A Type-coherent, Expressive Representation as an Initial Step to Language Understanding

Gene Louis Kim and Lenhart Schubert
Department of Computer Science, University of Rochester
{gkim21,schubert}@cs.rochester.edu

Abstract

A growing interest in tasks involving language understanding by the NLP community has led to the need for effective semantic parsing and inference. Modern NLP systems use semantic representations that do not quite fulfill the nuanced needs for language understanding: adequately modeling language semantics, enabling general inferences, and being accurately recoverable. This document describes underspecified logical forms (ULF) for Episodic Logic (EL), which is an initial form for a semantic representation that balances these needs. ULFs fully resolve the semantic type structure while leaving issues such as quantifier scope, word sense, and anaphora unresolved; they provide a starting point for further resolution into EL, and enable certain structural inferences without further resolution. This document also presents preliminary results of creating a hand-annotated corpus of ULFs for the purpose of training a precise ULF parser, showing a three-person pairwise interannotator agreement of 0.88 on confident annotations. We hypothesize that a divide-and-conquer approach to semantic parsing starting with derivation of ULFs will lead to semantic analyses that do justice to subtle aspects of linguistic meaning, and will enable construction of more accurate semantic parsers.

1 Introduction

Episodic Logic (EL) is a semantic representation extending FOL, designed to closely match the expressivity and surface form of natural language and to enable deductive inference, uncertain inference, and NLog-like inference (Morbini and Schubert, 2009; Schubert and Hwang, 2000; Schubert, 2014). Kim and Schubert (2016) developed a system that transforms annotated WordNet glosses into EL axioms which were competitive with state-of-the-art lexical inference systems while achieving greater expressivity. While EL is representationally appropriate for language understanding, the current EL parser is too unreliable for general text: The phrase structures produced by the underlying Treebank parser leave many ambiguities in the semantic type structure, which are disambiguated incorrectly by the hand-coded compositional rules; moreover, errors in the phrase structures can further disrupt the resulting logical forms (LFs). Kim and Schubert (2016) discuss the limitations of the existing parser as a starting point for logically interpreting glosses of WordNet verb entries. In order to build a better EL parser, it seems natural to take advantage of recent advances in corpus-based parsing techniques.

This document describes a type-coherent initial LF, or *unscoped logical forms* (ULF), for EL which captures the predicate-argument structure in the EL semantic types and is the first critical step in fully-resolved semantic interpretation of sentences. Montague's profoundly influential work (Montague, 1973) demonstrates that systematic assignments of appropriate semantic types to words and phrases allows us to view language as akin to formal logic, with meanings determined compositionally from syntactic structures. This view of language directly supports inferences, at least to the extent that we can resolve – or are prepared to tolerate – ambiguity, context-dependence, and indexicality, towards which semantic types are agnostic. ULF takes a minimal step across the syntax-semantics interface by doing exactly this – selecting the semantic types of words within EL. Thus ULFs are amenable to corpus-construction and statistical parsing using techniques similar to those used for syntax, and they enable generation of context-dependent structural inferences. The nature of these inferences is discussed in more detail in Section 3.4.

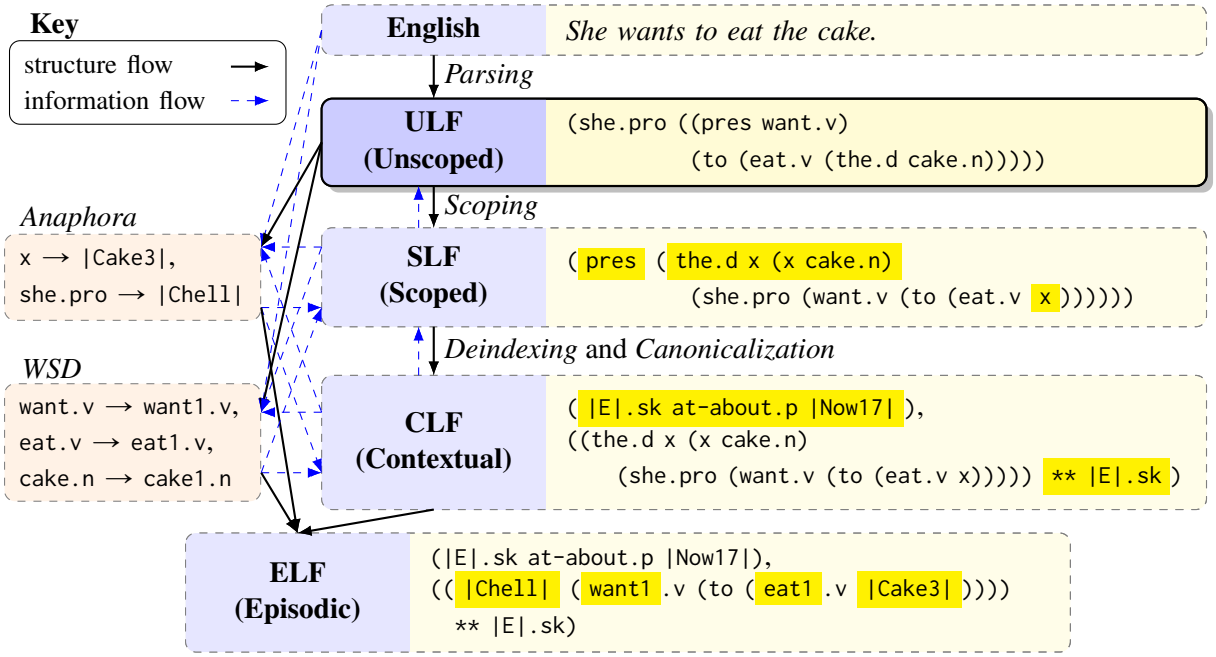

Figure 1: The semantic interpretation process, with the ULF step in the fore. Structurally dependent steps in the interpretation process are connected by solid black arrows and structurally independent information flow is represented with dashed blue arrows. The components that changed from the previous structural step are highlighted in yellow. Backward information arrows indicate that arriving at the optimal choice at a particular step may depend on "later" – or structurally dependent – steps.

Our working hypothesis in designing ULF is that a divide-and-conquer approach starting with preliminary surface-like LFs is a practical way to generate fully resolved interpretations of natural language in EL. Figure 1 shows a diagram of our divide-and-conquer approach, which is elaborated upon in Section 3.3. We also outline a framework for quickly and reliably collecting ULF annotations for a corpus in a multi-pronged approach. Our evaluation of the annotation framework shows that we achieve annotation speeds and agreement comparable to those for the *abstract meaning representation* (AMR) project, which has successfully built a large enough corpus to drive research into corpus-based parsing (Banarescu et al., 2013). Further resources relating to this project, including a more in-depth description of ULFs, the annotation guidelines, and related code are available from the project website `http://cs.rochester.edu/u/gkim21/ulf/`.

2 Episodic Logic

EL is a semantic representation that extends FOL to more closely match the expressivity of natural languages. It echoes both the surface form of language, and more crucially, the semantic types that are found in all languages. Some semantic theorists view the fact that noun phrases denoting both concrete and abstract entities can appear as predicate arguments (*Aristotle*, *everyone*, *the fact that there is water on Mars*) as grounds for treating all noun phrases as being of higher types (e.g., second-order predicates). EL instead uses a small number of reification operators to map predicate and sentence intensions to individuals. As a result, quantification remains first-order (but allows quantified phrases such as *most people who smoke*, or *hardly any errors*). Another distinctive feature of EL is that it treats the relation between sentences and episodes (including events, situations, and processes) as a *characterizing* relation, written '**'. This coincides with the Davidsonian treatment of events as extra variables of predicates (Davidson, 1967) when we restrict ourselves to positive, atomic predications. However, '**' also allows for logically complex characterizations of episodes, such as *not eating anything all day*, or *each superpower menacing the other with its nuclear arsenal* (Schubert, 2000).

EL defines a hierarchical ontology over the domain of individuals, $\mathcal{D}$. $\mathcal{D}$ includes simple individuals,

e.g. *John*, possible situations, $\mathcal{S}$, possible worlds, $\mathcal{W}\subset\mathcal{S}$, various numerical types, propositions, $\mathcal{P}$, and kinds, $\mathcal{K}$, as well as others that are not important for the purposes of this document. A complete description of the ontology is provided by Schubert and Hwang (2000). The types of some predicates are further restricted by these categories. For example, the predicate `claim.v` – as in *"I claim that grass is red."* – has the type $\mathcal{P} \rightarrow (\mathcal{D} \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))$, since its first argument is a proposition and the second argument is a simple individual (in the semantics of EL the agent argument is supplied last, though it precedes the predicate in the surface syntax).

