# Corpus Annotation, Parsing, and Inference for Episodic Logic Type Structure

by

Gene Louis Kim

Submitted in Partial Fulfillment of the

Requirements for the Degree

Doctor of Philosophy

Supervised by

Professor Lenhart K. Schubert

Department of Computer Science

Arts, Sciences and Engineering

Edmund A. Hajim School of Engineering and Applied Sciences

University of Rochester

Rochester, New York

2022

*To Kate*

# Table of Contents

**Biographical Sketch** vi

**Acknowledgments** viii

**Abstract** ix

**Contributors and Funding Sources** x

**List of Tables** xi

**List of Figures** xiv

**1 Introduction** 1

1.1 Chapter Descriptions . . . . . 5

1.2 Related Work . . . . . 6

1.3 Episodic Logic (EL) . . . . . 23

**2 Unscoped Logical Form (ULF)** 29

2.1 ULF Design . . . . . 29

2.2 Measuring Similarity . . . . . 58

**3 Annotation** **62**

3.1 Annotation Development & Interface . . . . . 62

3.2 Interannotator Agreement . . . . . 68

3.3 Annotation Release . . . . . 69

**4 Parsing** **73**

4.1 Background . . . . . 73

4.2 Transition System . . . . . 76

4.3 Model . . . . . 82

4.4 Parsing . . . . . 85

4.5 Experiments . . . . . 87

4.6 Conclusion . . . . . 91

**5 Inference** **92**

5.1 Generating Discourse Inferences from ULFs . . . . . 94

5.2 A Theoretical Framework for Natural Logic Inference with ULFs . . . . . 107

5.3 An Implementation of Natural Logic Inference with ULFs . . . . . 128

**6 Conclusion** **143**

**References** **145**

**Appendix A High-Fidelity Lexical Axiom Construction from Verb Glosses** **164**

A.1 Introduction . . . . . 164

A.2 Related Work . . . . . 165

A.3 Background . . . . . 168

A.4 Gloss Axiomatization . . . . . 170
A.5 Experiments . . . . . 177
A.6 Future Work and Conclusions . . . . . 180

**Appendix B Traditional Aristotelian Syllogisms in ULF** **182**

B.1 Scoping and Polarity Derivations . . . . . 182
B.2 Deriving the Syllogisms . . . . . 184

**Appendix C Transition Parser Appendices** **186**

C.1 Full ULF Alignment Details . . . . . 186
C.2 RoBERTa Handling Details . . . . . 187
C.3 Full Tables . . . . . 188
C.4 Parse Examples . . . . . 191

# Biographical Sketch

Gene Louis Kim was born in Gwangju, South Korea in 1993. He received his B.S. in Computer Science at the University of Washington in 2015 and began his doctoral studies in Computer Science the same year at the University of Rochester advised by Professor Lenhart Schubert. He earned his M.S. in Computer Science in 2017. He was awarded the Robert L. and Mary L. Sproull Fellowship in 2015 and was invited to the Heidelberg Laureate Forum as a young researcher in 2018.

The following publications were the result of the author's doctoral studies.

- **Gene Kim** and Lenhart Schubert. 2016. High-fidelity lexical axiom construction from verb glosses. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 34–44, Berlin, Germany. Association for Computational Linguistics
- **Gene Kim** and Lenhart Schubert. 2017. Intension, attitude, and tense annotation in a high-fidelity semantic representation. In *Proceedings of the Workshop Computational Semantics Beyond Events and Roles*, pages 10–15, Valencia, Spain. Association for Computational Linguistics
- **Gene Louis Kim** and Lenhart Schubert. 2019. A type-coherent, expressive representation as an initial step to language understanding. In *Proceedings of the 13th International Conference on Computational Semantics - Long Papers*, pages 13–30, Gothenburg, Sweden. Association for Computational Linguistics

- Lane Lawley, **Gene Louis Kim**, and Lenhart Schubert. 2019. Towards natural language story understanding with rich logical schemas. In *Proceedings of the Sixth Workshop on Natural Language and Computer Science*, pages 11–22, Gothenburg, Sweden. Association for Computational Linguistics

- **Gene Kim**, Benjamin Kane, Viet Duong, Muskaan Mendiratta, Graeme McGuire, Sophie Sackstein, Georgiy Platonov, and Lenhart Schubert. 2019. Generating discourse inferences from unscoped episodic logical formulas. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 56–65, Florence, Italy. Association for Computational Linguistics

- **Gene Louis Kim**, Mandar Juvekar, and Lenhart Schubert. 2020. Monotonic inference for underspecified episodic logic. In *Proceedings of the Workshop Natural Logic Meets Machine Learning*. Association for Computational Linguistics

- **Gene Louis Kim**, Mandar Juvekar, Junis Ekmekciu, Viet Duong, and Lenhart Schubert. 2021b. A (mostly) symbolic system for monotonic inference with unscoped episodic logical forms. In *Proceedings of the 1st and 2nd Workshops on Natural Logic Meets Machine Learning (NALOMA)*, pages 71–80, Groningen, the Netherlands (online). Association for Computational Linguistics

- **Gene Louis Kim**, Viet Duong, Xin Lu, and Lenhart Schubert. 2021a. A transition-based parser for unscoped episodic logical forms. In *Proceedings of the 14th International Conference on Computational Semantics (IWCS)*, pages 184–201, Groningen, The Netherlands (online). Association for Computational Linguistics

# Acknowledgments

I thank all of my collaborators for all that they have given me in both academic and personal terms. I am especially grateful to my advisor, Lenhart Schubert, for far more time, knowledge, and patience than I could have asked for, as well as to my partner, Keito Omokawa, whose support made this accomplishment possible. Finally, I would like to thank everyone in my family for helping me become who I am today and get to this point.

# Abstract

A growing interest in moving beyond lesser goals in the NLP community and moving to language understanding has led to the search for a semantic representation which fulfills its nuanced modeling and inferential needs. Current approaches to natural language understanding rely on neural networks with opaque internal representation of meaning and unpredictable failures or formal representations which fail to capture the full range of semantic capacity in natural language.

In this dissertation, I discuss the design, annotation, and use of Unscoped Logical Forms (ULFs) of Episodic Logic for the goal of building a system that can understand human language. ULF is designed to balance an unprecedented level of formal semantic expressivity with ease of annotation for corpus creation, derivability from English, and inferential capacity. The design of ULF retains a connection to surface syntax that enables sufficiently paced data collection despite the rich semantic information encoded in the type structure. By leveraging the systematic syntactic and semantic underpinnings of ULF we outperform extant state-of-the-art semantic parsers and overcome the limitations of modern data-hungry techniques on a more modestly-sized dataset. We also show how ULF supports important classes of discourse and natural logic inferences based on its retention of syntactic information while preserving semantic type coherence.

# Contributors and Funding Sources

This work was supervised by a dissertation committee consisting of Professors Lenhart Schubert, Daniel Gildea, James Allen, Aaron Steven White, and Scott Grimm. Apart from the collaborations listed below, the remainder of the work in this thesis was completed independently.

Most of Section 2.1 in Chapter 2 was originally published as Kim and Schubert (2019).

Chapter 4, Section 3.2, and Section 3.3 were originally published as Kim et al. (2021a); the co-authors are Viet Duong, Xin Lu, and Lenhart Schubert.

Section 5.1 in Chapter 5 was originally published as Kim et al. (2019); the co-authors are Benjamin Kane, Viet Duong, Muskaan Mendiratta, Graeme McGuire, Sophie Sackstein, Georgiy Platonov, and Lenhart Schubert.

Section 5.2 in Chapter 5 was originally published as Kim et al. (2020); the co-authors are Mandar Juvekar and Lenhart Schubert.

Section 5.3 in Chapter 5 was originally published as Kim et al. (2021b); the co-authors are Mandar Juvekar, Junis Ekmekciu, Viet Duong, and Lenhart Schubert.

Appendix A was originally published as Kim and Schubert (2016) and Fig. 1.6 was originally published in the supplementary document for Kim and Schubert (2016).

This material is based upon work supported by the NSF EAGER grant NSF IIS-1908595 and DARPA CwC subcontract W911NF-15-1-0542. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of above named organizations.

# List of Tables

1.1 ERG results for the second most recent Redwoods release (the Eighth Growth). Only a selection of segments (corpora) are included in this table, but the total results includes all sentences in the growth. . . . . 18

2.1 Predicate and sentence modifier forming operators in ULF along with examples and their semantic (positional) types. . . . . 50

2.2 A list of common rewriting macros in ULF with their expansion definitions and an example. $=_m$ is the macro defining operator and $\Rightarrow_m$ is a macro application. . . . . 55

3.1 Average timing of experimental ULF annotations. . . . . 64

3.2 Pairwise IAA scores, where the left score in each cell is over all annotations and the right score in each cell is only over annotations marked as *certain*. . . . . 69

4.1 Comparison to AMR parsers. . . . . 89

4.2 Statistics of model performances with constraints added—the average of 5 runs. . . 90

5.1 Statistics for each dataset on the result of the filtering process described in Section 5.1.7. **clause** and **counter** refer to clause-taking verbs and counterfactuals, respectively. **interest** refers to "interesting" sentences, that is, sentences that pass at least one of the filters. . . . . 98

5.2 Results of majority human evaluation of system generated inferences. Evaluation on 127 inferences with from the test set by 3 or 4 people per inference. **Correctness is evaluated on whether the sentence is a reasonable inference in conversation, allowing for some awkwardness in phrasing. Context, means the correctness is highly context-dependent.* The inference type labels in the header row are the same as in Fig. 5.5 except for the addition of breaking down questions to **q-pre** for question presuppositions and **q-act** for question act inferences. . . . . 104

5.3 Confusion matrix on the FraCaS dataset. . . . . 138

5.4 FraCaS performance of our system (Ours) compared against a majority class (ENT) baseline (BL) and several notable RTE systems: MN (Mineshima et al., 2015), LP (Abzianidze, 2016), HU (Hu et al., 2019), and HR (Haruta et al., 2020). Hu et al. (2019) and Haruta et al. (2020) only report the overall accuracy of their systems. . . 139

A.1 Mappings between dummy argument position, text, and EL atoms. . . . . 174

A.2 Performance against gold standard parses of 50 synsets. . . . . 178

A.3 Performance against gold standard parses of 50 synsets. . . . . 179

C.1 Ablation results without decoding constraints, mean and standard deviation of 5 runs. 188

C.2 Ablation results with the lexicon constraint, mean and standard deviation of 5 runs. $\Delta\bar{x}$ is the difference in the mean score between the test set results of the model with the lexicon constraint and without, i.e. Table C.1. We only list this for the full model, but the pattern of higher precision but lower scores on other metrics generally holds for the other variants as well. . . . . 188

C.3 Ablation results with the type composition constraint, mean and standard deviation of 5 runs. $\Delta\bar{x}$ is the difference in the mean score between the test set results of the model with the type constraint and without, i.e. Table C.1. We only ran the full model for this test because this constraint takes much longer to run. . . . . 189

C.4 Fragments per sentence on the test set decoding results for a subset of the ablated lexicon-constrained models (Table C.2). $\alpha$ is the original model and $\tau$ is with the type composition constraint. . . . . 189

C.5 Default model parameters. . . . . 190

# List of Figures

1.1 Parse for *"They tried to find the ice bucket"* using the vanilla dialogue model of TRIPS. . . . . 7

1.2 Screenshot of the PMB Explorer with analysis of the sentence *"The farm grows potatoes."* . . . . . 12

1.3 Screenshot of Redwoods treebanking environment. The left pane displays constituent parse trees of the remaining analyses to be disambiguated. The right pane displays the list of discriminants. . . . . 16

1.4 AMR representations for *"The girl wanted to believe herself"*. . . . . 19

1.5 Screenshot of the AMR Editor editing the sentence *"The girl wants to believe herself."* 21

1.6 If John stumbles, but doesn't fall, we can infer from the axioms extracted from WordNet verbs that he misses a step and nearly falls. This inference would lead to a contradiction with representations that use an intersective approach to predicate modification such as OWL-DL. . . . . 24

1.7 On the left, the ontology of basic individuals in EL. On the right, descriptions of select categories of individuals. . . . . 25

2.1 The semantic interpretation process with the ULF step brought to 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. 30

2.2 Example sentences with corresponding raw ULF annotations. Items (1) and (2) are from the Tatoeba database, Items (3) and (4) are from *The Little Prince*, and Item (5) is from the Web. Strictly speaking, `weak.a` in Item (4) is actually missing a type-shifting operator `mod-n`, a simplification discussed in Section 2.1.5. . . . . 32

2.3 An example of temporal deindexing for the sentence *"The employees will have been working for 15 hours"*. The clausal structure and type-shifter positions in ULF define the bounds for the operator scope order. These operators are then traversed top-down to construct tense trees and deindex the formula. The ↑**a**, ↑**b**, etc. subscripts in the scoped formula mark the positions where a node is added to the tense tree and the corresponding tense tree states are shown in Fig. 2.4. . . . . 45

2.4 Tense trees corresponding the progressive deindexing steps for the example in Fig. 2.3. The steps are labeled with ↑**label** followed by the operator corresponding to this deindexing step in parentheses, e.g., (pres). The trees are built and read top-down, with nodes at the same level read left-to-right. The tree has three branching options: *past* ↙, *perf* ↓, and *futr* ↘ (written in this document as *will.aux-s* to reflect ULF conventions). The tree also has two horizontal embedding options: *re-traversable* ⇢ and *non re-traversable* -→. The circle indicates the focal node of the tense tree, which situates the next event in the deindexing process. When interpreting multiple sentences, the focus returns to the utterance node at the top of the tense tree. . . . . 46

2.5 Example of syntactic mapping from EL to EL-SMATCH triple and graph representations for sentence *"I am very happy"*. . . . . 60

3.1 Timing results from ULF experimental annotations. . . . . 63

3.2 Current ULF annotator state with example annotation process. . . . . 67

4.1 State transition diagram of the node generative transition system. Nodes in the figure are phases and edges are actions. An unlabeled edge means that this state transition occurs no matter what action is taken in that phase. The transition system starts in the GEN phase. . . . . 77

4.2 Example run of the transition system running on the sentence *"Do you want to see me?"* from our parser. The left four columns show the parser configuration after taking the actions shown in the rightmost column. We make the following modifications for brevity. When a **WordGen** action takes place, it is always followed by one of **Name**, **Lemma**, or **Token** and then a **Suffix**($e$) action. Thus we omit the **WordGen** and **Suffix** actions and transfer the argument of **Suffix** to the **Name**, **Lemma**, or **Token** action. "Promote" is abbreviated as "P" (e.g., **PromoteSym** as **PSym**) and **PushIdx** as **Push**. Stack item indices $(i, v)$ are written as $v^i$ instead. $\chi$ and $\iota$ stand for COMPLEX and INSTANCE which are the special node and edge labels, respectively, for constructing non-atomic ULF operators in penman format. Edge labels arg0 and arg1 simply indicate the argument position in ULF. $E_n = \{e_i \mid 0 \le i < n\}$ where $e_0$ = (do.aux-s $\xleftarrow{arg0}$ pres), $e_1$ = (pres $\xleftarrow{\iota}$ $\chi_0$), $e_2$ = ($\chi_0$ $\xrightarrow{arg0}$ you), $e_3$ = (see.v $\xrightarrow{arg0}$ me.pro), $e_4$ = (to $\xrightarrow{arg0}$ see.v), $e_5$ = (want.v $\xrightarrow{arg0}$ to), $e_6$ = ($\chi_0$ $\xrightarrow{arg0}$ want.v), $e_7$ = ($\chi_0$ $\xleftarrow{\iota}$ $\chi_1$), $e_8$ = ($\chi_1$ $\xrightarrow{arg0}$ ?). . . . . 78

4.3 The model consists of a sentence-encoding BiLSTM, a symbol-encoding LSTM, and an action-decoding LSTM. New symbols generated in the GEN and PROMOTE phases of the transition system are appended to the symbol sequence. The transition system supplies hard attention pointers that select the relevant word and symbol embeddings. These are concatenated with the transition state feature vector and supplied as input to the action decoder, which predicts the next action that updates the transition system. . . . . 83

4.4 A histogram of oracle action lengths in the training set (note the log-scale on the y-axis) with the cumulative density in orange. Additionally, the green vertical line shows the maximum action length supported by our model configuration. . . . . 86

4.5 Ablation tests with standard deviation error bars of 5 runs of different random seeds. 88

5.1 Examples of the sorts of discourse inferences that we generate via ULFs with rules described in Section 5.1. . . . . 94

5.2 An example of an inference rule for inferring an underlying desire from a request. Infixed notation in the inference rule is marked with square brackets for readability. Generalizations and variants of the rule for handling extraneous sentence modifiers, such as *please*, are omitted for clarity. . . . . 97

5.3 Example shorthand regex patterns (Section 5.1.7) for filtering candidate sentences with matching sentences. . . . . 99

5.4 A diagram of the automatic ULF inference evaluation pipeline. . . . . 102

5.5 Results of automatic inference evaluation described in Section 5.1.13. . . . . 103

5.6 Upward (...)▲, downward (...)▼, and non- monotone (...)■ examples from the FraCaS and MED datasets. . . . . 108

5.7 The basic inference rules for Sánchez-Valencia's natural logic proof system and an example. . . . . 112

5.8 Example of a ULF scoping into an SLF for the sentence *"Ali does not know that I work with a dog"*. . . . 113

5.9 Examples of lexical monotonicity markings. . . . 114

5.10 A tree representation of the SLF for *"No scientist knows every scientific fact."* with all lexical monotonicity, local entailment context, and global polarity markings. . . . 116

5.11 Monotonicity orderings of the predicates that are involved in the polarity inference example. Lower predicates are subsets of predicates above them. Predicates can be replaced with those above them in positive polarity and below them in negative polarity. . . . 117

5.12 Inference example from the FraCaS dataset. . . . 129

5.13 A diagram of the inference system component dependencies. . . . 131

5.14 Several examples of inference successes and failures. The relevant polarity contexts for the final (or last two if not overlapping) inference step is marked with (...)▲ or (...)▼ and the relevant spans are underlined in the premises and hypothesis. Our inference system predicated UNK for each of the failure examples. . . . 141

A.1 Example of rule extraction from machine readable dictionaries for WordNet entry of *slam2.v*. . . . 165

A.2 Example gloss axiomatization process for WordNet entry *slam2.v*. The numbering corresponds to the subsections where these stages are discussed in detail. . . . 171

A.3 Example glosses demonstrating the treatment of arguments in glosses. Underlined words are arguments and italicized arguments indicate where an argument should exist, but does not in the gloss. . . . 174

A.4 Unified versions of WordNet glosses from Figure A.3. . . . 174

C.1 Several parse examples comparing behavior of our parser with the stronger baseline, the GS parser (Cai and Lam, 2020). Errors are written in red. If something from the gold parse is omitted, a red highlighted block marks the location. . . . . 192

# 1 Introduction

Language use and its understanding, in its full breadth, is a hallmark of human intelligence and has been a long-standing goal of the field of artificial intelligence research. The complexity of this problem has led many to consider it an AI-complete problem—that is, a problem which can only be fully solved by a computational system which solves the core of human intelligence, whether one considers this to be a single general problem solving and reasoning mechanism or a collection of human abilities that work together in a specific manner. The problem of natural language understanding requires not only the ability to parse the structure of language, but to contextualize utterances within knowledge of the world and the participants of the interaction.

After great success in more modest goals of structural interpretation of language, the natural language processing and computational linguistics communities have, in recent years, set their sights on language understanding. This new endeavor has led to a debate over the representation of language meaning that is best suited for this understanding task. On one hand, recent successes lead many to conclude that a distributional representation, either in the form of a vector embedding or the form of weights in a pretrained neural network, is best suited for this problem. Others argue that we need symbols to represent meaning in a precise and generalizable manner, which are interpreted in a way that relates to the world outside of the language.

This dissertation is firmly planted in the latter position that symbolic descriptions of meaning are an important component to language understanding. Symbolic systems have features that reflect

human interface with language that have not yet been shown to be possible with purely distributional systems. People can incorporate knowledge from individual utterances and interactions to develop complex, multi-step plans which require modeling of precise relationships between entities in their understanding. Furthermore, the reception of a single piece of information can affect major systematic change to a person's plan or even mental model of the world. As of yet, distributional systems still require much greater amounts of information to learn precise knowledge about the world than what we observe in people, even more so, children.

To build on top of the benefits of symbolic systems in general, an explicit modeling of objects in the world, and with it a type system that reflects distinctions that we see in human language, gets us a step further. In this vein, Bender and Koller (2020) argue for the need to have some real correspondence between the form of language and meaning in the world to truly instill a machine with the ability to understand language. Meaning in language is at its core, they argue, born out of the communicative intent of the speaker in the context of some model of the world such that a system learning language from only the distributions of forms of language (e.g., language models learning to predict word sequence distributions over large collections of webdata) will never *understand* language despite how well the system may mimic human responses in contexts that resemble data that it has seen.

Having a model of objects outside of language and a systematic relationship between the form of language and this model means that even when the language use extends beyond anything seen prior, the model of the objects in the world—if the model is in fact effectively representing the objects in the world—provides the capacity to generalize beyond known conversational contexts. To use an example from Bender and Koller's (2020) discussion, if you asked someone to help you design a coconut catapult (or something else that brings together typically unrelated objects in the world) they can hypothesize likely solutions given their understanding of these objects in the world and what the speaker may want to do with this. However, a system that has only ever interacted with language form is unlikely to have any meaningful distribution of this combination of words. Of course, this is not to say that distributional systems are not useful. Rather, when using them, we

need to be sure the overall manner in which we represent meaning is able to make a correspondence between language form and things in the world.

Now, building a system with a model of language meaning needs to also in fact resemble what we see in language. Given the purpose of language, to meet the communicative needs of human societies and represent the world in a useful ways for people, as well as the natural relationship language has with human cognition—that is, language must encode information in a way that is compatible with the way people typically think so as to be an effective means of information transfer—we can expect all languages to share some fundamental semantic types for representing the world. These types would be, based on what we know about well-studied languages at least: truth/falsity, predicates, identity, general quantifiers, sentence and predicate modifiers, sentence and predicate reifiers (thus allowing for attitudes and other forms of intensionality), event referrers, and comparatives. Due to the shared underlying biological cognitive framework, even if some specific types are not encoded directly in specific linguistic devices, we can expect that the language would support the means to communicate such distinctions.

With these observations about human language as motivating factors, here I explore the design and use of an expressive symbolic description of language that takes advantage of the recent technological innovations from statistical and neural systems for pattern recognition. The goal of this project is to develop a meaning representation that balances the necessary expressive and inferential capacities for properly modeling language meaning and its use with the engineering concern of the ease with which we can automatically obtain these representations.

We start with **Episodic Logic** (EL), an extended first-order logic designed to closely match the expressive capacity of human language with a model-theoretic basis for meaning and inference. The important feature of EL for us is that its underlying semantic formulation models all of the semantic types that we suppose will be common in human language. EL, however, is too far removed from the surface structure of language to successfully parse in one fell swoop using current techniques. Instead, we take a divide-and-conquer approach that distinguishes multiple, interrelated goals. These goals include assignment of semantic types to the logical-form constituents (predicates,

names, determiners, predicate modifiers, relativizers, etc.) which determine how they can coherently combine, scoping of tense, quantifiers, and coordinators, resolution of anaphora, inference of event structure, disambiguation of word senses, recovery of elided or presupposed information, and conversion to some canonical form.

This first goal of determining the semantic types and the ways in which they combine lays the foundation for integrating the other goals into a single logical form. This dissertation demonstrates that these type-coherent initial logical forms (so-called *unscoped logical forms* – ULFs) provide both a stepping stone to full capture of sentential meanings and enable interesting classes of inferences while supporting manual annotation collection and automatic parsing through its proximity to syntax. Even without further resolution, the semantic type structure, by the very nature of using the EL semantic model makes the distinctions that we need to navigate the space of meanings in interpreting human language. The further resolution into EL allows us to support inferences that are increasingly distanced from the surface form and concerned with the underlying model. However, preservation of syntactic signals in ULF gains us the ability to make inferences that EL may have trouble with. For one, ULF enables superficial inferences based on the form of the utterance (e.g., recognizing requests), generating presuppositional and implicative inferences which rely on the syntactic structure, and monotonicity inferences based on both the semantic operator-operand relationships and syntactic tree structure for determining polarity.

In the process of using ULFs for any of these types of inferences, the underlying semantic types and their definition formally defined relationships ensure that we avoid gross semantic errors. For example, ULF's support of intensional modification, ensures that we do not end up concluding that a small elephant is smaller than a big rat, or that we will have gotten rich by finding a pile of counterfeit bills. Similarly, the recognition of kinds as distinct (but related) from the entities that make them up ensures that we avoid concluding from *"dogs bark"* that every dog is currently barking or from *"mice are widespread"* that each mouse is widespread.

## 1.1 Chapter Descriptions

The rest of this chapter is devoted to covering the relevant background to contextualize this work and cover essential prior work on top of which this work has been done. First we will cover related efforts on developing language understanding systems, particularly focusing on the choice of semantic representation and methods of data annotation. We make explicit comparisons of the expressive capacity of ULF to these other approaches, showing that ULF in fact covers a greater range of semantic phenomena that are needed for language understanding than these other systems. Finally, this section describes Episodic Logic to the reader in preparation for an in-depth discussion of ULF semantics and its relation to EL in language understanding systems.

In Chapter 2, we describe the unscoped logical forms, its role in EL interpretation, its syntax, its semantics, and how ULF handles several linguistic phenomena to ground the general description. Afterwards, Section 2.2 describes similarity measures from the AMR literature how we adapt them to be compatible with ULFs which have non-atomic operators unlike AMR.

Chapter 3 describes the pilot ULF annotations and how they affected the development of both the ULF representation and the annotator for collecting ULF data, the annotation speed and interannotator agreement results, and the resulting first manually annotated dataset of ULF.

Chapter 4 then describes the transition system-based parser which we designed to exploit ULF structural features and overcome the problem of data sparsity. By designing a transition system that models the relationship between surface words and ULF atoms, as well as, the manner in which ULF uses single-argument types shifters to maintain semantic coherence our parser beats state-of-the-art semantic parsing models on our dataset of ULF annotations.

Chapter 5 covers methods of inference on ULFs. First, it describes our rule-based system for generating discourse-related inferences based on certain important structural signals: (1) implicatives, (2) questions & request presuppositions, (3) counterfactuals, and (4) attitudinal & communicative verbs. This demonstrates not only that such inferences are possible from ULFs with simple inference rules, but that by using a representation that can easily map to English, we can evaluate inferences

more readily while still maintaining internal semantic structure to restrict the types of inferences that are generated. Section 5.2 in Chapter 5 makes a correspondence between Sánchez Valencia's (1991a) framework and a proof-based inference system running over ULFs—theoretically ULF's ability to support natural logic-like monotonic inferences. Section 5.3 then goes on to implement a version of this and demonstrates experimentally that ULF indeed can support natural logic like inferences in a variety of contexts, handling both single- and multi-premise examples.

## 1.2 Related Work

There has been much activity in developing semantic parsers that derive logical representations, raising the possibility of making inferences with those representations (Popescu et al., 2004; Kate and Mooney, 2006; Kwiatkowski et al., 2011; Liang et al., 2011; Poon, 2013; Tellex et al., 2011; Artzi and Zettlemoyer, 2013; Howard et al., 2014; Artzi et al., 2015; Konstas et al., 2017). The techniques and formalisms employed are interesting (e.g., learning of CCG grammars that generate $\lambda$-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.[1] Acquisition of knowledge via language, or inferences about beliefs, intentions, etc., have generally not been addressed in any broad way.

Noteworthy examples of formal logic-based approaches, not targeting specific applications were done by Bos (2008) and Draicchio et al. (2013), whose hand-built semantic parsers respectively generate FOL formulas and OWL-DL expressions. But these representations preclude generalized quantifiers, modification, reification, attitudes, etc., and the route from natural language expressions to logic (CCG parsing → DRS → FOL, in the case of Bos' (2008) BOXER) often produces flawed representations. We are not aware of any work on inference generation of the type ULF targets based on these projects. We now turn to notable work in domain-general semantic parsing and

[1] For example, (Ross et al., 2018) develop a CCG-based semantic parser for action annotations in videos, representing sentences such as *"The woman walks by the table with a yellow cup"* in an approximate way: neglecting determiners and treating all entity references as variables.

annotation. Each one of these projects has been a major influence on a portion of this dissertation.

### 1.2.1 TRIPS

The TRIPS parser generates semantic parses in an underspecified semantic representation with scoping constraints (Allen and Teng, 2017; Allen et al., 2018). The nodes are grounded in an ontology, which is a single inheritance hierarchy built on a mixture of syntactic and semantic distinctions. Some levels of the ontology roughly correspond to VerbNet (Schuler, 2006) and FrameNet (Baker et al., 1998) classes, but the TRIPS ontology makes additional distinctions regarding temporal and entailment information. It has three equivalent formats, which roughly correspond to the three different formats in which AMR can be written: logical, graphical, and PENMAN. Figure 1.1 shows an example of a TRIPS parse in graphical format.

The TRIPS parser generates LFs using a bottom-up chart parser with a hand-built grammar, a syntax-semantic lexicon, and a semantic ontology. The parser's preferences are further tuned using the results from tokenizers, POS taggers, NER taggers, and constituent parsers as is relevant for the task at hand (Allen et al., 2018).

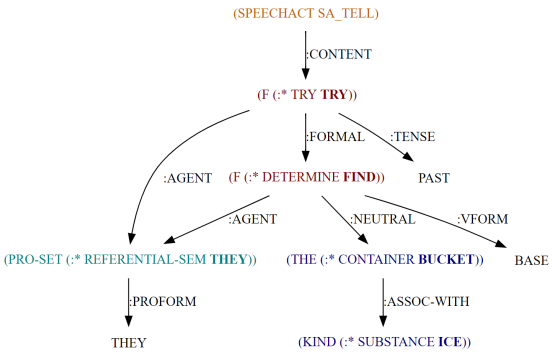



Figure 1.1: Parse for *"They tried to find the ice bucket"* using the vanilla dialogue model of TRIPS.

One of the greatest strengths of TRIPS is its ontology of English words which gives a unified relationship between words using semantic roles, aspectual class, and temporal/causal entailments. The benefits of this ontology appear during both task of parsing LFs and inference using LFs. During parsing the ontology provides semantic preferences in argument selection and word sense disambiguation. During inference the ontology provides paths between related words of varying degrees of specificity, information about time and causality in relation to the words at hand,

and structured semantic relationships between words for making nuanced similarity judgments between predications.

The TRIPS parser has been deployed in multiple tasks with minimal modifications: extracting knowledge from biology papers (Allen et al., 2015), conversational dialogue (Rhee et al., 2014), goal-oriented dialogue (Perera et al., 2017), and lexical knowledge acquisition (Allen et al., 2011).

### TRIPS LF

Allen et al. (2008) describe a graphical logical form representation derived from the representation used by the TRIPS dialogue system (Allen et al., 2007).[2] Each node in the LF has three components: *specifier*, *type*, and *word* which represent the node function, conceptual class, and surface word, respectively. The specifiers are semantic functions enumerated in the LF specification and the type is a conceptual class from the ontology used in conjunction with the LF. The nodes are connected using relations inspired by thematic roles (e.g., `:AGENT`, `:AFFECTED`, `:FORMAL`, etc.) (Gruber, 1965; Fillmore, 1968, among others). For example, consider the top-level node in Fig. 1.1, `(SPEECHACT SA_TELL)`. `SPEECHACT` is the specifier, `SA_TELL` is the type, and the word is empty because this top-level is generated from the general sentence structure assuming deployment in a dialogue setting. It participates in one relation `:CONTENT` with the node `(F (:* TRY TRY))` as the parent of the relation.

TRIPS LF descriptively captures modal constructions, generalized quantifiers, lambda abstractions, and dialogue semantics making it able to express a large subset of natural language phenomena. These features allow TRIPS LFs to make semantic distinctions that are unavailable to most other LFs that are currently commonplace in NLP systems.

Formally, TRIPS LF is an underspecified semantic representation which subsumes well-formed Minimal Recursion Semantics (MRS) (Copestake et al., 2005) and Hole Semantics (Bos, 1996), both of which are object-language agnostic, meta-level semantic representations (Allen et al., 2018).

[2]The full documentation for the LF is available at `http://trips.ihmc.us/parser/LF%20Documentation.pdf`.

Such meta-level representations are useful for managing underspecified, but constrained models of meaning in manners that are computationally simple and efficient. The problem with model-agnostic semantics, however, is that without grounding these representations into an object language, model-theoretic notions such as interpretation, truth, satisfaction, and entailment cannot be applied. Such model-theoretic semantics is what allows ULF to make formal claims about its inferential capabilities in Section 5.2 in relation both natural logic and EL inference—which itself supports its inference capabilities with model-theoretic episode semantics. Even beyond fully formalized inference rules, having a formal model of the underlying semantic relationships provides us with the semantic distinctions necessary to avoid major confusion in a system reasoning about language. Using a type system reflective of distinctions made in human language ensures that the inferences and distinctions that we make with the representation reflects those of actual language use; this puts limits on the sorts of errors that a system using this representation will make, for example we can avoid mistakenly concluding that *unicorns exist* from the sentence *"Bob dreamed of a unicorn"* or that *all lions are currently roaring* from the sentence *"lions roar"*.

One can ground these model-agnostic representations into an object language, but must be careful in their choice of object language. In a project extracting lexical knowledge from WordNet, Allen et al. (2013) mapped TRIPS LF to OWL-DL (OWL Working Group, 2004) from which subsumption-based inference was performed between WordNet definitions. While expressivity limitations did not arise in the subset of WordNet encountered in the project, OWL-DL struggles with common linguistic phenomena such as non-intersective modification, reification, self-reference, and uncertainty. Any attempt to generalize this project to the rest of WordNet would likely run into these fundamental limitations of OWL-DL. Of course, TRIPS doesn't need to be grounded into OWL-DL specifically. Though it would probably be a trickier endeavor, mapping TRIPS into a more expressive object language seems possible.

All of this does not mean that inference systems that operate directly on model-agnostic representations cannot be useful. In fact, I have already mentioned instances where TRIPS LF has been used to great effect. We simply cannot make model-theoretic claims about the inference until

the object language is defined. I will refrain from repeating such in-depth discussion of limitations in in the following sections, but these criticisms apply to any representation that is insufficiently expressive or model-agnostic in its semantics.

### 1.2.2 The Decompositional Semantics Initiative

Decomp[3] follows an approach to building up a model of language semantics by modeling every day language user annotations on focused phenomena. The idea is to pose semantic distinctions as questions that are quick and easy to judge by every day users. This allows the construction of large corpora from which to learn models of human language judgments. Then by building these models up for more and more semantic distinctions one can construct a general model of language semantics.

Much of the work in Decomp is built on top of PredPatt (White et al., 2016), a predicate-argument extractor built on top of universal dependencies. These are very low-level predicates, just above the surface language. See the example below (from the PredPatt github page[4]) which shows the predicate-argument structures that PredPatt outputs.

```
PredPatt extracts predicates and arguments from text .
?a extracts ?b from ?c
    ?a: PredPatt
    ?b: predicates
    ?c: text
?a extracts ?b from ?c
    ?a: PredPatt
    ?b: arguments
    ?c: text
```

[3] https://www.decomp.net

[4] https://github.com/hltcoe/PredPatt

Notice that the predicates include prepositions in them so they are not decomposed in the way semantic representations usually are. The goal was to provide a high-precision interface between syntax and semantics for reliable usage in downstream semantic tasks or further resolution. Zhang et al. (2017) showed that PredPatt has the best precision and recall among well-known open-domain information extraction systems on a large scale benchmark based on PropBank (Kingsbury and Palmer, 2002; Gildea and Palmer, 2002; Palmer et al., 2005).

Stengel-Eskin et al. (2020) bring together PredPatt predicate-argument structures with labels of a variety of semantic phenomena that were collected one at a time from crowd workers. These semantic labels include proto-roles (Reisinger et al., 2015; White et al., 2016), factuality (Rudinger et al., 2018), genericity (Govindarajan et al., 2019), entity types (White et al., 2016), and temporal relations (Vashishtha et al., 2019). The long term goals of this project align well with the goals of this thesis in generating high-fidelity semantics for language which enables commonsense reasoning. To paraphrase Ben Van Durme's comment during the CogSci Dinner on April 16th 2018, Decomp is trying to climb up the mountain of semantics one step at a time through modeling human judgments whereas the ULF project is trying to climb down the mountain of semantics from episodic logic to a tractable subset for feasible annotation and parsing. As such, I am sympathetic to the efforts of this initiative. At the moment, however, the semantic descriptions from this project are completely decoupled from the model-theoretic basis which drives much of this proposal.

### 1.2.3 Groningen/Parallel Meaning Bank

The Groningen Meaning Bank (GMB) (Basile et al., 2012) was collected at the University of Groningen led by Johan Bos and contains Discourse Representation Structures (Kamp, 1981) annotations of full documents. These annotations combine linguistic information from a variety of sources at different levels of semantic granularity, such as the thematic roles from VerbNet (Schuler, 2006) and word sense from WordNet (Miller, 1995). GMB used an annotation approach that they called *human-aided machine annotation*, where the annotation process first is automated then corrected by humans. This work was extended into the Parallel Meaning Bank (PMB) (Abzianidze

et al., 2017) which extends this annotation to multiple languages with semantic projection across sentence translations. I will focus my discussion on the English portion of the PMB since it is an improved version of GMB.

### Annotation Pipeline

The annotation pipeline is separated into layers from tokenizing to sentence and discourse level semantics that build on top of each other. These layers are automatically generated using custom taggers and parsers, then corrected by annotators to obtain gold-annotations. As annotations are accumulated, they are used to retrain the automatic systems. The modularity of the annotation layers and the correction-driven annotation procedure eases the burden on the annotator.

