Universal Semantic Parsing: Supplementary Material

Siva Reddy[†] **Oscar Täckström**[‡] **Slav Petrov**[‡] **Mark Steedman**[††] **Mirella Lapata**[††]
[†]Stanford University
[‡] Google Inc.
[††]University of Edinburgh
sivar@stanford.edu, {oscart, slav}@google.com, {steedman, mlap}@inf.ed.ac.uk

Abstract

This supplementary material to the main paper, provides an outline of how quantification can be incorporated in the UDEPLAMBDA framework.

1 Universal Quantification

Consider the sentence *Everybody wants to buy a house*,[1] whose dependency tree in the Universal Dependencies (UD) formalism is shown in Figure 1(a). This sentence has two possible readings: either (1) every person wants to buy a different house; or (2) every person wants to buy the same house. The two interpretations correspond to the following logical forms:

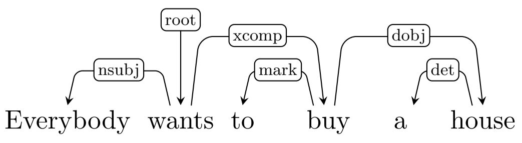
(1) $\forall x.\, \text{person}(x_a) \rightarrow$
$[\exists zyw.\, \text{wants}(z_e) \wedge \text{arg}_1(z_e, x_a) \wedge \text{buy}(y_e) \wedge \text{xcomp}(z_e, y_e) \wedge$
$\text{house}(w_a) \wedge\ \text{arg}_1(z_e, x_a) \wedge \text{arg}_2(z_e, w_a)]\ ;$

(2) $\exists w.\, \text{house}(w_a) \wedge (\forall x.\, \text{person}(x_a) \rightarrow$
$[\exists zy.\, \text{wants}(z_e) \wedge \text{arg}_1(z_e, x_a) \wedge \text{buy}(y_e) \wedge \text{xcomp}(z_e, y_e) \wedge$
$\text{arg}_1(z_e, x_a) \wedge \text{arg}_2(z_e, w_a)])\ .$

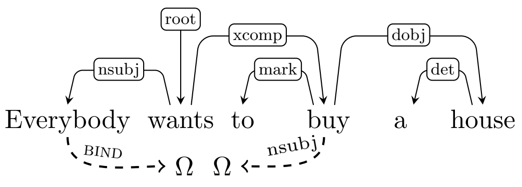
In (1), the existential variable w is in the scope of the universal variable x (i.e. the *house* is dependent on the *person*). This reading is commonly referred to as the *surface reading*. Conversely, in (2) the universal variable x is in the scope of the existential variable w (i.e. the *house* is independent of the *person*). This reading is also called *inverse reading*. Our goal is to obtain the surface reading logical form in (1) with UDEPLAMBDA. We do not aim to obtain the inverse reading, although this is possible with the use of Skolemization (Steedman, 2012).

In UDEPLAMBDA, lambda expressions for words, phrases and sentences are all of the form $\lambda x. \ldots$. But from (1), it is clear that we need to express variables bound by quantifiers, e.g. $\forall x$, while still providing access to x for composition. This demands a change in the type system since the

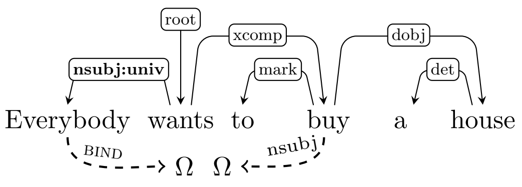
[1]Example borrowed from Schuster and Manning (2016).



(a) Original dependency tree.



(b) Enhanced dependency tree.



(c) Enhanced dependency tree with universal quantification.

