) \multimap t_2)) \multimap$
$(t_2 \multimap t_2 \multimap t_2) \multimap (v_2 \multimap t_2) \multimap t_2$

(31) **cc:** $\lambda P.\lambda\_.P :$
$((v_2 \multimap t_2) \multimap t_2) \multimap (t_2 \multimap t_2 \multimap t_2) \multimap (v_2 \multimap t_2) \multimap t_2$

(32) **and:** $\lambda p.\lambda q.p\,;q : t_2 \multimap t_2 \multimap t_2$

In addition we will have the meaning constructors for **he-laughed** and **smiled** in (33) and (34).

(33) **he-laughed:** $\lambda F.$ $\begin{array}{|l} \hline e_1\ x_1 \\ \hline \mathsf{laugh}(e_1) \\ \mathsf{nsubj}(e_1, x_1) \\ \hline \end{array}$ $; F(e_1) : (v_2 \multimap t_2) \multimap t_2$

(34) **smiled:** $\lambda F.$ $\begin{array}{|l} \hline e_1 \\ \hline \mathsf{smile}(e_1) \\ \hline \end{array}$ $; F(e_1) : (v_4 \multimap t_4) \multimap t_4$

We see that **cc** can take **he-laughed** as its first argument, and **conj** can take **smiled**. Next, **cc**(**he-laughed**) fits as the argument of **conj**(**smiled**). Finally, we can apply the result to **and**, which results in the merger of the two DRSs in (33) and (34), which gives the end result in (35).

(35)
$\begin{array}{|l} \hline x_1\ e_1\ e_2 \\ \hline \mathsf{pron.he}(x_1) \\ \mathsf{laugh}(e_1) \\ \mathsf{nsubj}(e_1, x_1) \\ \mathsf{smile}(e_2) \\ \hline \end{array}$

In effect, what has happened is that, since the UD annotation does not distinguish between VP and sentence coordination, we are forced to treat everything as sentence coordination. Our representation does not therefore capture the fact that *he* is the subject of both verbs.

5 Discussion

Let us take stock at this stage and compare the target meaning representations shown in Figure 5 with what our system gets us so far, shown in (26), (29) and (35). While we anticipate that these underspecified representations will be adequate for many purposes, they are of course lacking plenty of information present in Figure 5. We expect that much of this information can be recovered with some addition of language-specific lexical information at this late stage.

First of all, let us take the θ-role names. As alluded to above, these have simply been lifted from the respective UD edge labels and as such are uninformative. With the aid of meaning postulates such as those shown in (36), however, more informative thematic relations can be inferred.

(36)
$$\forall e \forall x((\textsf{persuade}(e) \wedge \textsf{nsubj}(e,x)) \rightarrow \textsf{agent}(e,x))$$
$$\forall e \forall x((\textsf{persuade}(e) \wedge \textsf{obj}(e,x)) \rightarrow \textsf{theme}(e,x))$$
$$\forall e \forall p((\textsf{persuade}(e) \wedge \textsf{xcomp}(e,p)) \rightarrow \textsf{content}(e,p))$$
$$\forall e \forall x((\textsf{bark}(e) \wedge \textsf{nsubj}(e,x)) \rightarrow \textsf{agent}(e,x))$$
$$\forall e \forall x((\textsf{laugh}(e) \wedge \textsf{nsubj}(e,x)) \rightarrow \textsf{agent}(e,x))$$
$$\forall e \forall x((\textsf{think}(e) \wedge \textsf{nsubj}(e,x)) \rightarrow \textsf{agent}(e,x))$$
$$\forall e \forall p((\textsf{think}(e) \wedge \textsf{ccomp}(e,p)) \rightarrow \textsf{content}(e,p))$$
$$\forall e \forall x((\textsf{admire}(e) \wedge \textsf{nsubj}(e,x)) \rightarrow \textsf{experiencer}(e,x))$$

Next, let us look at the control example. As mentioned above, what has happened in (26) is that the meaning constructor triggered by the token of *persuade* accompanied by an XCOMP dependent has introduced an xcomp relation between the persuading event e_1 and the proposition p_1 that there is a barking event e_2, and introduced an individual x_3 as the nsubj of e_2 and the controldep of e_1. To go further, we need lexical knowledge, specifically the knowledge that *persuade* is an object control verb. That knowledge can be encoded in the meaning postulate shown in (37).