The semantic types in EL are defined by recursive functions over individuals, $\mathcal{D}$, and truth values, $\{0,1\}$, written as $\mathbf{2}$. Semantic values of predicates applied to their surface arguments can yield a value in $\mathbf{2}$ at a given (possible) situation, or be *undefined* there (indicating irrelevance of the predication in the given situation). Most predicates in EL are of type $\mathcal{D}^n \rightarrow (\mathcal{S} \rightarrow \mathbf{2})$ (where $\mathcal{D}^2 \rightarrow \mathbf{2}$ abbreviates $\mathcal{D} \rightarrow (\mathcal{D} \rightarrow \mathbf{2})$, $\mathcal{D}^3 \rightarrow \mathbf{2}$ abbreviates $\mathcal{D} \rightarrow (\mathcal{D} \rightarrow (\mathcal{D} \rightarrow \mathbf{2}))$, and so on). That is, they are first-order intensional predicates.[1] Monadic predicates play a particularly important role in EL as well as ULF, and we will abbreviate their type $\mathcal{D} \rightarrow (\mathcal{S} \rightarrow \mathbf{2})$ as $\mathcal{N}$. In EL syntax, square brackets indicate infixed operators (i.e. $[\tau_n\ \pi\ \tau_1\ ...\ \tau_{n-1}]$ where π is the operator) and parentheses indicate prefixed operators (i.e. $(\pi\ \tau_1\ ...\ \tau_n)$ where π is the operator). Predicative formulas such as `[|Aristotle| famous.a]` or `[|Romeo| love.v |Juliet|]` are regarded as temporal and must be evaluated with respect to a situation via an episode-relating operator (e.g. '**') to supply the episode and thus produce an atemporal formula.

There are also a limited number of type-shifting operators in EL to map between some of these types. The kind operator, 'k', shifts a monadic predicate into a kind, $(\mathcal{D} \rightarrow (\mathcal{S} \rightarrow \mathbf{2})) \rightarrow \mathcal{K}$, and the operator , 'that', forms propositions from sentence intensions, $(\mathcal{S} \rightarrow \mathbf{2}) \rightarrow \mathcal{P}$. *"that grass is red"*, a segment of an earlier example, is formulated as `(that [(k grass.n) red.a])` in EL, uses both of these operators.

3 Unscoped logical form

ULFs are type-coherent initial LFs which provide a stepping stone to capturing full sentential EL meanings. They enable interesting classes of structural inferences that are of broader scope than those enabled by Natural Logic (NLog) (Sánchez Valencia, 1995), and unlike NLog inferences do not depend on prior knowledge of the propositions to be confirmed or refuted. ULF captures the full predicate argument structure of EL while leaving word sense, scope, and anaphora unresolved. Therefore, ULFs can be analyzed using the formal EL type system while taking the scopal ambiguities into account. There is not enough space here to exhaustively discuss how ULF handles various phenomena, so the discussion will be restricted to the broad framework of ULF and the most crucial aspects of the semantics. Please refer to `http://cs.rochester.edu/u/gkim21/ulf/` for complete information on ULF.

3.1 ULF Syntax

All atoms in ULF, with the exception of certain logical functions and syntactic macros, are marked with an atomic syntactic type. The atomic syntactic types are written with suffixed tags: `.v`, `.n`, `.a`, `.p`, `.pro`, `.d`, `.aux-v`, `.aux-s`, `.adv-a`, `.adv-e`, `.adv-s`, `.adv-f`, `.cc`, `.ps`, `.pq`, `.mod-n`, or `.mod-a`, except for names, which use wrapped bars, e.g. `|John|`. These are intended to echo the part-of-speech origins of the constituents, such as *verb, noun, adjective, preposition, pronoun, determiner,* etc., respectively; some of them contain further specifications as relevant to their entailments, e.g., `.adv-e` for locative or temporal adverbs (implying properties of events). The distinctions among predicates of sorts `.v`, `.n`, `.a`, `.p`, corresponding to English parts of speech, are often suppressed in other LFs for language, but are semantically important. For example, *"Bob danced"* can refer to a brief episode while *"Jill was a dancer"* generally cannot (and may suggest Jill is no longer alive); this is related to the fact that verbal predicates are typically "stage-level" (episodic) while nominal predicates are generally "individual-level" (enduring). Whereas in EL the bracket type specifies whether prefix or infix notation is being used, in ULF this distinction is inferred from the semantic types of the constituents and only parentheses are used.

[1] Some predicates allow for a monadic predicate complement such as *look* in *"They look happy"*.

(1) *Could you dial for me?*
```
(((pres could.aux-v) you.pro (dial.v {ref1}.pro (adv-a (for.p me.pro)))) ?)
```
(2) *If I were you I would be able to succeed.*
```
((if.ps (i.pro ((cf were.v) (= you.pro))))
 (i.pro ((cf will.aux-s) (be.v (able.a (to succeed.v))))))
```
(3) *Flowers are weak creatures*
```
((k (plur flower.n)) ((pres be.v) (weak.a (plur creature.n))))
```
(4) *Very few people still debate the fact that the earth is heating up*
```
(((fquan (very.mod-a few.a)) (plur person.n))
          (still.adv-s (debate.v
             (the.d (n+preds fact.n (= (that ((the.d |Earth|.n)
                                              ((pres prog) heat_up.v))))))))
```

Figure 2: Example sentences with corresponding raw ULF annotations. Examples (1) and (2) are from the Tatoeba database, (3) is from *The Little Prince*, and (4) is from the Web. Strictly speaking, `weak.a` in (3) is actually missing a type-shifting operator `mod-n`, a simplification discussed in Section 4.

Atoms that are implicit in the sentence or elided and thus supplied by the annotator are wrapped in curly brackets, such as `{ref}.pro` in example (1) of Figure 2.

For practical purposes we distinguish *raw ULF* from *postprocessed ULF*. In raw ULF we allow certain argument-taking constituents to be dislocated from their "proper" place, so as to adhere more closely to linguistic surface structure and thereby facilitate annotation. For example, sentence-level operators (of type `adv-s`) appearing mid-sentence may be left "floating" (e.g., `(|Alice| certainly.adv-s ((pres know.v) |Bob|))`), since they can be automatically lifted to the sentence-level; and verb-level adverbs (of type `adv-a`) can be interleaved with arguments (e.g., `((past speak.v) sternly.adv-a (to.p-arg |Bob|))`), even though semantically they operate on the whole verb phrase. Kim and Schubert (2017) presented this method of dislocated annotation for sentence-level operators. In postprocessed ULF, we can understand all atoms and subexpressions of well-formed formulas (wffs) as being one of the following ULF constituent types (modulo some following remarks):

entity, predicate, determiner, monadic predicate modifier, sentence, sentence modifier, connective, lambda abstract, or one of a limited number of type-shifting operators,

where the predicates and operators that act on predicates are subcategorized by whether the predicate is derived from a noun, verb, adjective, or preposition. These constituent types uniquely map to particular semantic types, i.e., are aliases for the formal types. Clausal constituents are combined according to their bracketing and semantic types.

A qualification of the above general claim is that unscoped tense operators, determiners, and coordinators remain in their surface position even in postprocessed ULF. For example, in `(|Bob| ((pres own.v) (a.d dog.n)))`, `pres` is actually an unscoped sentence-level operator (which, in conversion to EL, is deindexed to yield a characterization of an episode by the sentence, and a temporal predication about that episode). We also retain coordinated expressions such as `((in.p |Rome|) and.cc happy.a)`, where this will ultimately lead to a sentential conjunction in EL. Similarly, `(a.d dog.n)` is kept in argument position as if it were of semantic type $\mathcal{D}$ (thus, as if the determiner were of semantic type $\mathcal{N} \rightarrow \mathcal{D}$).[2] Such unscoped constituents do not disrupt type coherence, because the possible conversions to type-coherent EL are well-defined.