Figure 1: The dependency tree for *Everybody wants to buy a house* and its enhanced variants.

same variable cannot be lambda bound and quantifier bound—that is we cannot have formulas of the form $\lambda x \ldots \forall x \ldots$. In this material, we first derive the logical form for the example sentence using the type system from our main paper (Section 1.1) and show that it fails to handle universal quantification. We then modify the type system slightly to allow derivation of the desired surface reading logical form (Section 1.2). This modified type system is a strict generalization of the original type system.[2] Fancellu et al. (2017) present an elaborate discussion on the modified type system, and how it can handle negation scope and its interaction with universal quantifiers.

[2]Note that this treatment has yet to be added to our implementation, which can be found at https://github.com/sivareddyg/udeplambda.

1.1 With Original Type System

We will first attempt to derive the logical form in (1) using the default type system of UDEPLAMBDA. Figure 1(b) shows the enhanced dependency tree for the sentence, where BIND has been introduced to connect the implied `nsubj` of *buy* (BIND is explained in the main paper in Section 3.2). The s-expression corresponding to the enhanced tree is:

(nsubj (xcomp wants (mark
  (nsubj (dobj buy (det house a)) Ω) to))
 (BIND everybody Ω)) .

With the following substitution entries,

wants, *buy* $\in$ EVENT;
everybody, *house* $\in$ ENTITY;
a, *to* $\in$ FUNCTIONAL;
$\Omega = \lambda x.\, \mathrm{EQ}(x, \omega)$;
`nsubj` $= \lambda f g x.\, \exists y.\, f(x) \wedge g(y) \wedge \mathrm{arg}_1(x_e, y_a)$;
`dobj` $= \lambda f g x.\, \exists y.\, f(x) \wedge g(y) \wedge \mathrm{arg}_2(x_e, y_a)$;
`xcomp` $= \lambda f g x.\, \exists y.\, f(x) \wedge g(y) \wedge \mathrm{xcomp}(x_e, y_a)$;
`mark` $\in$ HEAD;
BIND $\in$ MERGE,

the lambda expression after composition becomes:

$$\lambda z.\ \exists xywv.\, \mathrm{wants}(z_e) \wedge \mathrm{everybody}(x_a) \wedge \mathrm{arg}_1(z_e, x_a) \wedge \mathrm{EQ}(x, \omega) \wedge \mathrm{buy}(y_e) \wedge \mathrm{xcomp}(z_e, y_e) \wedge \mathrm{arg}_1(y_e, v_a) \wedge \mathrm{EQ}(v, \omega) \wedge \mathrm{arg}_1(x_e, y_a) \wedge \mathrm{house}(w_a) \wedge \mathrm{arg}_2(y_e, w_a)\, .$$

This expression encodes the fact that x and v are in unification, and can thus be further simplified to:

(3) $\lambda z.\, \exists xyw.\, \mathrm{wants}(z_e) \wedge \mathrm{everybody}(x_a) \wedge \mathrm{arg}_1(z_e, x_a) \wedge \mathrm{buy}(y_e) \wedge \mathrm{xcomp}(z_e, y_e) \wedge \mathrm{arg}_1(y_e, x_a) \wedge \mathrm{arg}_1(x_e, y_a) \wedge \mathrm{house}(w_a) \wedge \mathrm{arg}_2(y_e, w_a)\, .$

However, the logical form (3) differs from the desired form (1). As noted above, UDEPLAMBDA with its default type, where each s-expression must have the type $\eta = \mathbf{Ind} \times \mathbf{Event} \rightarrow \mathbf{Bool}$, cannot handle quantifier scoping.

1.2 With Higher-order Type System

Following Champollion (2010), we make a slight modification to the type system. Instead of using expressions of the form $\lambda x. \ldots$ for words, we use either $\lambda f. \exists x. \ldots$ or $\lambda f. \forall x. \ldots$, where f has type η. As argued by Champollion, this higher-order form makes quantification and negation handling sound and simpler in Neo-Davidsonian event semantics. Following this change, we assign the following lambda expressions to the words in our example sentence:

everybody = $\lambda f.\, \forall x.\, \mathrm{person}(x) \rightarrow f(x)$;
wants = $\lambda f.\, \exists x.\, \mathrm{wants}(x_e) \wedge f(x)$;
to = $\lambda f.\, \mathrm{TRUE}$;
buy = $\lambda f.\, \exists x.\, \mathrm{buy}(x_e) \wedge f(x)$;
a = $\lambda f.\, \mathrm{TRUE}$;
house = $\lambda f.\, \exists x.\, \mathrm{house}(x_a) \wedge f(x)$;
$\Omega = \lambda f.\, f(\omega)$.