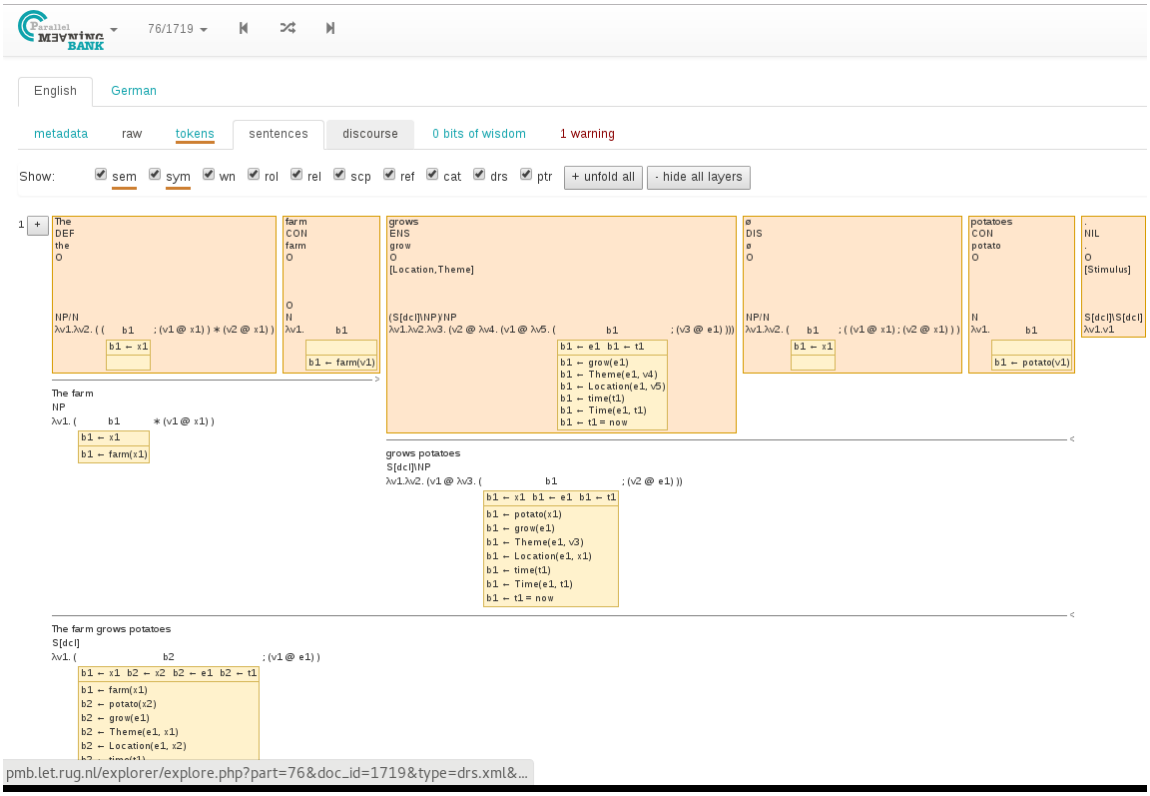



Figure 1.2: Screenshot of the PMB Explorer with analysis of the sentence *"The farm grows potatoes."*

### PMB Explorer

A highly-featured annotation website was developed to assist in PMB's structured annotation approach. Figure 1.2 shows a screenshot of the PMB Explorer displaying an automatic analysis. Below are notable features of the PMB Explorer.

- A modular, layer-wise annotation view – each can be marked correct separately.
- Correction tracking, revision history, and reversion.
- An integrated bug-tracker for annotator organization and communication.

### Annotated Texts

The texts selected for annotation in the PMB are mostly freely distributable and total in 11.3 million tokens and 285,154 documents (Abzianidze et al., 2017). They include the following sources.

- Totoeba[5]
- Newscommentary corpus (Tiedemann, 2012)
- Recognizing Textual Entailment (RTE) corpus (Giampiccolo et al., 2007)
- Sherlock Holmes stories[6]
- The Bible (Christodouloupoulos and Steedman, 2015)

The most recent release, v3.0.0 has 8,403 annotated documents, though this includes some unverified annotations that are generated automatically. Note that there are three other languages that are annotated in this release as well. While no inter-annotator agreement is reported, the Explorer allows annotation revisiting and correction between annotators so the gold-annotations are likely highly uniform and accurate.

[5] https://tatoeba.org

[6] http://gutenberg.org, http://etc.usf.edu/lit2go

#### Discourse Representation Structures

Discourse Representation Structure (DRS) is a representation developed by Kamp (1981) and Kamp and Reyle (1993) to handle anaphora, discourse structure, and presupposition. The key issue that DRS overcame is "donkey anaphora" (e.g., *"Every child who owns a dog loves it."*). DRT overcomes the challenges of determining the quantifier interpretation and scope in such examples with principles that determine the nature of anaphoric bindings.

DRS can also generate inferences via a mapping to FOL. The ability to perform this mapping, however, means that the formal expressivity of DRT equals that of FOL. FOL itself is not able to handle many important semantic phenomena such as generalized quantifiers and non-modal intensionality. An investigation of the manually corrected examples in PMB reveals that PMB instead interprets generalized quantifiers as adjective-like quantity descriptions of positive or negative polarity relying on an external silent operator to connect the propositions related to the restrictor and body of the quantification. This positive/negative polarity interprets distinct quantifiers such as *many* and *most* into the same logical representation.[7]

### 1.2.4 LinGO Redwoods Treebank

The LinGO Redwoods Treebank is a syntacto-semantic treebank with annotations of Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) with corresponding Minimal Recursion Semantics (MRS) (Copestake et al., 2005). This long-running project has developed a suite of resources which support the annotation project. This includes LinGO English Resource Grammar (ERG) (Bub et al., 1997), a hand-built HPSG-grammar, a fast grammar parser (Callmeier, 2001), and the `[incr tsdb()]` grammar profiling and annotation environment (Oepen, 2001).[8]

The Redwoods Treebank is annotated in ERG and has grown into a topically-varied collection of corpora over the years. The most recent release, the Ninth Growth, was released in 2016 and

[7] PMB document ids 00/3153, 01/2473, and 00/0855 are examples of this.

[8] All the tools can be found at `http://www.delph-in.net`.

consists of about 85,400 utterances in a broad range of genres (Oepen and Flickinger, 2019).

### Minimal Recursion Semantics (MRS)

MRS is a flat semantic representation which uses handles to encode scope effects and ambiguity. Its semantics is composed via unification by design since it was developed as a formalization of semantic composition in typed feature structure grammars (Copestake et al., 2001). In an effort to be an effective computational semantics, MRS is designed to satisfy the following criteria (Copestake et al., 2005).

- *Expressive Adequacy* – correctly express the full range of linguistic meanings
- *Grammatical Compatibility* – link cleanly to syntactic analysis;
- *Computational Tractability* – enable efficient processing of meaning; such as equivalence checking, and represent meaning in a computationally simple manner;
- *Underspecifiability* – allow the representation to leave ambiguous distinctions unresolved, which can be resolved in a simple and flexible manner.

Copestake et al. emphasize the last two criteria having found that previous work by computational linguists lack in those areas. Copestake et al. (1995) explain that MRS is not a complete model semantic theory, rather a representation for describing semantic structures in relation to HPSG. Thus it is unclear how we can make judgments about the formal semantic expressivity of MRS in relation to ERG at all. By design, then, MRSs are descriptively powerful while lacking a definite semantic interpretation and inference mechanism rooted in model-theoretic semantics.

### Annotation Procedure

Similar to the Parallel Meaning Bank, the Redwoods Treebank relies on human-guided machine annotations. The annotation procedure begins with generating the 500 highest-ranked analyses from a hand-built HPSG grammar (Flickinger et al., 2012). The correct analysis, if any, is manually

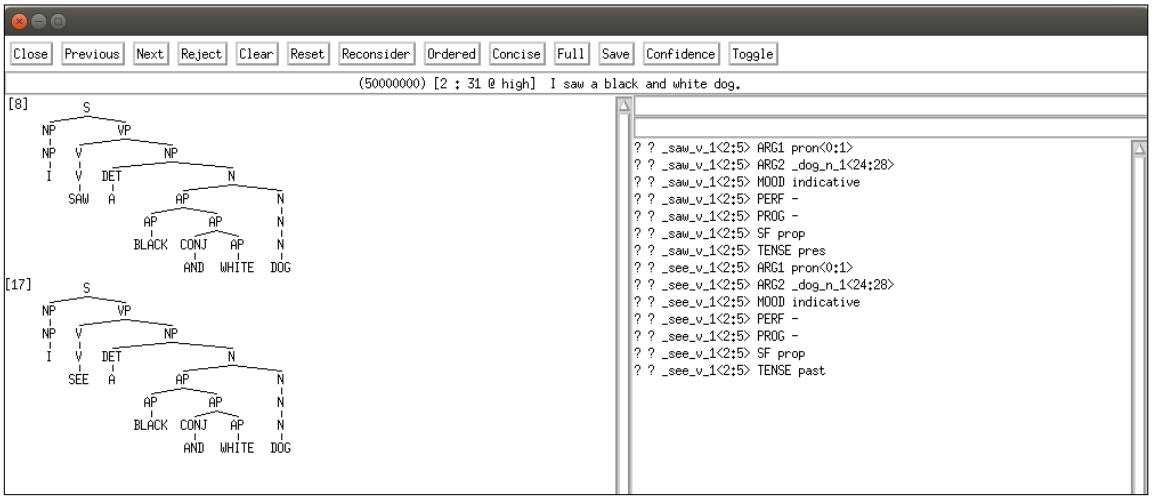



Figure 1.3: Screenshot of Redwoods treebanking environment. The left pane displays constituent parse trees of the remaining analyses to be disambiguated. The right pane displays the list of discriminants.

selected using a series of decisions of whether to include a candidate lexical or relational analysis. For example, `_see_v_1<2:5> TENSE past` which describes the analysis of the text span (2,5), "saw", as the first sense of the verb "see" (`_see_v_1`) with a past tense. The annotator then makes a decision as to whether this analysis should be included or excluded from the final form. See the screenshot in Fig. 1.3 for the environment state when only two candidate trees remain (there are no more trees below the frame) with over a dozen discriminants that the annotator can choose from to identify the correct analysis.

The `[incr tsdb()]` tool automatically prunes impossible analyses given the decision. The annotator decisions are saved so that upon updating the grammar, the annotation decisions can be rerun to identify and re-annotate ambiguities that arise from the changes in the grammar. This turns out to greatly reduce the re-annotation burden. Flickinger et al. (2012) report that this method required re-annotating 5-10% of the corpus in addition to the newly covered sentence when the grammar is improved. The limitation of this approach is that it assumes that the annotator was correct to begin with, so this cannot so easily be used to re-annotate the treebank efficiently to correct large-scale annotation mistakes identified during an error analysis.

### Challenges of a Hand-built Grammar

The annotation method of Redwoods Treebank is highly efficient but relies on a hand-built grammar. This grammar building turns out to take a long time as seen by the changes in coverage reported over the years of this project. The coverage statistics for the second most recent release of the Redwoods Treebank is shown in Table 1.1. It is quite impressive with an 87% verified coverage across several domains. Still, 13% is a considerable portion of text to leave unanalyzed. In 2004, Baldwin et al. (2004) analyzed ERG coverage on a sample of the British National Corpus (BNC) (Burnard, 2000). This is four years after the first paper describing ERG (Copestake and Flickinger, 2000). The results of the analysis can be summarized as follows:

- 32% of BNC sample have complete ERG lexical coverage
- Of those, 57% ERG could generate a parse for (18% of total sample)
- Of those, 83% contained a correct parse (15% of total sample)

When the errors are compounded for the final result ERG performance leaves much to be desired. This experiment shows the amount of work that it takes to hand-build a reliable grammar. A few years into development it could only generate a correct analysis for 15% of a sampling from the BNC. When developed for over a decade, a hand-built grammar can perform pretty well, but it is not a task that one should add to their project lightly. In order to overcome the 13% coverage gap of ERG, Zhang and Krieger (2011) developed a PCFG approximation of ERG which uses ERG rule names with feature structures and syntactic context as PCFG categories and a heuristic unifier to force unification. This PCFG parsed over 99% of sentences with a 2% reduction in F1. This has not been used in the annotation process so far.

### Discussion

The Redwoods Treebank project has many engineering successes that are reflective of the length of time that it has been under development and the great number of people collaborating on it. The

| Segment | # of sentences | Raw coverage | Verified coverage |
|---|---|---|---|
| Verbmobil | 12,393 | 96.46% | 92.18% |
| WSJ (00-21) | 43,541 | 93.87% | 85.17% |
| WeScience | 11,558 | 91.93% | 80.75% |
| SemCor | 3,000 | 93.81% | 85.11% |
| Total | 92,706 | 94.73% | 87.28% |

Table 1.1: ERG results for the second most recent Redwoods release (the Eighth Growth). Only a selection of segments (corpora) are included in this table, but the total results includes all sentences in the growth.

successes of the project in flexibility of analysis, particularly in syntax, also reflects the motivations described by early papers to overcome the limitations of annotations that focus on one type of information (e.g., phrase structure trees, semantic dependency graphs) with a fixed set of labels while losing information that could be gained concurrently in an integrated manner.

However, the lack of an inference mechanism makes it unsuitable for inferential language understanding. The recommended inference method with this representation is to use FOL theorem provers which limits their semantic expressive capacity to FOL. MRS use in applications so far have relied on task-specific, *ad hoc* methods: tree transformations for question generation (Yao and Zhang, 2010), semantic mapping rules for semantic-transfer machine translation (Copestake et al., 1995), or subsumption-based reasoning for recognizing textual entailment (Lien and Kouylekov, 2015)[9]

### 1.2.5 Abstract Meaning Representation

Abstract Meaning Representation (AMR) is a semantic representation that has made a big splash in the NLP community, leading to a large volume of papers published that present methods of parsing AMR from sentence text. AMR was developed as a semantic representation of language that abstracts away from morpho-syntactic idiosyncrasies (Banarescu et al., 2013). The idea is to introduce a simple, but unified semantic representation of sentences—bringing together argument

[9]Lien and Kouylekov use a representation that is a combination of OWL-DL (OWL Working Group, 2004) and Horn-like rules resulting in a similar level of expressive capacity.

structure identification, preposition attachment, etc.—similar to constituent trees for syntax. Its coverage of a wide range of linguistic phenomena is admirable but I expect that the expressive limitations of AMR representations will make it difficult to be used in a wide variety of applications. Furthermore, the decoupling of the representation from surface text makes AMR recovery from the surface text more difficult than semantic representations that have more transparent relationships to syntax.

AMRs are rooted, directed, acyclic graphs where the leaves act as node labels and relations and properties are defined between the nodes. AMR has a descriptively equivalent neo-Davidsonian (Davidson, 1969) logical format which introduces a implicitly existentially quantified variable for every entity, event, property, and state. The standard text representation is PENMAN format, which transparently represents a graphical structure in text using node-variables (Mathhiessen and Bateman, 1991). Figure 1.4 shows an example in all three formats.

AMR relations include core PropBank (Kingsbury and Palmer, 2002; Palmer et al., 2005) arguments, select semantic relations from the thematic roles literature, and relation for special domains such as quantities, dates, and lists. Node labels generalize across parts-of-speech and etymologically related words, which leads to the following four expressions all being annotated with the concept "fear-01".

- *"My fear of snakes"*
- *"I am fearful of snakes"*

**Logical format**

∃ w,g,b:

instance(w, want-01) ∧

instance(g, girl) ∧

instance(b, believe-01) ∧

arg0(w, g) ∧ arg1(w, b) ∧ arg0(b, g)

**AMR format**

```
(w / want-01
      :arg0 (g / girl)
      :arg1 (b / believe-01
            :arg0 g))
```

**Graph format**

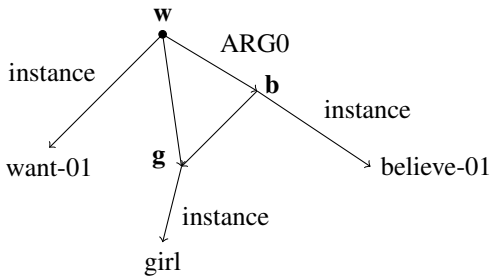



Figure 1.4: AMR representations for *"The girl wanted to believe herself"*.

- *"I fear snakes"*
- *"I'm afraid of snakes"*

Banarescu et al. (2013) and the AMR Specifications (Banarescu et al., 2019) discuss all the phenomena that AMR handle in detail.

### Semantics of AMR

Bos (2016) provides a model-theoretic analysis of AMR semantics. He shows that standard AMR representations are a subset of FOL with up to one universal quantifier[10] and presented a potential extension to AMR that would capture multiple universal quantifiers.

Providing AMRs with a model-theoretic interpretation is a step forward, but it remains a subset of FOL, which in itself is not expressive enough to appropriately describe many important natural language phenomena, such as intensionality and reification of quantified sentences (Schubert, 2015). Bender et al. (2015) point out concrete limitations of AMR expressivity with the following groups of sentences that generate the same AMR representations despite non-trivial differences in meaning.

1. (a) *"Every person failed to eat."*
   (b) *"No one ate."*
2. (a) *"The boy is responsible for the work."*
   (b) *"The boy is responsible for doing the work."*
   (c) *"The boy has the responsibility for the work."*

In the first pair of sentences, sentence 1a requires some sort of attempt, intention, or expectation to eat, whereas sentence 1b does not require such a context. In the second pair of sentences, sentence 2a, *"work"* may refer to the resulting object that already exists (such as a work of art) whereas sentences 2b and 2c cannot have such an interpretation.

[10]The AMR specifications note that universal quantifiers are not represented by AMR. Bos overcomes this issue by using a series of polarity operators to capture the semantics of a universal quantifier.

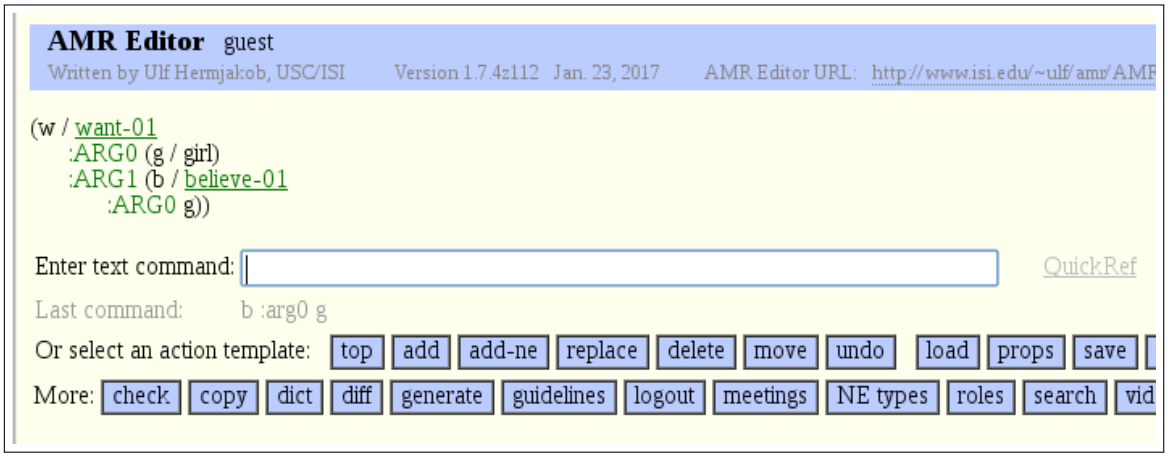



Figure 1.5: Screenshot of the AMR Editor editing the sentence *"The girl wants to believe herself."*

**AMR corpus & editor**

The most recent release of the multi-genre annotated AMR corpus, AMR Annotation Release 3.0, includes 59,255 sentences, available to Linguistic Data Consortium (LDC) members.[11] Hermjakob (2013) built a special editor for annotating AMR representations. The editor provides the following input methods to fit an annotator's preferences.

1. **Text Commands** – These are Unix-style text commands for editing the AMR graph. It includes a *last command* feature which is widely used in Unix shells.

2. **Templates** – The editor provides template versions of the text commands. Selecting a template opens a window with arguments slots to be filled out.

3. **Point and Click** – The annotator may point and click part of the annotation-in-progress to highlight a segment that they wish to delete or modify.

4. **Post Editing** – The AMR editor can automatically generate approximate AMR annotations from OntoNotes (Hovy et al., 2006) annotations.

[11] Available at `https://catalog.ldc.upenn.edu/LDC2020T02`.

The AMR editor also comes with annotator support in terms of links to the AMR Editor manual and FAQs, the AMR guidelines, the lists of roles in AMR, lists of named-entity types, and underlining some words with suggestions for concepts or properties. Figure 1.5 displays a screenshot of the editor in use.

### Multi-Sentence AMR

O'Gorman et al. (2018) build on top of AMR and unify AMRs from shared documents into into a shared representations of Multi-Sentence AMR (MS-AMR). This is accomplished by identifying coreference clusters of AMR variables among the multiple sentences and unifying them under a single variable. This resolves ambiguity from implicit roles and referential concepts (e.g., "he") which corefer to variables with explicit meanings in other sentences.

### Limitations

Beyond the semantic limitations already discussed at the start of this section, AMR also has limitations intrinsic in its design that limit its usefulness even in tasks that AMR set out to assist. For example, the lack of grammatical number, tense, aspect, quotation marks, etc. would make precise natural language generation challenging, which Banarescu et al. (2013) list as one of the tasks they expect an AMR corpus would help. This is so challenging, in fact, that translating AMRs into natural language has become a research problem of its own (Bai et al., 2020; Jin and Gildea, 2020; Wang et al., 2020; Song et al., 2018; Beck et al., 2018). Schneider et al. (2015) argue that AMR does not capture tense because it does not generalize to other languages at a sentence-level. This is a strange design decision considering AMR is not supposed to be an interlingua and is closely tied to English vocabulary anyway.

## 1.3 Episodic Logic (EL)

EL is a semantic representation extending FOL, designed to closely match the expressivity and form of natural language and to enable deductive inference, uncertain inference, and natural logic-like inference (Morbini and Schubert, 2009; Schubert and Hwang, 2000; Schubert, 2014). It echoes both the surface form of language, and more crucially, the semantic types that we can expect to find 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, allows for characterizations of episodes with sentences of arbitrary complexity, such as *"I did not eat anything all day"* or *"each superpower is menacing the other with its nuclear arsenal"* (Schubert, 2000).

EL also has modifiers of predicate and sentence intensions as well as operators for type-shifting predicate intensions into such modifiers which allows us to form modifiers out of complex expressions (for example *supposedly fake* can be a complex intensional modifier to the noun *diamond*). The importance of supporting intensional modification is demonstrated by Fig. 1.6 which illustrates a simple forward inference chain using an EL axiom from WordNet on the sentence *"John stumbles, but does not fall"*. This inference uses the axiom for *stumble2.v* and a hand written axiom schema asserting that statements conjoined with the connective "but" asserts the conjunction of

*Axioms*

A1. *stumble2.v* : *miss a step and fall or nearly fall*

```
(∀ x,e: [[x stumble2.v] ** e]
         [[(∃ z: [z step2.n] [x miss4.v z]) ∧
           [[x fall23.v] ∨ [x (nearly.adv fall23.v)]]] ** e])
```

A2. *If two statements are conjoined by "but", then both statements are true (i.e. conjunction)*

```
(∀wff x,y (∀ e: [[x but.cc y] ** e] [[x ∧ y] * e]))
```

*Inference*

Sentence: "John stumbles, but does not fall"

| | | |
|---|---|---|
| I1. | `[[John.name stumble2.v] but.cc ¬[John.name fall23.v]]` | Initial parsed sentence |
| I2. | `[[John.name stumble2.v] ∧ ¬[John.name fall23.v]]` | I1 & A2 |
| I3. | `[John.name stumble2.v],¬[John.name fall23.v]` | I2 |
| I4. | `[(∃ z: [z step2.n] [John.name miss4.v z]) ∧ [[John.name fall23.v] ∨ [John.name (nearly.adv fall23.v)]]]` | I3 & A1 |
| I5. | `[(∃ z: [z step2.n] [John.name miss4.v z]) ∧ [John.name (nearly.adv fall23.v)]]` | I3 & I4 |

*Contradiction using OWL-DL*

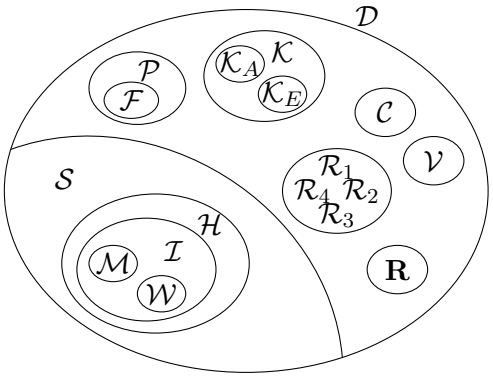
| | | |
|---|---|---|
| I6. | `[John.name (nearly.adv fall23.v)]` | I5 |
| I7. | $\forall_{of}$.(John.name) $\sqcap$ $\forall_{of}$-1.(nearly.adv) $\sqcap$ fall23.v | I6 (Represent in OWL-DL) |
| I8. | $\forall_{of}$.(John.name) $\sqcap$ fall23.v | I7 |
| I9. | `[John.name fall23.v]` | I8 (Represent in EL) |
| I10. | $\square$ | I3 & I9 (Contradiction) |

Figure 1.6: If John stumbles, but doesn't fall, we can infer from the axioms extracted from WordNet verbs that he misses a step and nearly falls. This inference would lead to a contradiction with representations that use an intersective approach to predicate modification such as OWL-DL.

the two statements as well.[12] Appendix A describes my work on extracting such axioms from EL automatically. When we use a meaning representation that only supports intersective modification, such as OWL-DL, the inference incorrectly leads to a contradiction.

The limitations of current EL parsing found while doing this axiom extraction work was one of the key motivations of this dissertation. While EL is representationally appropriate for language understanding in terms of expressive and inferential capacity, the current EL parser was

[12]This second axiom is an axiom schema since it uses substitutional quantification over well-formed formulas, $\forall_{wff}$. Substitutional quantification is part of what allows EL to represent information about its own syntax and is used for meta-syntactic reasoning. Substitutional quantification and meta-reasoning in EL is explained in detail by Morbini and Schubert (2008).




| Category | Description |
|---|---|
| $\mathcal{D}$ | The domain of individuals |
| $\mathcal{S}$ | Possible situations – for handling event structure and intension |
| $\mathcal{I}$ | Possible times – spatially maximal situations |
| $\mathcal{W}$ | Possible worlds – temporally and spatially maximal situations |
| $\mathcal{P}$ | Propositions – from reified sentences |
| $\mathcal{F}$ | Possible facts – consistent propositions |
| $\mathcal{K}$ | Kinds – from reified predicates |
| $\mathcal{K}_A$ | Kinds of actions |
| $\mathcal{K}_E$ | Kinds of events |
| $\mathcal{A}$ | Actions(*not a basic individual*) – the range of a pairing function of type $((\mathcal{D} \times \mathcal{S}) \rightarrow \mathcal{D})$ |

Figure 1.7: On the left, the ontology of basic individuals in EL. On the right, descriptions of select categories of individuals.

far too unreliable. 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). In order to build a better EL parser, it is only natural to take advantage of recent advances in corpus-based parsing techniques.

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. Figure 1.7 shows a diagram of the complete ontology. Schubert and Hwang (2000) provide a complete description of the ontology. 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 **2** at a given (possible) situation, or be *undefined* there, indicating irrelevance of the predication in the given situation. A complex type, $(A \rightarrow B)$, is constructed from arbitrary types $A$ and $B$. $\rightarrow$ is the only available connective. Two types $(A \rightarrow B)$ and $C$ can be composed together (regardless of order) if and only if $C \subseteq A$ resulting in the type $B$. For example, a general monadic predicate type, $(\mathcal{D} \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))$, can combine with a proposition type, $\mathcal{P}$, resulting in $(\mathcal{S} \rightarrow \mathbf{2})$ because a propositions part of the domain of individuals, $\mathcal{P} \subseteq \mathcal{D}$. On the other hand, $(\mathcal{P} \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))$ and $\mathcal{D}$ cannot compose together since $\mathcal{D} \nsubseteq \mathcal{P}$.

Most predicates in EL are of type $(\mathcal{D}^n \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))$, where $(A^2 \rightarrow B)$ abbreviates $(A \rightarrow (A \rightarrow B))$, $(A^3 \rightarrow B)$ abbreviates $(A \rightarrow (A \rightarrow (A \rightarrow B)))$, and so on. That is, they are first-order intensional predicates.[13] Monadic predicates play a particularly important role in EL as well as ULF, so 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 $\pi$ is the operator and $\tau_i$ are the arguments) and parentheses indicate prefixed operators (i.e. $(\pi\ \tau_1\ ...\ \tau_n)$ where $\pi$ is the operator and $\tau_i$ are the arguments). 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, which is formulated as `(that [(k grass.n) red.a])` in EL, uses both of these operators.

### 1.3.1 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, but these operators

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

are foundational to EL 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 '**' relation is directly interpreted in the model theory (and for positive atomic predications coincides with Davidsonian event semantics (Schubert, 2000)), while '*' and '@' are defined derivatively from '**'.

**'**' - 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,

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

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

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

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

Where $\sqsubseteq$ is an episode part-of relation. Hwang and Schubert (1993) provides a formal definition. Intuitively we can think of $s \sqsubseteq \eta$ to mean that $s$ is a subepisode of $\eta$, either a concurrent subepisode or temporally spanning just parts of $\eta$.

**'@' - 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 $\eta]$ $[\phi \ ** \ e]$). Formally, @ is defined

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

# 2 Unscoped Logical Form (ULF)

This chapter introduces Unscoped Logical Forms (ULFs) in detail, describing their syntax, their relationship to EL semantics, and highlighting how ULF addresses several key language phenomena. Then we discuss similarity measures for ULF pairs for quantifying annotator disagreement and parsing error. The current version of the annotation guidelines have been included in the supplemental files to the thesis in the ProQuest publication site. This version is also part of the supplemental files to Kim et al. (2021a) and the most recent versions will be maintained on the resources page of the ULF website.

## 2.1 ULF Design

*Unscoped logical forms* (ULFs) for EL capture the predicate-argument structure in the EL semantic type system and is the first critical step to producing fully-resolved semantic interpretation of sentences in the EL framework. Montague's (1973) profoundly influential work demonstrates that systematic assignments of appropriate semantic types to words and phrases allow 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

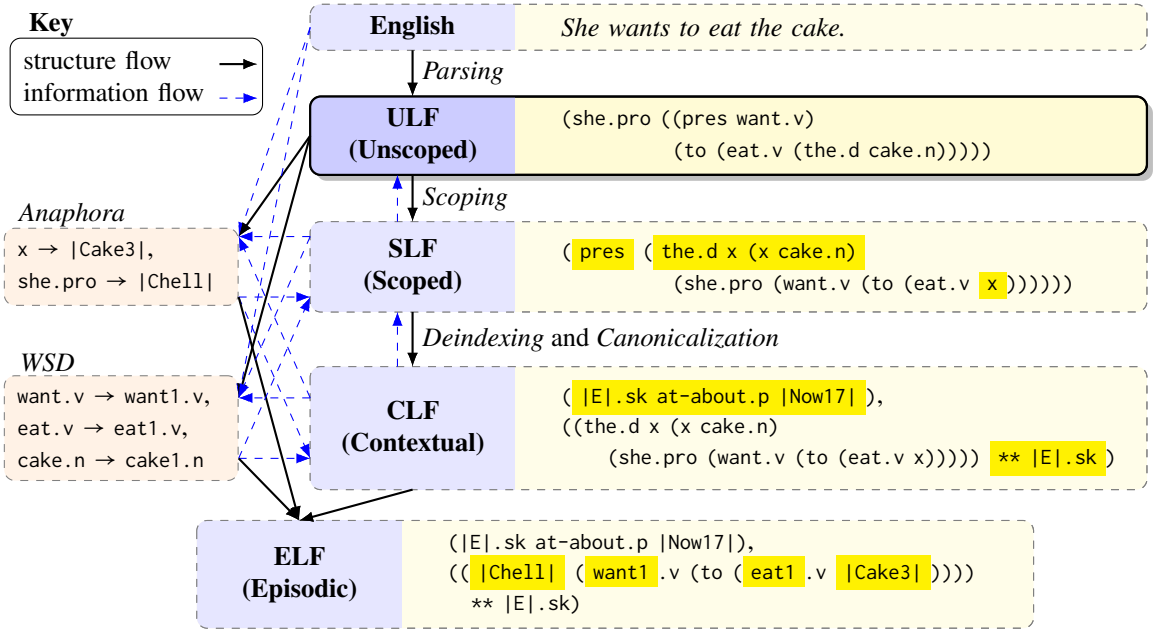



Figure 2.1: The semantic interpretation process with the ULF step brought to the fore. Structurally dependent steps in the interpretation process are connected by solid black arrows and structurally independent information flow is represented with dashed blue arrows. The components that changed from 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.

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.

These initial, type-coherent LFs capture the full EL meanings available for a lone sentence, which enable important classes of structural inferences which rely on signals from syntax: speech act inferences, presuppositional inferences, implicatives, and monotonicity inferences of the kind in natural logic. ULF captures the predicate-argument structure of EL while leaving word sense, scope, and anaphora unresolved. Therefore, ULFs can be analyzed using the EL semantic type system while taking these ambiguities into account.

ULF provides a preliminary surface-like logical form as a practical method of separating the various steps needed to generate fully resolved interpretations of natural language in EL. Figure 2.1 shows a diagram of how ULF fits into this multi-stage EL interpretation process. Section 2.1.3 elaborates on how ULF interacts with the remaining stages of meaning resolution after we have an understanding of how and what semantic information ULF captures.

### 2.1.1 ULF Syntax

As can be seen in Fig. 2.2, ULF structure quite closely reflects phrase structure. Atoms, typically, in ULF are marked with a type, which 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), using the stage- vs individual-level predicate distinction introduced by Carlson (1980). Though these types reflect syntactic, part of speech origins, they also correspond to semantic types (or a set of possible semantic types) in the EL type system. For example, the `.pro` suffix which corresponds to a pronoun also corresponds to the entity type, $\mathcal{D}$. A predefined set of logical functions and syntactic macro operators can appear as atoms in ULFs without any suffixed tag. These will be discussed further after describing the general syntax using the open-class atoms. Bracketing in ULF departs from EL by always using positions (in comparison to EL's use of square and angle brackets for infix and unscoped contexts) and simply inferring the operator position from the semantic types of the constituent. Atoms that are implicit and elided in the surface sentence and

(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) *He neglected three little bushes*

```
(he.pro ((past neglect.v) (three.d (little.a (plur bush.n)))))
```

(4) *Flowers are weak creatures*

```
((k (plur flower.n)) ((pres be.v) (weak.a (plur creature.n))))
```

(5) *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 2.2: Example sentences with corresponding raw ULF annotations. Items (1) and (2) are from the Tatoeba database, Items (3) and (4) are from *The Little Prince*, and Item (5) is from the Web. Strictly speaking, `weak.a` in Item (4) is actually missing a type-shifting operator `mod-n`, a simplification discussed in Section 2.1.5.

thus supplied by the annotator or parser are wrapped in curly brackets, for example, `{ref}.pro` in Item (1) of Fig. 2.2.

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`, `adv-e`, and `adv-f`) 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. Section 2.1.5 discusses how these dislocated annotations are identified and lifted to the appropriate scoping levels. 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 $(\mathcal{N} \rightarrow \mathcal{D})$).[1] 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 Item (5) in Fig. 2.2; this avoids immediate introduction of a $\lambda$-abstracted conjunction of predicates, simplifying the annotation task. Select phenomena, including macros, are discussed in more depth after first describing the semantic type structure in ULFs. Before moving on

[1] The actual semantic type of determiners in EL, after lambda-abstraction of the restrictor and body formula, is $(\mathcal{N} \rightarrow (\mathcal{N} \rightarrow (\mathcal{S} \rightarrow \mathbf{2})))$. See Section 2.1.6 for full details.

to discussing the underlying semantic type structure, here is a complete listing of suffix tags, logical operators, and macros with brief descriptions for a sense of the scope of ULF representations. For the suffix tags and logical operators, the corresponding semantic types or sets of types are listed. For sake of brevity, the sets of types are abbreviated using superscript notation for repeated arguments, e.g. $(\mathcal{D}^n \rightarrow \mathbf{2}), 0 \leq n \leq 2$ represents the set of types $\{\mathbf{2}, (\mathcal{D} \rightarrow \mathbf{2}), (\mathcal{D} \rightarrow (\mathcal{D} \rightarrow \mathbf{2}))\}$, and curly braces with pipes for alternatives, e.g. $(\{\mathcal{D}|\mathcal{N}\} \rightarrow \mathbf{2})$ represents the set of types $\{(\mathcal{D} \rightarrow \mathbf{2}), (\mathcal{N} \rightarrow \mathbf{2})\}$. These can be combined to indicate any combination of repeated alternatives. Here we only list *positional* types. The distinction between positional types and strict EL types is discussed in the next section (Section 2.1.2).[2]

**A Listing of Suffix Tags**

- `.n` : nominal predicate (*mouse*, *idea*, *domination*, etc.)

  $\mathcal{N}_N$ or $(\mathcal{D} \rightarrow (\mathcal{D} \rightarrow (\mathcal{S} \rightarrow \mathbf{2})))_N$ for relational nouns

- `.v` : verbal predicate (*run*, *love*, *laugh*, etc.)

  $(\{\mathcal{D}|\mathcal{N}\}^n \rightarrow (\mathcal{D} \rightarrow (\mathcal{S} \rightarrow \mathbf{2})))_V, 0 \leq n \leq 3$

- `.a` : adjectival predicate (*happy*, *green*, etc.)

  $(\mathcal{D}^n \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))_{ADJ}, 1 \leq n \leq 2$

- `.adv` : adverbial function

  - `.adv-a` : action/attribute modifying function (*quickly*, *angrily*, *confidently*, *very*, *quite*, *entirely, extremely*, etc.)

    $(\mathcal{N}_V \rightarrow \mathcal{N}_V)$

[2]The types listed here assume that syntactic features such as whether the expression is tensed are externally provided and thus do not model the propagation of such features. Further refining the types to fully model the interactions of syntactic types and features requires making additional distinctions regarding whether type compositions propagate the syntactic features of the argument or of the operator, as well as indicating when specific features are introduced or eliminated.