Finally, both raw ULFs and postprocessed ULFs can contain macros. For example, the macro operator `n+preds` is used for postmodified nominal predicates such as `(n+preds dog.n (on.p (a.d leash.n)))` – see also example (4) in Figure 2; this avoids immediate introduction of a λ-abstracted conjunction of predicates, simplifying the annotation task. Appendix C discusses macros further, including their formal definitions. Section 4 will ground the high-level discussions in this and the following section with a concrete discussion of modifiers.

[2]The actual semantic type of determiners in EL, after lambda-abstraction of the restrictor and matrix formula, is $\mathcal{N} \rightarrow (\mathcal{N} \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))$. See Appendix A for full details.

3.2 ULF Type Structure

The type-shifting operators mentioned in the previous section are crucial for type coherence in ULFs. In example (1) the phrase *"for me"* is coded as `(adv-a (for.p me.pro))`, rather than simply `(for.p me.pro)` because it is functioning as a *predicate modifier*, semantically operating on the verbal predicate `(dial.v {ref1}.pro)` (*dial a certain thing*). Let $\mathcal{N}_{ADJ}$, $\mathcal{N}_N$, and $\mathcal{N}_V$ be the sortal refinements of the monadic predicate type $\mathcal{N}$ corresponding to adjectives, nouns, and verbs, respectively. `(adv-a (for.p me.pro))` has type $\mathcal{N}_V \rightarrow \mathcal{N}_V$. Without the `adv-a` operator the prepositional phrase is just a 1-place predicate. Its use as a predicate is apparent in contexts like *"This puppy is for me"*. Note that semantically the 1-place predicate `(for.p me.pro)` is formed by applying the 2-place predicate `for.p` to the (individual-denoting) term `me.pro`. If we apply `(for.p me.pro)` to another argument, such as `|Snoopy|` (the name of a puppy), we obtain a sentence intension.[3] So semantically, `adv-a` is a *type-shifting operator* of type $\mathcal{N} \rightarrow (\mathcal{N}_V \rightarrow \mathcal{N}_V)$.

This brings up the issue of *intensionality*, which is preserved in ULF. Example (2) is a counterfactual conditional, and the consequent clause *"I would be able to succeed"* is not evaluated in the actual world, but in a possible world where the (patently false) antecedent is imagined to be true. ULF captures this with the '`cf`' operator in place of the tense and the EL formulas derived from it are evaluated with respect to *possible situations (episodes)*, whose maxima are possible worlds. The type of '`cf`' is $(\mathcal{S} \rightarrow \mathbf{2}) \rightarrow (\mathcal{S} \rightarrow \mathbf{2})$ after operator scoping to the sentence-level, but like tense operators is kept with the verb in raw ULF, essentially functioning as a predicate-level identity function, $(\lambda X.X)$, there.

'`to`' in (2), '`k`' in (3), and '`that`' in (4) are all operators that reify different semantic categories, shifting them to abstract individuals. '`to`' (synonym: `ka`) shifts a verbal predicate to a *kind (type) of action or attribute*, $\mathcal{N}_V \rightarrow \mathcal{K}_A$; '`k`' shifts a nominal predicate to a *kind* of thing, $\mathcal{N}_N \rightarrow \mathcal{K}$ (so the subject in example (3) is the abstract kind, *flowers*, whose instances consist of sets of flowers); and '`that`' produces a reified *proposition*, $(\mathcal{S} \rightarrow \mathbf{2}) \rightarrow \mathcal{P}$ (again an abstract individual) from a sentence meaning. Using these type shifts, EL and ULF are able to maintain a simple, classical view of predication, while allowing greater expressivity than the most widely employed LFs.

3.3 Role of ULF in Comprehensive Semantic Interpretation

ULFs are underspecified, but their surface-like form and the type structure they encode make them well-suited to reducing underspecification by using well-established linguistic principles and exploiting the distributional properties of language. Figure 1 shows the interpretation process for EL formulas and the role of ULFs in providing the first step into it. Due to the structural dependencies between the components in the interpretation process, the optimal choice at any given component depends on the overall coherence of the final interpretation; hence the backward arrows in the figure. Word sense disambiguation (WSD) and anaphora have no structural dependencies in the interpretation process so they are separated from and fully connected to the post-ULF components. These resolutions are depicted in the last step in the figure.

WSD & Anaphora: While `(weak.a (plur creature.n))` in example (3) does not specify which of the dozen WordNet senses of *weak* or three senses of *creature* is intended here, the type structure is perfectly clear: A predicate modifier is being applied to a nominal predicate. ULF also does not assume unique adicity of word-derived predicates such as `run.v`, since such predicates can have intransitive, simple transitive and other variants, but the adicity of a predicate in ULF is always clear from its structural context – we know that it has all its arguments in place when an argument (the "subject") is placed on its left, as in English.

Linguistic constraints (e.g. *binding constraints*) exist for coreference resolution. For example, in *"John said that he was robbed"*, *he* can refer to *John*; but this is not possible in *"He said that John was robbed"*, because in the latter, *he* C-commands *John*, i.e., in the phrase structure of the sentence, it is a sibling of an ancestor of *John*. ULF preserves this structure, allowing use of such constraints. While ULF

[3] `(for.p me.pro)` has type $\mathcal{D} \rightarrow (\mathcal{S} \rightarrow \mathbf{2})$ and `|Snoopy|` has type $\mathcal{D}$, so `(|Snoopy| (for.p me.pro))` has a type that resolves to $\mathcal{S} \rightarrow \mathbf{2}$ (i.e. a sentence intension).

constrains the word senses and coreferences through adicity and syntactic structure, WSD and anaphora resolution should not be applied to isolated sentences since word sense patterns and coreference chains often span multiple sentences.

Scoping: Unscoped constituents (determiners, tense operators, and coordinators) can generally "float" to more than one possible position. Following a view of scope ambiguity developed by Schubert and Pelletier (1982) elaborated by Hurum and Schubert (1986), these constituents always float to pre-sentential positions, and determiner phrases leave behind a variable that is then bound at the sentential level. The accessible positions are constrained by linguistic restrictions, such as *scope island* constraints in subordinate clauses (Ruys and Winter, 2010). Beyond this, many factors influence preferred scoping possibilities, with surface form playing a prominent role (Manshadi et al., 2013). The proximity of ULF to surface syntax enables the use of these constraints.

Deindexing and Canonicalization: Much of the past work relating to EL has been concerned with the principles of *deindexing* (Hwang, 1992; Hwang and Schubert, 1994; Schubert and Hwang, 2000). Deindexing corresponds to the introduction of event variables for explicitly characterizing the sentence it is linked to via the '**' operator (this variable becomes `|E|.sk` in Figure 1 after Skolemization). Hwang and Schubert's approach to tense-aspect processing, constructing *tense trees* for temporally relating event variables, is only possible if the LF being processed reflects the original clausal structure – as ULF indeed does. Canonicalization is the mapping of an LF into "minimal", distinct propositions, with top-level Skolemization. The CLF step in Figure 1 contains two separate formulas as a result of this process.

Episodic Logical Forms (ELF): When episodes have been made explicit and all anaphoric and word ambiguities are resolved the result is a set of *episodic logical forms*. These can be used in the EPILOG inference engine for reasoning that combines linguistic semantic content with world knowledge.[4] A variety of complex EPILOG inferences are reported by Schubert (2013), and Morbini and Schubert (2011) give examples of self-aware metareasoning. EPILOG also reasoned about snippets from the Little Red Riding Hood story, for example using knowledge about the world and goal-oriented behavior to understand why the presence of nearby woodcutters prevented the wolf from attacking Little Red Riding Hood when he first saw her (Hwang, 1992; Schubert and Hwang, 2000).

3.4 Inference with ULFs

An important insight of NLog research is that language can be used directly for inference, requiring only phrase structure analysis and upward/downward entailment marking (polarity) of phrasal contexts. This means that NLog inferences are *situated* inferences, i.e., their meaning is just as dependent on the utterance setting and discourse state as the linguistic "input" that drives them. This insight carries over to ULFs, and provides a separate justification for computing ULFs, apart from their utility in the process of deriving EL interpretations from language. The semantic type structure encoded by ULFs provides a more reliable and general basis for situated inference than mere phrase structure. Here, briefly, are some kinds of inferences we can expect ULFs to support with minimal additional knowledge due to their structural nature:

- *NLog inferences based on generalizations/specializations*. For example, *"Every NATO member sent troops to Afghanistan"*, together with the knowledge that France is a NATO member and that Afghanistan is a country entails that *France sent troops to Afghanistan* and that *France sent troops to a country*.