(37)
$$\forall e \forall x((\textsf{persuade}(e) \wedge \textsf{controldep}(e,x)) \rightarrow \textsf{obj}(e,x))$$

The DRS shown in (38) then follow logically from (26) and the meaning postulates given in (36)–(37). If we further assume thematic uniqueness, then we can infer that $x_2 = x_3$ in this case and hence derive a representation equivalent to (1) in Figure 5.

(38)

$x_1\ x_2\ x_3\ e_1\ p_1$
$\textsf{named}(x_1, \textsf{abrams})$, $\textsf{ant}(x_2)$
$\partial(\textsf{dog}(x_2))$, $\textsf{persuade}(e_1)$
$\textsf{agent}(e_1, x_1)$, $\textsf{theme}(e_1, x_2)$
$\textsf{theme}(e_1, x_3)$, $\textsf{content}(e_1, p_1)$
p_1 : $\begin{array}{\|l\|}\hline e_2 \\ \hline \textsf{bark}(e_2) \\ \textsf{agent}(e_2, x_3) \\ \hline \end{array}$

As for the bare relative clause example, if lexical information is to help in selecting the right interpretation of the two shown in (29), and then enriching dep to theme (possibly via obj), the way in which it does so will have to be somewhat less direct than simple entailments on the basis of meaning postulates. We can write one to the effect that every event of admiring has a theme (for example), but that won't in and of itself guarantee that x_1 is that theme, even if we know that it is some

dependent. A different kind of lexical information and/or reasoning process will be needed.

The situation with (2) is similar. We can write a meaning postulate to the effect that every smiling event has an agent, but to get from there to the inference that x_1 is that agent requires a bit more work. The different meanings of *on* are also a difficult case: while *rely* in (4) clearly subcategorizes for *on*, which does not contribute any meaning, it can be hard to reliably guess exactly what meaning of *on* is present in non-subcategorized examples, although the presence of the complement *Tuesday* is a robust cue for a temporal meaning. In all these cases, it might turn out useful to use default reasoning captured in a non-monotonic logic.

Ours is not the first work on semantics for UD; in particular, Reddy (2017) presents a much more developed system. The choice of Glue semantics has certain advantages and disadvantages in comparison with that system: on the plus side, with Glue there is no need for the UD trees to be binarized to get composition off the ground, and we get a flexible approach to scope taking yielding different readings that aren't derivable in a more rigid approach. However, that flexibility comes at a cost: practically it quicky becomes costly to compute lots of uninteresting scope differences, and in terms of design it can be hard to exclude non-existant readings.

In summary, the work described in this paper constitutes a proof of concept tested on carefully crafted examples, where we have applied LFG techniques (functional uncertainties) to enrich underspecified UD syntax, and applied Glue semantics to dependency structures. We have achieved some encouraging results, however we are very far from something practically useful: while we have basic coverage of the UD relations (though not yet VOCATIVE, DISLOCATED, CLF, LIST, PARATAXIS, ORPHAN), there has not yet been much work on interactions, special constructions or real data noise. These, and the limitations we have identified, provide plenty of opportunity for further work in this framework.

References

Andrews, Avery D. 2010. Propositional glue and the projection architecture of LFG. *Linguistics and Philosophy* 33(3). 141–170.

Asudeh, Ash & Richard Crouch. 2002. Coordination and parallelism in Glue semantics. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG02 conference*, 19–39. Stanford, CA: CSLI Publications.

Blackburn, Patrick & Johan Bos. 2006. Working with Discourse Representation Theory. Unpublished book draft. http://www.let.rug.nl/bos/comsem/book2.html.

Bos, Johan. 2008. Wide-coverage semantic analysis with Boxer. In *Proceedings of the 2008 conference on semantics in text processing* STEP '08, 277–286.

Stroudsburg, PA, USA: Association for Computational Linguistics. http://dl.acm.org/citation.cfm?id=1626481.1626503.

Champollion, Lucas. 2015. The interaction of compositional semantics and event semantics. *Linguistics and Philosophy* 38(1). 31–66. doi:10.1007/s10988-014-9162-8. https://doi.org/10.1007/s10988-014-9162-8.

Haug, Dag Trygve Truslew. 2014. Partial dynamic semantics for anaphora. *Journal of Semantics* 31. 457–511.

Kamp, Hans & Uwe Reyle. 1993. *From discourse to logic* (Studies in Linguistics and Philosophy 42). Dordrecht: Kluwer.

de Marneffe, Marie-Catherine, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre & Christopher D. Manning. 2014. Universal Stanford dependencies: A cross-linguistic typology. In *LREC*, Reykjavik, Iceland: ELRA.

Reddy, Siva. 2017. *Syntax-mediated semantic parsing*: University of Edinburgh dissertation.