  - `.adv-e` : event modifying function (*here*, *yesterday*, etc.)

    $((\mathcal{S} \to \mathbf{2}) \to (\mathcal{S} \to \mathbf{2}))$

  - `.adv-s` : sentence modifying function (*definitely*, *probably*, etc.)

    $((\mathcal{S} \to \mathbf{2}) \to (\mathcal{S} \to \mathbf{2}))$

  - `.adv-f` : sentence-frequency modifying function (*twice*, *regularly*, *usually*, etc.)

    $((\mathcal{S} \to \mathbf{2}) \to (\mathcal{S} \to \mathbf{2}))$

- `.mod` : predicate modifier

  - `.mod-a` : adjective modifier (*very*)

    $(\mathcal{N}_{ADJ} \to \mathcal{N}_{ADJ})$

  - `.mod-n` : noun modifier (*fake*)

    $(\mathcal{N}_N \to \mathcal{N}_N)$

- `.cc` : coordinator (*and*, *or*, *but*, etc.); however, the truth-functional connectives *and, or, not* can also be used as-is

  $(\alpha^n \to \alpha), n \geq 0$ for any type $\alpha$. *(takes any number of a single type and returns the same type)*

- `.p` : prepositional predicate, taking a noun complement (*in*, *with*, etc.)

  $(\mathcal{D} \to (\mathcal{D} \to (\mathcal{S} \to \mathbf{2})))$

- `.p-arg` : argument-marking preposition (as in *ask **for** ..., tamper **with** ..., etc.*)

  $(\mathcal{D} \to \mathcal{D})$

- `.ps` : sentential preposition, taking a sentential complement (*when, before, until, while, if, as though*, etc.); such words are also termed "subordinating conjunctions", but we use "preposition" because Treebank parsers label them as such.

  $((\mathcal{S} \to \mathbf{2}) \to ((\mathcal{S} \to \mathbf{2}) \to (\mathcal{S} \to \mathbf{2})))$

- `.pq` : single-word prepositional question phrase, taking an inverted sentential complement (*when, where*, meaning "at what time", "at what place", as in "When did he leave", "Where does he live?")[3]

  $((\mathcal{S} \rightarrow \mathbf{2}) \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))$

- `.pro` : pronoun (*him*, *I*, *it*, etc.)

  $\mathcal{D}$

- `.d` : determiner (*the*, *some*, *few*, etc.)

  $(\mathcal{N}_{\{N|P\}} \rightarrow \mathcal{D})$

- `.aux-s`, `aux-v` : auxiliary (*do*, *will*, *may*, etc. acting at the sentence or verb phrase level.)

  $((\mathcal{S} \rightarrow \mathbf{2}) \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))$ for `.aux-s` and $(\mathcal{N}_V \rightarrow \mathcal{N}_V)$ for `.aux-v`.

- `.rel` : relative pronoun (*who*, *that*)

  $\mathcal{D}$

- `|_|` : name (i.e. proper noun) (*Mary*, *Star Wars*, etc.); not really an extension, but marks the type of an atomic element in the logical form.

  $\mathcal{D}$

- `.x` : expletive (*Damn*, *Ow*, etc.)

  $(\mathcal{S} \rightarrow \mathbf{2})$

- `.gr` : greeting (*hi*, *hello*)

  $(\mathcal{S} \rightarrow \mathbf{2})$

- `{_}.[suffix]` : implicit referent

  *Associated type for suffix*

[3] To avoid complications in our type structure, we take a simplistic view of wh-nominals as "place-holder" variables to be filled in with non-wh nominals in answering a question (for a more sophisticated view see (Ginzburg and Sag, 2000)).

### A Listing of Logical Operators

These operators are written without a suffix tag. They represent operations that are marked morpho-syntactically in English (making it difficult to handle using symbols that correspond to words in the source sentence) or have notable and consistent interpretations in EL such as reifiers.

- `not`/`and`/`or` : truth functional operators

  $((\mathcal{S} \rightarrow \mathbf{2})^n \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))$, $n = 1$ for `not` and $n \geq 1$ for coordinators. *(Coordinators generalize to other types as described in the suffix tag list.)*

- `plur` : pluralizer

  $(\mathcal{N}_N \rightarrow \mathcal{N}_N)$

- `past`/`pres`/`cf` : tense operators

  $((\mathcal{S} \rightarrow \mathbf{2})_{\text{untensed}} \rightarrow (\mathcal{S} \rightarrow \mathbf{2})_{\text{tensed}})$ when scoped to the sentence, $(*_{V,\text{untensed}} \rightarrow *_{V,\text{tensed}})$ when scoped to the verb. $*_V$ stands for any verb type.

- `perf` : perfect aspect

  $((\mathcal{S} \rightarrow \mathbf{2}) \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))$ *after lifting to the sentence-level alongside sentential modifiers*

- `prog` : progressive aspect

  $((\mathcal{S} \rightarrow \mathbf{2}) \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))$ *after lifting to the sentence-level alongside sentential modifiers*

- `pasv` : passive voice

  $(\alpha_{V,\text{lexical}} \rightarrow \alpha_{V,\text{lexical}})$, where $\alpha$ is a predicate of any number of arguments. *(operates lexically; in other words, only on atomic verb predicates)*

- `k` : kind operator (predicate reifier)

  $(\mathcal{N}_N \rightarrow \mathcal{K})$, NB: $\mathcal{K} \subset \mathcal{D}$

- `ka/to` : kind of action operator (action reifier)

  $(\mathcal{N}_{V,\text{untensed}} \rightarrow \mathcal{K}_A)$, NB: $\mathcal{K}_A \subset \mathcal{D}$

- `ke` : kind of event operator (event reifier)

  $((\mathcal{S} \rightarrow \mathbf{2})_{\text{untensed}} \rightarrow \mathcal{K}_E)$, NB: $\mathcal{K}_E \subset \mathcal{D}$

- `that` : *that* operator (proposition forming operator)

  $((\mathcal{S} \rightarrow \mathbf{2})_{\text{tensed}} \rightarrow \mathcal{P})$, NB: $\mathcal{P} \subset \mathcal{D}$

- `whether/ans-to` : yes/no question reifier[4]

  $((\mathcal{S} \rightarrow \mathbf{2})_{\text{tensed}} \rightarrow \mathcal{D})$

- `fquan/nquan` : quantifier forming operators

  $(\mathcal{N} \rightarrow (\mathcal{N}_{\{N|P\}} \rightarrow \mathcal{D}))$

- `set-of` : set forming operator

  $(\mathcal{D}^n \rightarrow \mathcal{D}), n \geq 1$

- `=` : equality operator

  $(\mathcal{D} \rightarrow (\mathcal{D} \rightarrow (\mathcal{S} \rightarrow \mathbf{2})))_{\{A|P\}}$

- `voc`, `voc-0` : vocatives (`voc-0` is for vocatives explicitly marked with a "O")

  $((\mathcal{S} \rightarrow \mathbf{2}) \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))$

- `mod-*` : predicate modifier constructor

  $(\mathcal{N} \rightarrow (\mathcal{N}_{ADJ} \rightarrow \mathcal{N}_{ADJ}))$ for `mod-a` and $(\mathcal{N} \rightarrow (\mathcal{N}_N \rightarrow \mathcal{N}_N))$ for `mod-n`

[4] As with the `.pq` suffix, we take a simplistic view of wh-nominals as "place-holder" variables to be filled in with non-wh nominals in answering a question to avoid complications in our type structure. This leads to question reifiers having the same type as `that`. In a more sophisticated treatment of questions, `whether` and `ans-to` would take different types of arguments.

- `adv-*` : adverbial function constructor

  $(\mathcal{N} \to (\mathcal{N}_V \to \mathcal{N}_V))$ for `adv-a` and $(\mathcal{N} \to ((\mathcal{S} \to \mathbf{2}) \to (\mathcal{S} \to \mathbf{2})))$ for the rest.

- `poss-by` : possession relation

  $(\mathcal{D} \to (\mathcal{D} \to (\mathcal{S} \to \mathbf{2})))$

**A Listing of Macros**

*macros* allow movement and simplifications of logical constructions that are simply communicated in language.

- `n+preds` : post-nomial modification
- `np+preds` : post-pronominal modification
- `n+post` : generalized post-nominal modification
- `pu` : phrasal utterance
- `'s` : prenominal-possession
- `sub` : "substitute", topicalized (or left-moved) constituents
- `rep` : "replace", rightward-movement of constituents
- `qt-attr` : interleaved quote attribution
- `that.rel`/`who.rel` : relativizers
- `it-extra.pro` : it-extraposition
- `it-cleft.pro` : it cleft constructions
- `my.d`, `their.d`, etc. : possessive determiners

- `mine.a`, `theirs.a`, etc : possessive adjectives
- `mind.pro`, `theirs.pro`, etc.: possessive pronouns

### 2.1.2 ULF Semantic Type Structure

The semantic type structure of ULF is analyzed in relation to EL with a separate, but closely related type system in ULF that reflects the types that constituents *should* have given their position in the core ULF operator-operand structure (without macros or dislocated elements of raw ULFs). We call these *positional* semantic types, in contrast to *EL* semantic types, which are the types that the constituents will ultimately have when resolved to EL. For example, the *EL* type of an episode-modifying adverb such as `today.adv-e` is $(\mathcal{S} \rightarrow \mathbf{2})$ since it predicates over an episode ($[[$|Gene| study.v$]$ ** $e] \wedge [e$ before |Now|$] \wedge [e$ today.adv-e$]]$, translated *"Gene studies" characterizes* $e$*,* $e$ *happened in the past, and* $e$ *happened today*), whereas the *positional* type is $((\mathcal{S} \rightarrow \mathbf{2}) \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))$ since its position in ULF seems to indicate that it maps sentence intensions to sentence intensions, e.g., `(today.adv-e (i.pro (past study.v)))`. When speaking only within the context of ULFs, we will simply refer to positional types as semantic types. Another common example of this is quantifiers. Quantified noun phrases (e.g., *"all dogs"*) can occur in any position where an individual variable or entity can occur, and in post-processing are replaced by bound variables. Therefore, unscoped NPs have *positional* type $\mathcal{D}$ and determiners (e.g., `all.d`) have *positional* type $(\mathcal{N}_N \rightarrow \mathcal{D})$, i.e. a function from a noun predicate[5] to an individual. Ultimately, scoped determiners are interpreted in EL via satisfaction conditions over a quantified variable and two formulas (a restrictor formula and the nuclear scope). The introduction of *positional* types makes reasoning about ULF compositions simpler by allowing one to reason directly over the ULF constituents rather than considering result of resolving the formula to EL.

The type-shifting operators mentioned in the previous section are crucial for type coherence in ULFs. In Item (1) of Fig. 2.2 (repeated below),

[5] In practice determiners also allow prepositional predicates to handle cases such as *"all of them went home"*.

*Could you dial for me?*

```
(((pres could.aux-v) you.pro
  (dial.v {ref1}.pro (adv-a (for.p me.pro)))) ?)
```

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 $\mathcal{N}$, shorthand for $(\mathcal{D} \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))$, corresponding to adjectives, nouns, and verbs, respectively.[6] `(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. An example of its predicative use is *"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) expression `me.pro`.[7] If we apply `(for.p me.pro)` to another argument, such as `|Snoopy|` (the name of a puppy), we obtain a sentence intension.[8] So semantically, `adv-a` is a *type-shifting operator* of type $(\mathcal{N} \rightarrow (\mathcal{N}_V \rightarrow \mathcal{N}_V))$, type shifting a predicate to a verb modifier. Of course, the name `adv-a` is intended to suggest "adverbial", in recognition of the grammatical distinction between predicative and adverbial uses of prepositional phrases.

This brings up the issue of *intensionality*, which is preserved in ULF. Item (2) from Fig. 2.2 (repeated below),

*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))))))
```

is a counterfactual conditional, and the consequent clause *I would be able to succeed* is not evaluated

[6] These syntactic refinements, $\mathcal{N}_{ADJ}$, $\mathcal{N}_N$, and $\mathcal{N}_V$, simply boil down to $(\mathcal{D} \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))_{ADJ}$, $(\mathcal{D} \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))_N$, and $(\mathcal{D} \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))_V$, respectively. For atomic expressions, the subscripts $ADJ$, $N$, and $V$ mark the syntactic sort of the expression. For composed expressions, they mark the syntactic sort of the operator that generated this type.

[7] Viewing $n$-place predicates as successively applied to their arguments, each time reducing the adicity, is in keeping with the traditions of Curry, Schönfinkel, Church, Montague, and others—hence "curried" predicates.

[8] `(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).

in the actual world, but in a possible world where the (patently false) antecedent is imagined to be true (Veltman, 1976; Kratzer, 1979, 1981). 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.

ULF has operators that reify different semantic categories, such as `to` in Item (2) of Fig. 2.2, `k` in Item (4) (repeated below),

*Flowers are weak creatures*

```
((k (plur flower.n)) ((pres be.v) (weak.a (plur creature.n))))
```

and `that` in Item (5) (repeated below),

*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))))))))
```

shifting these semantic categories 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 Item (4) 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—for example enabling generalized quantification (as in Item (5)), modification, reification, and other forms of intensionality.

### 2.1.3 Role of ULF in Comprehensive Semantic Interpretation

Now that we have an understanding of the ULF syntax and underlying type system we can discuss its place in the full EL interpretation process. ULF's surface-like form and the type structure that it encodes makes it well-suited to reducing underspecification by using well-established linguistic principles and exploiting the distributional properties of language. Figure 2.1 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 while resolving 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 Item (4) does not specify which of the dozen WordNet senses of *weak* or three senses of *creature* are intended here, the type structure is perfectly clear: A predicate modifier is being applied to a nominal predicate. A slightly subtler issue is the adicity of predicates. ULF does not assume unique adicity of word-derived predicates such as `run.v`—only restricting an operator's adicity to the possibilities that its suffix-subcategorized constituent type makes available—since such predicates can have intransitive, simple transitive, and other variants (e.g., *run quickly* vs. *run an experiment*); 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, such that the full sentence forms a sentence intension, when an argument (the "subject") is placed on its left, as in English.

Linguistic constraints—e.g., *binding constraints* (Chomsky, 1995)—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 *he* is a sibling of an ancestor of *John*. ULF preserves the syntactic structure necessary to make such determinations, making the use of such constraints possible. While

ULF 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 or whole documents.

While access to ULFs is needed in order to apply surface-structural constraints, substitutions of constants (possibly from earlier sentences) for pronouns and definite descriptions may be applied at any step of the EL interpretation process. We indicate this in Fig. 2.1 by placing this resolution parallel to the ordered steps in EL interpretation and incorporating the results at the final step.

**Scoping.** Unscoped constituents (determiners, tense operators, sentence modifiers, and coordinators) can generally "float" to more than one possible position. Following a view of scope ambiguity developed by (Schubert and Pelletier, 1982) and 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, 2011; Barker, 2021). 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*, i.e. making events and situations explicit (Hwang, 1992; Hwang and Schubert, 1994; Schubert and Hwang, 2000). Deindexing is the process of removing context dependence (*indexicality*) in the representation. Here, I will restrict the usage of "deindexing" to event-indexicality, such as tense and aspect, since anaphora was already discussed. This process corresponds to the introduction of event variables for explicit linking to the characterizing sentence via the '**' operator (this variable becomes `|E|.sk` in Fig. 2.1 after Skolemization). Hwang and Schubert's approach to tense-aspect processing is to construct *tense trees* for temporally relating event variables which yields multiple related reference events for sentences such as *"The employees*

**Sentence**

*"The employees will know that they had been working for 15 hours"*

**ULF**

```
((the.d (plur (employee-of.n {ref}.pro)))
 ((pres will.aux-s)
  (know.v
   (that (they.pro ((past perf)
                    (prog (work.v
                           (adv-e (for.p (k (15.a (plur hour.n))))))))))))
```

**Scoping + Anaphora Resolution**

(pres$_{\uparrow \mathbf{a}}$ (will.aux-s$_{\uparrow \mathbf{b}}$
(the.d $x$: ($x$ (plur (employee-of.n {ref}.pro)))
($x$ (know.v (that$_{\uparrow \mathbf{c}}$ (past$_{\uparrow \mathbf{d}}$ (perf$_{\uparrow \mathbf{e}}$ (prog ((adv-e (for.p (k (15.a (plur hour.n)))))
($x$ work.v)))))))))))

**Deindexed**

($\exists u_0$ : [$u_0$ same-time Now$_0$]
[[|Speaker| tell.v |Hearer| (that
($\exists e_0$ : [$e_0$ at-about $u_0$]
[($\exists e_1$ : [$e_1$ after $e_0$]
[(the.d $x$ :
[$x$ (plur (employee-of.n {ref}.pro))]
[$x$ know.v (that
($\exists e_2$ : [$e_2$ before $e_1$]
[($\exists e_3$ : [$e_3$ until $e_2$]
[[[$e_3$ lasts-for (k (15.a (plur hour.n)))]$\wedge$
[$x$ work.v]] ** $e_3$])
** $e_2$]))])
** $e_1$])
** $e_0$]))]
** $u_0$])

Figure 2.3: An example of temporal deindexing for the sentence *"The employees will have been working for 15 hours"*. The clausal structure and type-shifter positions in ULF define the bounds for the operator scope order. These operators are then traversed top-down to construct tense trees and deindex the formula. The $\uparrow \mathbf{a}$, $\uparrow \mathbf{b}$, etc. subscripts in the scoped formula mark the positions where a node is added to the tense tree and the corresponding tense tree states are shown in Fig. 2.4.

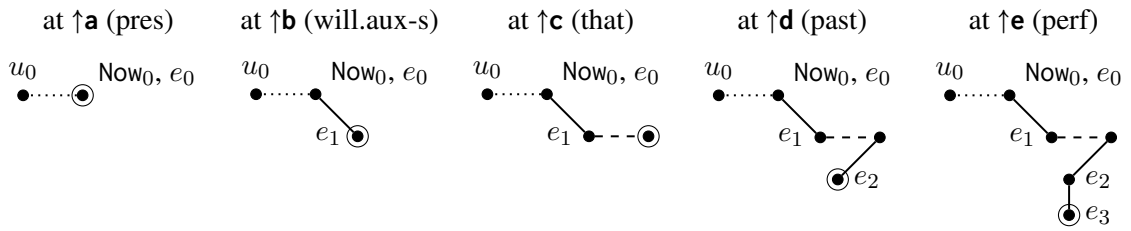



Figure 2.4: Tense trees corresponding the progressive deindexing steps for the example in Fig. 2.3. The steps are labeled with ↑**label** followed by the operator corresponding to this deindexing step in parentheses, e.g., (pres). The trees are built and read top-down, with nodes at the same level read left-to-right. The tree has three branching options: *past* ↙, *perf* ↓, and *futr* ↘ (written in this document as *will.aux-s* to reflect ULF conventions). The tree also has two horizontal embedding options: *re-traversable* ⇢ and *non re-traversable* -→. The circle indicates the focal node of the tense tree, which situates the next event in the deindexing process. When interpreting multiple sentences, the focus returns to the utterance node at the top of the tense tree.

*will know that they had been working for 15 hours"*. The compositional construction and use of tense trees is only possible if the LF being processed reflects the original clausal structure—as ULF indeed does. Figure 2.3 shows a deindexing example with the previously mentioned sentence and Fig. 2.4 shows the corresponding tense trees.

The clausal structure and type-shifter positions in ULF define the bounds for the operator scope order. In this example, the sentence reifier, `that`, acts as a scoping barrier for scoping operators in its operand. Otherwise, tense scopes at the top-level and other temporal operators scope in the original clausal order. This scoped formula is then traversed top-down to construct tense trees and derive a deindexed form. If the sentence under consideration were part of a longer discourse, the deindexing operations would have involved re-traversal of the tense tree resulting from the preceding sentences, adjoining the new event variables to those already at the nodes of that tense tree, and adding discourse-dependent `orients` relations between certain ones of these adjoined events. The progressive aspect and temporal adverbial are not reflected in the tense tree, but lead to predications about the relevant events. The general interpretation of the `perf` operator as an `impinges-on` relation is narrowed down to an `until` relation (as opposed to `before`) because `prog` specifies an unbounded aspectual type of the formula. More straightforwardly, the adverbial *"for 15*

*hours"* leads to a predication about how long the modified event lasts. For further details regarding the deindexing process, please refer to Hwang (1992) and Hwang and Schubert (1994).

Canonicalization is the mapping of an LF into "minimal", distinct propositions, with top-level Skolemization (and occasionally $\lambda$-conversions). The CLF step in Fig. 2.1 contains two separate formulas as a result of this process.[9]

**Episodic Logical Forms.** 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.[10] 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).

### 2.1.4 Handling Copulas

For the sake of uniformity, the copula, `be.v`, is always a binary operator that takes an entity type argument as the subject and a predicate type argument as its object, $(\mathcal{D} \rightarrow (\mathcal{N} \rightarrow (\mathcal{S} \rightarrow \mathbf{2})))_V$. Most of the time, this is perfectly natural, as in Item (1). This leads to a perhaps unintuitive annotation for sentences where the object of the copula is a noun phrase with an existential quantifier, as in Item (2).

(1) *"I am happy"*

```
(i.pro ((pres be.v) happy.a))
```

[9] Prior to Skolemization, in the representation immediately following deindexing (not shown in Fig. 2.1), an existentially scoped variable would characterize the sentence and a logical conjunction would join the two propositions.

[10] EPILOG, at its release, was competitive against the state-of-the-art FOL theorem provers of the time (Morbini and Schubert, 2009).

(2) *"I am a lawyer"*

```
(i.pro ((pres be.v) (= (a.d lawyer.n))))
```

An equality operator for Item (2) ensures type consistency without deleting the determiner.

### 2.1.5 Predicate and Sentence Modification

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. $(\mathcal{N} \to \mathcal{N})$, 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; *"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. Ultimately in EL, `adv-a`, `adv-e`, and `adv-f` will be reconstrued as predications

(not modifications) over actions and events via meaning postulate inferences. 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 (more generally, subject) of a sentence and the episode it characterizes.[11] 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—leaving *"he"* and *"with a dog"* indexical for simplicity and clarity—[[[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 formula—again leaving the modifier indexical for simplicity—[[[she.pro eat.v] ** E1.sk] and.cc [(pair she.pro E1.sk) (at.p (a.d cafe.n))]].

A formula or nonatomic verbal predicate (a verb phrase) 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|))`.

[11] Such agent-episode pairs are needed because episodes (and relatedly reified propositions) in EL are unstructured. They are individuals in the domain of discourse that are related to formulas through specific operators: the episodic operators such as '**' (Section 1.3.1) for episodes and the sentence reifier `that` for reified propositions. The structured proposition approaches to semantic interpretation instead include the internal structures of sentences directly in the propositional representations resulting from their interpretations (Kaplan, 1989; Soames, 1985, 1987, 1989; Salmon, 1986b,a).

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))))`.

A type of modification not covered above is exemplified by *"Seattle skyline"*. For the corresponding type, $(\mathcal{D} \rightarrow (\mathcal{N}_N \rightarrow \mathcal{N}_N))$, we provide type-shifter `nnp`, indicating premodification of a nominal predicate by an entity type (in this case a proper noun), e.g.,

`((nnp |Seattle|) skyline.n)`.

All of the operators just discussed are listed alongside a ULF example, the positional type, and the EL type in Table 2.1. 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.

Table 2.1: Predicate and sentence modifier forming operators in ULF along with examples and their semantic (positional) 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-a 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)))` | $((\mathcal{S} \rightarrow \mathbf{2}) \rightarrow (\mathcal{S} \rightarrow \mathbf{2}))$ |

In practice, we're able to drop the `mod-a`, `mod-n`, and `nnp` type-shifters during annotation unless the modifying predicate is a verb since we can post-process them with the appropriate type-shifter to make the composition valid. This is not generally possible with modifiers that are verbs since

verbs may take a predicate argument, leading to ambiguity as to whether the verb is meant to be acting as a predicate or a modifier. If not a verb, 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, it does not necessarily indicate that the modification is non-intersective.[12] 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))))))).
```

Thus ULF distinguishes postnominal modifiers, which are always intersective, from prenominal ones, which are often non-intersective. Section 2.1.7 explains `n+preds` further and discusses this in more depth. The EL type system also makes the distinction between intensional and extensional modification for sentence modifiers. This is in terms of whether the modifiers predicate over the episodes (e.g., *"in the forest"*, *"after dinner"*) or map sentence intensions to sentence intensions (e.g., *"probably"*, *"according to the report"*), in the EL types of course. In the positional types all of these have the same sentence intension modifying type. These two cases are annotated using `adv-e` and `adv-s` operators for extensional and intensional modification, respectively. Both have lexical and modifier forming variants just like `adv-a`.

[12] For example, we would annotate *"triangular container"* as `((mod-n triangular.a) container.n)`, but since this relation is intersective, given the correct background knowledge, we would infer that this has the same meaning as `(λx ((x triangular.a) and.cc (x container.n)))`.

**Predicate vs. Sentence Modifiers**

Annotation of modifiers in ULF requires distinguishing predicate modifiers from sentence modifiers, since these have different semantic types and need to be post-processed to the correct position where it operates over an argument of the appropriate type. If a modal auxiliary or modal adverb(ial) modifies a sentence without affecting what the sentence-level predicate says about the subject (e.g., *A major earthquake {may, could} occur; {perhaps, surprisingly, in my opinion} there is no life on Mars*), then it is a sentence-level modifier, thus annotated with `adv-s`, `adv-e`, or `adv-f` type if it an adverbial phrase and `aux-s` if it is an auxiliary. If instead the modal auxiliary or adverb(ial) alters the property attributed to the subject, then it must be a predicate-level modifier (e.g., *The cadet must (i.e., is obligated to) obey; the skater {nearly, awkwardly} fell*) and annotated as `adv-a`, `mod-a`, or `mod-n` if it is a predicate modifier, and `aux-v` if it is an auxiliary.These operators are then lifted to an appropriate position in postprocessing (e.g., `adv-s` and `aux-s` to sentence-level, `adv-a` to the complete verb phrase level, etc.).

This distinction can be quite subtle since it is dependent on both the lexical entry and the syntax. Consider the following sentences:

(a) *"Mary confidently spoke up"*

(b) *"Mary undoubtedly spoke up"*

(c) *"Koko is surprisingly intelligent"*

(d) *"Surprisingly, Koko is intelligent"*

In sentence (a) *confidently* is a predicate modifier whereas in sentence (b) *undoubtedly* is a sentence modifier. Clearly, this is entirely determined by the lexical entry since the syntax trees of the two sentences are identical. Then compare sentence (c) and sentence (d). The only difference between them is the placement of the modifier *surprisingly*, which changes its semantic type.

### 2.1.6 Quantifier Semantics

Quantified expressions can occur in any position where 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, $\mathcal{D}$. Thus, we can treat determiners such as `every.d` in ULF as if they were of type $(\mathcal{N}_{\{N|P\}} \rightarrow \mathcal{D})$—i.e. a function from a (nominal or prepositional) 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 nominal monadic predicate, $\mathcal{N}_N$, we can infer that the *positional* type of the determiner in this example is $(\mathcal{N}_N \rightarrow \mathcal{D})$. In general, since determiners can also take prepositional phrase arguments, e.g., *"All of us run"*, determiners have positional type $(\mathcal{N}_{N,P} \rightarrow \mathcal{D})$. This will be transformed after scoping into a formula of the form $(\delta\ v : \phi\ \psi)$, where $\delta$ is the determiner, and $\phi$ and $\psi$ correspond to the formulas resulting from substituting the scoped variable into the restrictor and body 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"*

$(\text{most}\ v : \phi\ \psi)^{\mathcal{MU}} = 1$ iff for most $d \in \mathcal{D}$ for which $\phi^{\mathcal{MU}_{v:d}} = 1$, $\psi^{\mathcal{MU}_{v:d}} = 1$

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

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

where '**' is the operator relating a sentence to the episode it *characterizes* (describes as a whole),

which is discussed in Section 1.3. $(\delta\ v : \phi\ \psi)$ can equivalently be rewritten as $((\delta\ (\lambda v\ \phi))\ (\lambda v\ \psi))$ where we define $\delta$ as a second-order intensional predicate of type $(\mathcal{N} \rightarrow (\mathcal{N} \rightarrow (\mathcal{S} \rightarrow \mathbf{2})))$ similar to the approach used in generalized quantifier theory (Barwise and Cooper, 1981).

### 2.1.7 Macros

ULF macros are syntactic rewriting operators for reducing the annotator burden of encoding complex, but regular, semantic structures or for avoiding unnecessary word reordering. The development of macros was closely tied to the development of the annotator and dataset construction, which is elaborated upon in Section 3.1. In addition to reducing annotation burden, they bring ULFs closer to surface form making them easier to parse and suitable for applying syntactic constraints (e.g., *island constraints*) to analysis or inference, without sacrificing nuance in the underlying semantic relationships.

Table 2.2 lists the definitions and simple examples of the several commonly used 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 answered the door"*.

`n+preds` and `np+preds` are macros for handling post-nominal and post-pronominal modification. `n+preds` modifies the noun predicate in its first argument position intersectively with the following predicates and returns the resulting complex noun. `np+preds` modifies the entity in its first argument position with the following predicates and returns the modified entity. Intuitively, `np+preds` handles non-restrictive modifiers, whereas `n+preds` handles restrictive modifiers. This makes sense since for a quantified phrase of the form `(det.d (n+preds A.n B.prd))` the modification from `n+preds` is

Table 2.2: A list of common rewriting macros in ULF with their expansion definitions and an example. $=_m$ is the macro defining operator and $\Rightarrow_m$ is a macro application.

| **Name** | **Definitions** | **Example** |
|---|---|---|
| sub | (sub C S[*h]) $=_m$ S[*h←C] | (sub A (B *h)) $\Rightarrow_m$ (B A) |
| rep | (rep S[*p] C) $=_m$ S[*p←C] | (rep (A *p) B) $\Rightarrow_m$ (A B) |
| n+preds | (n+preds N $P_1$ ... $P_n$) $=_m$<br>($\lambda$x ((x N) and.cc (x $P_1$) ... (x $P_n$))) | (n+preds dog.n red.a) $\Rightarrow_m$<br>($\lambda$x ((x dog.n) and.cc (x red.a))) |
| np+preds | (np+preds NP $P_1$ ... $P_n$) $=_m$<br>(the.d ($\lambda$x ((x = NP) and.cc<br>(x $P_1$) ... (x $P_n$)))) | (np+preds he.pro red.a) $\Rightarrow_m$<br>(the.d ($\lambda$x ((x = he.pro) and.cc<br>(x red.a)))) |

applied before the determiner operates over it, thus the modifiers are included in the restrictor of the quantification.

The 's macro handles prenominal genitive constructions in English (e.g., *"John's dog"*). It is defined separately for two cases. The first case directly relates the prenominal NP to the relational noun and the second is a more general possessive relation to the noun.

((NP 's) (_-of.n *s)) $=_m$ (the.d (_-of.n NP))

((NP 's) N) $=_m$ (the.d (n+preds N (poss-by NP)))

The first case is used in cases where possessive occurs with a relational noun that directly relates to the prenominal NP, using *s to mark of position where the prenominal NP should go. For example, *"Aarav's mother"* is annotated and expanded as follows.

((|Aarav| 's) (mother-of.n *s)) $\Rightarrow_m$ (the.d (mother-of.n |Aarav|))

Via this macro expansion, *"Aarav's mother"* is rewritten to the equivalent expression, *"the mother of Aarav"*.

The more general case modifies the noun with a poss-by relationship. poss-by is a binary predicate indicating general, unspecified possession between two entities which is only distinguished from relational possession, which is lexicalized in the relational predicate. For example, *"Lorena's dog"* is annotated and expanded follows.

((|Lorena| 's) dog.n) $\Rightarrow_m$ (the.d (n+preds dog.n (poss-by |Lorena|)))

This shows that for possession relations in more general cases involve a silent possession operator. That is, *"Lorena's dog"* translates to *"the dog possessed by Lorena"*. Notice that this definition uses a macro itself. We can recursively use macro definitions for simplifying the interface from the syntax to the underlying semantic representation.

An additional layer is built for possessive determiners, pronouns, and adjectives to further lexicalize and simplify the annotations. Possessive determiners get defined to their expanded counter part. For example, *"my"* is expanded to *"me's"*.

`my.d` $=_m$ `(me.pro 's)`

`my.d` and `me.pro` can be replaced by any corresponding pair of possessive determiner and personal pronoun, e.g., *your*/*you*, *their*/*they*, etc.

Similarly for possessive pronouns and possessive adjectives,

`mine.a` $=_m$ `(poss-by me.pro)`

`mine.pro` $=_m$ `((me.pro 's) {ref}.n)`

`mine.a`, `mine.pro`, and `i.pro` can be replaced by corresponding possessive pronoun and personal pronoun. Here are basic examples of their use.

*"That is mine"*

`(that.pro ((pres be.v) mine.a))` $\Rightarrow_m$ `(that.pro ((pres be.v) (poss-by me.pro)))`

*"Mine is red"*

`(mine.pro ((pres be.v) red.a))`

$\Rightarrow_m$`(((i.pro 's) {ref}.n) ((pres be.v) red.a))`

$\Rightarrow_m$`((the.d (n+preds {ref}.n (poss-by i.pro))) ((pres be.v) red.a))`

Here we've only address macros to the extent of describing their general purpose in ULF. Please refer to the annotation guidelines (released by Kim et al. (2021a) alongside the annotated dataset as supplementary material and included as a supplementary document to this thesis) for an explanation of how every macro interacts with specific language phenomena.

### 2.1.8 Topicalization & Relative Clauses

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 suffix. Relative clauses present an interesting case where multiple macros interact in a regular manner. For example, *"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)))))
```

which uses n+preds to indicate that the relative clause is an intersective modifier, that.rel to handle the relativizer, and sub to place the relativizer in the appropriate position. During macro-expansion and post-processing, the embedded sentence in which the .rel variable lies is $\lambda$-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)).

### 2.1.9 Conclusion

This concludes the description of ULF in this dissertation. This section focused on describing the core syntax and semantics of ULF and highlighting the relationship between surface form and the semantic structure that ULF models. This information should be sufficient to understand the remaining chapters in this dissertation regarding work on parsing and inference, we well as understand the arguments motivating its use in developing language understanding systems: that is, the rich compositional semantic type information which ensures that a language understanding system makes appropriate semantic distinctions and avoids major pitfalls in inference which may happen with an insufficiently expressive model of meaning.

This is far from an exhaustive presentation of ULF. Please refer to the annotation guidelines in the supplemetary materials (access locations described in the introduction of Chapter 2) for a thorough discussion of annotation subtleties and a description of all language phenomena that ULF can currently handle. In addition, please see the ULF project website, currently at `http://cs.rochester.edu/u/gkim21/ulf/`, to stay up-to-date on developments in ULF. Now we move on to describing evaluation techniques which are used across many projects in this dissertation and the long-term process of manually annotating a dataset of English-ULF pairs before moving on to the next chapter.

## 2.2 Measuring Similarity

In order to make quantitative judgments about annotation and parsing quality, we need ways to measure the similarity between two ULFs. On this front we take inspiration from AMR. Here I discuss two standard methods of comparing AMR parses and how we can make them compatible with use on EL formulas and ULFs. One key benefit of using existing metrics is that we can make rough comparisons with similar work in the AMR literature, whether that be interannotator agreement statistics or parser performance, to measure progress on this research project. This is

especially important for the initial stages of the research project when the methods and systems are in their infancy and cannot be evaluated on their use in downstream tasks.

### 2.2.1 EL-SMATCH

EL-SMATCH is a generalized formulation of SMATCH (Cai and Knight, 2013), the standard evaluation metric for AMR parsing. SMATCH represents each logical form as a conjunction of triples of three types.

1. *instance(variable, type)*
2. *relation(variable, variable)*
3. *attribute(variable, value)*

Every node instance of the logical form is associated with a variable, and the nodes are described and related to each other using the above triples. Thus, *type* and *value* can both only be atomic constants. The logical format in Fig. 1.4 shows how a conjunction of such triple representations with existentially quantified variables represents an AMR. The SMATCH score is then defined as the *maximum f-score (of triples) obtainable via a one-to-one matching of variables between the two formulas* (Cai and Knight, 2013). In order to capture non-atomic operators in EL and ULF, we introduce an additional triple,

*instance(variable, variable)*.

The first variable argument is associated with the instance, and the second variable argument, with the type.

With this addition to the representation, we can syntactically map EL formulas and ULFs into a conjunction of triples by introducing a node variable for every component of the formula and then describing and relating the components using the triples. That is, we can generate left-to-right arcs in the order that they appear (:ARG0, :ARG1, etc.), assuming the leftmost constituent is the

parent. Since the representation used by SMATCH is the same as that of AMR, we can map the triple representation into a graph representation in the same manner as AMR formulas. Figure 2.5 shows an example of the use of the new instance triple in mapping the EL formula for *"I am very happy"* into these representations. Note, however, that this mapping does not relate the semantics of EL or ULF to AMR since the interpretation of the triples differ for the two formalisms.

In practice, instead of using the new *instance(variable, variable)* triple, we use represent this with a `COMPLEX` node label and an `:INSTANCE` edge marking the identity of a non-atomic operator. This allows us to reuse systems designed for AMR without extending it to support a new triple type, but requires discounting the extra node that gets introduced.

**EL**

```
[me.pro (very.mod-a happy.a)]
```

**EL-SMATCH Triple Representation**

```
instance(a, very.adv) ∧
instance(b, happy.a) ∧
instance(d, me.pro) ∧
ARG0(a, b) ∧
instance(c, a) ∧
ARG0(c, d)
```

**EL-SMATCH Graph Representation**

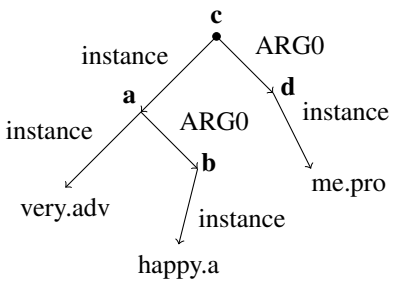



Figure 2.5: Example of syntactic mapping from EL to EL-SMATCH triple and graph representations for sentence *"I am very happy"*.