- *Inferences based on implicatives*. For example, *"She managed to quit smoking"* entails that *She quit smoking* (and the negation of the premise leads to the opposite conclusion). Inferences of this sort have been demonstrated for headlines using ELFs by Stratos et al. (2011).

- *Inferences based on attitudinal and communicative verbs*. For example, *"John denounced Bill as a charlatan"* entails that *John probably believes that Bill is a charlatan*, that *John asserted to his*

[4] EPILOG is competitive against state-of-the-art FOL theorem provers (Morbini and Schubert, 2009).

listeners (or readers) that Bill is a charlatan, and that *John wanted his listeners (or readers) to believe that Bill is a charlatan*. These inferences would be hard to capture within NLog, since they are partially probabilistic, require structural elaboration, and depend on constituent types.

- *Inferences based on counterfactuals*. For example, *"If I were rich, I would pay off your debt"* and *"I wish I were rich"* both implicate that *the speaker is not rich*. This depends on recognition of the counterfactual form, which is distinguished in ULF.

- *Inferences from questions and requests*. For example, *"When are you getting married?"* enables the inferences that the addressee will get married (in the foreseeable future), that the questioner wants to know the expected date of the event, and that the addressee probably knows the answer and will supply it. Similarly an apparent request such as *"Could you close the door?"* implies that the speaker wants the addressee to close the door, and expects that he or she will do so.

4 Predicate and Sentence Modification in Depth

Here we ground the general description of ULF given so far with an in-depth discussion of how ULF handles modification. This is done with the purpose of demonstrating how the core syntax of ULF, its syntactic looseness, and semantic types fit together in practice. EL semantic types represent *predicate modifiers* as functions from *monadic* intensional predicates to *monadic* intensional predicates, i.e., $\mathcal{N} \rightarrow \mathcal{N}$, which enables handling of intersective, subsective, and intensional modifiers such as in the examples

`((mod-n wooden.a) shoe.n)`, `((mod-n ice.n) pick.n)`, `(fake.mod-n ruby.n)`,
`((mod-a worldly.a) wise.a)`, `(very.mod-a fit.a)`, `(slyly.adv-a grin.v)`.

Modifier extensions `.mod-n`, and `.mod-a` respectively reflect the linguistic categories of noun-premodifying (attributive) adjectives and adjective-premodifying adverbs; correspondingly, operators `mod-n`, and `mod-a` type-shift prenominal predicates to modifiers applicable to predicates of sorts `.n` and `.a` respectively. Modifier extension `.adv-a` reflects the linguistic category of VP adverbials, and operator `adv-a` creates such modifiers from predicates. Thus, *"walk with Bob"* is represented in raw and postprocessed ULF respectively as

`(walk.v (adv-a (with.p |Bob|)))` and `((adv-a (with.p |Bob|)) walk.v)`.

Adverbial modifiers of the sort `.adv-a` intuitively modify actions, experiences, or attributes, as distinct from events. Thus *"He lifted the child easily"* refers to an action that was easy for the agent, rather than to an easy event. Actions, experiences, and attributes in EL are individuals comprised of agent-episode pairs, and this allows modifiers of the sort `.adv-a` to express a constraint on both the agent and the episode it characterizes. As such, actions are not explicitly represented in ULF but rather derived during deindexing when event variables are introduced.

A formula or nonatomic verbal predicate in ULF may contain sentential modifiers of type $(\mathcal{S} \rightarrow \mathbf{2}) \rightarrow (\mathcal{S} \rightarrow \mathbf{2})$: `.adv-s`, `.adv-e`, and `.adv-f`. Again there are type-shifting operators that create these sorts of modifiers from monadic predicates. Ones of the sort `.adv-s` are usually modal (and thus opaque), e.g.,

`perhaps.adv-s`, `(adv-s (without.p (a.d doubt.n)))`;

However, negation is transparent in the usual sense – the truth value of a negated sentence depends only of the truth value of the unnegated sentence. Modifiers of sort `.adv-e` are transparent, typically implying temporal or locative constraints, e.g.,

`today.adv-e`, `(adv-e (during.p (the.d drought.n)))`, `(adv-e (in.p |Rome|))`;

these constraints are ultimately cashed out as predications about episodes characterized by the sentence being modified. (This is also true for the `past` and `pres` tense operators.) Similarly any modifier of sort `.adv-f` is transparent and implies the existence of a multi-episode (characterized by the sentence as a whole) whose temporally disjoint parts each have the same characterization (Hwang and Schubert, 1994); e.g.,

`regularly.adv-f`, `(adv-f (at.p (three.d (plur time.n))))`;

The earlier *walk with Bob* example shows how in ULF the operator and operand can be inferred from the constituent types. Consider the types for `play.v` and `(adv-a (with.p (the.d dog.n)))`. Since they

have types $\mathcal{N}_V$ and $\mathcal{N}_V \rightarrow \mathcal{N}_V$, respectively, we can be certain that `(adv-a (with.p (the.d dog.n)))` is the operator while `play.v` is the operand.

In practice, we're able to drop the `mod-a`, `mod-n`, and `nnp` type-shifters during annotation since we can post-process them with the appropriate type-shifter to make the composition valid. We assume in these cases that the prefixed predicate is intended as the operator, which reflects a common pattern in English. Thus, "burning hot melting pot" would be hand annotated as

```
((burning.a hot.a) (melting.n pot.n))
```

which would be post-processed to

```
((mod-n ((mod-a burning.a) hot.a)) ((mod-n melting.n) pot.n))
```

While the prefixed predicate modification allows us to formally model non-intersective modification, there are modification patterns in English that force an intersective interpretation, e.g., post-nominal modification and appositives, and we annotate them accordingly. *"The buildings in the city"* is annotated

```
(the.d (n+preds (plur building.n) (in.p (the.d city.n))))
```

which is equivalent (via the `n+preds` macro) to

```
(the.d (λx ((x (plur building.n)) and.cc (x (in.p (the.d city.n))))))
```

.

5 Annotating a ULF Corpus

The syntactic relaxations in ULF and the annotation environment work hand-in-hand to enable quick and consistent annotations. ULF syntax relaxations are designed to: (1) Preserve surface word order and (2) Make the annotations match linguistic intuitions more closely. As a result, annotating a sentence with its ULF interpretation boils down to marking the words with their semantic types, bracketing the sentence according to the operator-operand relations, then introducing macros and logical operators as necessary to make the ULF type-consistent. The annotation environment is designed to assist in this process by improving the readability of long ULFs and catching mistakes that are easy to miss. The environment is shared across annotators with certainty marking so that more experienced annotators can correct and give feedback to trainees. This streamlines the training process and minimizes the mistakes entering into the corpus. Here are the core annotator features.[5]

1. **Syntax and bracket highlighting.** Highlights the cursor location and the closing bracket, unmatched brackets and quotes, operator keywords, and badly placed operators.

2. **Sanity checker.** Alerts the annotator to invalid type compositions and suggests corrections for common mistakes.

3. **Certainty marking.** Annotators can mark whether they are certain of an annotation's correctness so that partial progress can be made while preserving the integrity of the corpus.

4. **Sentence-specific comments.** Annotators can record their thoughts on partially complete annotations so that others can pick up where they left off.

The ULF type system makes it possible to build a robust sanity checker for the annotator. The type system severely restricts the space of valid ULF formulas and usually when an annotator makes an error in annotation, it leads to a type inconsistency.

6 Experimental Results and Current Progress

We ran a timing study and an interannotator agreement (IA) study to quantify the efficacy of the presented annotation framework. We timed 80 annotations of the Tatoeba dataset and found the average annotation speed to be 8 min/sent with 4 min/sent among the two experts and 11 min/sent among the three trainees that participated. AMRs reportedly took on average 10 min/sent (Hermjakob, 2013). In the IA study five

[5]The annotator can be accessed from the ULF project website and a screenshot of it is in Appendix D.

annotators each annotated between 18 and 23 sentences from the same set of 23 sentences, marking their certainty of the annotations as they normally would. The sentences were sampled from the four datasets listed in Table 1. The mean and standard deviation of sentence length were 15.3 words and 10.8 words, respectively.