Here *everybody* is assigned universal quantifier semantics. Since the UD representation does not distinguish quantifiers, we need to rely on a small (language-specific) lexicon to identify these. To encode quantification scope, we enhance the label `nsubj` to `nsubj:univ`, which indicates that the subject argument of *wants* contains a universal quantifier, as shown in Figure 1(c).

This change of semantic type for words and s-expressions forces us to also modify the semantic type of dependency labels, in order to obey the single-type constraint of DEPLAMBDA (Reddy et al., 2016). Thus, dependency labels will now take the form $\lambda P Q f. \ldots$, where P is the parent expression, Q is the child expression, and the *return expression* is of the form $\lambda f. \ldots$. Following this change, we assign the following lambda expressions to dependency labels:

`nsubj:univ` $= \lambda P Q f.\, Q(\lambda y.\, P(\lambda x.\, f(x) \wedge \mathrm{arg}_1(x_e, y_a)))$;
`nsubj` $= \lambda P Q f.\, P(\lambda x.\, f(x) \wedge Q(\lambda y.\, \mathrm{arg}_1(x_e, y_a)))$;
`dobj` $= \lambda P Q f.\, P(\lambda x.\, f(x) \wedge Q(\lambda y.\, \mathrm{arg}_2(x_e, y_a)))$;
`xcomp` $= \lambda P Q f.\, P(\lambda x.\, f(x) \wedge Q(\lambda y.\, \mathrm{xcomp}(x_e, y_a)))$;
`det`, `mark` $= \lambda P Q f.\, P(f)$;
BIND $= \lambda P Q f.\, P(\lambda x.\, f(x) \wedge Q(\lambda y.\, \mathrm{EQ}(y, x)))$.

Notice that the lambda expression of `nsubj:univ` differs from `nsubj`. In the former, the lambda variables inside Q have wider scope over the variables in P (i.e. the universal quantifier variable of *everybody* has scope over the event variable of *wants*) contrary to the latter.

The new s-expression for Figure 1(c) is

(nsubj:univ (xcomp wants (mark
  (nsubj (dobj buy (det house a)) Ω) to))
 (BIND everybody Ω)) .

Substituting with the modified expressions, and performing composition and simplification leads to the expression:

(6) $\lambda f.\, \forall x\, .\mathrm{person}(x_a) \rightarrow [\exists zyw.\, f(z) \wedge \mathrm{wants}(z_e) \wedge \mathrm{arg}_1(z_e, x_a) \wedge \mathrm{buy}(y_e) \wedge \mathrm{xcomp}(z_e, y_e) \wedge \mathrm{house}(w_a) \wedge \mathrm{arg}_1(z_e, x_a) \wedge \mathrm{arg}_2(z_e, w_a)]\, .$

This expression is identical to (1) except for the outermost term λf. By applying (6) to $\lambda x.\mathrm{TRUE}$, we obtain (1), which completes the treatment of universal quantification in UDEPLAMBDA.

References

Lucas Champollion. 2010. Quantification and negation in event semantics. *Baltic International Yearbook of Cognition, Logic and Communication* 6(1):3.

Federico Fancellu, Siva Reddy, Adam Lopez, and Bonnie Webber. 2017. Universal Dependencies to Logical Forms with Negation Scope. *arXiv Preprint* .

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics* 4:127–140.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English Universal Dependencies: An Improved Representation for Natural Language Understanding Tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*. European Language Resources Association (ELRA), Paris, France.

Mark Steedman. 2012. *Taking Scope - The Natural Semantics of Quantifiers*. MIT Press.