### 2.2.2 SEMBLEU

SEMBLEU (Song and Gildea, 2019) is an alternative metric for measuring similarity of AMRs. This metric extends BLEU (Papineni et al., 2002) to node- and edge-labeled graphs. BLEU is designed for text and measured over token subsequences, n-grams, of the two texts. It is defined as follows for two token sequences, $s_g$ for the gold sequence and $s_p$ for the prediction sequence

$$\text{BLEU}(s_g, s_p) = \text{BP} \exp \sum_{k=1}^{n} w_k \log p_k(s_g, s_p)$$

where BP is the brevity penalty

$$\text{BP} = \exp \min(1 - \frac{|s_g|}{|s_p|}, 0),$$

$w_k$ is the weight for classification of n-grams of length $k$, and $p_k$ is a measure of overlap between $k$-grams between the two sequences which is normalized by the $k$-gram counts of the prediction.

SEMBLEU substitutes graph path segments for n-grams in the BLEU equations and makes the same computation. It has been shown to have better agreement than SMATCH with human judgments of AMR similarity. An important feature of SEMBLEU is the use of bigger subgraphs than SMATCH (which is built completely out of counting single edge or node matches) which results in scores that are sensitive to the connectedness of the graphs. For example, a graph with no edges but all of the correct nodes will achieve 0.5 SMATCH score, but near 0 SEMBLEU score. This leads to a more discriminating metric which is clearly reflected in our parser evaluations in Chapter 4 where the highest SEMBLEU scores were almost 4 times the lowest whereas the highest EL-SMATCH scores were less than twice the lowest scores.

We use the same trick as EL-SMATCH to make this metric compatible with ULFs. That is, decomposing ULF into individual instance, attribute, and edge nodes with an addition instance definition method for representing non-atomic operators.

# 3 Annotation

This chapter details the components of manually annotating ULFs and verifying their quality: the annotation interface and its development, interannotator agreement, and information about the released dataset.

## 3.1 Annotation Development & Interface

The development of the annotation interface and the look of ULFs were driven by preliminary pilot annotation projects which revealed major limitations in the original ULF and annotator designs. This section will discuss the pilot annotations and the resulting changes chronologically before describing the final annotator used for gathering the manually annotated ULF dataset.

We performed two pilot annotations early in the project, one in the Fall of 2016 and one in the Fall of 2017. We measured annotation difficulty in terms of interannotator agreement (IAA) and annotation speed and received feedback on the annotation interface from annotators.

### 3.1.1 Pilot Annotation 1, Fall 2016

We performed our first exploratory annotation effort early in the project after developing an initial version of the annotation guidelines and a command-line based annotation tool inspired by the AMR

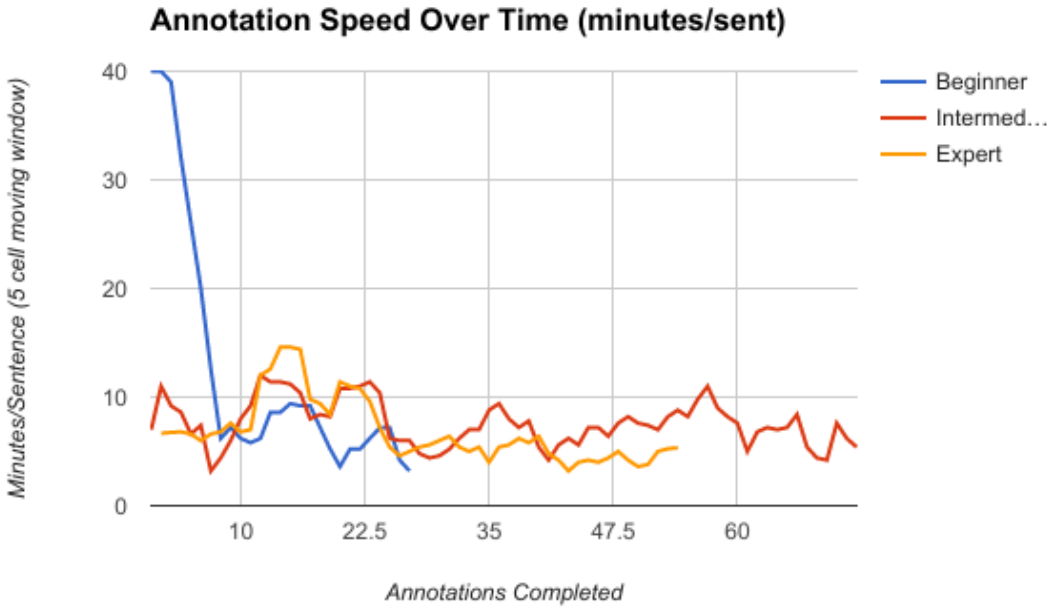



Figure 3.1: Timing results from ULF experimental annotations.

Editor (Hermjakob, 2013). We had three annotators of varying degrees of expertise in EL: one expert, one intermediate, and one beginner. The sentences we annotated were randomly sampled from the Brown corpus, filtered to limit the sentence length to 17 words (the average sentence length in the corpus). Each annotator annotated between 27 and 72 sentences from the same set of sentences to measure correlation.

We found that annotators can learn to annotate quickly after practicing on a fairly small set of sentences and that the sentences could be annotated fast enough for building a corpus to be feasible. Figure 3.1 shows a plot of timing results by annotator and the number of annotation completed with a 5-cell moving window average to control for variable individual sentence difficulty. We can see that the intermediate and expert annotators have relatively consistent annotation speeds as a function of annotations completed, as we would expect from their familiarity with the formulation. The beginner on the other hand, whose only annotation experience was with artificial and short sentences

from the annotation tutorial, developed an annotation speed comparable to the intermediate and expert with 10 examples.

Table 3.1 shows that a practiced annotator averaged less than 8 minutes per sentence. This is promising since it is less than the 10 minutes/sentence annotation speed reported on the AMR project website[1], which has been successful in annotating over 40,000 sentences.

| **Annotator** | **Minutes/Sentence** |
|---|---|
| Beginner | 12.67 |
| Beginner (- first 10) | 6.83 |
| Intermediate | 7.70 |
| Expert | 6.87 |

Table 3.1: Average timing of experimental ULF annotations.

We also found that ULF did not run into sentences that could not be handled by ULF's expressive capacity in the Brown corpus. There were only a couple of cases where the expert and intermediate annotators were unable to generate a confident interpretation in terms of normal sentence semantics. The annotators were stumped in cases of ill-formed sentences such as fragments.

The expert and intermediate annotators had 53 annotations that were common between them, from which we computed IAA using EL-SMATCH. The 53 annotation pairs had an interannotator agreement of 0.48. This was insufficient, but expected from exploratory annotations. A review of the discrepancies showed that they were caused by the following issues (in order of severity).

1. The annotators could not consistently annotate phenomena that require movement of large phrases, such as prepositional modifiers which usually post-modify in English, but form prefix-operators in ULF.
2. Ill-formatted text, such as fragments, were not discussed in the preliminary guidelines.
3. Language phenomena that were not carefully discussed in the guidelines, so the standard representation was unclear to the annotator.

[1] http://amr.isi.edu/editor.html. Visited 4/24/2017.

Additionally, we found that the AMR style annotator was not helpful because the similarity between ULF and surface form means that it's easier to directly annotate on the surface word than building up a graph.

Based on these results, we changed the annotator style to that closer to GMB (Bos et al., 2017). The annotation collection was changed to be collaborative so that all annotators have a shared view and experts can review and correct annotations. This setup reduces annotation error and streamlines the annotator training process. Additionally, each sentence annotation was broken up into multiple stages. Here is an example of the stages for the sentence *"Mary loves to solve puzzles"*:

1. Group syntactic constituents (NPs, ADJPs, VPs, etc) using round brackets:
   `(Mary (loves (to (solve puzzles))));`

2. Run a POS tagger over the original sentence:
   `(nnp Mary) (vbz loves) (to to) (vb solve) (nns puzzles);`

3. Make any necessary corrections to tags, and then use them as dot-extensions in the bracketed sentence (the tag to dot-extension will be automated):
   `(Mary.nnp (loves.vbz (to.to (solve.vb puzzles.nns))));` *No corrections needed*

4. Convert POS extensions to logical-types, and separate tense and plural as operators:
   `(|Mary| ((pres love.v) (to (solve.v (plur puzzle.n)))));`
   (`|_|` ↔ name (proper noun); `.v` ↔ verbal predicate; `.n` ↔ nominal predicate; `to` without an extension is a special reifying operator);

5. Add any necessary implicit operators (typically, type-shifting operators):
   `(|Mary| ((pres love.v) (to (solve.v (k (plur puzzle.n))))));`
   (`k` converts a predicate that is true of ordinary singular or plural entities into a kind – i.e. an abstract individual whose instances are ordinary entities; it is applied whenever we have a common noun phrase lacking a determiner (a so-called "bare noun phrase").)

The multi-stage annotation enabled us to incorporate multi-stage gold-standardizing as done in the Parallel Meaning Bank, so we could take a more incremental approach to annotation if deemed necessary.

The results of this experiment motivated the focus on development of syntactic relaxations (Section 2.1.1) and macros (Section 2.1.7). This way we can minimize the movement of phrases, implicit operator insertions, and the manual insertion of complex lambda expressions that led to annotation errors and discrepancies. We also added guidelines for annotating sentence fragments: annotation with an operator, pu, to mark phrasal utterances for fragments that could reasonably act as standalone utterances, and guidance to skip sentences that could not reasonably act as a standalone utterance in a dialogue.

### 3.1.2 Pilot Annotation 2, Fall 2017

This pilot annotation used a team of eight annotators and annotated the Tatoeba dataset. 270 sentences were annotated, 80 of which were timed. The average annotation speed was 8 minutes per sentence with 4 minutes per sentence among the two experts and 11 minutes per sentence among the three trainees that participated in the timed annotations. We did not time the reviewing time of annotations done by non-experts.

After this annotation pilot we decided to reduce the annotation interface to just syntactic parsing and final conversion to logical form. Annotators are still taught using the five-step process outlined above, but during annotation time this is no longer required. We found the overhead for separating out each step was not worth the simplification and that more experienced annotators naturally skipped the intermediate stages of annotation, performing those steps all at once.

We 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). Section 3.2 describes the IAA study that was done as part of these pilot annotations.

### 3.1.3 Annotator

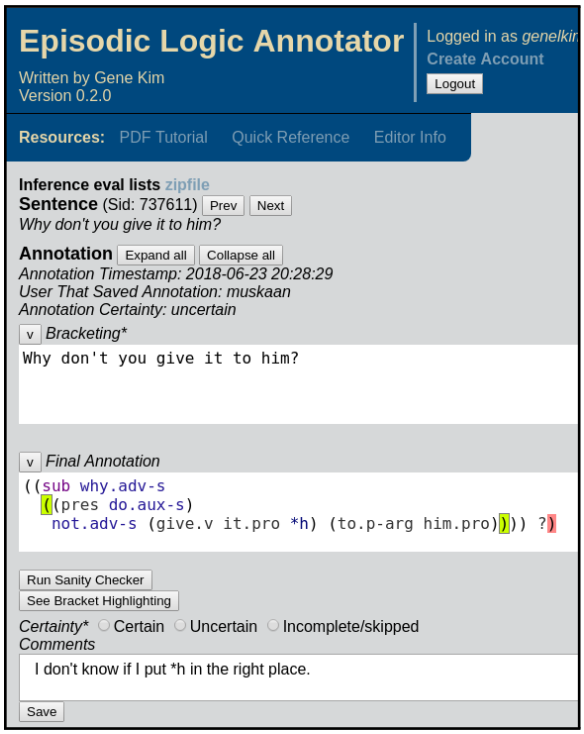



Figure 3.2: Current ULF annotator state with example annotation process.

The surface modifications to ULF and the annotation environment work hand-in-hand to enable quick and consistent annotations. ULF modifications were 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 in the face of indirect relationships between the syntactic and semantic structure. 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.

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 Fig. 3.2 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. ULF's compositional 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. In addition, we can identify common mistakes by analyzing the incorrect types that were annotated and point annotators in the right direction.

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 Fig. 3.2 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 Fig. 3.2 is the sentence-specific comment window. These comments are viewable by all annotators when accessing this sentence.

## 3.2 Interannotator Agreement

We ran an interannotator agreement (IAA) study to quantify the efficacy of the annotation framework. This experiment uses the annotation interface described in Section 3.1, which includes (1) syntax and bracket highlighting, (2) a sanity checker based on the underlying type grammar, and (3) uncertainty marking (which in a collaborative setting triggers a review by a second annotator). Five 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 Section 3.3.1. The mean and standard deviation of sentence lengths were 15.3 words and 10.8 words, respectively.

We compute a similarity score between two annotations using EL-SMATCH (Section 2.2.1).[2] Annotators were given the option to mark whether they were *certain* of their annotation, *uncertain*

[2] SEMBLEU had not yet been published at the time of this study.

of their annotation, or were unable to complete the annotation (*incomplete*). The document-level EL-SMATCH score between all annotated sentence pairs is 0.70. When the analysis is restricted to only annotations marked as *certain*, the agreement rises to 0.78. The complete pairwise scores are shown in Table 3.2. Notice that annotators 1, 2, and 3 have 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 are marked as *uncertain* or *incomplete*. In comparison, AMR annotations reportedly have annotator vs consensus IAA of 0.83 for newswire and 0.79 for web text (Tsialos, 2015).

Table 3.2: Pairwise IAA scores, where the left score in each cell is over all annotations and the right score in each cell 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 |

This study also demonstrates that the certainty marking indeed reflects the quality of the annotation, and thus performs the role we intend. 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 at the time of this experiment.

## 3.3 Annotation Release

The released dataset of hand-annotated ULFs totals 1,738 sentences (1,378 train, 180 dev, 180 test).[3] The dataset is a mixture of sentences from crowd-sourced translations, news text, a question dataset, and novels. The text sources for this dataset were selected to reduce genre-effects and provide good

[3] Available at https://www.cs.rochester.edu/u/gkim21/ulf/resources/.

coverage of all the phenomena we are investigating. Some of these datasets include annotations which were used only to identify sentence and token boundaries. The sentence length statistics of the dataset are a mean of 10.275 tokens, a median of 8 tokens, a minimum of 2 tokens, and a maximum of 128 tokens.

In order to mitigate the issue of low agreement of less experienced annotators which we observed in the IAA study (Section 3.2), I reviewed each annotation in our dataset and corrected as necessary. This step was also important to ensure that all of the annotations followed the most up-to-date annotation guidelines. In addition, there was an open discussion among annotators to clear up uncertainty and handle tricky cases during both the original annotation process and the reviewing step. With these two differences from the IAA study (which has completely independent annotations by single annotators) the dataset annotations should be more consistent than the IAA study would suggest.

### 3.3.1 Data Sources

- **The Tatoeba dataset**[4] consists of crowd-sourced translations from a community-based educational platform. People can request the translation of a sentence from one language to another on the website and other members will provide the translation. Due to this pedagogical context, the sentences are fluent, simple, and highly-varied. The English portion downloaded on May 18, 2017 contains 687,274 sentences.

- **The Discourse Graphbank** (Wolf, 2005) is a discourse annotation corpus created from 135 newswire and WSJ texts. We use the discourse annotations to perform sentence delimiting. This dataset is on the order of several thousand sentences.

- **Project Gutenberg**[5] is an online repository of texts with expired copyright. We downloaded the top 100 most popular books from the 30 days prior to February 26, 2018. We then ignored

[4] https://tatoeba.org/eng/

[5] https://www.gutenberg.org

books that have non-standard writing styles: poems, plays, archaic texts, instructional books, textbooks, and dictionaries. This collection totals to 578,650 sentences.

- **The UIUC Question Classification dataset** (Li and Roth, 2002) consists of questions from the TREC question answering competition. It covers a wide range of question structures on a wide variety of topics, but focuses on factoid questions. This dataset consists of 15,452 questions.

Most of the sentences for the dataset were selected by randomly sampling individual sentences or some contiguous sequence of (on the order to 10s of) sentences. Part of the annotated dataset comes from the annotated sentences used for the inference experiments regarding questions, requests, counterfactuals, and clause-taking verbs which are described in Section 5.1. Therefore, the dataset has a bias towards having these phenomena at a higher frequency than expected from a random selection of English text.

The dataset includes many short sentences, especially from the Tatoeba and UIUC Question Classification datasets, but the sentences cover a wide range of styles and topics. The Tatoeba dataset is built from a crowd-sourced translation community, so the sentences are not limited in genre and style and has a bias toward sentences that give people trouble when learning a second language. We consider this to be valuable for a parsing dataset since they vary widely in topic and tend to focus on tricky phenomena that give language-learners—and likely parsers—trouble. Sentences from the Discourse Graphbank (news text) and Project Gutenberg (novels) further widen the scope of genres and styles in the dataset. This mitigates the problem of models overfitting to dataset distributions on at least one important aspect. The dataset also has a considerable representation of longer sentences (~10% of the sentences are more than 20 words long) including dozens of sentences exceeding 40 words, reaching up to 128 words.

### 3.3.2 Dataset Splits

The data is split by segmenting the dataset into 10 sentence segments and distributing them in a round-robin fashion, with the training set receiving eight chunks in each round. This splitting method is designed to allow document-level topics to distribute into splits while limiting any performance inflation of the dev and test results that can result when localized word-choice and grammatical patterns are distributed into all splits.

The Tatoeba dataset further exacerbates the issue of localized word-choice and grammatical patterns since multiple sentences using the same phrase or grammatical construction often appear back-to-back. We suspect that this is because the Tatoeba dataset is ordered chronologically and users often submit multiple similar sentences in order to help understand a particular phrase or grammatical pattern in a language that they are learning.

# 4 Parsing

Now that we have the manually annotated ULFs we move on to training and evaluating a model using modern semantic parsing methods. The core challenge is that our annotated dataset is smaller than what is needed by modern data-hungry techniques. In this chapter I will discuss a carefully designed transition system, which exploits known features of ULF to reduce the amount of training data necessary to perform well.

This is the first system that learns to parse ULFs of English sentences from an annotated dataset. We evaluate the parser using SEMBLEU (Section 2.2.2) and EL-SMATCH (Section 2.2.1), establishing a baseline for future work. An initial effort in learning a parser producing a representation as rich as ULF is bound to face a data sparsity issue. Thus a major goal in our choice of a transition-system-based parser is to reduce the search space of the model. We investigate three additional methods of tackling this issue: (1) constraining actions in the decoding phase based on faithfulness to the ULF type system, (2) using a lexicon to limit the possible word-aligned symbols that the parser can generate, and (3) defining learnable features of the transition system state.

## 4.1 Background

This ULF parser development draws inspiration from the recent advances in AMR parsing (Peng et al., 2018; Zhang et al., 2019a). The core organization of our parser is based on Peng et al. (2018),

which uses a sequence-to-sequence model to predict the transition action sequence for a cache transition system with transition system features and hard attention alignment.

There are many transition-based parsers that were developed for parsing meaning representations (Zhang et al., 2016; Buys and Blunsom, 2017; Damonte et al., 2017; Hershcovich et al., 2017). These are mainly based on an arc-eager parsing method, termed by Abney and Johnson (1991). Arc-eager parsing greedily adds edges between nodes before full constituents are formed, which keeps the partial graph as connected as possible during the parsing process (Nivre, 2004). Our transition system can be considered a modification of bottom-up arc-standard parsing due to restrictions on arc formation. While this leads to a longer action sequence for parsing, the parser's access to complete constituents allows promotion-based symbol generation for unary operators such as type shifters and standard bottom-up type analysis for constrained parsing.

### 4.1.1 The Cache Transition System for AMR Parsing

Our transition system is a modification of the cache transition system (Gildea et al., 2018) which has been shown to be effective in AMR parsing (Peng et al., 2018). Here we briefly describe the organization of Peng et al.'s (2018) system for AMR parsing in preparation for the presentation of our modified version for ULF parsing. The cache transition parser has a running configuration of the following form.

$$C = (\sigma, \eta, \beta, G_p) \tag{4.1}$$

$\sigma$ is the stack, $\eta$ is the cache, $\beta$ is the buffer, and $G_p$ is the partial graph. The parser is initialized with the configuration ([],[$, ..., $],[$v_1, \ldots, v_n$],$\varnothing$), that is, an empty stack, the cache with null values ($), the buffer with the input sequence of graph vertices ($v_i$), and an empty partial graph, $G_p = (V_p, E_p)$. The transition system has the following four transitions:

1. **Pop** pops $(i, v)$ from $\sigma$ where $i$ is the index of $\eta$ which $v$ came from. $v$ is placed at index $i$ of $\eta$ and shifts the appropriate elements to the right, discarding the rightmost element in $\eta$.

2. **Shift** signals that we will start processing the next input vertex, which will become a new vertex in the output graph. That is, $v_i$ at the front of $\beta$ is added to $V_p$ and the front of the buffer is shifted over one, to $v_{i+1}$.

3. **PushIndex(**$i$**)** pushes $(i, v)$ onto $\sigma$, where $v$ is the vertex currently at index $i$ of $\eta$. Then it places the last vertex shifted from the buffer (during the **Shift** transition) in $\eta$ at index $i$.

4. **Arc(**$i, d, l$**)** forms an arc with label $l$ in direction $d$ (i.e. left or right) between the vertex at index $i$ of the cache and the rightmost vertex in the cache. The **NoArc** action is used if no arc is made.

These transitions are chosen during three phases, SHIFTORPOP, PUSHINDEX, and ARC, which cycle in order until the termination condition—$\sigma$ and $\beta$ are both empty and $\eta$ is in the initial state, $[\$, \ldots, \$]$. The phases restrict when each transition may be chosen.

The sequence of actions that lead to the *known* desired final graph is generated using an oracle algorithm. Then a model is trained to predict the action sequences without knowledge of the desired final graph. Given the desired graph, $G_g = (V_g, E_g)$, the oracle algorithm uses the following rules, organized by phase, until the termination condition is satisfied.

1. SHIFTORPOP phase: choose the **Pop** transition when there is no edge $(v_i, v_j) \in E_g$ such that for the vertex at the front of $\beta$, $v_k$, vertex $j >= k$, or chooses the **Shift** transition otherwise.

2. PUSHINDEX phase: choose $(i, v)$ in $\eta$ such that $v$ is the vertex from $\eta$ whose closest neighbor in $\beta$ is furthest from the front of it. The oracle chooses transition **PushIndex(**$i$**)**, which stores $(i, v)$ in $\sigma$ for later processing.

3. ARC phase: between the rightmost cache vertex and each vertex in $\eta$, check whether there an edge between them exists in $E_g$. If there is an edge, the oracle chooses the corresponding direction and label based on $E_g$ and the $\eta$ state.

In order to generate the input sequence of vertices, Peng et al. (2018) use Flanigan et al.'s (2016) semi-Markov model to segment the sentence into a sequence of AMR-concept-relevant spans and

map each span into an AMR subgraph. Each subgraph is treated as a single vertex by the transition system, where subgraphs with more than one vertex are collapsed to single vertices with labels corresponding to a small set of categories (named entities, dates, numbers, phrases, and multiple oncepts). These mappings are saved and the subgraphs are recovered at the end using heuristic rules to disambiguate between multiple instances of the same subgraph category.

## 4.2 Transition System

A key distinctive aspect of our modified transition system is that it generates nodes that are derived, but distinct, from the input sequence. It is a node generative transition system. This eliminates the two-stage parsing framework of Peng et al. (2018). Our transition system also restricts the parses to be bottom-up to enable node generation and decoding constraints that depend on the available constituents since ULF has an bottom-up compositional type system. The transition parser configuration has the same high-level structure as the cache transition system for AMR parsing, Eq. (4.1), but rather than initializing the buffer with graph vertices, it is initialized with the input sequence of words, $\beta = [w_1, \ldots, w_n]$, where each word is a token, lemma, POS tuple, $w_i = (t_i, l_i, p_i)$. Additionally, the word-symbol alignment is captured internally by including it in the vertices. $G_p = (V_p, E_p)$, where $V_p$ is ordered and a vertex, $v_i = (s_i, a_i) \in V$, is a pair of a ULF symbol $s_i$, and its alignment $a_i$—the index of the word from which $s_i$ was produced. We will refer to the leftmost element in $\beta$ as $w_{\text{next}}$.

While the size of the cache is a hyperparameter that can be set for the cache transition parser, we restrict the cache size to 2 in order to keep the oracle simple despite the newly added actions. This means that only tree structures can be parsed. In describing the transition system, we differentiate between *phases* and *actions*. Phases are classes of states in the transition system and the actions move between states. Figure 4.1 shows the full state transition diagram and shows how the phases dictate which actions can be taken and how actions move between phases. Actions may take variables to specify how to move into the next phase. Phases also determine which features go into

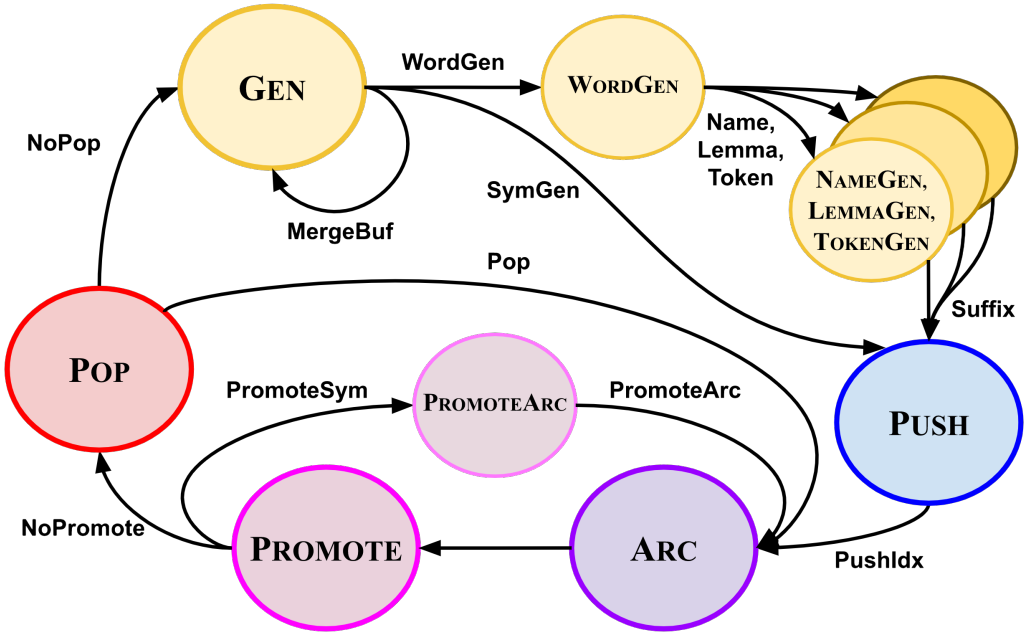



Figure 4.1: State transition diagram of the node generative transition system. Nodes in the figure are phases and edges are actions. An unlabeled edge means that this state transition occurs no matter what action is taken in that phase. The transition system starts in the GEN phase.

the determining the next action. We will write phases in small caps (e.g. GEN) and actions in bold (e.g. **TokenGen**) for clarity.

The GEN and PROMOTE phases are novel to our transition system. The GEN phase generates graph vertices that are transformations of the buffer values. This allows us to put words of the input sentence in $\beta$ instead of a pre-computed ULF atom sequence. The PROMOTE phase enables context-sensitive symbol generation. It generates unaligned symbols in the context of an existing constituent in the partial graph; use of logical operators without word alignments only makes sense with respect to something for the operators to act on. We now describe each of the actions in the transition system. Figure 4.2 shows an example of using the transition system to parse a sentence, which can be used for reference as we go through the details of each action. The following are parser actions that were almost directly inherited from the vanilla cache transition parser (Section 4.1.1). They are minimally modified to reflect the GEN phase which replaces the **Shift** action.

| **Stack** ($\sigma$) | **Cache** ($\eta$) | **Buffer** ($\beta$) | **Edges** | **Actions taken** |
|---|---|---|---|---|
| [] | [$, $] | [Do, you, want, to, see, me, ?] | $\varnothing$ | — |
| [$^0$] | [$, do.aux-s] | [you, want, to, see, me, ?] | $\varnothing$ | **Lemma**(aux-s);**Push**(0) |
| [$^0$] | [$, pres] | [you, want, to, see, me, ?] | $E_1$ | **NoArc**;**PSym**(pres);**PArc**(arg0) |
| [$^0$] | [$, $\chi_0$] | [you, want, to, see, me, ?] | $E_2$ | **NoArc**;**PSym**($\chi_0$);**PArc**($\iota$) |
| [$^0$, $^0$] | [$\chi_0$, you.pro] | [want, to, see, me, ?] | $E_2$ | **NoArc**;**NoP**;**Lemma**(pro);**Push**(0) |
| [$^0$] | [$, $\chi_0$] | [want, to, see, me, ?] | $E_3$ | **Arc**(0, R, arg0);**NoP**;**Pop** |
| [$^0$, $^0$] | [$\chi_0$, want.v] | [to, see, me, ?] | $E_3$ | **NoArc**;**NoP**;**Lemma**(v);**Push**(0) |
| [$^0$, $^0$, $\chi_0^0$] | [want.v, to] | [see, me, ?] | $E_3$ | **NoArc**;**NoP**;**Lemma**($\varnothing$);**Push**(0) |
| [$^0$, $^0$, $\chi_0^0$, want.v$^0$] | [to, see.v] | [me, ?] | $E_3$ | **NoArc**;**NoP**;**Lemma**(v);**Push**(0) |
| [$^0$, $^0$, $\chi_0^0$, want.v$^0$, to$^0$] | [see.v, me.pro] | [?] | $E_3$ | **NoArc**;**NoP**;**Token**(pro);**Push**(0) |
| [$^0$, $^0$, $\chi_0^0$, want.v$^0$] | [to, see.v] | [?] | $E_4$ | **Arc**(0, R, arg0);**NoP**;**Pop** |
| [$^0$, $^0$, $\chi_0^0$] | [want.v, to] | [?] | $E_5$ | **Arc**(0, R, arg0);**NoP**;**Pop** |
| [$^0$, $^0$] | [$\chi_0$, want.v] | [?] | $E_6$ | **Arc**(0, R, arg0);**NoP**;**Pop** |
| [$^0$] | [$, $\chi_0$] | [?] | $E_7$ | **Arc**(0, R, arg1);**NoP**;**Pop** |
| [$^0$] | [$, $\chi_1$] | [?] | $E_8$ | **NoArc**;**PSym**($\chi_1$);**PArc**($\iota$) |
| [$^0$, $^0$] | [$\chi_1$, ?] | [] | $E_8$ | **NoArc**;**NoP**;**Lemma**($\varnothing$);**Push**(0) |
| [$^0$] | [$, $\chi_1$] | [] | $E_9$ | **Arc**(0, R, arg0);**NoP**;**Pop** |
| [] | [$, $] | [] | $E_9$ | **NoArc**;**NoP**;**Pop** |

Figure 4.2: Example run of the transition system running on the sentence *"Do you want to see me?"* from our parser. The left four columns show the parser configuration after taking the actions shown in the rightmost column. We make the following modifications for brevity. When a **WordGen** action takes place, it is always followed by one of **Name**, **Lemma**, or **Token** and then a **Suffix**($e$) action. Thus we omit the **WordGen** and **Suffix** actions and transfer the argument of **Suffix** to the **Name**, **Lemma**, or **Token** action. "Promote" is abbreviated as "P" (e.g., **PromoteSym** as **PSym**) and **PushIdx** as **Push**. Stack item indices $(i, v)$ are written as $v^i$ instead. $\chi$ and $\iota$ stand for COMPLEX and INSTANCE which are the special node and edge labels, respectively, for constructing non-atomic ULF operators in penman format. Edge labels arg0 and arg1 simply indicate the argument position in ULF. $E_n = \{e_i \mid 0 \le i < n\}$ where $e_0$ = (do.aux-s $\xleftarrow{arg0}$ pres), $e_1$ = (pres $\xleftarrow{\iota}$ $\chi_0$), $e_2$ = ($\chi_0$ $\xrightarrow{arg0}$ you), $e_3$ = (see.v $\xrightarrow{arg0}$ me.pro), $e_4$ = (to $\xrightarrow{arg0}$ see.v), $e_5$ = (want.v $\xrightarrow{arg0}$ to), $e_6$ = ($\chi_0$ $\xrightarrow{arg0}$ want.v), $e_7$ = ($\chi_0$ $\xleftarrow{\iota}$ $\chi_1$), $e_8$ = ($\chi_1$ $\xrightarrow{arg0}$ ?).

- **PushIndex**($i$) pushes $(i, v)$ onto $\sigma$, where $v$ is the vertex currently at index $i$ of $\eta$. Then it moves the vertex generated by the prior GEN phase to index $i$ in $\eta$.

- **Arc**($i, d, l$) forms an arc with label $l$ in direction $d$ (i.e. left or right) between the vertex at index $i$ of the cache and the rightmost vertex in the cache. The **NoArc** action is used if no arc is made.

- **Pop** pops $(i, v)$ from $\sigma$ where $i$ is the index of $\eta$ which $v$ came from. $v$ is placed at index $i$ of $\eta$ and shifts the appropriate elements to the right.

We introduce two sets, $S_p$ and $S_s$, which define the vocabulary of the two unaligned symbol generation actions: **PromoteSym** and **SymGen**, respectively. $S_p$ consists of logical and macro operators that do not align with English words. $S_s$ consists of symbols that could not be aligned in the training set and are not members of $S_p$.

### 4.2.1 Promotion-based Symbol Generation

PROMOTE includes a subordinate PROMOTEARC phase for modularizing the parsing decision. The following parsing actions are in this phase.

- **PromoteSym($s_p$)** generates a promotion symbol, $s_p \in S_p$, appends the vertex $(s_p, \text{NONE})$ to $V_p$, and proceeds to the PROMOTEARC phase.
- **NoPromote** skips the PROMOTE phase and proceeds to the POP phase.
- **PromoteArc($l$)** makes an arc from the last added vertex, $v_p$, to the vertex at the rightmost position of the cache, $v_{\eta_r}$, by adding $(v_p, v_{\eta_r}, l)$ to $E_p$. $v_p$ then takes the place of $v_{\eta_r}$ in the cache and $v_{\eta_r}$ is no longer accessible by the transition system. The system proceeds to the ARC phase.

### 4.2.2 Sequential Symbol Generation

We replace the **Shift** action with the GEN phase to generate ULF atoms based on the tokenized text input. This phase allows the parser to generate a symbol using $w_{\text{next}}$ as a foundation, or generate an arbitrary symbol that is not aligned to any word in $\beta$. GEN includes subordinate phases WORDGEN, LEMMAGEN, TOKENGEN, and NAMEGEN for modularizing the decision process.

- **WordGen** proceeds to WORDGEN phase, in which the following actions are available.

1. **Name** proceeds to the NAMEGEN phase.
2. **Lemma** proceeds to the LEMMAGEN phase.
3. **Token** proceeds to the TOKENGEN phase.

- **Suffix(**$e$**)** is the only action available in the NAMEGEN, LEMMAGEN, and TOKENGEN phases. It generates a symbol $s$ consisting of a stem and suffix extension $e$ from $w_{\text{next}}$. In the NAMEGEN phase, the stem is $t_{\text{next}}$ with surrounding pipes; in the TOKENGEN phase, the stem is $t_{\text{next}}$; and in the LEMMAGEN phase, the stem is $l_{\text{next}}$. $(s, i)$ where $i$ is the index of $w_{\text{next}}$ is added to $V_p$ and we move forward one word in $\beta$. The system proceeds to the PUSH phase.
- **SymGen(**$s$**)** adds an unaligned symbol $(s, \text{NONE})$ to $V_p$ and proceeds to the PUSH phase.
- **SkipWord** skips word in $\beta$ and returns to the GEN phase.
- **MergeBuf** takes $w_{\text{next}}$ and merges it with the word after it $w_{\text{next+1}}$. This is stored at the front of the buffer as a pair $(v_\beta, v_{\beta+1})$. This forms a single stem with a space delimiter in the NAMEGEN phase and an underscore delimiter in the LEMMAGEN and TOKENGEN phases. The system returns to the GEN phase. This is used to handle multi-word expressions (e.g. *"had better"*).

The transition system begins in the GEN phase.

### 4.2.3 Oracle Extraction Algorithm

In order to train a model of the parser actions, we need to extract the desired action sequences from gold graphs. We modify the oracle extraction algorithm for the vanilla cache transition parser, described by Gildea et al. (2018). The oracle starts with a gold graph $G_g = (V_g, E_g)$ and maintains the partial graph $G_p = (V_p, E_p)$ of the parsing process, where $V_g$ is sequenced by the preorder traversal of $G_g$. The oracle maintains $s_{\text{next}}$, the symbol in the foremost vertex of $V_g$ that has not yet been added to $G_p$. The oracle begins with a transition system configuration, $C$, initialized with the input sequence, $w_1, ..., w_n$.

The oracle is also provided with an approximate alignment, $A = \{(w_i, v_j) \mid 1 \leq i \leq n, 1 \leq j \leq m\}$, between the input sequence, $w_{i:n}$, and the nodes in the gold graph, $V_g, |V_g| = m$, which is generated with a greedy matching algorithm. The matching algorithm uses a manually-tuned similarity heuristic built on the superficial similarity of English words, POS, and word order to the stems, suffixes, and preorder positions of the corresponding ULF atoms. A complete description of the alignment algorithm is in appendix C.1. This alignment is not necessary to maintain correctness of the oracle, but it is used to cut the losses when the input words become out of sync with the gold graph vertex order.[1] Steps 5-7 of the GEN phase uses the alignments to identify whether the buffer or the vertex order is ahead of the other and appropriately sync them back together.

The oracle uses the following procedure, broken down by parsing phase, to extract the action sequence to build the $G_p = G_g$ with $C$ and $A$.