We computed a similarity score between two annotations using *EL-smatch* (Kim and Schubert, 2016), a generalization of *smatch* (Cai and Knight, 2013) which handles non-atomic operators. The document-level EL-smatch score between all annotated sentence pairs was 0.70. When we restricted the analysis to just annotations that were marked *certain*, the agreement rose to 0.78. The complete pairwise scores are shown in Table 2. Notice that annotators 1, 2, and 3 had very high agreement with each other. If we restrict the agreement to just those three annotators, the full and certain-subset scores are 0.79 and 0.88, respectively. Out of all the annotations, less than a third were marked as uncertain or incomplete. AMR annotations reportedly have annotator vs consensus IA of 0.83 for newswire and 0.79 for web text (Tsialos, 2015).

Table 1: Current sentence annotation counts broken down by dataset and certainty. DG and PG are the Discourse Graphbank and Project Gutenberg, respectively. The *Old* column annotations are from before we added the certainty feature.

	Cert.	*Unc.*	*Inc.*	*Old*	*All*
Tatoeba	533	66	24	396	1019
DG	102	37	4	0	143
UIUC QC	179	50	0	0	229
PG	113	59	17	0	189
Total	**927**	212	45	396	1580

This study also demonstrates that the certainty marking indeed reflects the quality of the annotation, thus performing the role we intended. Also, based on the high agreement between annotators 1, 2, and 3, we can conclude that consistent ULF annotations across multiple annotators is possible. However, the lower scores of annotators 4 and 5, even in annotations marked as certain, indicates room for improvement in the annotation guidelines and training of some annotators.

Table 2: Pairwise IA scores, where the left score is over all annotations and the right score is only over annotations marked as certain.

	2	3	4	5
1	0.80/0.88	0.79/0.89	0.69/0.77	0.63/0.75
2	-	0.77/0.86	0.72/0.77	0.62/0.75
3	-	-	0.69/0.75	0.63/0.73
4	-	-	-	0.62/0.71

We have so far collected 927 certain annotations and have 1,580 in total. The full annotation breakdown is in Table 1. We started with the English portion of the Tatoeba dataset (`https://tatoeba.org/eng/`), a crowd-sourced translation dataset. This source tends to have shorter sentences, but they are more varied in topic and form. We then added text from Project Gutenberg (`http://gutenberg.org`), the UIUC Question Classification dataset (Li and Roth, 2002), and the Discourse Graphbank (Wolf, 2005). Preliminary parsing experiments on a small dataset (900 sentences) show promising results and we expect to be able to build an accurate parser with a moderately-sized dataset and representation-specific engineering (Kim, 2019).

7 Related Work

A notable development in general representations of semantic content has been the design of AMR (Banarescu et al., 2013) followed by numerous research studies on generating AMR from English and on using it for downstream tasks. AMR is intended as a kind of intuitive normal form for the relational context of English sentences in order to assist in machine translation. Given this goal, AMR deliberately neglected issues such as articles, tense, the distinction between real and hypothetical entities, and non-intersective modification. In the context of inference, this risks making false conclusions such as that a *"big ant"* is bigger than a *"small elephant"*.

Still, this development was an inspiration to us in terms of both the quest for broad coverage and methods of learning and evaluating semantic parsers. There has also been much activity in developing semantic parsers that derive logical representations, raising the possibility of making inferences with those representations (Artzi et al., 2015; Artzi and Zettlemoyer, 2013; Howard et al., 2014; Kate and

Mooney, 2006; Konstas et al., 2017; Kwiatkowski et al., 2011; Liang et al., 2011; Poon, 2013; Popescu et al., 2004; Tellex et al., 2011). The techniques and formalisms employed are interesting (e.g., learning of CCG grammars that generate λ-calculus expressions), but the targeted tasks have generally been question-answering in domains consisting of numerous monadic and dyadic ground facts ("triples"), or simple robotic or human action descriptions.[6]

Noteworthy examples of formal logic-based approaches, not targeting specific applications are Bos' (2008) and Draiccio et al.'s (2013), whose hand-built semantic parsers respectively generate FOL formulas and OWL-DL expressions. But these representations preclude generalized quantifiers, modification, reification, attitudes, etc. Manshadi and Allen (2012) presented an intuitive graphical representation, like AMR, but allowing for modals, generalized quantifiers, etc., and not attempting to canonicalize meanings in the way AMR does. The difference from ULF is that it focuses on binary structural relations such as *restrictor*, *body*, or *modifier* between semantic components, rather than operator-operand type structure. It is not directly intended for inference, but readily lends itself to incremental disambiguation. We are not aware of any work on inference generation of the type ULFs targets, based on these projects.

A couple of yet-unmentioned but notable semantic annotation projects are the Groningen Meaning Bank (Bos et al., 2017), with discourse representation structure (DRS) annotations (Kamp, 1981) and the Redwoods treebank (Flickinger et al., 2012; Oepen et al., 2002) with Minimal Recursion Semantics (MRS) (Copestake et al., 2005) annotations. DRSs have the same representational limitations as Bos' (2008) system. MRS is descriptively powerful and linguistically motivated, with significant resources including a hand-built grammar, multiple parsers, and a large annotated dataset (Bub et al., 1997; Callmeier, 2001). Given that MRS and Manshadi and Allen's graphical representation are object-language agnostic, meta-level semantic representations, inference systems cannot be built directly for them based on model-theoretic notions of interpretation, truth, satisfaction, and entailment. However, the lack of an object-language leaves open the possibility of forming a correspondence between these representations and ULF that fully respects both formalisms. Finally, the use of unscoped LFs in a rule-to-rule framework was first introduced by Schubert and Pelletier (1982) and a similar approach to scope ambiguity was taken by the Core Language Engine (Alshawi and van Eijck, 1989).

8 Conclusion & Future Work

ULF, the underspecified initial representation for EL described in this document, captures a subset of the semantic information of EL that allows it to be annotated reliably, participate in the complete resolution to EL, and form the basis for structural inferences that are important for language understanding tasks. We will continue this work by expanding the corpus of ULF annotations and training a statistical parser over that corpus. Automatic ULF parses could then be used as the backbone for a complete EL parser or as the core representation for NLP tasks that require sentence-level formal semantic information or structural inferences.

9 Acknowledgements

We would like to thank Burkay Donderici, Benjamin Kane, Lane Lawley, Tianyi Ma, Graeme McGuire, Muskaan Mendriatta, Akihiro Minami, Georgiy Platonov, Sophie Sackstein, and Siddharth Vashishta for raising thoughtful questions in the development of this work. We are grateful to the anonymous reviewers for their helpful feedback. This work was supported by DARPA CwC subcontract W911NF-15-1-0542.

[6]For example, Ross et al. (2018) develop a CCG-based semantic parser for action annotations in videos, representing sentences in an approximate way—neglecting determiners and treating all entity references as variables.

References

Alshawi, H. and J. van Eijck (1989, June). Logical forms in the core language engine. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, Vancouver, British Columbia, Canada, pp. 25–32. Association for Computational Linguistics.

Artzi, Y., K. Lee, and L. Zettlemoyer (2015, September). Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, pp. 1699–1710. Association for Computational Linguistics.

Artzi, Y. and L. Zettlemoyer (2013). Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics 1*(1), 49–62.

Banarescu, L., C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider (2013, August). Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, Sofia, Bulgaria, pp. 178–186. Association for Computational Linguistics.

Barwise, J. and R. Cooper (1981). Generalized quantifiers and natural language. In *Philosophy, language, and artificial intelligence*, pp. 241–301. Springer.

Bos, J. (2008). Wide-coverage semantic analysis with Boxer. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, STEP '08, Stroudsburg, PA, USA, pp. 277–286. Association for Computational Linguistics.

Bos, J., V. Basile, K. Evang, N. Venhuizen, and J. Bjerva (2017). The Groningen Meaning Bank. In N. Ide and J. Pustejovsky (Eds.), *Handbook of Linguistic Annotation*, Volume 2, pp. 463–496. Springer.

Bub, T., W. Wahlster, and A. Waibel (1997, Apr). Verbmobil: the combination of deep and shallow processing for spontaneous speech translation. In *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Volume 1, pp. 71–74 vol.1.

Cai, S. and K. Knight (2013, August). Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Sofia, Bulgaria, pp. 748–752. Association for Computational Linguistics.

Callmeier, U. (2001). Efficient parsing with large-scale unification grammars. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany.

Copestake, A., D. Flickinger, C. Pollard, and I. A. Sag (2005). Minimal Recursion Semantics: An introduction. *Research on Language and Computation 3*(2), 281–332.

Davidson, D. (1967). The logical form of action sentences. In N. Rescher (Ed.), *The Logic of Decision and Action*. University of Pittsburgh Press.

Draicchio, F., A. Gangemi, V. Presutti, and A. Nuzzolese (2013). FRED: From natural language text to rdf and owl in one click. In *P. Cimiano et al. (eds.) , ESWC 2013*, pp. 263–267. Springer.

Flickinger, D., Y. Zhang, and V. Kordoni (2012). DeepBank: A dynamically annotated treebank of the wall street journal. In *Proceedings of the Eleventh International Workshop on Treebanks and Linguistic Theories*, pp. 85–96. EdiÃğÃţes Colibri.

Hermjakob, U. (2013). AMR Editor: A tool to build abstract meaning representations.

Howard, T. M., S. Tellex, and N. Roy (2014). A natural language planner interface for mobile manipulators. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6652–6659.

Hurum, S. and L. Schubert (1986, May). Two types of quantifier scoping. In *Proc. 6th Can. Conf. on Artificial Intelligence (AI-86)*, Montreal, Canada, pp. 19–43.

Hwang, C. and L. Schubert (1993). Episodic Logic: A situational logic for natural language processing. In P. Aczel, D. Israel, Y. Katagiri, and S. Peters (Eds.), *Situation Theory and its Applications 3 (STA-3)*, pp. 307–452. CSLI.

Hwang, C. H. (1992). *A logical approach to narrative understanding*. Ph. D. thesis, University of Alberta.

Hwang, C. H. and L. K. Schubert (1994). Interpreting tense, aspect and time adverbials: A compositional, unified approach. In *Proceedings of the First International Conference on Temporal Logic*, ICTL '94, London, UK, pp. 238–264. Springer-Verlag.

Kamp, H. (1981). A theory of truth and semantic representation. In J. A. G. Groenendijk, T. M. V. Janssen, and M. B. J. Stokhof (Eds.), *Formal Methods in the Study of Language*, Volume 1, pp. 277–322. Amsterdam: Mathematisch Centrum.

Kate, R. J. and R. J. Mooney (2006, July). Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia, pp. 913–920. Association for Computational Linguistics.

Kim, G. and L. Schubert (2016, August). High-fidelity lexical axiom construction from verb glosses. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, Berlin, Germany, pp. 34–44. Association for Computational Linguistics.

Kim, G. and L. Schubert (2017, April). Intension, attitude, and tense annotation in a high-fidelity semantic representation. In *Proceedings of the Workshop Computational Semantics Beyond Events and Roles*, Valencia, Spain, pp. 10–15. Association for Computational Linguistics.

Kim, G. L. (2019). Towards parsing unscoped episodic logical forms with a cache transition parser. In *the Poster Abstracts of the Proceedings of the 32nd International Conference of the Florida Artificial Intelligence Research Society*.

Konstas, I., S. Iyer, M. Yatskar, Y. Choi, and L. Zettlemoyer (2017, July). Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada, pp. 146–157. Association for Computational Linguistics.

Kwiatkowski, T., L. Zettlemoyer, S. Goldwater, and M. Steedman (2011, July). Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, UK., pp. 1512–1523. Association for Computational Linguistics.

Li, X. and D. Roth (2002). Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, Stroudsburg, PA, USA, pp. 1–7. Association for Computational Linguistics.

Liang, P., M. Jordan, and D. Klein (2011, June). Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA, pp. 590–599. Association for Computational Linguistics.

Manshadi, M. and J. Allen (2012, May). A universal representation for shallow and deep semantics. In *Joint ISA-7 Workshop on Interoperable Semantic Annotation SRSL-3 Workshop on Semantic Representation for Spoken Language I2MRT Workshop on Multimodal Resources and Tools*, pp. 52.

Manshadi, M., D. Gildea, and J. Allen (2013, August). Plurality, negation, and quantification:towards comprehensive quantifier scope disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Sofia, Bulgaria, pp. 64–72. Association for Computational Linguistics.

Montague, R. (1973). The proper treatment of quantification in ordinary English. In K. J. J. Hintikka, J. Moravcsic, and P. Suppes (Eds.), *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics*, pp. 221–242. Dordrecht: Reidel.

Morbini, F. and L. Schubert (2009, June). Evaluation of Epilog: A reasoner for Episodic Logic. In *Proceedings of the Ninth International Symposium on Logical Formalizations of Commonsense Reasoning*, Toronto, Canada.

Morbini, F. and L. Schubert (2011, January). Metareasoning as an Integral Part of Commonsense and Autocognitive Reasoning. In M. T. Cox and A. Raja (Eds.), *Metareasoning: Thinking about thinking*. MIT Press.

Oepen, S., K. Toutanova, S. Shieber, C. Manning, D. Flickinger, and T. Brants (2002). The LinGo Redwoods Treebank: Motivation and preliminary applications. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 2*, COLING '02, Stroudsburg, PA, USA, pp. 1–5. Association for Computational Linguistics.

Poon, H. (2013, August). Grounded unsupervised semantic parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Sofia, Bulgaria, pp. 933–943. Association for Computational Linguistics.

Popescu, A.-M., A. Armanasu, O. Etzioni, D. Ko, and A. Yates (2004). Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ross, C., A. Barbu, Y. Berzak, B. Myanganbayar, and B. Katz (2018, October 31 - November 4). Grounding language acquisition by training semantic parsers using captioned videos. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Brussels, Belgium, pp. 2647–2656.

Ruys, E. and Y. Winter (2010). Quantifier scope in formal linguistics. In D. M. Gabbay and F. Guenthner (Eds.), *Handbook of Philosophical Logic*, pp. 159–225. Springer, Dortrecht.

Sánchez Valencia, V. (1995). Natural logic: parsing driven inference. *Linguistic Analysis 25*, 258–285.

Schubert, L. (2013). NLog-like inference and commonsense reasoning. In A. Zaenen, V. de Paiva, and C. Condoravdi (Eds.), *Perspectives on Semantic Representations for Textual Inference, special issue of Linguistic Issues in Language Technology (LiLT 9)*, Volume 9, pp. 1–26.

Schubert, L. (2014, June). From treebank parses to Episodic Logic and commonsense inference. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, Baltimore, MD, pp. 55–60. Association for Computational Linguistics.

Schubert, L. and F. Pelletier (1982). From English to logic: Context-free computation of 'conventional' logical translations. *Am. J. of Computational Linguistics 8 [now Computational Linguistics] 8*, 26–44.

Schubert, L. K. (2000). The situations we talk about. In J. Minker (Ed.), *Logic-based Artificial Intelligence*, pp. 407–439. Norwell, MA, USA: Kluwer Academic Publishers.

Schubert, L. K. and C. H. Hwang (2000). Episodic Logic meets Little Red Riding Hood: A comprehensive natural representation for language understanding. In L. M. Iwańska and S. C. Shapiro (Eds.), *Natural Language Processing and Knowledge Representation*, pp. 111–174. Cambridge, MA, USA: MIT Press.

Stratos, K., L. K. Schubert, and J. Gordon (2011). Episodic Logic: Natural Logic + reasoning. In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development (KEOD)*.

Tellex, S., T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy (2011). Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI Conference on Artificial Intelligence*.

Tsialos, A. (2015, March). Abstract meaning representation for sembanking. Available at `www.inf.ed.ac.uk/teaching/courses/tnlp/2014/Aristeidis.pdf`, accessed December 8, 2018.

Wolf, F. (2005). *Coherence in natural language : data structures and applications*. Ph. D. thesis, Massachusetts Institute of Technology, Dept. of Brain and Cognitive Sciences.