- GEN phase: Let $b = \text{Stem}(s_{\text{next}})$, $e = \text{Suffix}(s_{\text{next}})$, $n = \text{IsName}(s_{\text{next}})$.[2]

  1. If $n$ and $t_{\text{next}} = b$, **NameGen(**$e$**)**
  2. If not $n$ and $t_{\text{next}} =_i b$, **TokenGen(**$e$**)**
  3. If not $n$ and $l_{\text{next}} =_i b$, **LemmaGen(**$e$**)**
  4. **MergeBuf** if

     $n$ and $\text{Pre}(\text{Concat}(t_{\text{next}}, \text{“ ”}, t_{\text{next+1}}), b)$ or

     not $n$ and $\text{Pre}_i(\text{Concat}(l_{\text{next}}, \text{“_”}, l_{\text{next+1}}), b)$ or

     not $n$ and $\text{Pre}_i(\text{Concat}(t_{\text{next}}, \text{“_”}, t_{\text{next+1}}), b)$
  5. If $(w_i, v_{\text{next}}) \in A$ for $w_i$ before $w_{\text{next}}$ or $v_{\text{next}} \in S_s$, then **SymGen(**$v_{\text{next}}$**)**
  6. If $(w_{\text{next}}, v_j) \in A$ for $v_j$ which comes after $v_{\text{next}}$ or $v_j \in V_p$, then **SkipWord**
  7. Otherwise, **SymGen(**$v_{\text{next}}$**)**

[1] When the words become out-of-sync with the gold graph the oracle must rely on **SymGen** to generate the graph nodes. Since **SymGen** requires selecting the correct value out the entire vocabulary of ULF atoms, it is much more difficult to predict correctly than **NameGen**, *TokenGen*, and *LemmaGen* which require only selecting the correct type tag.

[2] $=$ is string match, $=_i$ is case-insensitive string match, Pre determines whether its first argument is a prefix of the second and $\text{Pre}_i$ is the case-insensitive counterpart.

Step 5-7 allow the oracle to handle the generation of symbols that are not in word order, by skipping any words that come earlier than the symbol order; and generating symbols that cannot be aligned with **SymGen** for any reason.

- PUSH phase: The push phase of the vanilla cache transition parser's oracle—viz., choosing the cache position whose closest edge into $\beta$ is farthest away—is extended to account not only for direct edges, but also for paths that include only unaligned-symbols.[3]

- ARC phase: The vanilla cache transition system's rule of generating the $ARC$ action for any edge, $e \in E_g \wedge e \notin E_p$ between the rightmost cache position and the other positions, is extended to also require the child vertex to be fully formed. That is, for the vertex $v_{\text{child}}$, |descendants($v_{\text{child}}, G_g$)| = |descendants($v_{\text{child}}, G_p$)|. This enforces bottom-up parsing, which is necessary for both the promotion-based symbol generation and type composition constraint.

- PROMOTE phase: If the vertex in the rightmost cache position, $v_{\eta_r}$, is fully formed (|descendants($v_{\eta_r}, G_g$)| = |descendants($v_{\eta_r}, G_p$)|) and has a parent node in the PROMOTE lexicon (label(parent($v_{\eta_r}, G_g$)) $\in S_p$), then the parser generates the action sequence **PromoteSym**(parent($v_r, G_g$)), **PromoteArc**($l_p$) where $l_p$ is the label for the edge from the parent of $v_{\eta_r}$ to $v_{\eta_r}$ in $G_g$ (EdgeLabel(parent($v_{\eta_r}, G_g$), $v_{\eta_r}, G_g$)).

## 4.3 Model

Our model has three basic components: (1) a word sequence encoder, (2) a ULF atom sequence encoder, and (3) an action decoder, all of which are LSTMs. During decoding, the transition system

[3]The motivation for this is that if only unaligned symbols exist in the path, the full path can be made without changing the relative status of any other node in the transition system. Let $v_1$ and $v_2$ be the end points of the path. With $v_1$ in the cache and the word aligned to $v_2$, $w_{v_2} = w_{\text{next}}$, **SymGen** and PROMOTE can generate all nodes in the path without interacting with the rest of the transition system.

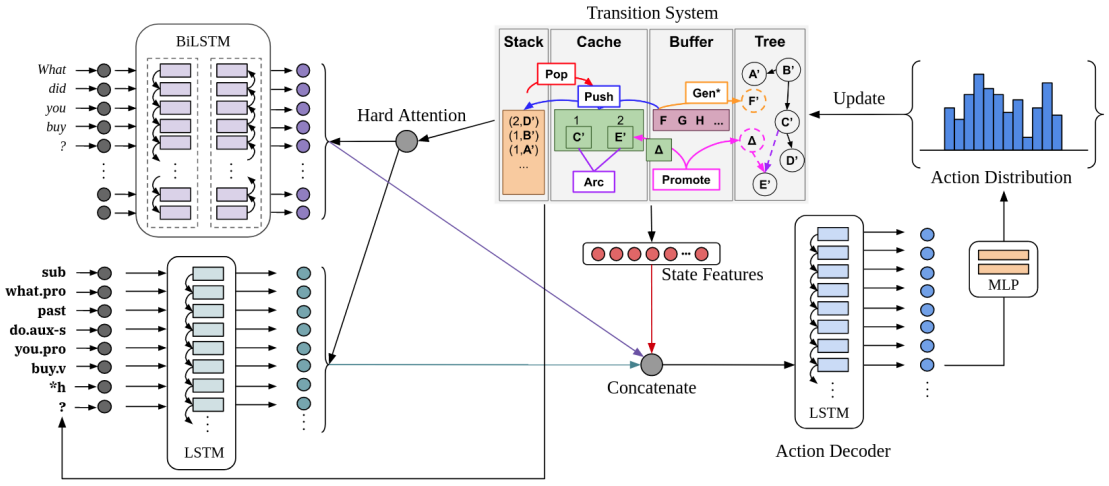



Figure 4.3: The model consists of a sentence-encoding BiLSTM, a symbol-encoding LSTM, and an action-decoding LSTM. New symbols generated in the GEN and PROMOTE phases of the transition system are appended to the symbol sequence. The transition system supplies hard attention pointers that select the relevant word and symbol embeddings. These are concatenated with the transition state feature vector and supplied as input to the action decoder, which predicts the next action that updates the transition system.

configuration, $C$, is updated with decoded actions and used to organize the action decoder inputs using the sequence encoders. The system models the following probability

$$P(a_{1:q}|w_{1:n}) = \prod_{t=1}^{q} P(a_t|a_{1:t-1}, w_{1:n}; \theta) \tag{4.2}$$

where $a_{1:q}$ is the action sequence, $w_{1:n}$ is the input sequence, and $\theta$ is the set of model parameters. Figure 4.3 is a diagram of the full model structure.

### 4.3.1 Word and Symbol Sequence Encoders

The input word embedding sequence $w_{1:n}$ is encoded by a stacked bidirectional LSTM (Hochreiter and Schmidhuber, 1997) with $L_w$ layers. Each word embedding sequence is a concatenation of embeddings of GloVe (Pennington et al., 2014), lemmas, part-of-speech (POS) and named entity

(NER) tags, RoBERTa (Liu et al., 2019), and features learned by a character-level convolutional neural network (CharCNN, Kim et al., 2016). As ULF symbols are generated during the parsing process, the symbol embedding sequence $s_{1:m}$, which is the concatenation of a symbol-level learned embedding and the CharCNN feature vector over the symbol string, is encoded by a stacked *uni*directional LSTM of $L_s$ layers.

### 4.3.2 Hard Attention

Peng et al. (2018) found that for AMR parsing with cache transition systems, a hard attention mechanism, tracking the next buffer node position and its aligned word, works better than a soft attention mechanism for selecting the embedding used during decoding. We take this idea and modify the tracking mechanism to find the most relevant word, $w_i$, and symbol, $s_j$, for each phase.

- ARC and PROMOTE*: The symbol $s_j$ in the rightmost cache position and aligned word $w_i$.
- PUSH: The symbol $s_j$ generated in the previous action and aligned word $w_i$.
- Otherwise: The last generated symbol $s_j$ and the word $w_i$ in the leftmost $\beta$ position.

This selects the output sequences $\boldsymbol{h}_{w_i}^{L_w}$ and $\boldsymbol{h}_{s_j}^{L_s}$ from the encoders for the action decoder.

### 4.3.3 Transition State Features

Similar to Peng et al. (2018), we extract features from the current transition state configuration, $C$, to feed into the decoder as additional input in the form of learned embeddings

$$\boldsymbol{e}_f(C) = [\boldsymbol{e}_{f_1}(C); \boldsymbol{e}_{f_2}(C); ...; \boldsymbol{e}_{f_l}(C)] \tag{4.3}$$

where $\boldsymbol{e}_{f_k}(C)$ $(k = 1, ..., l)$ is the $k$-th feature embedding, with $l$ total features. Our features, which are heavily inspired by Peng et al. (2018), are as follows.

- Phase: An indicator of the phase in the transition system.

- POP, GEN features: *Token features*[4] of the rightmost cache position and the leftmost buffer position; the number of rightward dependency edges from the cache position word and the first three of their labels; and the number of outgoing ULF arcs from the cache position and their first three labels.
- ARC, PROMOTE features: For the two cache positions, their token features and the word, symbol[5], and dependency distance between them; furthermore, their first three outgoing and single incoming dependency arc labels and their first two outgoing and single incoming ULF arc labels.
- PROMOTEARC features: Same as the PROMOTE features but for the rightmost cache position use the node/symbol generated in the preceding **PromoteSym** action.
- PUSH features: Token features for the leftmost buffer position and all cache positions.

### 4.3.4 Action Encoder/Decoder

The action sequence is encoded by a stacked unidirectional LSTM with $L_a$ layers where the action input embeddings, $\boldsymbol{h}_{a_{1:q}}$ are concatenations of the word and symbol encodings.

$$\boldsymbol{h}_{a_k} = [\boldsymbol{h}_{w_i}^{L_w}; \boldsymbol{h}_{s_j}^{L_s}; \boldsymbol{e}_f(C)] \tag{4.4}$$

The state features $\boldsymbol{h}_{a_k}^{L_a}$ are then decoded into prediction weights with a linear transformation and ReLU non-linearity.

## 4.4 Parsing

The model is trained on the cross-entropy loss of the model probability (4.2) using the oracle action sequence. Both training and decoding are limited to a maximum action length of 800. For the

[4]The token features are the ULF symbol and the word, lemma, POS, and NER tags of the aligned index of the input.
[5]Symbol distance is based on the order in which the symbols are generated by the parser.

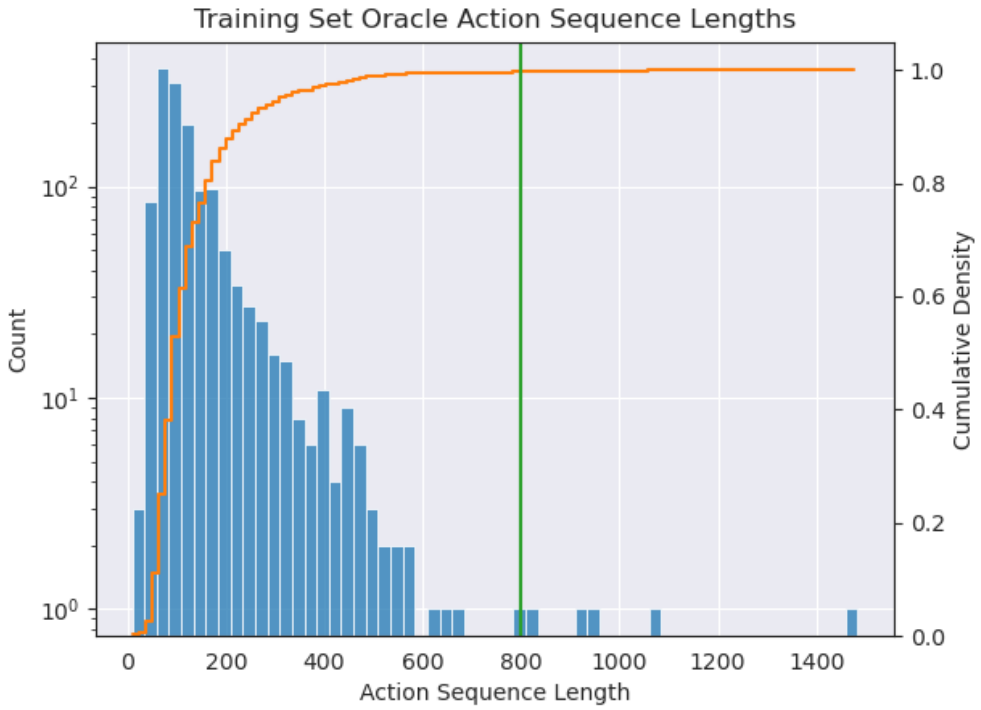



Figure 4.4: A histogram of oracle action lengths in the training set (note the log-scale on the y-axis) with the cumulative density in orange. Additionally, the green vertical line shows the maximum action length supported by our model configuration.

training set the oracle has an average action length of 134 actions and a maximum action length of 1477. Figure 4.4 shows a histogram of oracle action lengths, with only a handful of examples that exceed an action length of 800 and are only partially solved due to the maximum action length supported by our model.

### 4.4.1 Constrained Decoding

We investigate two methods of constraining the decoding process with prior knowledge of ULF to overcome the challenge of using a small dataset. These automatic methods filter out clearly incorrect choices at the cost of some decoding speed and further tailor the parser to ULFs.

**ULF Lexicon.** To improve symbol generation, we introduce a lexicon with possible ULF atoms for each word. Nouns, verbs, adjectives, adverbs, and preposition entries are automatically converted from the Alvey lexicon (Carroll and Grover, 1989) with some manual editing. Pronouns, determiners, and conjunctions entries are extracted from Wiktionary[6] category lists. Auxiliary verbs entries are manually built from our ULF annotation guidelines.[7] When generating a word-aligned symbol the stem is searched in the lexicon. If the string is present in the lexicon, only corresponding symbols in the lexicon are allowed to be generated. Since the lexicon is not completely comprehensive, this constraint may lead to some additional errors.

**Type Composition.** The type system constraint adds a list of types, $T_v$, to accompany $|V_p|$ (the vertices of the partial graph), which stores the ULF type of each vertex. When a vertex, $v$, is added to $G_p$, its ULF type, $t_v$ is added to $T_v$. This ULF type system is generalized with placeholders for macros and each stage in processing them. When the parser predicts an arc action during decoding, the types source, $t_s$, and target, $t_t$ nodes are run through a type composition function. If the types can compose, $t_c = (t_s.t_t), t_c \neq \varnothing$, the type of the source node is replaced with $t_c$. Otherwise, the resulting $C$ is not added to the search beam.

## 4.5 Experiments

All experiments use the first ULF annotation release (Section 3.3).

**Setup.** We primarily evaluate the model with SEMBLEU (Section 2.2.2) and also measure ELSMATCH (Section 2.2.1) for analysis of the model since it has F1, precision, and recall components. The tokens, lemmas, POS tags, NER tags, and dependencies are all extracted using the Stanford

[6] https://en.wiktionary.org/

[7] Manual construction of the other closed class category portions of the lexicon is possible and ultimately preferred for optimal performance. Our use of external sources for prepositions, pronouns, determiners, and conjunctions was a means to quickly obtain a generous lexicon which would include entries that are uncommon or even a bit out-of-date. Our manual construction for auxiliary verbs was due to a list already existing from the annotation guidelines.

CoreNLP toolkit (Manning et al., 2014). In all experiments the model was trained for 25 epochs. Starting at the 12th epoch we measured the SEMBLEU performance on the dev split with beam size 3. Hyperparameters were tuned manually on the dev split performance of a smaller, preliminary version of the annotation corpus. We use RoBERTa-Base embeddings with frozen parameters, 300 dimensional GloVe embeddings, and 100 dimensional $t_i$, $l_i$, $p_i$, action, and symbol embeddings. The word encoder is 3 layers. The symbol encoder and action decoder are 2 layers. Experiments were run on a single NVIDIA Tesla K80 or GeForce RTX 2070 GPU. Training the full model takes about 6 hours. The full tables of results and default parameters are available in appendix C.3.

In our ablation tests, the model from the training epoch with the highest dev set SEMBLEU score is evaluated on the test split with beam size 3.[8] The results are shown in Fig. 4.5.

**Ablations.** CharCNN and RoBERTa are the least important components—to the point that we cannot conclude that they are of any benefit to the model due to the large overlap in the performance of models with and without them. The GloVe, POS, and feature embeddings are more important. The importance of POS is not surprising given the tight correspondence between POS tags and ULF type tags.

**Comparison to Baselines.** We compare our parser performance against two AMR parsers with minimal AMR-specific assumptions. The major recent efforts by the research community in AMR parsing make these parsers strong baselines. Specifically, we compare against the sequence-to-graph (STOG) parser (Zhang et al., 2019a) and Cai and Lam's (2020) graph-

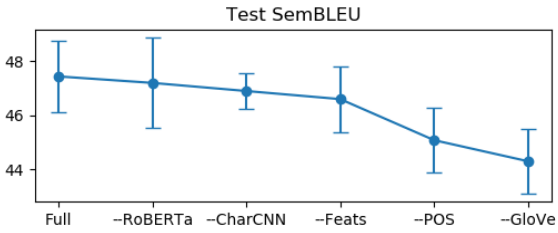



Figure 4.5: Ablation tests with standard deviation error bars of 5 runs of different random seeds.

[8] Our initial experiments re-evaluated the top-5 choices with a beam size of 10, but we found that the performance consistently degraded and abandoned this step.

sequence iterative inference (GS) parser.[9] The ULF dataset is preprocessed for these parsers by stripping pipes from names to support the use of a copy mechanism and splitting node labels with spaces into multiple nodes to make the labels compatible with their data pipelines. Table 4.1 shows the results.[10] The STOG parser fares poorly on both metrics. A review of the results revealed that the parser struggles with node prediction in particular. This is likely the result of the dataset size not properly supporting the parser's latent alignment mechanism.[11] The GS parser performs better than the STOG parser by a large margin, but is still far from our parser's performance. The GS parser also struggles with node prediction, but is more successful in maintaining the correct edges in spite of incorrect node labels.

Table 4.1: Comparison to AMR parsers.

| **Model** | SEMBLEU | EL-SMATCH |
|---|---|---|
| (Zhang et al., 2019a) | 12.3 | 34.3 |
| (Cai and Lam, 2020) | 34.2 | 52.6 |
| Our best model | **47.4** | **59.8** |

Investigating the dev set results reveals that our parser is quite successful in node generation, since by design the node generation process reflects the design of ULF atoms. Despite the theoretical capacity to generate node labels without a corresponding uttered word or phrase, our parser only does this for common logical operators such as reifiers and modifier constructors. The GS parser on the other hand, is relatively successful on node labels without uttered correspondences, correctly generating the elided "you" in imperatives and the logical operators `!` and `multi-sent` which indicate imperatives and multi-sentence annotations, respectively. Our parser also manages to correctly generate a variety of verb phrase constructions, but fails to recognize reified infinitives as arguments of less frequent clausal verbs such as "neglect", "attach", etc. (as opposed to "have",

[9] We do not compare our model against the existing rule-based ULF parsers since they are domain specific and cannot handle the range of sentences that appear in our dataset.

[10] Our parser gets the exact ULF for 6 out of the 180 sentences (3.3%). They were all yes-no questions which tend to be a bit shorter than informative declarative sentences (e.g. *"Can't you do something?"*).

[11] The STOG parser is improved by (Zhang et al., 2019b) with about 1 point of improvement on SMATCH. Unfortunately, the code for this parser is not released to the public.

"tell") and instead interprets "to" as either an argument-marking prepositions or reification of an already reified verb. Examples of parses and a discussion of specific errors are omitted here due to space constraints and provided in appendix C.4.

**Constrained Decoding** When evaluating decoding constraints, we select the model by re-running the five best performing epochs with constraints. When using the type composition constraint, we additionally increase the beam size to 10 so that the parser has backup options when its top choices are filtered out. Table 4.2 presents these results. We see a increase in precision for +Lex, but a greater loss in recall. +Type reduces performance on all metrics. Due to the bottom-up parsing procedure, a filtering of choices can cascade into fragmented parses. The outputs for an arbitrarily selected run of the model has on average 2.9 fragments per sentence when decoding with the type constraint and 1.4 without. This and the relative performance on the precision metric suggest that constraints improve individual parsing choices, but are too strict, leading to fragmented parses.

Table 4.2: Statistics of model performances with constraints added—the average of 5 runs.

| | SEMBLEU | EL-SMATCH | | |
|---|---|---|---|---|
| | | F1 | Precision | Recall |
| Full | **47.4** | **59.8** | 60.7 | **59.0** |
| +Lex | 46.2 | 57.5 | **61.5** | 54.1 |
| +Type | 40.0 | 55.8 | 59.1 | 52.8 |

**Dependence on Length.** To investigate the performance dependence on the problem size, we partition the test set into quartiles by oracle action length. Our full model with random seed 0[12] has SEMBLEU scores of 52, 47, 48, and 31 on the quartiles of increasing length. As expected, the parser performs better on shorter tasks. The parser performance is relatively stable until the last quartile. This is likely due to a long-tail of sentence lengths in our dataset. This last quartile includes sentences with oracle action length ranging from 148 to 1474.

[12]This analysis was performed based on the suggestion of an anonymous reviewer, at which point we no longer had the trained models available for all five random seeds.

## 4.6 Conclusion

This chapter presented the first ULF parser trained on an annotated dataset. We showed that our parser is a strong baseline, out-performing existing semantic parsers from a similar task. Surprisingly, our experiments showed that even in this low-resource setting, constrained decoding with a lexicon or a type system does more harm than good. However, the symbol generation method and features designed for ULFs result in a performance lead over using an AMR parser with minimal representational assumptions. We see many remaining potential avenues of improvement. The type grammar opens up many promising possibilities: sampling of silver data from the type grammar in conjunction with reliable ULF to English generation (Section 5.1.11), use of the type grammar as a weighted constraint, or direct incorporation of the type grammar into a parsing model to avoid the pitfalls we observed in our simple approach to semantic type enforcement.

# 5 Inference

We now move on to the problem of using ULFs for inference. Even with the ambiguities that remain in ULF, the information offered by the argument structure enable important classes of inference for language understanding and dialogue systems. An important insight of natural logic 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 natural logic 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.

Another key aspect this chapter highlights is forward inferencing, as opposed to inference guided by propositions to be confirmed or disconfirmed (as in most textual entailment and natural logic studies to date (Angeli and Manning, 2014; Tian et al., 2014)). This is important, because human language understanding seems to involve continuous forward inferencing. For example, if according to your news source *"Police reported that the vehicle struck several cars"*, you will conclude that the reported event almost certainly happened, and further, that the cars involved were all damaged. The first of these inferences requires only a small amount of knowledge about communication, to the effect that *reporting* typically involves reporting of facts; whereas the latter depends on very

specific world knowledge. The work in this chapter focuses on the former kinds of inference, since accumulation of sufficient world knowledge for enabling the latter kinds of inference remains out of reach in the short run. Here are some kinds of inferences we can expect ULF to support with minimal additional knowledge due to its structural nature and will be demonstrated in this chapter.

1. **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 EL by Stratos et al. (2011).

2. **Attitudinal & 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 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 natural logic, since they are partially probabilistic, require structural elaboration, and depend on constituent types.

3. **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.

4. **Questions & 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.

5. **Natural Logic.** 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*.

Section 5.1 describes experiments demonstrating the capacity to generate inferences from Items 1 to 4 which only require access to the ULFs. These experiments were run before the development

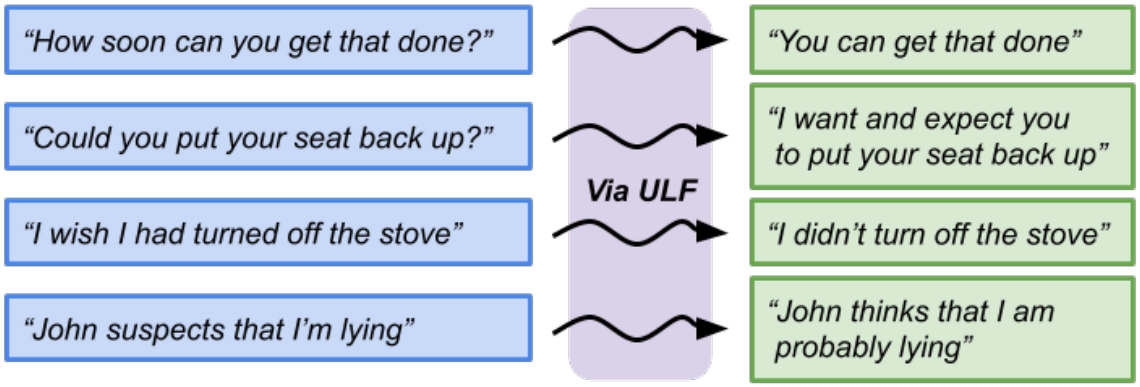



Figure 5.1: Examples of the sorts of discourse inferences that we generate via ULFs with rules described in Section 5.1.

of the ULF parsers described in Chapter 4 and were run on manually annotated ULFs. Section 5.2 describes a theoretical formulation of performing natural logic inference with ULF as the underlying representation. The implementation of this formulation and automated experiments are described in Section 5.3.

## 5.1 Generating Discourse Inferences from ULFs

### 5.1.1 Introduction

ULF, being a pre-canonicalized semantic form, makes available many possible structures for similar meanings, which leads to a challenge in formulating generalizable inferences. This pre-canonicalized nature of ULF, though structurally relatively intricate, has some advantages over fully canonicalized representations for use in natural language tasks. One is that it allows direct translation of intuitions about warranted textual inferences into inference rules (much as in Natural Logic). As well, the ability to accurately generate the English sentences corresponding to a ULF formula and choose how and when to modify the surface form allows a more natural interface with the end task. This feature allows us to evaluate inferences generated by ULF directly over English text rather than using an artificially structured interface, such as classification.

We present a method of generating inferences from ULFs from a small set of interpretable inference rules by first defining general semantic predicates over ULF clauses and tree transformations that correspond to natural semantic operations in ULF. We then evaluate these inferences on the first four inferential classes described at the start of this chapter (Items 1 to 4) over a multi-genre dataset. A common feature among all of these inferences is that they are highly dependent on a combination of the predicate-argument and syntactic structures. Also, these are inferences that come naturally and spontaneously to speakers during conversation and are important for generating natural dialogues by setting up the appropriate conversational context.

The near surface structure of ULF allows us to incorporate a strict paraphrase rewrite module and then perform direct string comparisons of English generated from ULFs to human generated inferences. Human evaluations show that 68.5% of these generated inferences are acceptable and an error analysis of the system shows that many of the errors can be corrected with some refinement to the inference rules and the ULF-to-English generation system.

### 5.1.2 ULF Inference Rules

The inference rules that we define are tree transductions that respect the EL type system in both the antecedent and consequent clauses, ensuring semantic coherence in the concluded formulas. By using high-level predicates and transformations over ULF expressions, these are simple and interpretable at the top level. We use TTT (Purtee and Schubert, 2012) to define our tree-transductions rules as it provides a powerful and flexible way to declare tree transductions and supports custom predicate and mapping functions.

### 5.1.3 Named ULF Expression Predicates

The foundation of the interpretable predicates correspond to the ULF semantic types with syntactic features, e.g., `lex-pronoun?` which is true for any atom with a `.pro` suffix—a ULF pronoun. In line with TTT notation, we indicate predicates by ending the name with a question mark, `?`.

These are defined over the possible compositions of ULF expressions which includes, for example, `verb?` and `tensed-verb?` that match arbitrary untensed and tensed verb phrases in ULF. This extends to all distinct ULF constituent types: `noun?`, `adv?`, `term?`, `plural-term?`, `sent?`, etc. We supplement these with predicates that correspond to patterns or enumerations of ULFs that correspond specifically to the inference task in question. For example, `aux-indicating-request?` is a predicate that is true for eight ULF auxiliary forms that may indicate a request.[1]

### 5.1.4 Named ULF Expression Transformations

High-level tree transformation rules which correspond to natural semantic modifications are also defined and named. These are defined for transformations where the indexical nature and the looser syntactic constraints of ULF lead to non-trivial interactions with the syntactic structure. In other words, these rules are indexical and syntax-sensitive variants of simple EL inference rules. This includes rules such as `non-cf-vp!`, which transforms a counterfactual verb phrase (VP) to the corresponding factual VP, `negate-vp!`, which negates a VP, and `uninvert-sent!`, which transforms an subject-auxiliary inverted sentence, e.g., a question, to the uninverted form. We indicate transformation rules by ending the name with an exclamantion mark, `!`. Here are a couple of examples of `negate-vp!` transformations for clarity.

(1) *left the house → did not leave the house*

```
  ((past leave.v) (the.d house.n))
→((past do.aux-s) not (leave.v (the.d house.n)))
```

(2) *had met before → had not met before*

```
  ((past perf) meet.v before.adv-e)
→((past perf) not (meet.v before.adv-e))
```

[1] `can.aux-v`, `can.aux-s`, `will.aux-v`, `will.aux-s`, `would.aux-v`, `would.aux-s`, `could.aux-v`, and `could.aux-s`.

*"Can somebody help me?"*
(((pres can.aux-v) somebody.pro
(help.v me.pro)) ?)

$\Longrightarrow$

*"I want somebody to help me."*
(i.pro ((pres want.v) somebody.pro
(to (help.v me.pro))))

**Inference Rule**

```
(∀a,r,v [[[a aux-indicating-request?] ∧
          [r request-personal-pronoun?] ∧
          [v verb?] ∧
          (((pres a) r v) ?)]
         → (i.pro ((pres want.v) r (to v)))])
```

Figure 5.2: An example of an inference rule for inferring an underlying desire from a request. Infixed notation in the inference rule is marked with square brackets for readability. Generalizations and variants of the rule for handling extraneous sentence modifiers, such as *please*, are omitted for clarity.

Items (1) and (2) show that the way negation modifies a ULF verb phrase is dependent on the presence or absence of auxiliaries and aspectual operators (i.e. perfect and progressive aspect). And if this process results in a new head verb, the tense operator would need to be moved accordingly. In order to avoid directly managing these idiosyncratic syntactic phenomena in the inference rules, the VP negation is encapsulated into a single transformation rule.

### 5.1.5 Defining Inference Rules

The inferences rules are simple if-then relations defined over a structure where the predicates can appear in the antecedent and the named transformations can appear in the consequent. Figure 5.2 shows an inference rule for simple requests, written as a universal quantifier over ULF expressions. In practice, this rule is implemented using a TTT tree transduction rule. These rules can be formulated as EL meta-axioms (Morbini and Schubert, 2008) generalized with the named ULF expression predications and transformations to interface with the looser syntax of ULF and its representational idiosyncrasies inherited from English. Since the inferential categories we are exploring are a mixture of entailments, presuppositions, and implicatures their use in a general inference framework warrants additional management of projecting presuppositions and defusing

| Dataset | clause | counter | request | question | interest | ignored |
|---|---|---|---|---|---|---|
| Discourse Graphbank | 1,987 | 110 | 2 | 47 | 2,030 | 1,122 |
| Project Gutenberg | 264,109 | 31,939 | 2,900 | 60,422 | 303,306 | 275,344 |
| UIUC QC | 3,711 | 95 | 385 | 15,205 | 15,251 | 201 |
| Tatoeba | 291,905 | 15,650 | 11,201 | 100,412 | 353,656 | 333,618 |

Table 5.1: Statistics for each dataset on the result of the filtering process described in Section 5.1.7. **clause** and **counter** refer to clause-taking verbs and counterfactuals, respectively. **interest** refers to "interesting" sentences, that is, sentences that pass at least one of the filters.

implicatures.

### 5.1.6 Dataset Construction

We chose a variety of text sources for constructing this dataset to reduce genre-effects and provide good coverage of all the phenomena we are investigating. Some of these datasets include annotations, which we use only to identify sentence and token boundaries. The data sources are the same as those from the first release of ULF annotations (Section 3.3.1).

### 5.1.7 Pattern-based Filtering

As the phenomena that we want to focus on are relatively infrequent, we wrote filtering patterns to reduce the number of human annotations needed to get a sufficient amount of data for evaluation. Requests, for example, occur once in roughly every 100 to 1000 sentences, depending on the genre. The filtering is performed by first sentence-delimiting and tokenizing the source texts then matching these tokenized sentences over linguistically augmented regular expression patterns. The filtering patterns are designed for near-full recall of the targeted sentence types by retaining sentences that superficially look like they could be of those types.

The sentence-delimiters and tokenizers are hand-built for each dataset for a couple of reasons. First, general purpose models are likely to fail systematically on our multi-genre dataset and relatively infrequent phenomena, leading to unintended changes in the dataset distribution. Second,

**Basic if-then** `"<begin?>(if|If)<mid>(was|were|had|<past>|<ppart>)<mid?>(<futr>) .+"`
*If I thought this would make it difficult for the family , I would n't do it , " he said .* – Discourse Graphbank

**Inverted if-then** `"<begin?>(<futr>)<mid>if<mid>(was|were|had|<past>|<ppart>) .+"`
*Tom would n't have married Mary if he 'd known she had spent time in prison .* – Tatoeba

Figure 5.3: Example shorthand regex patterns (Section 5.1.7) for filtering candidate sentences with matching sentences.

the datasets have common patterns and existing annotations which can be exploited in a hand-built system. For example, the Discourse Graphbank follows the ends of sentences with a newline and in the Tatoeba and UIUC datasets each line is a sentence. The transparency of the rules also have the benefit that we can interpretably fix errors in their performance in the future.

These filtering patterns are written in augmented regex patterns. Figure 5.3 shows two such augmented regex patterns for plain and inverted if-then counterfactual constructions. The regexes are augmented with *tags* written in angle brackets, e.g., `<begin?>`. These tags refer to regex fragments that are reusable and conceptually coherent to people. `<begin?>` matches either the beginning of the string or space separated from previous text. `<mid>` matches words that are padded with spaces on the sides (i.e. separate tokens from what's defined next to it) and `<mid?>` is a variant that allows just a space as well. `<past>` and `<ppart>` are alternative lists of past tense and past participle verb forms. `<futr>` is an alternatives list of different conjugations of the word *will*. Tags for closed classes of words and shorthands for common non-word patterns were hand-curated. Tags for open classes such as `<past>` and `<ppart>` are generated from the XTAG morphological database (Doran et al., 1994) with minor edits during the development process.

### 5.1.8 Sentence selection

After performing filtering, we still have far too many sentences to feasibly annotate, so we build a balanced set of 800 sentences split evenly among the four sentence types we filtered for, *clause-taking verbs*, *counterfactuals*, *requests*, and *questions*. For each sentence type, we select the sentence round-robin between the four datasets to balance out the genres. Some types of sentences appear

more that 200 times in this sampling because some sentences pass multiple filters. For example, *"Could you open the door?"* passes both the *request* and *question* filters.

### 5.1.9 Inference Annotation

As we discuss in Section 5.1.17, evaluating automated forward inferences effectively is a major challenge. Every sentence leads to many inferences at various levels of discourse, certainty, and context-dependence. This is exacerbated by the ability to paraphrase the inferred statements. By limiting ourselves to inferences of particular general structures, we are able to elicit natural responses from people that are restricted to the particular phenomena that we are interested in investigating.

The annotations are separated into the same four categories as the filtering: *clause-taking verbs*, *counterfactuals*, *questions*, and *requests*. The annotator is first asked to select the structural inference pattern that holds for the given sentence and write down the corresponding inferred sentence. For example, say there is the sentence *"If I were rich, I would own a boat"*. The annotator would select an inference template along the lines of `(if <x> were <pred>, <x> would <q>)` → `(<x> is not <pred>)` and write down the inference *I am not rich*. This way we can get a fluent inference, but push the annotator to think about the inferences structurally. The annotators are additionally instructed to keep the inferred sentence as fluent as possible, preserve the original sentence as much as possible, and keep the perspective of the speaker of the sentence. We also included an option for annotators to add new rules, to extend the dataset into categories we did not anticipate. This category will be referred to as *Other*.

The annotations were performed by members of our research group, including some of the authors in the original publication. These were completed before starting the development of the inference system. There is the possibility of development being skewed by knowledge of the annotated data, but we expect this factor to be quite small since the core inference system was built by only a couple of the annotators and the bulk of this development was done several months after completion of the annotations. The annotations totaled 698 inferences from 406 sentences.[2]

[2]This is half of the original 800 sampled sentence after filtering sentences that had duplicates due to dataset artifacts

### 5.1.10 Evaluation

We developed the inference rules based on a set of 40 sentences randomly sampled from the annotated dataset. The correctness of these inferences is evaluated both through an automatic evaluation over the whole dataset and a human evaluation of a sample of the inferences. Both evaluations are done directly over English sentences by automatically translating the ULF inferences to English sentences. The automatic evaluation also involves a ULF rewriting module to handle semantically equivalent inference variants. All of these components are fine-tuned on the 40 sentence devset. In all of the experiments we start with human ULF annotations as a reliable ULF parser was not yet available at the time of this experiment.

### 5.1.11 ULF to English

The ULF-to-English translation is done in the following multi-stage process:

1. Analyzing the ULF type of each subexpression,
2. Incorporating morphological inflections based on the type analysis,
3. Filtering out purely logical operators, and
4. Mapping logical symbols to surface form counterparts.

The proximity of ULF to syntax and its preservation of most word order makes hand-building a robust function for this reasonably simple. The verb conjugations and noun pluralizations are performed using the `pattern-en` python package (Smedt and Daelemans, 2012).[3] The code for mapping ULFs to English is available at `https://github.com/genelkim/ulf2english`.

---

we failed to notice at the sentence selections stage and sentences that could not be annotated given the current ULF guidelines.

[3]The package code is currently available at `https://github.com/clips/pattern`.

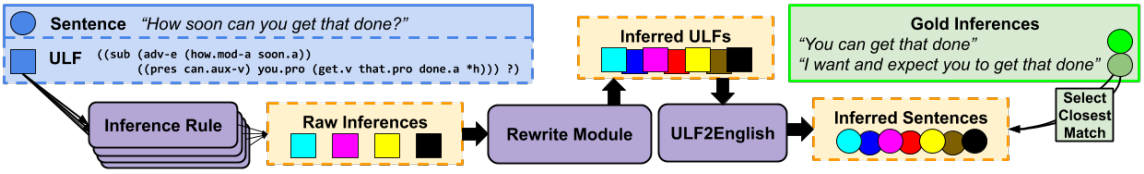



Figure 5.4: A diagram of the automatic ULF inference evaluation pipeline.