A Quantifier Semantics

Noun phrases can occur in any position here an individual variable or constant can occur, and in post-processing are replaced by bound variables. Therefore the *positional* types of noun phrases are individuals, $\mathcal{D}$. Therefore, we can treat determiners such as `every.d` in ULF as if they were of type $(\mathcal{N} \to \mathcal{D}$, i.e. a function from a predicate to an individual. For example consider the ULF formula `((every.d dog.n) (pres run.v))`. `(every.d dog.n)` seems to be able to occur in any place that `|John|` and `they.pro` can occur.

```
((every.d dog.n) (pres run.v)),  (i.pro ((pres like.v) (every.d dog.n))),
        (|John| (pres run.v)),  (i.pro ((pres like.v) |John|)),
      (they.pro (pres run.v));  (i.pro ((pres like.v) they.pro));
```

Semantically we consider `they.pro` and `them.pro` to be the same, as they only differ in syntactic position. Then since `dog.n` (and any other argument of a determiner) is a monadic predicate, we can infer that the *positional* type of determiners is $\mathcal{N} \to \mathbf{2}$. This will be transformed after scoping into a formula of the form $(\delta v : \phi\ \psi)$, where δ is the determiner, and ϕ and ψ correspond to the formulas resulting from substituting the scoped variable into the restrictor and matrix predicates, respectively. These formulas are interpreted in EL via satisfaction conditions over the quantified variable and two formulas (a restrictor formula and the nuclear scope), e.g., for an sentence such as *"Most car crashes are due to driver error"*,

$(\text{most } v : \phi\ \psi)^{\mathcal{M}\,\mathcal{U}} = 1$ iff

for most $d \in \mathcal{D}$ for which $\phi^{\mathcal{M}\,\mathcal{U}_{v:d}} = 1$, $\psi^{\mathcal{M}\,\mathcal{U}_{v:d}} = 1$

where $\mathcal{M}$is the model, $\mathcal{U}$is the variable assignment function, and $\mathcal{U}_{v:d}$ is the same as $\mathcal{U}$ except that its value for variable v is d. When this formula is evaluated with respect to an episode, it corresponds to a formula of the form

$[(\text{several } v : \phi\ \psi) **\ \eta]$,

where '$**$' is the operator relating a sentence to the episode it *characterizes* (describes as a whole), which is discussed in Section 2. $(\delta v : \phi\ \psi)$ can equivalently be rewritten as $(\delta\ (\lambda v\ \phi)\ (\lambda v\ \psi))$ and we can define δ as a second-order intensional predicate of type $\mathcal{N} \to \mathcal{N} \to \mathcal{S} \to \mathbf{2}$ similar to the approach used in generalized quantifier theory (Barwise and Cooper, 1981).

B Episodic Operators

'$**$', '$*$', and '@' are *episodic* operators, which relate formulas to episode variables in Episodic Logic. They do not appear in ULFs since ULFs do not have explicit episode variables. However, these operators

are foundational to Episodic Logic semantics in handling event structure and intensional semantics. All formulas in EL must be evaluated with respect to one of these operators to obtain a truth value since sentence intensions in EL have the type $\mathcal{S} \rightarrow \mathbf{2}$.

- '**' - the *characterizing* operator

  '**' relates an episode variable to a formula that *characterizes* it. In other word, the formula describes the episode as a whole, or the nature of the episode, rather than a tangential part or a temporal segment of it. This, however, does not mean that the characterizing formula must describe *every* detail of the episode. It can in fact be quite abstract. For instance, *"John had a car accident"* and *"John hit some black ice and his car skidded into a tree"* might characterize the same event. As such, for most news stories the headline and the first sentence of the article are likely to both characterize the same event even though the headline is much shorter. Formally,

  $$[\phi \ast\ast \eta]^{\mathcal{M}\mathcal{U}} = 1 \text{ iff } \phi^{\mathcal{M}\mathcal{U}}(\eta^{\mathcal{M}\mathcal{U}}) = 1;$$

  $$[(\text{not } \phi) \ast\ast \eta]^{\mathcal{M}\mathcal{U}} = 1 \text{ iff } \phi^{\mathcal{M}\mathcal{U}}(\eta^{\mathcal{M}\mathcal{U}}) = 0.$$

  The semantic type of ϕ is $\mathcal{S} \rightarrow \mathbf{2}$ (a sentence intension) and the semantic type of η is $\mathcal{S}$, a situation. Therefore, η characterizes ϕ just in the case that the interpretation of ϕ with respect to the model $\mathcal{M}$ and variable assignment function $\mathcal{U}$ evaluated over the interpretation of η with respect to $\mathcal{M}$ and $\mathcal{U}$ is true.

- '*' - the *truth* operator

  '*' relates an episode variable to a formula that is *true* in that episode. This is a weaker operator than '**' in that a formula that is '*'-related can be a just a segment or an incidental aspect of the episode to be true. Therefore, $[\phi \ast\ast \eta]$ entails $[\phi \ast \eta]$, but not the other way. Therefore, *"There was black ice on the road"* and *"John was driving"* could both be '*'-related to the episode characterized by the example given in for the '**' operator. Formally,

  $$[\phi \ast \eta]^{\mathcal{M}\mathcal{U}} = 1 \text{ iff there is an episode } s \sqsubseteq \eta^{\mathcal{M}\mathcal{U}} \text{ such that } \phi^{\mathcal{M}\mathcal{U}}(s) = 1.$$

  Where $\sqsubseteq$ is an episode part-of relation. It's formal definition is given by Hwang and Schubert (1993). Intuitively we can think of $s \sqsubseteq \eta$ to mean that s is a subepisode of η.

- '@' - the *concurrent* operator

  '@' relates an episode variable to a formula characterizes another episode that runs concurrent with it. So this operator can be rewritten in the following way. $[\phi \ @ \ \eta]$ entails and is entailed by (some e : $[e$ same-time $\eta]$ $[\phi \ast\ast e]$). Formally, @ us defined as

  $$[\phi \ @ \ \eta]^{\mathcal{M}\mathcal{U}} = 1 \text{ iff there is an episode } s \in \mathcal{S} \text{ with } \mathit{time}(s) = \mathit{time}(\eta^{\mathcal{M}\mathcal{U}}) \text{ such that } \phi^{\mathcal{M}\mathcal{U}}(s) = 1.$$

C More About Macros

ULF macros are different syntactic rewriting operators to reduce the annotator burden of encoding complex, but regular, semantic structures or avoid unnecessary word reordering. Table 3 lists the definitions and simple examples of the basic ULF macros. The `sub` macro is the *substitution* macro which performs a simple substitution of its first argument into the position of `*h` within the second argument. This is used for topicalization, such as *"Swiftly, the fox ran away"*, which topicalizes *"Swiftly"* from the sentence *"The fox swiftly ran away"*. The `rep` macro is the *replace* operator and the exact same as `sub` with the arguments swapped and using `*p` instead of `*h` as the placeholder variable. This is used for rightward-displaced clauses, such as, *"A man answered the door with a white beard"*, in which *with a white beard* is really displaced from the expected post-nominal position, i.e *"A man with a white beard ..."*.

Table 3: List of basic rewriting macros in ULF. $=_m$ is the macro defining operator.

Name	*Definitions*	*Example*
sub	(sub C S[*h]) $=_m$ S[*h←C]	(sub A (B *h)) $=_m$ (B A)
rep	(rep S[*p] C) $=_m$ S[*p←C]	(rep (A *p) B) $=_m$ (A B)
n+preds	(n+preds N P_1 ... P_n) $=_m$ (λx ((x N) and.cc (x P_1) ... (x P_n)))	(n+preds dog.n red.a) $=_m$ (λx ((x dog.n) and.cc (x red.a)))
np+preds	(np+preds NP P_1 ... P_n) $=_m$ (the.d (λx ((x = NP) and.cc (x P_1) ... (x P_n))))	(np+preds he.pro red.a) $=_m$ (the.d (λx ((x = he.pro) and.cc (x red.a))))
's	((NP 's) N) $=_m$ (the.d ((poss-by NP) N))	((|John| 's) dog.n) $=_m$ (the.d ((poss-by |John|) dog.n))

Next, n+preds and np+preds are macros for handling post-nominal modification. n+preds modifies a noun and returns a noun, whereas np+preds modifies an entity and returns a modified entity. Intuitively, np+preds handles non-restrictive modifiers, whereas n+preds handles restrictive modifiers. This makes sense since the modifying predicates in n+preds are added before the determiner, thus introduced into the restrictor of the quantification.

The 's macro is for handling possession using an appended marker to the possessor just as is done in English (e.g. "John's dog"). Formally, this maps to a pre-modifying possession relation. So *"John's dog"* is hand-annotated as ((|John| 's) dog.n), which expands out to (the.d ((poss-by |John|) dog.n)). poss-by is a binary predicate relating two entities, semantic type $\mathcal{D} \rightarrow (\mathcal{D} \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))$. so (poss-by |John|) resolves to semantic type of a predicate, $\mathcal{N}$. Notice that this is a predicate-noun pair so as discussed in Section 4 the mod-n type-shifter is automatically introduced, resulting in (the.d ((mod-n (poss-by |John|)) dog.n)).

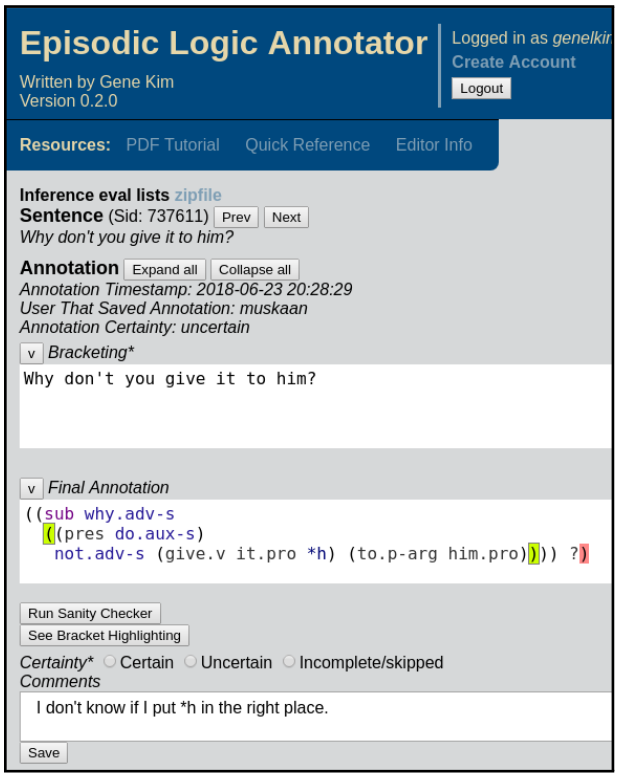

Figure 3: Current ULF annotator state with example annotation process.

D Additional Annotator Info

Here we reiterate the annotator features as described in Section 5 with reference to an image of it in Figure 3.

1. **Syntax and bracket highlighting.** Highlights the cursor location and the closing bracket, unmatched brackets and quotes, operator keywords, and badly placed operators. The "Final Annotation" window in Figure 3 shows the cursor matching bracket in yellow-green highlighting, an unmatched bracket in red, the sub macro in purple, and sentence-level operators in blue.

2. **Sanity checker.** Alerts the annotator to invalid type compositions and suggests corrections for common mistakes.

3. **Certainty marking.** Annotators can mark whether they are certain of an annotation's correctness so that partial progress can be made while preserving the integrity of the corpus. The bottom of

Figure 3 shows radio buttons for selecting the certainty of the annotation.

4. **Sentence-specific comments.** Annotators can record their thoughts on partially complete annotations so that others can pick up where they left off. The bottom-most window in view in Figure 3 is the sentence-specific comment window. These comments are viewable by all annotators when accessing this sentence.

E Additional Grounding Examples

Here are a couple of additional sections that ground the high-level ULF background in concrete examples.

E.1 More Resources on Predicate Modifiers

A type of modification not covered in the main document is entity-predicate modification. The type shifter from an individual to a nominal predicate modifier is named `nnp` and has semantic type, $\mathcal{D} \rightarrow (\mathcal{N}_N \rightarrow \mathcal{N}_N)$. It is for indicating premodification of a common noun by a proper noun; e.g.,
`((nnp |Seattle|) skyline.n)`.
All of the operators discussed in Section 4 and here are listed alongside a ULF example, and its semantic type in Table 4.

Table 4: Predicate and sentence modifier forming operators in ULF along with examples and their semantic types.

Name	*Example*	*Semantic Type*
`mod-a`	`((mod-a worldly.a) wise.a)`	$\mathcal{N} \rightarrow (\mathcal{N}_{ADJ} \rightarrow \mathcal{N}_{ADJ})$
`mod-n`	`((mod-n (very.mod-n happy.a)) dog.n)`	$\mathcal{N} \rightarrow (\mathcal{N}_N \rightarrow \mathcal{N}_N)$
`adv-a`	`(play.v (adv-a (with.p (a.d dog.n))))`	$\mathcal{N} \rightarrow (\mathcal{N}_V \rightarrow \mathcal{N}_V)$
`nnp`	`((nnp |Seattle|) skyline.n)`	$\mathcal{D} \rightarrow (\mathcal{N}_N \rightarrow \mathcal{N}_N)$
`adv-s`	`(show_up.v (adv-s (to.p (my.d surprise.n))))`	$(\mathcal{S} \rightarrow \mathbf{2}) \rightarrow (\mathcal{S} \rightarrow \mathbf{2})$
`adv-e`	`(eat.v (adv-e (at.p (a.d cafe.n))))`	$(\mathcal{S} \rightarrow \mathbf{2}) \rightarrow (\mathcal{S} \rightarrow \mathbf{2})$
`adv-f`	`(run.v (adv-f (very.mod-a often.a)))`	$(\mathcal{S} \rightarrow \mathbf{2}) \rightarrow (\mathcal{S} \rightarrow \mathbf{2})$

Ultimately in EL, `adv-a`, `adv-e`, and `adv-f` will be reconstrued as predications over actions and events via meaning postulate inferences. Agent-episode pairs that intuitively represent actions, experiences, or attributes are distinct from events. For example, *"He fell painfully"* refers to a painful experience rather than to a painful event and *"He excels intellectually"* refers an intellectual attribute rather than to an intellectual event or situation. `.adv-a` type modifiers constrain both the agent and the episode in the pair. No sharp or exhaustive classification of such pairs into actions, experiences, and attributes is presupposed by this – the point is just to make available the subject of sentences in working out entailments of VP-modification. Since actions are formed by pairing an agent with an event variable, they are not explicitly represented in ULF. The meaning postulate inferences on `.adv-a` type modifiers would infer from `(he.pro (play.v (adv-a (with.p (a.d dog.n)))))` the following deindexed ULF `[[[he.pro play.v] ** E1.sk] and.cc [(pair he.pro E1.sk) (with.p (a.d dog.n))]]`. The meaning postulate inference of `.adv-e` type modifiers to predications over events is also straightforward. The ULF formula `(she.pro (eat.v (adv-e (at.p (a.d cafe.n)))))` leads to the deindexed, inferenced formula `[[[she.pro eat.v] ** E1.sk] and.cc [(pair she.pro E1.sk) (at.p (a.d cafe.n))]]`.

E.2 Topicalization & Relative Clauses in ULF

The `sub` macro was introduced to reduce the amount of lexical reordering by annotators when annotating sentences with syntactic movement such as topicalization. `sub` takes two constituents, the second of which must contain the symbol `*h`. When the operator is evaluated the first argument is inserted into the

position of *h in the second argument. *"Swiftly, the fox ran away"* for example would be annotated as (in raw ULF form)

(sub swiftly.adv-a ((the.d fox.n) ((past run.v) away.adv-a *h)))

and when the sub macro is evaluated, becomes

((the.d fox.n) ((past run.v) away.adv-a swiftly.adv-a)).

For relative clauses we introduce one extra post-processed element which is the relativizer, annotated with a .rel extension. *"The coffee that you drank"* is annotated in raw ULF with macros as

(the.d (n+preds coffee.n (sub that.rel (you.pro ((past drink.v) *h)))))

During post-processing, the embedded sentence in which the .rel variable lies is λ-abstracted and the lambda variable replaces the .rel variable. Post-processing that.rel leads to

(the.d (n+preds coffee.n (λx (sub x (you.pro ((past drink.v) *h))))))

Now if we evaluate both n+preds and sub, and perform one lambda reduction we get

(the.d (λy ((y coffee.n) and.cc (you.pro ((past drink.v) y)))))

which is exactly the meaning that is expected that is expected from the relative clause. That is, *"The coffee that you drank"* is a coffee ((y coffee.n)) and is something that you drank ((you.pro ((past drink.v) y))).