### 5.1.12 Rewriting Rules

The rewriting rules capture alternative ways to represent the same sentence without changing the meaning. This includes clausal restructuring (e.g., *"I expect that you come here"* to *"I expect you to come here"* or *"I expect you come here"*[4]), merging inferences (e.g., *"I want you to get that done"* and *"I expect you to get that done"* to *"I want and expect you to get that done"*) and others of this sort that are extremely unlikely to change the meaning of the sentence.

### 5.1.13 Automatic Inference Evaluation

A diagram of the automatic evaluation pipeline is shown in Fig. 5.4. The pipeline for a given source sentence and ULF proceeds as follows:

1. Use the inference rules (Section 5.1.2) to generate a set of raw inferences from the source ULF.

2. Generate a complete set of possible realizations of the inferred ULFs by rewriting the raw inferences into possible structural variations (Section 5.1.12).

3. Translate inferred ULFs into English to get a set of inferred sentences (Section 5.1.11).

[4]While this particular example *"I expect you come here"* is ungrammatical, in general *that* can often be omitted in propositional arguments of clausal verbs, e.g., *"I know ~~that~~ you like coffee"*. Here we opted for this general rule for clausal restructuring. A more nuanced alternative would be to use acceptability judgments of clausal verbs, i.e. the MegaAcceptability dataset (White and Rawlins, 2020, 2016), to filter out unacceptable combinations of clausal verbs and syntactic frames.

4. For each human inference elicited from the current source sentence, find the system-inferred sentence that has the smallest edit distance.

5. Report recall over human inferences with a max edit distance threshold.[5]

We use an edit distance threshold of 3. This allows minor English generation errors such as verb conjugations and pluralizations, but does not allow simple negation insertion/deletion (a difference of a space-separated *"not"* token). Figure 5.5 lists the results of this evaluation. The numerical values are fairly low, but this may be expected given the evaluation procedure. A trivial baseline such as most frequent devset inference or copying the source sentence would lead to a score of 0 or very close to 0 as these are very unlikely to be within a 3-character edit from the inferences in the dataset.

| | **counterfactual** | **clause-taking** | **request** | **question** | **other** | **all** |
|---|---|---|---|---|---|---|
| Recall | 1/13 (8%) | 1/33 (3%) | 33/97 (34%) | 69/316 (22%) | 7/130 (5%) | 112/662 (18%) |

Figure 5.5: Results of automatic inference evaluation described in Section 5.1.13.

### 5.1.14 Human Inference Evaluation

The human inference evaluation was performed over 127 raw ULF inferences. This set was constructed of 100 randomly sampled inferences with the addition of every counterfactual and clause-taking inference as they are not as common. Each inference was then translated to English and presented alongside the source sentence to 3 to 4 independent human judges. The judges evaluated correctness of the discourse inference and the grammaticality of the output sentence. Table 5.2 presents the results of this. 87 of the 127 inferences were marked as correct by a majority of judges and only 21 were marked as incorrect by a majority of judges, for the remaining 19

[5]We do not report precision over automatic inferences because missing inferences are common in our dataset. This could be alleviated in the future by explicitly splitting the inference elicitation task into smaller subtasks and/or incorporating a reviewing stage where initial inferences are reviewed, corrected, and possibly added to by a second person.

| | counterfactual | clause-taking | request | q-pre | q-act | other | all |
|---|---|---|---|---|---|---|---|
| Correct* | 11/27 | 2/5 | 17/19 | 13/21 | 31/39 | 13/16 | 68.5% |
| Incorrect* | 9/27 | 3/5 | 0/19 | 3/21 | 3/39 | 3/16 | 16.5% |
| Context* | 7/27 | 0/5 | 2/19 | 5/21 | 5/39 | 0/16 | 15.0% |
| Grammar | 20/27 | 1/5 | 19/19 | 12/21 | 33/39 | 14/16 | 78.0% |

Table 5.2: Results of majority human evaluation of system generated inferences. Evaluation on 127 inferences with from the test set by 3 or 4 people per inference. **Correctness is evaluated on whether the sentence is a reasonable inference in conversation, allowing for some awkwardness in phrasing. Context, means the correctness is highly context-dependent.* The inference type labels in the header row are the same as in Fig. 5.5 except for the addition of breaking down questions to **q-pre** for question presuppositions and **q-act** for question act inferences.

inferences judges either disagreed completely or a majority judged it as context-dependent. 99 of the 127 inferences were judged grammatical by a majority of judges, which demonstrates the efficacy of the ULF-to-English translation system.[6] The system seems to struggle most with counterfactual and clause-taking inferences.

### 5.1.15 Evaluation of Rewriting Rules

In order to verify that the rewriting rules in fact preserve the semantic meanings, we gathered a sample of 100 system-inferred sentences that were closest to a gold inference (Item 4 in Section 5.1.13). Each inferred sentence is judged as whether it is a valid rewrite of one or more of the raw inferences. A valid rewrite is one that does not introduce new semantic information. 91 out of the 100 were judged as valid by a majority of three human judges. As such, the rewriting system is not abusively over-generating sentences that are semantically different and match to gold inferences, increasing the recall score.

[6] Some inferences marked as ungrammatical were also marked as correct, indicating that the ULF-to-English failures can be minor enough to be easily understood.

### 5.1.16 Analysis and Discussion

The human inference evaluation (Section 5.1.14) showed that the system struggles most with counterfactual and clause-taking verb constructions. This is largely because the sampling procedures for these constructions are not as effective, leading to fewer positive examples in our dataset. In turn, our development set of 40 sentences only included a handful of examples of each inference, so the rules remained brittle after adjusting to the development set. In fact, two of the three incorrect clause-taking verb inferences are a result of a simple mistake of allowing arbitrary terms rather than only reified sentences and verbs in the antecedent.

Some of the automatic inferences were impossible to handle using our inference rules because of disagreements among human elicited inferences on what circumstances warrant particular inferences and how precisely an inference should be expressed. For example, the distinction between the presence or absence of the word “probably” is best handled with a separate confidence metric. In conversations, the distinction between highly probable statements and simply true statements is blurred. One could choose to include or omit “probably" for statements where the possibility of the plain sentence being false is small. Still, we would not want to add this as a rewriting rule since strictly speaking, such hedges do affect the meaning. Similarly, human elicited inferences disagreed on whether requests warrant a question act inference (e.g., *“Could you open the door?” → “You know whether you could open the door”*). We opted to avoid generating these inferences in building our rules, which significantly affected the recall score in the automatic evaluation.

The ULF-to-English generation system is remarkably accurate given its fairly simple pipeline approach and given that this is the first real use of this system. 78% grammaticality shows room for improvement and a cursory review of the errors show that there are some ULF macros that still need handling and that verb conjugations need to be made more robust.

Given these results, improvements to the filtering system for counterfactual and clause-taking verb constructions, gathering a larger dataset with a more robust collection procedure, and another set of experiments with the larger dataset would be valuable next steps in more precisely measuring

the use of ULF in generating discourse inferences.

### 5.1.17 Related Work

Inference demonstrations have been performed in the past for various symbolic semantic representations (e.g., those discussed in Section 1.2), showing their respective strengths. Similarly, distributional representations have been shown to be effective for RTE, such as in the SNLI and MultiNLI datasets (Bowman et al., 2015; Williams et al., 2018). These datasets are much larger than previous RTE datasets and both provide classification tasks supporting the use of an implicit distributional representation in a neural network system. The discourse inferences that are demonstrated in this section with ULFs as well our evaluation method based on reliable English generation, are a challenge to all of the semantic representations discussed in Section 1.2 because of their relative remoteness from syntax. Some of our inferences require access to syntactic cues and evaluations using English are difficult with models that are abstracted too far away from the surface form and syntax.

In the realm of evaluation methods, our work has similarities with the TAC KBP slot-filling task, which defines specific types of information that the system is meant to extract from the text without knowledge of the possible correct answers (Ellis et al., 2015). But TAC KBP focuses on restricted types of factoids, whereas our evaluation focuses on structure-based sentential inferences. In recent years inference evaluations have typically been posed as either a classification tasks similar to RTE (Bowman et al., 2015; Williams et al., 2018) or multiple-choice question answering (Clark et al., 2018). This knowledge of possible alternatives allows systems to avoid modeling inferences explicitly and to exploit statistical artifacts (Jia and Liang, 2017; Poliak et al., 2018; Gururangan et al., 2018). The inference model trained on the ATOMIC commonsense dataset was evaluated without providing a set of possible choices by using BLEU (Sap et al., 2019). Though BLEU scores tend to correlate with correct inferences in practice, using it as a metric of evaluation is fraught with danger. Small changes that dramatically alter the meaning of a sentence (e.g., negation) are not reflected in the BLEU scores, and for structurally oriented inferences, incorrect inferences are likely

to have misleadingly high scores.

### 5.1.18 Conclusion

This section presented a method of generating discourse inferences from ULF and an evaluation of such inferences. We also presented a method of collecting human elicitations of restricted categories of structural inferences, allowing a novel forward inference evaluation. We used these elicited inferences to automatically evaluate the generated inferences with promising results. Human judgments on a sample of generated inferences showed that 68.5% of the inferences are reasonable discourse inferences, 16.5% were unreasonable, and 15% were context-dependent or had disagreements between judges. Our experiments also demonstrate some of the advantages of using a semantic representation closer to the syntactic form like ULF—reliable translation to English and access to syntactic signals—though this comes at the cost of a more complicated interface with the semantic patterns. In the following sections of this chapter we will focus on generating natural logic-like monotonicity inferences from ULFs.

## 5.2 A Theoretical Framework for Natural Logic Inference with ULFs

### 5.2.1 Introduction

Natural Logic is an approach to generating inferences from language directly over the grammatical structure through knowledge of entailment monotonicity in the lexicon. An important fragment of natural logic is monotonicity calculus which operates using syntactic structure and the knowledge of polarity inducing elements and monotonicity relationships. A monotone function between partially ordered sets either preserves or inverts the ordering of argument values. More precisely, a function $f$ is said to be upward monotone if $x \leq y$ implies $f(x) \leq f(y)$. Similarly, $f$ is said to be downward monotone if $x \leq y$ implies $f(x) \geq f(y)$. If neither of these hold, $f$ is said to be non-monotone. When used in the context of subset relations and entailment, monotonicity can be a

| | |
|---|---|
| **Up** (FraCaS) | *Some delegates (finished the survey on time)*$^{\blacktriangle}$ ⇒ *Some delegates finished the survey* |
| **Down** (MED) | *I never had a (girlfriend)*$^{\blacktriangledown}$ *before* ⇒ *I never had a girlfriend taller than me before* |
| **Non-** (MED) | *Exactly 12 aliens read (magazines)*$^{\blacksquare}$ ⇏⇎ *Exactly 12 aliens read (news magazines)*$^{\blacksquare}$ |

Figure 5.6: Upward (...)$^{\blacktriangle}$, downward (...)$^{\blacktriangledown}$, and non- monotone (...)$^{\blacksquare}$ examples from the FraCaS and MED datasets.

tool for making natural language inferences. For instance, consider the second example in Fig. 5.6. *Never* is downward monotone in entailment, since it flips the entailment ordering of (1) *I had a girlfriend taller than me before* entails (2) *I had a girlfriend before* to (2) ***I never*** *had a girlfriend before* entails (1) ***I never*** *had a girlfriend taller than me before*. Natural logic is an approach to generating natural language inferences based on syntactic structure and knowledge of the semantic properties of the lexical items and local constructions (Van Benthem et al., 1986; Sánchez Valencia, 1991b). It can be seen as an extension of Aristotelian syllogistic reasoning (Van Benthem et al., 1986) and was first formally related to higher-order logic entailments by Sánchez Valencia (1991a). Icard and Moss (2014) and Icard et al. (2017) later construed Natural Logic as a formal system of its own, independent of a separate logical formalism. Figure 5.6 shows the three basic cases of monotonicity inference, upward, downward, and non- monotone contexts leading to different entailment conditions.

Here we present a proof-based Natural Logic inference formalism for ULF. We show that this method covers inferences presented by Sánchez Valencia (1991a) and can support Rule Instantiation from Episodic Logic (EL) inference for nested polarity inference. The contributions of this section are two-fold: (1) this marks the first formalized inference procedure for ULF and (2) we present an alternative to parsing full logical formulas in symbolic Natural Logic inference through the use of an underspecified representation.

### 5.2.2 Natural Logic

We will limit our discussion of Natural Logic to that presented by Sánchez Valencia (1991a), which I will refer to as NLog), and follow his notation and terminology.[7] The semantics of NLog is rooted in an undirected, typed lambda calculus constructed from derivations of Lambek cum Permutation Calculus (Lambek, 1988). The primitives semantic types are $e$ and $t$ (for entities and truth values) which denote sets, with complex types of the form $\langle a, b\rangle$ where $a$ and $b$ are semantic types in the language. There is one inference rule over these, $\langle \alpha, \beta\rangle$, $\alpha \rightarrow \beta$, where the order of the functor and the argument do not matter. Note that similarly, the ordering of operator-argument relations in ULF can have the operator in the first or second position, disambiguated by the types of the participating expressions. The EL type system has the same single function application for combining types, $\langle A, B\rangle, A \rightarrow B$. Here are the formal definitions for monotonicity.

**Definition 5.2.1.** Monotonicity is defined over the partial ordering relation $\leq_a$ which is defined as follows, where $a$ is a semantic type and $\mathcal{D}_a$ is the corresponding set:

- If $\alpha, \beta \in \mathcal{D}_e$ then $\alpha \leq_e \beta$ iff $\alpha = \beta$.
- If $\alpha, \beta \in \mathcal{D}_t$ then $\alpha \leq_t \beta$ iff $\alpha = \bot$ or $\beta = \top$.
- If $\alpha, \beta \in \mathcal{D}_{(c,d)}$ then $\alpha \leq_{(c,d)} \beta$ iff for each $\kappa \in \mathcal{D}_c$ , $\alpha(\kappa) \leq_d \beta(\kappa)$.

**Definition 5.2.2.** Monotonicity for functions $f \in \mathcal{D}_{(a,b)}$ is defined over this ordering as follows:

- $f \in \mathcal{D}_{(a,b)}$ is *upward monotone* iff for all $x, y \in \mathcal{D}_a$, $x \leq_a y$ entails $f(x) \leq_b f(y)$.
- $f \in \mathcal{D}_{(a,b)}$ is *downward monotone* iff for all $x, y \in \mathcal{D}_a$, $x \leq_a y$ entails $f(y) \leq_b f(x)$.
- $f \in \mathcal{D}_{(a,b)}$ is *non-monotone* if it is neither upward or downward monotone.

**Definition 5.2.3.** Monotonicity of an occurrence $M$ in $N$ is defined relative to its semantic interpretation, where $I$ is the interpretation function such that:

[7] Much of the recent work in Natural Logic uses the terminology and notation introduced by Icard and Moss (2014).

- $M$ is *upward monotone* in $N$ iff $I(M) \leq I(M')$ entails $I(N) \leq I(N\{M/M'\})$ for all models and assignments.
- $M$ is *downward monotone* in $N$ iff $I(M) \leq I(M')$ entails $I(N\{M/M'\}) \leq I(N)$ for all models and assignments.[8]

Using this, Sánchez Valencia (1991b) proves that positive and negative polarity items from the prior Natural Logic literature corresponds to upward and downward monotone occurrences. And from this correspondence the soundness of substituting supersets for subsets in positive polarities and vice versa is realized.

**Inference with Natural Logic**

Sánchez Valencia (1991a) reasons using a tableau proof system (Beth, 1955) with nodes of the form

$$a_1, a_2, ..., a_n \bullet b_1, b_2, ..., b_m$$

where $a_i, 1 \leq i \leq n$ and $b_j 1 \leq j \leq m$ are English expressions with corresponding Lambek derivations $a_i'$ and $b_j'$, which supply grammatical information (scoping and polarity) to the proof. The proof starts with the premises on the left side ($a_i$) and the desired conclusions on the right side ($b_j$). The proof concludes when all paths of the proof tree are closed. A path is closed when the leaf node of the path has the same statement (including scope markings) on both sides of the node. For those unaware of the notation of tableau systems, this node can be interpreted as the following well-formed formula.

$$(A_1 \wedge A_2 \wedge ... \wedge A_n) \rightarrow (B_1 \vee B_2 \vee ... \vee B_m)$$

A tableau step, e.g.,

[8] $N\{M/M'\}$ is shorthand for $M'$ substitutes for $M$ in $N$.

$$\begin{array}{l} a \\ a' \end{array} \bullet\!\!-\!\!\bullet \begin{array}{l} b \\ b' \end{array}$$

can be interpreted as the formula

$$(A \to B) \longleftrightarrow (A' \to B').$$

### 5.2.3 Scope and Polarity Marking in NLog

Sánchez Valencia's (1991a) monotonicity and scope marking rules are

$$\begin{array}{l} X \\ X \end{array} \bullet\!\!-\!\!\bullet \begin{array}{l} A, Y \\ A^*, Y \end{array} \qquad\qquad \begin{array}{l} A, X \\ A^*, X \end{array} \bullet\!\!-\!\!\bullet \begin{array}{l} Y \\ Y \end{array}$$

where $A^*$ is a monotonicity or scope marking of $A$ provided by the fixed Lambek analysis of $A$. Monotonicity can take values + and − and scoping is marked with $()^{\#}$ where the parentheses circumscribe the words associated with the top-level scope.

Now that we know how to read both the tableau steps and the scoping rules we can take a look at a simple example in Fig. 5.7. Figure 5.7 also lists the basic inference rules for Sánchez Valencia's natural logic proof system. We will now describe corresponding inference and scoping mechanisms for ULF and define corresponding inference rules to demonstrate the same inference capacity using ULF as a basis.

### 5.2.4 Scope Marking ULFs

Rather than using a Lambek analysis for identifying the operator scopes and as a result the polarities, scoped logical forms (SLFs) are used, which are ULFs with disambiguated scopes.[9] The conversion from ULF to SLF can be denoted mid-proof so that a specific scoping does not need to be committed to at the start of the proof.

[9] See Section 2.1.3 for details of how SLF fits into the full EL interpretation process.

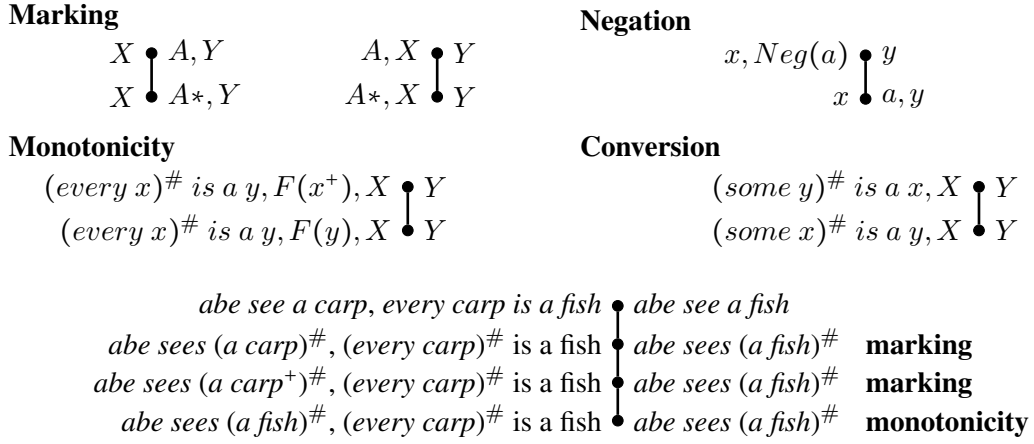



Figure 5.7: The basic inference rules for Sánchez-Valencia's natural logic proof system and an example.

### Mapping ULFs to SLFs

Scoping for ULFs comes in two flavors:

(S1) **Independent scoped operators.** An independent scoped operator is one that simply raises up to any wff level and introduces some information to only that scope of the overall formula. This includes tense, aspect, and sentence-level adverbials operators. These operators add temporal, locative, or general additional contextual information to the wff.

```
(|Abelard| ((past see.v) him.pro yesterday.adv-e))
```

↓ scoping

(past (yesterday.adv-e (|Abelard| (see.v him.pro))))

(S2) **Determiners with restrictors.** When determiners are scoped, they bring with them the restrictor predicate. A variable is introduced which is placed in the position of the lifted constituent and this variable is quantified with the lifted determiner and restricted by the restrictor predicate.

(|Abelard| (see.v (a.d carp.n)))

↓ scoping

(a.d $x$: ($x$ carp.n) (|Abelard| (see.v $x$)))

Figure 5.8 shows an example of a ULF, SLF pair with both types of scopings.

(|Ali| (do.aux-s **not** (know.v (that
(i.pro (work.v (adv-a (with.p **(a.d dog.n)** ))))))))
↓ **scoping**
**(not** (|Ali| (do.aux-s (know.v (that
**(a.d x: (x dog.n)**
(i.pro (work.v (adv-a (with.p **x** )))) ) )))) **)**

Figure 5.8: Example of a ULF scoping into an SLF for the sentence *"Ali does not know that I work with a dog"*.

### Scope Marking SLFs

In accordance with Sánchez-Valencia's treatment, we will only perform scope marking on the (S2) classes of scoping operators. Fortunately, this is the more interesting one from a structural perspective. The scoping of ULFs with (S2) classes leads to a more decoupled representation of the constituent, so we must define a correspondence between these components that allows us to still mark the scoping of the ULF based on the fixed scoped realization.

The key to making a correspondence between the (S2)-type constituent in ULF and SLF is the quantified variable of the SLF. For a ULF, $\psi$ which contains a quantified expression $\wr$ of form the ($\delta$ $\pi$) where $\delta$ is a determiner and $\pi$ is a predicate, the corresponding formula with ($\delta$ $\pi$) at the top-level scope is ($\delta$ $x$: ($x$ $\pi$) $\psi\{\wr/x\}$), where $\phi\{a/b\}$ represents a substituting $b$ for all instances of $a$ in $\phi$.

**Top-level scope marking process.** Knowing this correspondence, we have a path to marking the ULF quantified expression constituent that is the top-level scope. Given the SLF which defines the

scope ordering, the constituent of the form ($\delta$ $\pi$) in $\psi$ at the position of $x$ in $\psi\{\wr/x\}$ is marked as the top scope of $\psi$. Note that $\wr$ is not an alias for the pattern ($\delta$ $\pi$). Rather it refers to a unique constituent of $\psi$ which has the form ($\delta$ $\pi$). This is an important distinction in order to properly handle sentences with multiple constituents of the same form, e.g. *"A dog greets a dog"*. Below is an example to help illustrate the mapping.

*"Abelard sees a carp"*

**SLF** (a.d $x$: ($x$ carp.n) (|Abelard| (see.v x)))

**Marked ULF** `(|Abelard| (see.v (a.d carp.n)#))`

$\boldsymbol{\delta}$: `a.d`, $\boldsymbol{\pi}$: `carp.n`

In practice, we will not use the actual these scope markings for inference since we do not use the tableau method for inference. Rather, we use this process to identify the ULF constituent with the top-level scope on the fly using a process which retains the same inferential capacity to the marking in tableau method.

### 5.2.5 Polarity marking in ULF

We perform polarity marking in a two stage process that mirrors the process used by Sánchez Valencia (1991a). First we classify lexical entries according to their monotonicity properties, called lexical monotonicity markings—in which direction they place entailment contexts on their arguments—and mark them in the SLF with subscripts. For example, the determiner `no.d`, which places both its restrictor and scope (its two arguments) in a downward entailment context, is marked as $\texttt{no.d}_{(-,-)}$. The

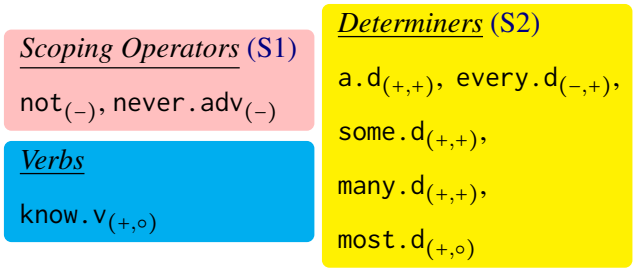



Figure 5.9: Examples of lexical monotonicity markings.

possible entailment options are + for upward, − for downward, and ∘ for flat. Clausal verbs overload this notation, for example $\mathsf{know.v}_{(+,\circ)}$, for their monotonicity properties over their subject (here +) and clausal argument (here ∘). Figure 5.9 provides more examples of lexical monotonicity markings.

Unmarked lexical entries are assumed to place all of their arguments in upward entailment context. Using the lexical annotations, we mark the local entailment direction of the argument constituents in the SLF using subscripts again. For example, the SLF with lexical monotonicity markings for *"no scientist knows every scientific fact"*,

$$\begin{aligned}&(\text{no.d}_{(-,-)}\ x\text{: }(x\ \text{scientist.n})\\ &\qquad(\text{every.d}_{(-,+)}\ y\text{: }(y\ (\text{scientific.a fact.n}))\\ &\qquad\qquad(x\ (\text{know.v}\ y)))),\end{aligned}$$

gets its arguments marked as follows.[10]

$$\begin{aligned}&(\text{no.d}_{(-,-)}\ x\text{: }(x\ \text{scientist.n})_{-}\\ &\qquad(\text{every.d}_{(-,+)}\ \text{y: }(y\ (\text{scientific.a fact.n}))_{-}\\ &\qquad\qquad(x\ (\text{know.v}\ y))_{+})_{-})\end{aligned}$$

Finally, the global polarity contexts are derived from these local entailment directions and marked with superscripts. The global polarity is derived by traversing the SLF from the root and counting the number of occurrences of negative and flat entailments. The global polarity is computed form this using the following rules, applied in order.

1. Node $a$ has no polarity if any node in the path from the root to $a$ is marked with ∘ (flat local entailment).

2. Else, node $a$ has negative polarity if there are an odd number of nodes between the root and $a$ (inclusive) marked with − (downward local entailment).

3. Otherwise, node $a$ has positive polarity.

[10]This example ignores the cases with default lexical monotonicity marking values for readability.

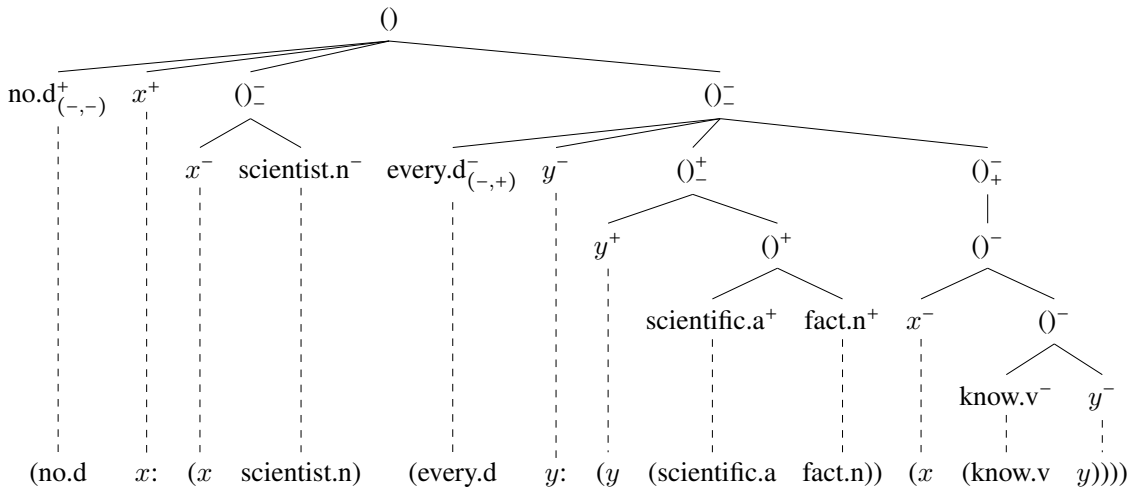



Figure 5.10: A tree representation of the SLF for *"No scientist knows every scientific fact."* with all lexical monotonicity, local entailment context, and global polarity markings.

Following these rules the argument marked SLF gets the following global polarity marking—limiting global polarity marking to just nodes with local entailment marking for readability.

(no.d$_{(-,-)}$ $x$: ($x$ scientist.n)$^{-}_{-}$
(every.d$_{(-,+)}$ $y$: ($y$ (scientific.a fact.n))$^{+}_{-}$
($x$ (know.v $y$))$^{-}_{+}$)$^{-}_{-}$)

Figure 5.10 shows the polarity propagation in a tree diagram, which makes it easier to follow.

The global polarity marking in the ULF is set according to the corresponding constituent in the SLF. The following is the resulting ULF for the above SLF, only marking constituents that correspond to predicates in predicative position in the SLF.

```
((no.d scientist.n⁻) (know.v⁻ (every.d (scientific.a fact.n)⁺)))
```

We now show that these markings result in inferences that follow our intuition. From the sentence *"No scientists knows every scientific fact"* we want to infer

(1) a. No physicist knows every scientific fact

b. No scientist knows every fact

but not

(2) a. *No person knows every scientific fact

b. *No scientist knows every physics fact

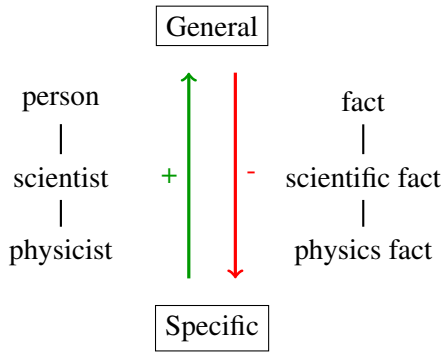



Figure 5.11: Monotonicity orderings of the predicates that are involved in the polarity inference example. Lower predicates are subsets of predicates above them. Predicates can be replaced with those above them in positive polarity and below them in negative polarity.

Figure 5.11 shows the monotonicity orderings of the predicates that are involved in this example and the direction of warranted inference in each polarity. In positive polarity, we are warranted inference upward on this diagram, and in negative polarity, inference downward. In our sentence, *scientist* is in negative polarity which warrants replacement by *physicist*, which is indeed intuitively warranted in English. As is desired from the motivating sentences, *scientist* is in negative polarity and can be replaced with *physicist* (1a) and *scientific fact* is in positive polarity and can be replaced with *fact* (1b).

### 5.2.6 Inferences using ULFs

Now we will go over how to make inferences using ULFs with the scope and polarity markings. Where appropriate, we also make correspondences to Sánchez Valencia's (1991a) inference rules.

#### Negation

Sánchez Valencia's (1991a) negation rule is

$$\begin{array}{l} x, Neg(a) \\ x \end{array} \begin{array}{l} \bullet\ y \\ \bullet\ a, y \end{array}$$

whic in simple logical terms, this rule is the following.

$$((X \wedge \neg A) \rightarrow Y) \leftrightarrow (X \rightarrow (A \vee Y)) \tag{5.1}$$

Negation of ULFs is performed by applying the logical `not` operator so no special rule is necessary. For instance, the negation of the ULF `(|Abelard| walk.v)` is `(not (|Abelard| walk.v))`.

Some identities that are useful for inferences are listed below, in the form of inference rules. Proofs for these identities can be given using the formal definitions of the respective generalized quantifiers. The rules are given using SLFs for structural simplicity.

$$\frac{\text{(not (not } \phi\text{))}}{\phi} \qquad \frac{\phi}{\text{(not (not } \phi\text{))}}$$

$$\frac{\text{(not (some.d } \nu\text{: } \psi\ \phi\text{))}}{\text{(no.d } \nu\text{: } \psi\ \phi\text{)}} \qquad \frac{\text{(no.d } \nu\text{: } \psi\ \phi\text{)}}{\text{(not (some.d } \nu\text{: } \psi\ \phi\text{))}}$$

$$\frac{\text{(not (a.d } \nu\text{: } \psi\ \phi\text{))}}{\text{(no.d } \nu\text{: } \psi\ \phi\text{)}} \qquad \frac{\text{(no.d } \nu\text{: } \psi\ \phi\text{)}}{\text{(not (a.d } \nu\text{: } \psi\ \phi\text{))}}$$

**Handling `be.v`**

While `be.v` are included in ULFs for simplifying the interface to natural language since the copula can act as an anchor for modifications from adverbial phrases and temporal information from its conjugation, we can consider it to be semantically void with respect to its arguments.

**SLF Rule 1 (`be.v` Elimination).**

$$\frac{\text{(a.d } y\text{: } (y\ P)\ (x\ \text{(be.v (= } y\text{))))}}{(x\ P)}$$

where $P$ is an arbitrary unary predicate and $x$ is an arbitrary entity.

**Proof**

| | |
|---|---|
| 1. (a.d $y$: ($y$ $P$) ($x$ (be.v (= $y$)))) | Assumption |
| 2. $I$((a.d $y$: ($y$ $P$) ($x$ (be.v (= $y$))))) | Interpretation function |
| 3. There exists $d \in \mathcal{D}$ s.t. $d \in I(P)$ and $I((x \text{ (be.v (= } y\text{)))})^{\mathcal{U}_{y:d}}$ | Satisfaction conds of $\exists$ |
| 4. There exists $d \in \mathcal{D}$ s.t. $d \in I(P)$ and $I((x \text{ (= } y\text{))})^{\mathcal{U}_{y:d}}$ | Definition of be.v |
| 5. There exists $d$ s.t. $d \in I(P)$ and $(I(x) = d)$ | $I$(=) |
| 6. $I(x) \in I(P)$ | Variable substitution |
| 7. ($x$ $P$) | Interpretation function of predication |

With this inference rule we can easily derive a predicate subset defining inference rule which is necessary for polarity inferences.

$$\frac{\text{(every.d } x\text{: (}x\ P1\text{) (a.d } y\text{: (}y\ P2\text{) (}x\text{ (be.v (= }y\text{)))))}}{\text{(every.d } x\text{: (}x\ P1\text{) (}x\ P2\text{))}}$$

where $P1$ and $P2$ are arbitrary unary predicates.

For example, from the initial SLF for *"every carp is a fish"* we get a nice relationship between the predicates carp.n and fish.n.

$$\text{(every.d } x\text{: (}x \text{ carp.n) (a.d } y\text{: (}y \text{ fish.n) (}x \text{ (be.v (= } y\text{)))))}$$
$$\Downarrow$$
$$\text{(every.d } x\text{: (}x \text{ carp.n) (}x \text{ fish.n))}$$

In this formula, every.d can generalize to other universal quantifiers such as all.d and a.d can generalize to other existential quantifiers such as some.d and an.d.

**Monotonicity Inference**

The basic monotonicity inference rule in natural logic takes a subset relationship between two predicates, $P1 \subseteq P2$, and a formula, $f$, where something of type $P1$ occurs in positive polarity.

Then we can assert $f'$ which is the same as $f$ except that $P2$ is substituted for $P1$. We can state a direct analog of this rule using SLFs.

We also formulate a similar rule which takes the subset relationship $P1 \subseteq P2$ and a formula $g$, where something of type $P2$ appears in negative polarity. In this case we can assert $g'$ which is the same as $g$ except that $P1$ is substituted for $P2$. Using SLFs the inferences looks as follows.

**SLF Rule 2 (Monotonicity Inference, SMI).**

$$\frac{\phi[P1^{+}],\ \text{(every.d } y\text{: } (y\ P1)\ (y\ P2))}{\phi\{P1/P2\}} \qquad \frac{\phi[P2^{-}],\ \text{(every.d } y\text{: } (y\ P1)\ (y\ P2))}{\phi\{P2/P1\}}$$

The SLFs are necessary for keeping track of the outer scope and determining the polarities, but the core inference can be written using ULFs, closer to surface form, with the SLFs acting as auxiliary information to ensure consistency of the formulas. Using ULFs and chaining SMI and `be.v` elimination we get the following inferences (again, these can be generalized to allow other universal and existential quantifiers).

**ULF Rule 1 (Monotonicity Inference, UMI).**

$$\frac{\phi[P1^{+}],\ \text{((every.d } P1)\ \text{(be.v (= (a.d } P2))))}{\phi\{P1/P2\}} \qquad \frac{\phi[P2^{-}],\ \text{((every.d } P1)\ \text{(be.v (= (a.d } P2))))}{\phi\{P2/P1\}}$$

It is worth noting that if the restrictor of a determiner is a conjunction of predicates restricting the variable, then due to the upward entailing nature of the $\wedge$ operator we can propagate the polarity induced by the determiner on its restrictor to each term in the conjunction. Also, we know that the $\wedge$ operator preserves subset relations, that is if $x$ satisfies predicates $P1$ and $P2$ and if every element of $P1$ is also in $Q$, then $x$ must satisfy the predicates $Q$ and $P2$. Therefore, in a case where a variable is restricted by a conjunction of predicates, it is possible to use the monotonicity inference rule on individual predicates in the restrictor. This is particularly useful when dealing with extensionally modified predicates (Section 5.2.6).

**Example 1.** Now we will use the presented ULF/SLF marking and inference rules to perform an inference over generalized quantifiers that Sánchez Valencia (1991a) demonstrated: from *"Abelard*

*sees a carp"* and *"Every carp is a fish"* we will conclude *"Abelard sees a fish"*. Before we start the inference, we walk through the scoping and polarity derivation of assumption 1, which will be used in the inference. This derivation takes the place of the Lambek derivations used by Sánchez Valencia to get polarity and scoping information into the proof.

**Scoping and Polarity Derivation**

| | |
|---|---|
| 1. `(\|Abelard\| (see.v (a.d carp.n)))` | ULF |
| 2. (a.d $x$: ($x$ carp.n) (\|Abelard\| (see.v $x$))) | Only possible scoping |
| 3. (a.d$_{(+,+)}$ $x$: ($x$ carp.n) (\|Abelard\| (see.v $x$))) | a.d lexical monotonicity |
| 4. (a.d $x$: ($x$ carp.n)$_+$ (\|Abelard\| (see.v $x$))$_+$) | Local entailment contexts |
| 5. (a.d $x$: ($x$ carp.n)$^+$ (\|Abelard\| (see.v $x$)$^+$)$^+$) | Global polarity |

Now for the actual proof.

**Proof**

| | |
|---|---|
| 1. `(\|Abelard\| (see.v (a.d carp.n)))` | Assumption |
| 2. `((every.d carp.n) (be.v (= (a.d fish.n))))` | Assumption |
| 3. `(\|Abelard\| (see.v (a.d carp.n)`$^+$`))` | Polarity marking, 1. |
| 4. `(\|Abelard\| (see.v (a.d fish.n)))` | UMI, 2.,3. |

**Inferences with Predicate Modifiers**

Let $P'$ be an extensional modification of a predicate $P$ and let $P_m$ be the modifying predicate. Since the modification is extensional, we know that an entity $x$ satisfies $P'$ if and only if it satisfies both $P$ and $P_m$. Hence we get the following rule.

**SLF Rule 3.**

$$\frac{(\delta\ x\text{: }(x\ P')\ \phi(x))}{(\delta\ x\text{: }((x\ P) \wedge (x\ P_m))\ \phi(x))}$$

where $P'$ is an extensional modification of the predicate $P$ with modifying predicate $P_m$, and $\delta$ is a determiner.

This rule can then be combined with monotonicity inference to get

**ULF Rule 2.**

$$\frac{\phi[(\delta\ (M\ P1))^{+}],\ ((\text{every.d}\ P1)\ (\text{be.v}\ (=\ (\text{a.d}\ P2))))}{\phi[(\delta\ (M\ P2))]}$$

where $M$ is an extensional modifier and $\delta$ is a determiner. A similar rule for negative polarities can be written as well.

This allows us to make another inference demonstrated by Sánchez Valencia (1991a).

**Example 2.** From *"Abelard sees a male carp"* and *"Every carp is a fish"*, we will conclude *"Abelard sees a male fish"*.

**Scoping and Polarity Derivation**

| | |
|---|---|
| 1. `(|Abelard| (see.v (a.d (male.a carp.n))))` | ULF |
| 2. (a.d $x$: ($x$ (male.a carp.n)) (|Abelard| (see.v $x$))) | Only possible scoping |
| 3. (a.d $x$: (($x$ male.a) $\wedge$ ($x$ carp.n)) (|Abelard| (see.v $x$))) | Assume intersective modification |
| 4. (a.d$_{(+,+)}$ $x$: (($x$ male.a) $\wedge$ ($x$ carp.n)) (|Abelard| (see.v $x$))) | a.d lexical monotonicity |
| 5. (a.d $x$: (($x$ male.a) $\wedge$ (x carp.n))$_{+}$ (|Abelard| (see.v $x$))$_{+}$) | Local entail. context |
| 6. (a.d $x$: (($x$ male.a)$_{+}$ $\wedge$ ($x$ carp.n)$_{+}$) (|Abelard| (see.v $x$))$_{+}$) | Upward entail. of $\wedge$ |
| 7. (a.d $x$: (($x$ male.a)$^{+}$ $\wedge$ ($x$ carp.n)$^{+}$) (|Abelard| (see.v $x$)$^{+}$)$^{+}$) | Global polarity |

**Proof**

| | |
|---|---|
| 1. `(|Abelard| (see.v (a.d (male.a carp.n))))` | Assumption |
| 2. `((every.d carp.n) (be.v (= (a.d fish.n))))` | Assumption |
| 3. `(|Abelard| (see.v (a.d (male.a carp.n$^{+}$))))` | Polarity marking, 1. |
| 4. `(|Abelard| (see.v (a.d (male.a fish.n))))` | UMI, 2.,3. |

The need for addressing the intersective nature of the modification in *male fish* brings up a benefit of using ULFs as a basis for the inferences. Since ULF is explicitly underspecified, the assumptions made during the inference process must be stated. The corresponding proof presented by

Sánchez Valencia hides the assumption of intersective modification in the lexical monotonicity marking of *male* (as $((e,t)^+,(e,t))$). In ULF, modifications are assumed to be intensional unless otherwise assumed, so the intersective nature of the modifier `male.n` must be explicitly stated.

### Conversion Rules

Sánchez Valencia's (1991a) conversion rule is

$$\begin{array}{l}(some\ y)^{\#}\ is\ a\ x, X \\ (some\ x)^{\#}\ is\ a\ y, X\end{array} \begin{array}{l}\bullet\ Y \\ \bullet\ Y\end{array}$$

Before stating the corresponding rule for ULFs, we note that the rule also works for the determiners `a.d` and `no.d`. Thus we state the ULF conversion rules as the following.

**SLF Rule 4 (Conversion, SCI).**

$$(d\ x\text{: }(x\ P)\ (x\ Q)) \leftrightarrow (d\ y\text{: }(y\ Q)\ (y\ P))$$

where $d \in \{\text{some.d, a.d, no.d}\}$.

Correctness of this rule can be argued using the definitions of the generalized quantifiers `some.d`, `a.d`, and `no.d` and subset relations under interpretation. Using ULFs and chaining this rule with `be.v` inference, we get the following rule.

**ULF Rule 3 (Conversion, UCI).**

$$((d1\ P)\ (\text{be.v}\ (=\ (d2\ Q)))) \leftrightarrow ((d1\ Q)\ (\text{be.v}\ (=\ (d2\ P))))$$

or as a deduction rule

$$\frac{((d1\ P)\ (\text{be.v}\ (=\ (d2\ Q))))}{((d1\ Q)\ (\text{be.v}\ (=\ (d2\ P))))}$$

where $d1 \in \{\text{some.d, a.d, no.d}\}$ and $d2 \in \{\text{some.d, a.d}\}$.

### 5.2.7 Boolean Connectives

Sánchez Valencia (1991a) handles generalized boolean connectives by allowing connectives (*and*, *or*) to have the type $(a, (a, a))$, where $a$ is any complex category ending in $t$. Then the inference rules appropriately substitute one of the connective constituents for the entire phrase. The rules, which we will not list here, have a few versions depending on the position of the connective due to the left side of the tableau nodes being interpreted as connected with conjunctions and the right side with disjunctions.

ULF handles this similarly, but without the tableau-specific details, by allowing connectives to have the type $\langle A, \langle A, A\rangle\rangle$ for an arbitrary type $A$. They are interpreted as generalized lambda expressions.

**Definition 5.2.4** (ULF Generalized Connective)**.**

$$(A\ \chi\ B) \leftrightarrow (\lambda\ x_1, ..., x_n\text{: } ((A\ x_1 ... x_n)\ \chi\ (B\ x_1 ... x_n)))$$

where $\chi \in \{\text{and.cc, or.cc}\}$ and both $A$ and $B$ are prefix operators with arity $n$. Infix operators are defined in the equivalent way while respecting the predicate position relative to the arguments.

This, along with the observation that the following formulas hold true through the intersective and unionistic nature of conjunction and disjunction, respectively, allow us to use simple monotonicity rules in the context of boolean connectives.

$$(\lambda\ x_1, ..., x_n\text{: } ((A\ x_1 ... x_n)\ \text{and.cc}\ (B\ x_1 ... x_n))) \subseteq A, B$$

$$A, B \subseteq (\lambda\ x_1, ..., x_n\text{: } ((A\ x_1 ... x_n)\ \text{or.cc}\ (B\ x_1 ... x_n)))$$

### 5.2.8 EL Rule Instantiation & Monotonicity

It turns out that the monotonicity rules (ULF Rule 1, UMI) are special cases of EL inference rules, which operate on substitution under arbitrarily nested polarity contexts (Schubert and Hwang, 2000).

EL supports two forward inference rules and two goal-based inference rules. First, the forward inference rules, called Rule Instantiation (RI):

$$\frac{MAJ(\phi^{-}), MIN(\phi'^{+})}{MAJ_{\sigma}(\neg MIN_{\sigma}(\bot^{+})^{-})}\ \text{RI-1} \qquad \frac{MAJ(\phi^{-}), MIN(\phi'^{+})}{MIN_{\sigma}(MAJ_{\sigma}(\top^{-})^{+})}\ \text{RI-2}$$

where RI-1 is sound if the only variables in the matching expression, $\phi'$, of the minor premise, $MIN$, are "matchably bound"—bound within $\phi'$ or by a universal quantifier in positive polarity context or existential quantifier in negative polarity context—and RI-2 is sound if the only variables in the matching expression, $\phi$, of the major premise, $MAJ$, are "matchably bound."

Here is an example to demonstrate monotonicity inference over SLFs which for this inference are sufficiently disambiguated. This example is a special case RI-2.

**Proof**

| | |
|---|---|
| 1. *Every carp is a fish* | Assumption |
| 2. *Abelard sees a carp* | Assumption |
| 3. (every.d $x$: ($x$ carp.n) (a.d $y$: ($y$ fish.n) (be.v (= y)))) | SLF for 1. |
| 4. (a.d $x$: ($y$ carp.n) (\|Abelard\| (see.v $y$))) | SLF for 2. |
| 5. (every.d $x$: ($x$ carp.n) ($x$ fish.n)) | be.v Elim, 3. |
| 6. (every.d $x$: ($x$ carp.n)$^{-}$ ($x$ fish.n)$^{+}$) | Polarity marking, 5. |
| 7. (a.d $y$: ($y$ carp.n)$^{+}$ (\|Abelard\| (see.v $y$))$^{+}$) | Polarity marking, 4. |
| 8. (a.d $y$: ($y$ fish.n)$^{+}$ (\|Abelard\| (see.v $y$))$^{+}$) | RI-2, 6.,7. (see 5.2.8) |
| 9. *Abelard sees a fish* | English for 8. |

**Step-by-step RI-2 application**

| | |
|---|---|
| 1. (every.d $x$: ($x$ carp.n)$^{-}$ ($x$ fish.n)$^{+}$) | $MAJ$ |
| 2. (a.d $y$: ($y$ carp.n)$^{+}$ (\|Abelard\| (see.v $y$))$^{+}$) | $MIN$ |
| 3. $\top \rightarrow$ ($y$ fish.n)$^{+}$ | Converted $MAJ$, $\{x/y\}$, 1. |
| 4. $\bot \vee$ ($y$ fish.n)$^{+}$ | $\rightarrow$ Def, 3. |
| 5. ($y$ fish.n)$^{+}$ | $\bot$ Elim, 4. |

6. (a.d $y$: ($y$ fish.n)$^+$ (|Abelard| (see.v $y$))$^+$) — Subst. of converted $MAJ$ , 2.,5.

Notice that this proof holds for an arbitrary predicates in place of *carp* and *fish* and an arbitrary sentence where *carp* occurs in positive polarity context in place of *Abelard sees a carp*. Thus, RI-2 is a generalization of Sánchez's monotonicity rule.

$$\begin{array}{r} (every\ x)^{\#}\ is\ a\ y, F(x^+), X \\ (every\ x)^{\#}\ is\ a\ y, F(y), X \end{array} \Big\} \begin{array}{l} Y \\ Y \end{array}$$

Note that RI-2 can also handle inferences where the major premise is a more complex construction than "every $p$ is a $q$". In episodic logic, RI-2 can be used to conclude *Something is a cap or pretty if Little Red Riding Hood wears it* from *Every dress or hood that Little Red Riding Hood wears is pretty* and *Something is a cap or a hood* (Schubert and Hwang, 2000).

Additionally, RI-1 is a generalization of the reverse inference: substituting in more specific predicates when in negative polarity.

**Step-by-step RI-1 application**

1. (every.d $x$: ($x$ carp.n)$^-$ ($x$ fish.n)$^+$) — $MIN$
2. (no.d $y$: ($y$ fish.n)$^-$ (|Abelard| (see.v $y$))$^-$) — $MAJ$
3. (no.d $x$: ($x$ fish.n)$^-$ (|Abelard| (see.v $x$))$^-$) — Converted $MAJ$, $\{y/x\}$, 2.
4. $\neg((x$ carp.n)$^- \rightarrow \bot^+)$ — Converted $MIN$ , 1.
5. $\neg(\neg(x$ carp.n)$^- \vee \bot^+)$ — $\rightarrow$ Def, 4.
6. $\neg\neg(x$ carp.n)$^- \wedge \top^+$ — de Morgan, 5.
7. ($x$ carp.n)$^- \wedge \top^+$ — $\neg$ Elim, 6.
8. ($x$ carp.n)$^-$ — $\top$ Annih, 7.
9. (no.d $x$: ($x$ carp.n)$^-$ (|Abelard| (see.v $x$))$^-$) — Subst. of converted $MIN$ , 3.,8.

Now that we have seen that RI subsumes the specialized monotonicity inference presented by Sánchez Valencia (1991a), we will show that in first order contexts RI also subsumes a more general

presentation of natural logic inference by Sánchez Valencia (1991b). The upward monotonicity inference in positive contexts is written by Sánchez Valencia as

$$\frac{[\![M]\!] \leq [\![M']\!] \quad [\![N(M)]\!]}{[\![N(M')]\!]}$$

where $[\![]\!]$ is the denotation function and $\leq$ is the monotonicity ordering from Definition 5.2.1. We will refer to this rule as SVMI. First, we show that the RI-2 inference in first order contexts can be interpreted in this form.

Since RI-2 substitutes $MAJ_\sigma(\top)$ for $\phi'^+$ in $MIN(\phi'^+)$, if we can show that $[\![\phi']\!] \leq [\![MAJ_\sigma(\top)]\!]$, then RI-2 can be justified through SVMI. As $MAJ(\phi)$ is assumed in RI-2, if $\phi = \top$, then for all models satisfying the assumptions, $MAJ(\top) = \top$.[11] Basically, $MAJ(\phi), (\phi = \top) \rightarrow MAJ(\top)$. This by definition of $\leq$ (5.2.1) satisfies $[\![\phi]\!] \leq [\![MAJ(\top)]\!]$: in any case where $\phi$ is true, so is $MAJ(\top)$. Since $\phi$ and $\phi'$ are matchably bound, their differences are irrelevant in the above justification and can be substituted for each other. Thus, for any application of RI-2 $[\![\phi']\!] \leq [\![MAJ_\sigma(\top)]\!]$ holds, and therefore the inference can be justified through SVMI.

Now, we show that for any SVMI inference where $M$ and $M'$ are wffs, it can be written in the form of RI-2. $[\![M]\!] \leq [\![M']\!]$ can be restated as $(\forall_x(M \rightarrow M'))$, which we identify as $MAJ(\phi^-)$, where $M$ is $\phi$ and we know that $M$ is in a negative polarity context due to $\forall$. $N(M)$ is identified as $MIN(\phi'^+)$ where $M$ is $\phi'$ and we know is in a positive polarity context by assumption. $MAJ(\top) = \top \rightarrow M' = M'$ so $MIN(MAJ(\top)) = N(M')$.

I conjecture that this generalizes to all $M$ and $M'$ that are not wffs. The monadic predicate case seems simple enough through a connection with modus ponens, but proofs for cases such as determiners and variables are more elusive.

[11] This step requires the first-order context. In intensional contexts, the substitution via equality is not justified.

### 5.2.9 Conclusion

This section presented a proof-based formalism for making natural logic inferences from ULFs with in-proof scoping assumption declarations using less ambiguous, scoped LFs. We showed that this formalism enables all of the core natural logic inferences described by Sánchez Valencia (1991a) as well as inferences in arbitrarily nested polarity contexts through the use of Rule Instantiation from the EL inference toolkit. This inferential capacity of ULF, in conjunction with its ease of parsing, positions ULF as a promising representational basis for automatically generating natural logic inferences.

This symbolic system for inference easily integrates with well-researched machine learning paradigms, e.g., semantic parsing, sequence labeling (polarities), word sense disambiguation, etc., and the proof-based search process allows the systematic backtracking of assumptions so that we are not reliant on the machine learning systems always generating the correct labels as the top result. Additionally, the explicit tracking of labels and assumptions makes the inference result easily interpretable. The next section describes an implementation of this inference system to demonstrate the efficacy of this approach empirically.

## 5.3 An Implementation of Natural Logic Inference with ULFs

Now in this section we present an implementation of a proof-based inference system using ULFs as a basis for making natural logic-like inferences. Our system achieves a high precision on monotonicity problems using a small number of sound inference rules on a variety of inference cases. We thereby experimentally demonstrate that ULF is in fact capable of handling the five kinds of inference outlined at the start of this chapter.

Figure 5.12 is an simple inference example from the FraCaS dataset—the actual output of our system—which demonstrates a simple use of the UMI inference rule.[12] This example also

[12]Irrelevant polarity marking symbols are omitted for brevity and clarity.

shows some differences between our system and the theoretical method presented in the last section (Section 5.2). Namely, our UMI rules generalize to variants of *every A is a B* (in this case *all As are Bs*), our initial polarity marking method circumvents the need for SLFs (Section 5.3.2), and we haves rules to generate monotonicity relations from intersective predicate modification (Section 5.3.2).

**Inference Example (FraCaS Problem 24)**

1. `((many.d (plur delegate.n))`
   `((past obtain.v)`
   `(k (interesting.a (plur result.n)))`
   `(adv-a (from.p (the.d survey.n)))))` — Assumption
2. `((all.d (interesting.a (plur result.n)))`
   `((pres be.v) (= (k (plur result.n)))))` — Intersective modifier relation, 1.
3. `((many.d (plur delegate.n)`■`)`
   `((past obtain.v)`
   `(k (interesting.a (plur result.n))`▲`)`
   `(adv-a (from.p (the.d survey.n)))))` — Polarity marking 1.
4. `((many.d (plur delegate.n))`
   `((past obtain.v) (k (plur result.n))`
   `(adv-a (from.p (the.d survey.n)))))` — UMI 2.,3.

Figure 5.12: Inference example from the FraCaS dataset.

### 5.3.1 Background

Building computational approaches to natural logic inference—distinct from general natural language inference—is an active area of research (Angeli and Manning, 2014; Tian et al., 2014; Mineshima et al., 2015; Abzianidze, 2016; Hu et al., 2019; Haruta et al., 2020). In order to evaluate our monotonicity-specific inference system fairly, we focus on the FraCaS dataset (Cooper et al., 1996) which carefully presents monotonicity-based entailments for evaluation and aim to show competence on monotonicity, rather than state-of-the-art (SOTA) performance. In our experiments (Section 5.3.4) we compare against a few notable systems that were previously evaluated on the same parts of the FraCaS dataset.

Mineshima et al. (2015) and Abzianidze (2016) extend first-order lambda logical forms with higher-order terms (e.g most, many, half of, etc.) and augment first-order inference with rules geared towards those terms. Haruta et al. (2020) achieve SOTA performance by employing degree and event semantics to approximate key higher-order logic features presented in different linguistic phenomena. Hu et al. (2019) differs from the others by running directly on the natural language text, with a combinatory categorical grammar (CCG)-based monotonicity labeling system.

Our approach most resembles Hu et al.'s (2019) system because our logical form closely matches the form and expressiveness of natural language, which enables monotonic reasoning using a relatively compact set of inference rules and we also use an auxiliary representation to obtain monotonicity labels. Our goal is not to achieve the SOTA performance on this dataset; rather the SOTA system results are provided to contextualize our system's performance with respect to the wider research efforts on this dataset.

### 5.3.2 System Description

Our inference system starts with a set of premise sentences and a hypothesis sentence in English which are automatically converted to ULF and then used to determine an *entailment*, *contradiction*, or *unknown* relationship between the premises and the hypothesis through a forward inference search from the premises. The inference process is modeled after the theoretical framework described in Section 5.2 which uses SLFs for identifying the polarity operator scopes and computing the global polarity context of each sub-expression. These polarities are then mapped back to the corresponding ULFs which are used as the basis for the inference rules.

We simplify the theoretical framework in two ways. First, we do not include the scoping possibilities in the proof process. That is, we compute a single SLF for each ULF by cross-referencing the effect of known negative polarity operators with an external word-level polarity marking system and assume that the one SLF is the correct scoping. Second, we introduce variations of the monotonicity and conversion inference rules that correspond to specific ULF macros and syntax. These reduce two steps of inference (expanding the macro and then applying the inference

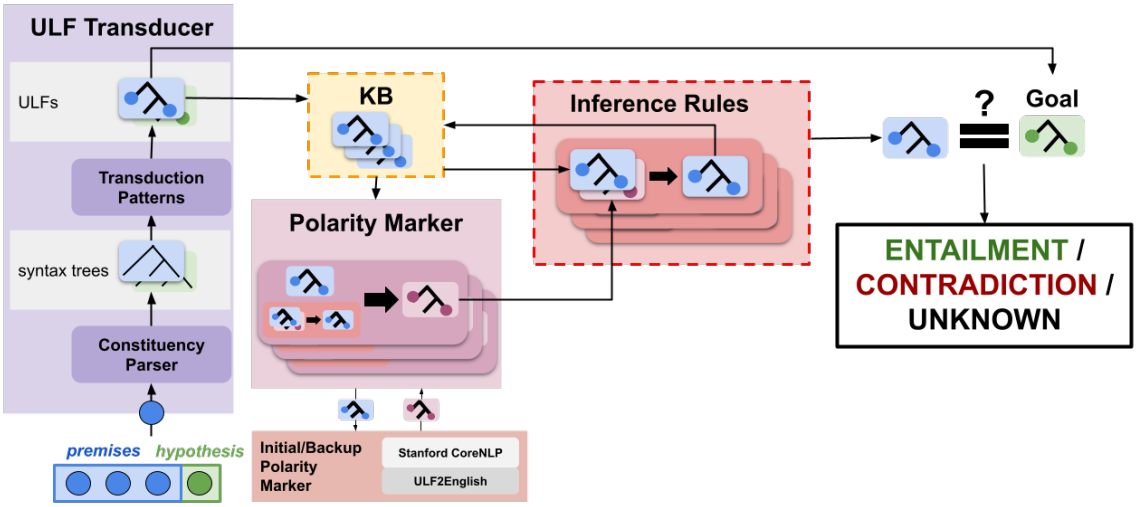



Figure 5.13: A diagram of the inference system component dependencies.

rule) to a single step. Both of these simplifications are introduced to reduce the search space and speed up the inference process.

Here we describe an example of the second simplification. We directly extract the monotonicity relation from the complex expression containing the nominal predicate with its premodifiers and postmodifiers (i.e., all but the determiner or kind-forming operator of the term derived from a noun phrase). In postnominal modification, the operands of the `n+preds` macro—combining a nominal predicate with postmodifying predicates—are used directly for inference, without expansion of the macro construct to a conjunctive lambda predicate. For example, for a postmodified noun, as in the phrase *a dog in the park*, we directly extract the entailments *every dog in the park is a dog* and *every dog in the park is in the park*, rather than first converting the phrase to *something that is a dog and that is in the park* and indirectly computing the rules via explicit predicate intersection. For an intersective prenominal modifier, as in the phrase *a happy dog*, we directly extract the entailments *every happy dog is a dog* and *every happy dog is happy*, rather than first converting the phrase to *something that is happy and that is a dog* first.

The inference system has the following high-level components:

- a heuristic-based inference search function,
- a constituency parse to ULF tree transducer,
- a global polarity marking function,
- inference rules with polarity propagators, and
- external knowledge resources.

Figure 5.13 shows a diagram of the component dependencies. While most of the inference system is symbolic, the initial constituency parses and initial polarity marking—used for ULF transduction and scope selection, respectively—are computed using neural network and statistical methods. Furthermore, the machine learning-based polarity marking is used when the symbolic polarity propagation methods fail or take too long.

#### Search Process

Our inference process is guided by a simple heuristic forward search. Algorithm 1 describes this process in detail. In order to retain completeness while using fast and naive heuristic functions, the search process alternates between heuristic guided search and breadth-first search every several steps. This is a generic search process, where $h$ is a heuristic function which estimates the distance from some formula, $x$, to the goal formula, $\epsilon$ is a small positive number which is used to gives preference formulas reached earlier in the search process in cases of ties, and $c$ is the number of search steps in a row that the search process uses heuristic search before switching to BFS and vice-versa. Section 5.3.3 specifies the values of these parameters that we use in our experiments.

#### ULF Transducer

The ULF transducer converts constituency parses into ULFs with a series of simple correspondences from the phrase structure and POS tags to ULF expressions. This technique is the same as those

**Algorithm 1** Heuristic search. Inference rules map a set ULF premises to a set of ULF inferences.

**Inputs:** $\Phi$, a set of premises; $\psi$, a goal ULF; $h$, a heuristic function; $M$, a search depth limit.
**Outputs:** The entailment classification.
**Global Constants:** $U$, a list of unary rules; $B$, a list of binary rules; $\varepsilon$, a small positive number; $c$, a step count for search method change.
**Procedure:**
Initialize $n \leftarrow 0$, $\mathrm{KB} \leftarrow \Phi$, $Q_{\mathrm{h}} \leftarrow$ empty priority queue, $Q_{\mathrm{bfs}} \leftarrow$ empty basic queue, $Q \leftarrow Q_{\mathrm{h}}$, and $Q_{\mathrm{other}} \leftarrow Q_{\mathrm{bfs}}$.
**loop**
  If $n > M$ or $Q = \varnothing$, **return** UNKNOWN.
  If $\psi \in \mathrm{KB}$, **return** ENTAILMENT.
  If $\neg\psi \in \mathrm{KB}$, **return** CONTRADICTION.
  $\nu \leftarrow Q.\mathrm{pop}()$.
  $t_{\mathrm{unary}} \leftarrow U \times \nu$ and $t_{\mathrm{binary}} \leftarrow B \times ((\nu \times \nu) \cup (\nu \times \mathrm{KB}) \cup (\mathrm{KB} \times \nu))$.
  Push all results $x$ of computing the tuples in $t_{\mathrm{unary}}$ and $t_{\mathrm{binary}}$ that are not contained in KB to $Q_{\mathrm{h}}$ with key $h(x) + n\varepsilon$ and $Q_{\mathrm{bfs}}$.
  $\mathrm{KB} \leftarrow \mathrm{KB} \cup \nu$ and $n \leftarrow n + 1$.
  **if** $n \mod c = 0$ **then**
    $\mathrm{tmp} \leftarrow Q$.
    $Q \leftarrow Q_{\mathrm{other}}$.
    $Q_{\mathrm{other}} \leftarrow \mathrm{tmp}$.
  **end if**
**end loop**

used in the initial stages of prior transduction-based EL parsers (Schubert, 2002; Schubert and Tong, 2003; Gordon and Schubert, 2010; Schubert, 2014), but modified for the modern ULF specification (Chapter 2). Some transduction rules add type assumptions that are not necessarily true, but are unlikely to affect the monotonicity inferences. For example, ULF makes a semantic distinction between event modifiers (e.g. *today*) and proposition modifiers (e.g. *surprisingly*) which is not relevant for monotonicity inferences. If the parser fails to eliminate one of these options, it assumes that it is an event modifier.[13]

We use the Berkeley neural parser (Kitaev and Klein, 2018) to get the constituency trees.[14]

[13]The transduction rules are written in a combination of the tree-to-tree transduction language (Purtee and Schubert, 2012) and a simplified variant.

[14]The version 0.2.0 release and the `benepar_en3` model available at https://github.com/nikitakit/self-attentive-parser/.

We opt not to use the transition parser from Chapter 4 because sentences in the FraCaS dataset tend to be fairly short and follow written English syntax. The transition parser is more robust to language length and variety, but in our development datasets we found a symbolic transduction to be more reliable. Additionally, our symbolic transductions have more predictable and regular errors. This allows monotonicity inferences to succeed even with minor errors. The predictability allows us to write post-processing rules that “fix" the parse by adding additional assumptions that we know won’t usually affect monotonicity inferences (e.g. turning all unknown verb modifiers to action modifiers—as opposed to event modifiers). The regularity of errors means that even if the premise and hypothesis parses are erroneous the inferences may succeed if the error is not specific to triggering the necessary inference (e.g. preposition is attached at the wrong constituent or the incorrect modifier type is used).

**Polarity Marking**

We delegate the initial polarity marking problem to a component of the Natlog and NaturalLi systems (MacCartney and Manning, 2008; Angeli and Manning, 2014) which runs over raw English text.[15] We then align the polarities of each token to the corresponding ULF sub-expression. Rather than using the actual English premises and hypothesis we use the output of the ULF-to-English system (Section 5.1.11) so that we can use its subroutines to assist in subexpression alignment.

This alignment is then used to select the scoping by finding the possible SLF that minimizes the number of polarity discrepancies between the NatLog polarity labels and the labels inferred from the scoping and a manually curated list of negative polarity operators. The inference rules propagate the polarities so this is only performed on the input sentences (Section 5.3.2). During the inference process, this polarity marking is reserved as a fallback in cases where polarity propagation via inference rules fails or takes too long.

Possible SLFs are computed by generating every possible scope configuration while accounting for island constraints. We roughly model scope island constraints with the following rule: *Scoping*

[15]This is available through the Natural Logic component of Stanford CoreNLP.

*operators cannot scope outside of ancestors that are ULF type-shifters*. This rule handles complex modifiers (which are shifted from predicates to modifiers) and reified clausal complements (e.g., *I believe that everyone thinks.*) and is implemented trivially with the ULF type system. This is an approximation of the full range of actual island constraints, which come in various classes and with nuances that are still under active investigation in linguistics research. Our rule tends to be stricter than actual scope island constraints leading to some losses in expressive capacity, such as exceptions to commonly accepted island constraints (Barker, 2021) and the *de dicto* / *de re* distinction for clausally-embedded indefinite quantifiers (Donnellan, 1966; Burge, 1977).[16] However, this is only a limitation of our implementation of scoping and polarity propagation. A more nuanced treatment of available scopings can be accommodated by the underlying theoretical inference framework as discussed in Section 5.2.

**Inference Rules**

All of our inference rules fall under one of four categories.

1. *Monotonicity Substitution*

   This is the core monotonicity inference. Given the premise *Every A is a B*, *B* is substituted for *A* in positive polarity contexts and *A* is substituted for *B* in negative polarity contexts. In order to reduce the proof lengths, we suppress ULF macro expansion rules and extract monotonicity relations directly from macro instances.

2. *Conversion*

   *Some A is a B* $\Leftrightarrow$ *Some B is an A*

3. *Conservativity*

[16] For example, the referential reading of *someone* in the sentence *I know that someone lied* is not available if the indefinite quantifier is not allowed to take wide scope over the sentence.

*δ As are Bs* ⇔ *δ As are As that/who are Bs*, where *δ* is a determiner. This is a category of inferences in the FraCaS dataset and a commonly used inference step for introducing and eliminating relative clauses in simple quantified expressions.

4. *Equivalences*

   This includes equivalent determiner substitutions (e.g., *Every dog is happy* ⇔ *All dogs are happy*) and predicate synonym substitutions (e.g., *I saw the accident* ⇔ *I witnessed the accident*).

We have 9 total inference rules when accounting for specializations for macros—though some of these inference rules themselves include several distinct transduction patterns to account for minor syntactic variations.

In order to identify whether a modification is intersective, we use the non-subsective adjective list by Nayak et al. (2014) expanded to words in the WordNet (Miller, 1995) synsets.

**Polarity Propagation.** For computational efficiency, each inference rule has a corresponding polarity propagation function. The polarity propagation function takes the premise ULF formulas, their polarity markings, and the conclusion and computes the polarity marking of the conclusion.

As a concrete example, consider the polarity propagation function for the UMI inference rule with the premises and conclusions based on FraCaS problem 24 (Figure 5.12) from the beginning of this section. The premises are steps 1 (*many delegates obtained interesting results from the survey*) and 2 (*all interesting results are results*) in the inference example and the conclusion is step 4 (*many delegates obtained results from the survey*). The polarity marking[17] for step 1 is step 3 of the proof and the polarity marking for step 2 is

```
((all.d (interesting.a (plur result.n)))▼
 ((pres be.v) (= (k (plur result.n)▲)))).
```

[17] Omitting irrelevant polarities.

The propagation function identifies that (plur result.n)▲ is the polarized version of the subexpression that substituted for (interesting.a (plur result.n)) in the step 1 premise. Thus, most polarity markings are transferred over from the step 1 polarity marking except the marking for (plur result.n) in the substituted subexpression. This leads to the following polarity marking of the conclusion.[18]

```
((many.d (plur delegate.n)■) ((past obtain.v) (k (plur result.n)▲)
                                (adv-a (from.p (the.d survey.n)))))
```

Most of these propagation functions can be implemented efficiently without accessing the corresponding SLFs because the inference context eliminates the possibility of polarity operators interacting outside of the localized expression substitution due to scope island constraints (Fodor and Sag, 1982; Park, 1995; Ruys and Winter, 2011; Barker, 2015). For example, the conversion rule substitutes two nominal predicates for each other in sentences with the main verb *be*, an indefinitely quantified subject, and a nominal subject complement. In this case, any quantifier embedded within either nominal predicate is constrained by the Complex NP Constraint (Ross, 1967).

A notable exception is the monotonicity substitution of determiners: the polarity propagation function must have access to the SLFs and cannot be implemented as efficiently because the new determiner may induce different polarities in its restrictor and body than the replaced determiner.[19] Properly computing the global polarity from this requires access to the quantifier scopes.

### 5.3.3 Experimental Setup

In our experiments we allow a maximum of 50 inference steps and use a leaf label F1 heuristic (LL-F1) which alternates with breadth-first search (BFS) every 5 inference steps. LL-F1 computes the F1 score between the leaf labels of the new formula and the goal formula, ignoring order, but preserving repetitions. This is turned into a cost ranging 0-to-1 by subtracting it from 1.

[18] Again, omitting irrelevant polarities.

[19] For example, in positive contexts, *the* may be replaced with *a*, as in, *I saw the dog* ⇒ *I saw a dog*. *The* imposes a flat entailment context on its restrictor whereas *a* imposes a positive entailment context which warrants a fresh computation of the global polarity markings.

The FraCaS dataset is a set of entailment questions related to specific semantic phenomena that were curated by semanticists (Cooper et al., 1996). It contains 346 problems, of which 12 do not have well-defined answers. We focus on the most relevant section of the FraCaS dataset, section 1: Generalized Quantifiers (GQs). This is also the largest section, making up almost a quarter of the dataset.

Due to the small size of the FraCaS dataset and the challenging phenomena it contains, prior research has trained and tested models on the same problems, focusing on the capacity of their systems to perform such inferences, rather than their competence in learning and generalizing to a larger scale. This aligns nicely with our goal to demonstrate the capacity to use ULFs as the basis for monotonic inferences, rather than present a system to compete with the state-of-the-art on entailment tasks.

### 5.3.4 Results

Our experiments show that our system is able to precisely cover a variety of semantic phenomena and constructions, but, as expected from a demonstration system, does not achieve the robustness of SOTA entailment systems.

Table 5.3 shows the confusion matrix of our system on the FraCaS dataset. Our system shows very high precision (it is never incorrect when it makes a definitive conclusion—not UNK) because of the soundness of our inference rules. While our system fails to correctly identify any contradictions, this is not an inherent limitation of the system. It was simply the case that parser errors led to the inability to match the inferred negated formula with the hypothesis in the 5 problems that have contradiction labels.

Table 5.3: Confusion matrix on the FraCaS dataset.

| Gold\Pred. | ENT | CON | UNK |
|---|---|---|---|
| ENT | **22** | 0 | 15 |
| CON | 0 | **0** | 5 |
| UNK | 0 | 0 | **32** |

In Table 5.4, the accuracy of our system is compared to the majority class baseline and other natural logic systems that focus on monotonicity and FraCaS inferences. A variety of methods

prove effective at monotonic reasoning over a variety of linguistic phenomena. LP and HR perform notably well and both rely on CCG parses for obtaining the representation and theorem provers for managing inferences. Although our system falls short of the performance of SOTA systems on FraCaS, we still perform noticeably better than the majority class baseline. Investigating the error cases of our system makes clear that the shortfalls of our system are not inherent in the theoretical approach—rather they are due to syntactic and inference cases that were not addressed in this exploratory inference system.

Table 5.4: FraCaS performance of our system (Ours) compared against a majority class (ENT) baseline (BL) and several notable RTE systems: MN (Mineshima et al., 2015), LP (Abzianidze, 2016), HU (Hu et al., 2019), and HR (Haruta et al., 2020). Hu et al. (2019) and Haruta et al. (2020) only report the overall accuracy of their systems.

| Section | Accuracy % | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Single-premise | | | | Multi-premise | | | | Overall | | | | | |
| | BL | Ours | MN | LP | BL | Ours | MN | LP | BL | Ours | MN | LP | HU | HR |
| 1 GQs | 45 | 73 | 82 | 93 | 57 | 67 | 73 | 93 | 50 | 70 | 78 | 93 | 88 | 99 |

The polarity propagation system used the fallback system (the polarity marking component of the Natlog system) in 42 out of the 3,109 (1.3%) total polarity propagation calls made in the GQs section of the FraCaS evaluation.

### Qualitative Analysis

Figure 5.14 shows three distinct success and three distinct failure cases of our system. First looking at the successes, example 18 is a multi-premise entailment problem which requires conservativity inference and multiple UMI applications in both positive and negative contexts. Example 59 has two distinct components—first the determiner *a few* must be generalized to *at least a few*, second *female* must be recognized as an intersective modifier and removed to generalize the nominal predicate in positive polarity context. Example 60 again has the intersective modifier *female*, but must not trigger an inference because of the negative polarity context.

Now taking a look at the failures, example 25 requires the recognition of *in major national newspapers* as an adjunct that may be dropped for a more general meaning. Our system parses the premise incorrectly—specifically "results published in major national newspapers" is parsed as a single kind-of-event[20] argument rather than an argument and an adjunct. This is can be addressed by an improvement of the ULF parser, e.g., an expansion of the verb subcategorization frames known by the ULF transduction rules. Example 48 requires the introduction of the phrase *a lot of* in negative polarity context, since it acts as a specializing modifier. Our system does not recognize *a lot of* as specializing modifier and this sort of multi-word idiosyncratic syntactic construction for a specializing modifier needs to be addressed specifically in the grammar. Finally, example 76 is a reversal of the intersective modifier *female* that was in the successful example 59. Because we use a forward inference framework, the proof-system does not have access to the modifier *female*. This could be handled by extraction of necessary intersective monotonicity rules from the hypothesis or more generally keeping a lexicon of intersective modifiers—though, the latter approach would be less efficient if implemented naively.

### 5.3.5 Conclusion

We have presented a simple implementation of forward monotonic inference starting with English sentences and using ULFs as the representational basis. Our system shows a high degree of precision on a variety of monotonicity phenomena, empirically confirming the final class of inferences discussed at the start of the chapter, alongside a suite of pragmatics-oriented inference capabilities of ULF that were demonstrated in Section 5.1.

The specifics of this demonstration system and the results point to many possible avenues of improvement. Beyond direct improvements to the ULF parser, operator scoping, and inference rules to cover more constructions, the proof-search process can be expanded to explicitly include alternate

[20] A kind-of-event is a type in the domain of discourse in EL semantics corresponding to generic events. For example, in the sentence "The news reporting on a missing kitten was unexpected", *unexpected* is a predicate over the kind-of-event "The news reporting on a missing kitten". This is distinct from similar EL types of events, which are particular instances, and propositions, which are statements that may be true or false.

**Successes**

| ID | Correct inference |
|---|---|
| 18 | *Every European has the right to live in Europe*;<br>*Every European is a person*;<br>*Every* (*person who has the right to live in Europe*)$^{\blacktriangledown}$ *can travel freely within Europe*<br>⇒ *Every European can travel freely within Europe* |
| 59 | (*A few*)$^{\blacktriangle}$ (*female committee members*)$^{\blacktriangle}$ *are from Scandinavia*<br>⇒ *At least a few committee members are from Scandinavia* |
| 60 | *Few* (*female committee members*)$^{\blacktriangledown}$ *are from southern Europe*<br>↛ *Few committee members are from southern Europe* |

**Failures**

| ID | Correct inference |
|---|---|
| 25 | *Several delegates* (*got the results published in major national newspapers*)$^{\blacktriangle}$<br>⇒ *Several delegates got the results published* |
| 48 | *At most ten commissioners spend* (*time*)$^{\blacktriangledown}$ *at home*<br>⇒ *At most ten commissioners spend a lot of time at home* |
| 76 | *Few* (*committee members*)$^{\blacktriangledown}$ *are from southern Europe*<br>⇒ *Few female committee members are from southern Europe* |

Figure 5.14: Several examples of inference successes and failures. The relevant polarity contexts for the final (or last two if not overlapping) inference step is marked with (...)$^{\blacktriangle}$ or (...)$^{\blacktriangledown}$ and the relevant spans are underlined in the premises and hypothesis. Our inference system predicated UNK for each of the failure examples.

parsing and scoping choices thereby enabling proper exploration of ambiguous constructions. For example, each particular English-to-ULF parse and each scoping choice leading to a distinct SLF can be formulated as an inference rule that can be explored during the proof search. The inference rules can also be made more flexible by implementing the RI-1 and RI-2 rules that generalize UMI as discussed in Section 5.2.8. The direct access to syntactic structure from ULF leaves room for a much more sophisticated treatment of linguistic constraints (notably island constraints as discussed in Section 5.3.2) and the logical type structure makes ULF theoretically capable of inferences from disjunctive conclusions; e.g., *Alice has a dog or a cat*, given that *Alice has a furry pet* and *Furry pets are either dogs or cats*. Finally, in the vein of joining machine learning and deep learning approaches

with symbolic approaches, ULF can be reliably translated back into English (Section 5.1.11) so that these other approaches that work over raw English text can be accessed and used in conjunction with the symbolic rules with little to no modification. In fact, the polarity marking component of our system (Section 5.3.2) is precisely an example of such a bridging of methods. A more labor intensive, but structurally informed, approach would be to design graph encoders that translate the structural information in ULFs into vector representations, which can then be integrated into deep learning systems as inputs or intermediate features.

# 6 Conclusion

In this dissertation I presented ULFs, an underspecified, first-step representation of EL, for the purpose of building a human language understanding system. The design of ULF makes it suited for scalable annotation collection and automatic parsing. ULF captures the underlying semantic type structure while maintaining a transparent relationship with the syntax by retaining scope, anaphoric, and word sense ambiguity and introducing operators to model more opaque relationships between the surface form and the underlying meaning such as syntactic movement and post-nominal genitives.

ULF as a semantic representation and as a parsing system currently formally captures the core semantic phenomena of language—the subtle distinctions that need to be made systematically to avoid fundamental confusion about the world—more fully than any prior semantic parsing framework, and as such can serve as a starting point for deeper language understanding and inference than has yet been achieved. This work makes a substantial advance towards this grand goal by gathering a dataset of annotations, training an automatic parser, and building inference systems that demonstrate the breadth of ULF inference capabilities which uses not only the underlying semantic types, but also the syntactic information that is carefully retained in the representation. Further developments of along these fronts have the potential to lead to a unified language understanding system with a precise symbolic representation and unprecedented formalized expressive capacity.

As ULFs use details in the syntax and the ways it relates to meaning to simplify the annotation

and parsing, the presented work can only be used for English. However, we can expect all languages to share fundamental semantic types or means to express such type distinctions (in a way that may not directly reflect the semantic structure) based on the fact that all human languages reflect a combination of the nature of human cognition, descriptive adequacy for the world, and communicative needs of societies. These types would include, at least: truth/falsity, predicates, identity, general quantifiers, sentence and predicate modification, sentence and predicate reification (thus allowing for attitudes and other forms of intensionality), event reference, and comparatives. Though some language may not have direct ways to represent all of these categories—encoded in individual lexical or grammatical mechanisms—we could expect ways more roundabout ways to communicate such ideas just as some languages have single words to describe ideas that take full sentences in others (see the innumerable articles about "X untranslatable words from other languages" that proceed to give wonderfully nuanced translations using multiple sentences). Given such universal semantic types in languages, developing ULFs for other languages will be a matter of finding the appropriate mechanisms for relating the less straight-forward relationships between the particular language's manner of communicating certain ideas to their underlying semantic structure. This will be unlikely to change much of the core ULF representation that is needed. Rather the bulk of the difference will be in developing the right syntactic relaxations and macros to address the pragmatic issues related to data collection and computability similar to our process described in Section 3.1.

The development of a robust ULF parser and general inference systems open up fresh opportunities for research that move beyond the goals of this dissertation. Robust automated systems open up a new horizon in ULF and EL work: expanding the scope of systems that have so far relied on manual annotations and thus been limited to preliminary investigation. Future work can be scaled to datasets that overcome the cognitive limitations of individual human analysis and collect ULF data from a more diverse set of sources.

The use of ULF for downstream applications is already underway. Even with the still limited parsing capabilities, the flexible nature of ULFs have led to its use in both dialogue systems and schema representations. ULF is an integral part of a history-aware spatial reasoning system

because of the ease of its conversion to English and because its inferences enable natural question answering (Kane et al., 2020; Platonov et al., 2020). ULFs with selective ambiguity resolution are used as the internal meaning representation of a general schema-learning framework, which requires the nuanced semantic distinctions of ULF and EL for purpose of building a general, analogical reasoning system that reflects human capacity to learn from stories (Lawley et al., 2019).

## References

Steven P Abney and Mark Johnson. 1991. Memory requirements and local ambiguities of parsing strategies. *Journal of Psycholinguistic Research*, 20(3):233–250.

Lasha Abzianidze. 2016. Natural solution to FraCaS entailment problems. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 64–74, Berlin, Germany. Association for Computational Linguistics.

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The Parallel Meaning Bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 242–247, Valencia, Spain. Association for Computational Linguistics.

James Allen, Will de Beaumont, Lucian Galescu, Jansen Orfan, Mary Swift, and Choh Man Teng. 2013. Automatically deriving event ontologies for a CommonSense knowledge base. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 23–34, Potsdam, Germany. Association for Computational Linguistics.

James Allen, Will de Beaumont, Lucian Galescu, and Choh Man Teng. 2015. Complex event extraction using DRUM. In *Proceedings of BioNLP 15*, pages 1–11, Beijing, China. Association for Computational Linguistics.

James Allen, William de Beaumont, Nate Blaylock, George Ferguson, Jansen Orfan, and Mary Swift. 2011. Acquiring commonsense knowledge for a cognitive agent. In *Proceedings of the AAAI Fall Symposium Series: Advances in Cognitive Systems (ACS 2011)*, Arlington, VA, USA.

James Allen, Myroslava Dzikovska, Mehdi Manshadi, and Mary Swift. 2007. Deep linguistic processing for spoken dialogue systems. In *ACL 2007 Workshop on Deep Linguistic Processing*, pages 49–56, Prague, Czech Republic. Association for Computational Linguistics.

James Allen and Choh Man Teng. 2017. Broad coverage, domain-generic deep semantic parsing. In *AAAI Spring Symposium Series*.

James F. Allen, Omid Bahkshandeh, William de Beaumont, Lucian Galescu, and Choh Man Teng. 2018. Effective broad-coverage deep parsing. In *AAAI Conference on Artificial Intelligence*.

James F. Allen, Mary Swift, and Will de Beaumont. 2008. Deep semantic analysis of text. In *Semantics in Text Processing. STEP 2008 Conference Proceedings*, pages 343–354. College Publications.

Hiyan Alshawi. 1989. Analysing the dictionary definitions. In Bran Boguraev and Ted Briscoe, editors, *Computational Lexicography for Natural Language Processing*, pages 153–169. Longman Publishing Group, White Plains, NY, USA.

Gabor Angeli and Christopher D. Manning. 2014. NaturalLI: Natural logic inference for common sense reasoning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 534–545, Doha, Qatar. Association for Computational Linguistics.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, Lisbon, Portugal. Association for Computational Linguistics.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62.

Xuefeng Bai, Linfeng Song, and Yue Zhang. 2020. Online back-parsing for AMR-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1206–1219, Online. Association for Computational Linguistics.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*.

Timothy Baldwin, Emily M. Bender, Dan Flickinger, Ara Kim, and Stephan Oepen. 2004. Road-testing the English Resource Grammar over the British National Corpus. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 2047–2050.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2019. Abstract meaning representation (AMR) 1.2.6 specifications. https://github.com/amrisi/amr-guidelines/blob/master/amr.md. Accessed: 2021-10-06.

Chris Barker. 2015. Scope. In Shalom Lappin and Chris Fox, editors, *Handbook of Contemporary Semantics*, 2 edition, chapter 2, pages 40–76. Wiley Blackwell.

Chris Barker. 2021. Rethinking scope islands. *Linguistic Inquiry*, pages 1–55.

Jon Barwise and Robin Cooper. 1981. Generalized quantifiers and natural language. In *Philosophy, language, and artificial intelligence*, pages 241–301. Springer.

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC 2012)*, pages 3196–3200, Istanbul, Turkey.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 273–283, Melbourne, Australia. Association for Computational Linguistics.

Emily M. Bender, Dan Flickinger, Stephan Oepen, Woodley Packard, and Ann Copestake. 2015. Layers of interpretation: On grammar and compositionality. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 239–249, London, UK. Association for Computational Linguistics.

Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics.

Evert Willem Beth. 1955. Semantic entailment and formal derivability. In *Proceedings of the Section of Sciences*, 18, page 309–342. Koninklijke Nederlandse Akademie van Wentenschappen.

Johan Bos. 1996. Predicate logic unplugged. In *Proceedings of the 10th Amsterdam Colloquium*, pages 133–143.

Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In *Semantics in Text Processing. STEP 2008 Conference Proceedings*, pages 277–286. College Publications.

Johan Bos. 2016. Squib: Expressive power of Abstract Meaning Representations. *Computational Linguistics*, 42(3):527–535.

Johan Bos, Valerio Basile, Kilian Evang, Noortje Venhuizen, and Johannes Bjerva. 2017. The Groningen Meaning Bank. In Nancy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*, volume 2, pages 463–496. Springer.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

T. Bub, W. Wahlster, and A. Waibel. 1997. 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, pages 71–74 vol.1.

Tyler Burge. 1977. Belief de re. *The Journal of Philosophy*, 74(6):338–362.

Lou Burnard. 2000. User reference guide for the British National Corpus. Technical report, Oxford University Computing Services.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1215–1226, Vancouver, Canada. Association for Computational Linguistics.

Deng Cai and Wai Lam. 2020. AMR parsing via graph-sequence iterative inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1290–1301, Online. Association for Computational Linguistics.

Shu Cai and Kevin Knight. 2013. 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)*, pages 748–752, Sofia, Bulgaria. Association for Computational Linguistics.

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

Nicoletta Calzolari. 1984. Detecting patterns in a lexical data base. In *10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics*, pages 170–173, Stanford, California, USA. Association for Computational Linguistics.

G.N. Carlson. 1980. *Reference to kinds in English*. Ph.D. thesis, University of Massachussetts, Amherst, New York.

J. Carroll and C. Grover. 1989. The derivation of a large computational lexicon of english from LDOCE. In Boguraev B. and Briscoe E., editors, *Computational Lexicography for Natural Language Processing*, pages 117–134. Longman, Harlow, UK.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: Mining the web for fine-grained semantic verb relations. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 33–40, Barcelona, Spain. Association for Computational Linguistics.

Martin S. Chodorow, Roy J. Byrd, and George E. Heidorn. 1985. Extracting semantic hierarchies from a large on-line dictionary. In *23rd Annual Meeting of the Association for Computational Linguistics*, pages 299–304, Chicago, Illinois, USA. Association for Computational Linguistics.

Noam Chomsky. 1995. *The Minimalist Program*. MIT Press.

Christos Christodouloupoulos and Mark Steedman. 2015. A massively parallel corpus: the bible in 100 languages. *Language Resources and Evaluation*, 49(2):375–395.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, and Steve Pulman. 1996. Using the framework. Technical Report LRE 62-051 D-16, The FraCaS Consortium.

Ann Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. In *IN PROCEEDINGS OF LREC 2000*, pages 591–600.

Ann Copestake, Dan Flickinger, Rob Malouf, Susanne Riehemann, and Ivan Sag. 1995. Translation using Minimal Recursion Semantics. In *Proceedings of the Sixth Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Katholieke Universiteit, Leuven.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal Recursion Semantics: An introduction. *Research on Language and Computation*, 3(2):281–332.

Ann Copestake, Alex Lascarides, and Dan Flickinger. 2001. An algebra for semantic construction in constraint-based grammars. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 140–147, Toulouse, France. Association for Computational Linguistics.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for Abstract Meaning Representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 536–546, Valencia, Spain. Association for Computational Linguistics.

Donald Davidson. 1967. The logical form of action sentences. In Nicholas Rescher, editor, *The Logic of Decision and Action*. University of Pittsburgh Press.

Donald Davidson. 1969. The individuation of events. In Nicholas Rescher, editor, *Essays in Honor of Carl G. Hempel: A Tribute on the Occasion of his Sixty-Fifth Birthday*, pages 216–234. Springer Netherlands, Dordrecht.

Keith S Donnellan. 1966. Reference and definite descriptions. *The philosophical review*, 75(3):281–304.

Christy Doran, Dania Egedi, Beth Ann Hockey, B. Srinivas, and Martin Zaidel. 1994. XTAG system - a wide coverage grammar for English. In *COLING 1994 Volume 2: The 15th International Conference on Computational Linguistics*.

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

Joe Ellis, Jeremy Getman, Dana Fore, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie Strassel. 2015. Overview of linguistic resources for the TAC KBP 2015 evaluations: Methodologies and results. In *Proceedings of the Eighth Text Analysis Conference (TAC 2015)*.

C.J. Fillmore. 1968. The case for the case. In E. Bach and R. Harms, editors, *Universals in Linguistic Theory*. Rinehart and Winston, New York.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206, San Diego, California. Association for Computational Linguistics.

Dan Flickinger, Yi Zhang, and Valia Kordoni. 2012. DeepBank. a dynamically annotated treebank of the Wall Street Journal. In *Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories*, pages 85–96.

J. Fodor and I. Sag. 1982. Referential and quantificational indefinites. *Linguistics and Philosophy*, 5:355–398.

Aldo Gangemi, Nicola Guarino, Claudio Masolo, Alessandro Oltramari, and Luc Schneider. 2002. Sweetening ontologies with DOLCE. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, EKAW '02, pages 166–181, London, UK. Springer-Verlag.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague. Association for Computational Linguistics.

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 239–246, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Daniel Gildea, Giorgio Satta, and Xiaochang Peng. 2018. Cache transition systems for graph parsing. *Computational Linguistics*, 44(1):85–118.

Jonathan Ginzburg and Ivan A. Sag. 2000. *Interrogative Investigations: the form, meaning, and use of English interrogatives*. CSLI Lecture Notes 123, Stanford, CA.

Jonathan Gordon and Lenhart Schubert. 2010. Quantificational sharpening of commonsense knowledge. In *Proceedings of the AAAI 2010 Fall Symposium on Commonsense Knowledge*.

Venkata Govindarajan, Benjamin Van Durme, and Aaron Steven White. 2019. Decomposing generalization: Models of generic, habitual, and episodic statements. *Transactions of the Association for Computational Linguistics*, 7:501–517.

Jeffrey S. Gruber. 1965. *Studies in Lexical Relations*. Ph.D. thesis, Massachusetts Institute of Technology.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Sanda M. Harabagiu, George A. Miller, and Dan I. Moldovan. 1999. WordNet 2 - a morphologically and semantically enhanced resource. In *SIGLEX99: Standardizing Lexical Resources*.

Izumi Haruta, Koji Mineshima, and Daisuke Bekki. 2020. Combining event semantics and degree semantics for natural language inference. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1758–1764, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ulf Hermjakob. 2013. Amr editor: A tool to build abstract meaning representations. `https://amr.isi.edu/papers/amr-editor-ulf2013a.pdf`. Accessed: 2021-10-07.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1127–1138, Vancouver, Canada. Association for Computational Linguistics.

Jerry R. Hobbs. 1985. Ontological promiscuity. In *23rd Annual Meeting of the Association for Computational Linguistics*, pages 60–69, Chicago, Illinois, USA. Association for Computational Linguistics.

Jerry R. Hobbs. 2008. Deep lexical semantics. In *Computational Linguistics and Intelligent Text Processing, 9th International Conference, CICLing Proceedings*, volume 4919 of *Lecture Notes in Computer Science*, pages 183–193, Haifa, Israel. Springer.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, New York City, USA. Association for Computational Linguistics.

Thomas M. Howard, Stefanie Tellex, and Nicholas Roy. 2014. A natural language planner interface for mobile manipulators. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6652–6659.

Hai Hu, Qi Chen, and Larry Moss. 2019. Natural language inference with monotonicity. In *Proceedings of the 13th International Conference on Computational Semantics - Short Papers*, pages 8–15, Gothenburg, Sweden. Association for Computational Linguistics.

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

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

Chung Hee Hwang. 1992. *A logical approach to narrative understanding*. Ph.D. thesis, University of Alberta.

Chung Hee Hwang and Lenhart 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, pages 238–264, London, UK. Springer-Verlag.

Thomas Icard, Lawrence Moss, and William Tune. 2017. A monotonicity calculus and its completeness. In *Proceedings of the 15th Meeting on the Mathematics of Language*, pages 75–87, London, UK. Association for Computational Linguistics.

Thomas F Icard and Lawrence S Moss. 2014. Recent progress on monotonicity. In *Linguistic Issues in Language Technology*. Citeseer.

Nancy Ide and Jean Véronis. 1993. Knowledge extraction from machine-readable dictionaries: an evaluation. In *Third International EAMT Workshop: Machine Translation and the Lexicon*, pages 19–34, Heidelberg, Germany. Springer Berlin Heidelberg.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Lisa Jin and Daniel Gildea. 2020. Generalized shortest-paths encoders for AMR-to-text generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2004–2013, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Hans Kamp. 1981. A theory of truth and semantic representation. In J. A. G. Groenendijk, T. M. V. Janssen, and M. B. J. Stokhof, editors, *Formal Methods in the Study of Language*, volume 1, pages 277–322. Mathematisch Centrum, Amsterdam.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*, volume 42 of *Studies in Linguistics and Philosophy*. Springer, Dordrecht.

Benjamin Kane, Georgiy Platonov, and Lenhart Schubert. 2020. Registering historical context for question answering in a blocks world dialogue system. In *Text, Speech, and Dialogue*, pages 487–494, Cham. Springer International Publishing.

David Kaplan. 1989. Demonstratives: An essay on the semantics, logic, metaphysics and epistemology of demonstratives and other indexicals. In Joseph Almog, John Perry, and Howard Wettstein, editors, *Themes From Kaplan*, pages 481–563. Oxford University Press.

Rohit J. Kate and Raymond J. Mooney. 2006. 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*, pages 913–920, Sydney, Australia. Association for Computational Linguistics.

Gene Kim, Benjamin Kane, Viet Duong, Muskaan Mendiratta, Graeme McGuire, Sophie Sackstein, Georgiy Platonov, and Lenhart Schubert. 2019. Generating discourse inferences from unscoped episodic logical formulas. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 56–65, Florence, Italy. Association for Computational Linguistics.

Gene Kim and Lenhart Schubert. 2016. High-fidelity lexical axiom construction from verb glosses. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 34–44, Berlin, Germany. Association for Computational Linguistics.

Gene Louis Kim, Viet Duong, Xin Lu, and Lenhart Schubert. 2021a. A transition-based parser for unscoped episodic logical forms. In *Proceedings of the 14th International Conference on Computational Semantics (IWCS)*, pages 184–201, Groningen, The Netherlands (online). Association for Computational Linguistics.

Gene Louis Kim, Mandar Juvekar, Junis Ekmekciu, Viet Duong, and Lenhart Schubert. 2021b. A (mostly) symbolic system for monotonic inference with unscoped episodic logical forms. In *Proceedings of the 1st and 2nd Workshops on Natural Logic Meets Machine Learning (NALOMA)*, pages 71–80, Groningen, the Netherlands (online). Association for Computational Linguistics.

Gene Louis Kim, Mandar Juvekar, and Lenhart Schubert. 2020. Monotonic inference for underspecified episodic logic. In *Proceedings of the Workshop Natural Logic Meets Machine Learning*. Association for Computational Linguistics.

Gene Louis Kim and Lenhart Schubert. 2019. A type-coherent, expressive representation as an initial step to language understanding. In *Proceedings of the 13th International Conference on Computational Semantics - Long Papers*, pages 13–30, Gothenburg, Sweden. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Paul Kingsbury and Martha Palmer. 2002. From TreeBank to PropBank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, Las Palmas, Canary Islands - Spain. European Language Resources Association (ELRA).

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. 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)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Angelika Kratzer. 1979. Conditional necessity and possibility. In Rainer Bäuerle, Urs Egli, and Arnim von Stechow, editors, *Semantics From Different Points of View*, pages 117–147. Springer Verlag.

Angelika Kratzer. 1981. Partition and revision: The semantics of counterfactuals. *Journal of Philosophical Logic*, 10(2):201–216.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1512–1523, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Joachim Lambek. 1988. Categorial and categorical grammars. In *Categorial grammars and natural language structures*, pages 297–317. Springer.

Lane Lawley, Gene Louis Kim, and Lenhart Schubert. 2019. Towards natural language story understanding with rich logical schemas. In *Proceedings of the Sixth Workshop on Natural Language and Computer Science*, pages 11–22, Gothenburg, Sweden. Association for Computational Linguistics.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 590–599, Portland, Oregon, USA. Association for Computational Linguistics.

Elisabeth Lien and Milen Kouylekov. 2015. Semantic parsing for textual entailment. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 40–49, Bilbao, Spain. Association for Computational Linguistics.

Ken Litkowski. 2004. Senseval-3 task: Automatic labeling of semantic roles. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 9–12, Barcelona, Spain. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 521–528, Manchester, UK. Coling 2008 Organizing Committee.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Mehdi Manshadi, Daniel Gildea, and James Allen. 2013. 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)*, pages 64–72, Sofia, Bulgaria. Association for Computational Linguistics.

Christian M.I.M. Mathhiessen and John A. Bateman. 1991. *Text Generation and Systemic-Functional Linguistics*. Pinter Publishers, London, UK.

George A. Miller. 1995. WordNet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.

Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2015. Higher-order logical inference with compositional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2055–2061, Lisbon, Portugal. Association for Computational Linguistics.

Dan Moldovan and Vasile Rus. 2001. Logic form transformation of WordNet and its applicability to question answering. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 402–409, Toulouse, France. Association for Computational Linguistics.

Richard Montague. 1973. The proper treatment of quantification in ordinary English. In K. J. J. Hintikka, J. Moravcsic, and P. Suppes, editors, *Approaches to Natural Language*, pages 221–242. Reidel, Dordrecht.

Fabrizio Morbini and Lenhart Schubert. 2008. Metareasoning as an integral part of commonsense and autocognitive reasoning. In *AAAI-08 Workshop on Metareasoning*.

Fabrizio Morbini and Lenhart Schubert. 2011. Metareasoning as an Integral Part of Commonsense and Autocognitive Reasoning. In Michael T. Cox and Anita Raja, editors, *Metareasoning: Thinking about thinking*. MIT Press.

Fabrizio Morbini and Lenhart K. Schubert. 2009. Evaluation of EPILOG: a reasoner for Episodic Logic. In *Proceedings of the Ninth International Symposium on Logical Formalizations of Commonsense Reasoning*, Toronto, Canada.

Nasrin Mostafazadeh and James F. Allen. 2015. Learning semantically rich event inference rules using definition of verbs. In *Computational Linguistics and Intelligent Text Processing - 16th International Conference, CICLing Proceedings, Part I*, volume 9041 of *Lecture Notes in Computer Science*, pages 402–416, Cairo, Egypt. Springer.

Neha Nayak, Mark Kowarsky, Gabor Angeli, and Christopher D Manning. 2014. A dictionary of nonsubsective adjectives. Technical Report CSTR 2014-04, Department of Computer Science, Stanford University.

Ian Niles and Adam Pease. 2001. Towards a standard upper ontology. In *Proceedings of the International Conference on Formal Ontology in Information Systems - Volume 2001*, FOIS '01, pages 2–9, New York, NY, USA. ACM.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain. Association for Computational Linguistics.

Stephan Oepen. 2001. [incr tsdb()] – competence and performance laboratory. user manual. Technical report, Computational Linguistics, Saarland University, Saarbrücken, Germany.

Stephan Oepen and Dan Flickinger. 2019. The ERG at MRP 2019: Radically compositional semantic dependencies. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 40–44, Hong Kong. Association for Computational Linguistics.

Tim O'Gorman, Michael Regan, Kira Griffitt, Ulf Hermjakob, Kevin Knight, and Martha Palmer. 2018. AMR beyond the sentence: the multi-sentence AMR corpus. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3693–3702, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Jansen Orfan and James Allen. 2015. Learning new relations from concept ontologies derived from definitions. In *Proceedings of the AAAI 2015 Spring Symposium Series on Logical Formalizations of Commonsense Reasoning*.

W3C OWL Working Group. 2004. *OWL Web Ontology Language Guide*. W3C Recommendation. Available at `https://www.w3.org/TR/2004/REC-owl-guide-20040210`.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Jong C. Park. 1995. Quantifier scope and constituency. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 205–212, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Xiaochang Peng, Linfeng Song, Daniel Gildea, and Giorgio Satta. 2018. Sequence-to-sequence models for cache transition systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1842–1852, Melbourne, Australia. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Ian Perera, James Allen, Lucian Galescu, Choh Man Teng, Mark Burstein, Scott Friedman, David McDonald, and Jeffrey Rye. 2017. Natural language dialogue for building and learning models and structures. In *AAAI Conference on Artificial Intelligence*.

Georgiy Platonov, Lenhart Schubert, Benjamin Kane, and Aaron Gindi. 2020. A spoken dialogue system for spatial question answering in a physical blocks world. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 128–131, 1st virtual meeting. Association for Computational Linguistics.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

C. Pollard and I. A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago/CLSI Publications.

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

Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. 2004. Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 141–147, Geneva, Switzerland. COLING.

Adam Purtee and Lenhart Schubert. 2012. TTT: A tree transduction language for syntactic and semantic processing. In *Proceedings of the Workshop on Applications of Tree Automata Techniques in Natural Language Processing*, pages 21–30, Avignon, France. Association for Computational Linguistics.

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. 2015. Semantic proto-roles. *Transactions of the Association for Computational Linguistics*, 3:475–488.

Hyekyun Rhee, James Allen, Jennifer Mammen, and Mary Swift. 2014. Mobile phone-based asthma self-management aid for adolescents (masmaa): a feasibility study. *Patient Preference and Adherence*, 8:63.

Candace Ross, Andrei Barbu, Yevgeni Berzak, Battushig Myanganbayar, and Boris Katz. 2018. Grounding language acquisition by training semantic parsers using captioned videos. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2647–2656, Brussels, Belgium. Association for Computational Linguistics.

John Robert Ross. 1967. *Constraints on Variables in Syntax*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Rachel Rudinger, Adam Teichert, Ryan Culkin, Sheng Zhang, and Benjamin Van Durme. 2018. Neural-Davidsonian semantic proto-role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 944–955, Brussels, Belgium. Association for Computational Linguistics.

Eddy G Ruys and Yoad Winter. 2011. Quantifier scope in formal linguistics. In *Handbook of philosophical logic*, pages 159–225. Springer.

Nathan Salmon. 1986a. Reflexivity. *Notre Dame Journal of Formal Logic*, 27(3):401–429.

Nathan U. Salmon. 1986b. *Frege's Puzzle*. Ridgeview.

Victor Sánchez Valencia. 1991a. Categorial grammar and natural logic. ILTI Prepublication: Logic, Philosophy and Linguistics (LP) Series.

Victor Sánchez Valencia. 1991b. *Studies on natural logic and categorial grammar*. Ph.D. thesis, University of Amsterdam.

Maarten Sap, Ronan LeBras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019. ATOMIC: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*. AAAI Press.

Nathan Schneider, Jeffrey Flanigan, and Tim O'Gorman. 2015. The logic of AMR: Practical, unified, graph-based sentence semantics for NLP. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, pages 4–5, Denver, Colorado. Association for Computational Linguistics.

Lenhart Schubert. 2002. Can we derive general world knowledge from texts? In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 94–97, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Lenhart Schubert. 2013. NLog-like inference and commonsense reasoning. In *Perspectives on Semantic Representations for Textual Inference, special issue of Linguistic Issues in Language Technology (LiLT 9)*, volume 9, pages 1–26.

Lenhart Schubert. 2014. From treebank parses to episodic logic and commonsense inference. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 55–60, Baltimore, MD. Association for Computational Linguistics.

Lenhart Schubert. 2015. Semantic representation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 4132–4138. AAAI Press.

Lenhart Schubert and Matthew Tong. 2003. Extracting and evaluating general world knowledge from the brown corpus. In *Proceedings of the HLT-NAACL 2003 Workshop on Text Meaning*, pages 7–13.

Lenhart K. Schubert. 2000. The situations we talk about. In Jack Minker, editor, *Logic-based Artificial Intelligence*, pages 407–439. Kluwer Academic Publishers, Norwell, MA, USA.

Lenhart K. Schubert and Chung Hee Hwang. 2000. Episodic logic meets little red riding hood: A comprehensive natural representation for language understanding. In Lucja M. Iwańska and Stuart C. Shapiro, editors, *Natural Language Processing and Knowledge Representation*, pages 111–174. MIT Press, Cambridge, MA, USA.

Lenhart K. Schubert and Francis Jeffry Pelletier. 1982. From English to logic: Context-free computation of 'conventional' logical translation. *American Journal of Computational Linguistics*, 8(1):27–44.

Karin Kipper Schuler. 2006. *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania.

Tom De Smedt and Walter Daelemans. 2012. Pattern for python. *Journal of Machine Learning Research*, 13(66):2063–2067.

Scott Soames. 1985. Lost innocence. *Linguistics and Philosophy*, 8(1):59–71.

Scott Soames. 1987. Direct reference, propositional attitudes, and semantic content. *Philosophical Topics*, 15(1):47–87.

Scott Soames. 1989. Semantics and semantic competence. *Philosophical Perspectives*, 3:575–596.

Linfeng Song and Daniel Gildea. 2019. SemBleu: A robust metric for AMR parsing evaluation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4547–4552, Florence, Italy. Association for Computational Linguistics.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for AMR-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626, Melbourne, Australia. Association for Computational Linguistics.

Elias Stengel-Eskin, Aaron Steven White, Sheng Zhang, and Benjamin Van Durme. 2020. Universal decompositional semantic parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8427–8439, Online. Association for Computational Linguistics.

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

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI Conference on Artificial Intelligence*.

Ran Tian, Yusuke Miyao, and Takuya Matsuzaki. 2014. Logical inference on dependency-based compositional semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 79–89, Baltimore, Maryland. Association for Computational Linguistics.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

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

Johan Van Benthem et al. 1986. *Essays in Logical Semantics*. Springer.

Benjamin Van Durme, Phillip Michalak, and Lenhart Schubert. 2009. Deriving generalized knowledge from corpora using WordNet abstraction. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 808–816, Athens, Greece. Association for Computational Linguistics.

Siddharth Vashishtha, Benjamin Van Durme, and Aaron Steven White. 2019. Fine-grained temporal relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2906–2919, Florence, Italy. Association for Computational Linguistics.

Frank Veltman. 1976. Prejudices, presuppositions and the theory of counterfactuals. In Jeroen Groenendijk and Martin Stokhof, editors, *Amsterdam papers in formal grammar*, volume 1, pages 248–281. University of Amsterdam.

Piek Vossen, Willem Meijs, and M. den Broeder. 1989. Meaning and structure in dictionary definitions. In *Computational Lexicography for Natural Language Processing*, pages 171–192. Longman Publishing Group, White Plains, NY, USA.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020. AMR-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*, 8:19–33.

Hila Weisman, Jonathan Berant, Idan Szpektor, and Ido Dagan. 2012. Learning verb inference rules from linguistically-motivated evidence. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 194–204, Jeju Island, Korea. Association for Computational Linguistics.

Aaron Steven White and Kyle Rawlins. 2016. A computational model of S-selection. *Semantics and Linguistic Theory*, 26:641–663. Number: 0.

Aaron Steven White and Kyle Rawlins. 2020. Frequency, acceptability, and selection: A case study of clause-embedding. *Glossa: a journal of general linguistics*, 5(1):105. Number: 1 Publisher: Ubiquity Press.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on Universal Dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1713–1723, Austin, Texas. Association for Computational Linguistics.

Yorick Wilks, Dan Fass, Cheng-ming Guo, James E. McDonald, Tony Plate, and Brian M. Slator. 1989. A tractable machine dictionary as a resource for computational semantics. In *Computational Lexicography for Natural Language Processing*, pages 193–228. Longman Publishing Group, White Plains, NY, USA.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

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

Xuchen Yao and Yi Zhang. 2010. Question generation with minimal recursion semantics. In *Proceedings of the Third Workshop on Question Generation: Shared Task Evaluation Challenge*.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Transition-based neural word segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 421–431, Berlin, Germany. Association for Computational Linguistics.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019a. AMR parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94, Florence, Italy. Association for Computational Linguistics.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019b. Broad-coverage semantic parsing as transduction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3786–3798, Hong Kong, China. Association for Computational Linguistics.

Sheng Zhang, Rachel Rudinger, and Benjamin Van Durme. 2017. An evaluation of PredPatt and open IE via stage 1 semantic role labeling. In *IWCS 2017 — 12th International Conference on Computational Semantics — Short papers*.

Yi Zhang and Hans-Ulrich Krieger. 2011. Large-scale corpus-driven PCFG approximation of an HPSG. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 198–208, Dublin, Ireland. Association for Computational Linguistics.

# Appendix A High-Fidelity Lexical Axiom Construction from Verb Glosses

## A.1 Introduction

Words encapsulate a great deal of knowledge, and in conjunction with language syntax, allow human beings to construct sentences that convey novel ideas to one another. Any system intended for broad natural language understanding will need to be able to perform inferences on the words that are the building blocks of language. For this reason, there have been many attempts to transduce informal lexical knowledge from machine readable dictionaries into a formally structured form (Calzolari, 1984; Chodorow et al., 1985; Harabagiu et al., 1999; Moldovan and Rus, 2001; Hobbs, 2008; Allen et al., 2013).

Consider an example of the types of knowledge these approaches seek to extract in Figure A.1. WordNet defines *slam2.v*, i.e., sense 2 of the verb *slam*, as “strike violently". This gloss states an implication that if “x slams y" characterizes an event $e$, then “x strikes y violently" also characterizes event $e$. All language phenomena must be able to be represented and reasoned about for such axioms to be useful in a language understanding system. This is where previous approaches share a common shortcoming: the logical representations that the lexical knowledge is mapped into are insufficient for representing many common natural language devices or for performing inference.

The contributions of this paper are the following:

**Gloss** – *slam2.v* : “strike violently"
**Axiom** – ((x slam2.v y) ** e)
→ ((x (violently.adv (strike.v y))) ** e)

Figure A.1: Example of rule extraction from machine readable dictionaries for WordNet entry of *slam2.v*.

- We demonstrate limitations in previous approaches to extracting lexical knowledge from machine readable dictionaries, particularly in their choices of logical representation.
- We present an approach to extracting lexical axioms in EL, which is a logical representation that overcomes these limitations. Our approach includes novel preprocessing and information synthesis strategies for making precise axioms.
- We present *EL-smatch*, a generalized *smatch* scoring metric for partial scoring of semantic parses with complex operators and predicates.

The remainder of the paper presents related work in Section 2, background in Section 3, then a description of our semantic parsing approach in Section 4. A description of *EL-smatch* is presented in Section 5, followed by experimental results in Section 6, and future work and conclusions in Section 7.

## A.2 Related Work

There have been many approaches in the past to extracting lexical information from machine-readable dictionaries. Early approaches to this problem focused on surface-level techniques, including hypernym extraction (Calzolari, 1984; Chodorow et al., 1985), pattern matching (Alshawi, 1989; Vossen et al., 1989; Wilks et al., 1989), and co-occurrence data extraction (Wilks et al., 1989).

In an evaluation of such methods, Ide & Veronis (1993) identified key challenges that thwart progress on this problem—challenges that persist to this day. Among these are the fact that dictionary glosses are often abstract, sometimes miss important information (such as arguments), and may be

inconsistent with one another. Evidently there is a need for sophisticated extraction techniques to acquire accurate and consistent knowledge from dictionaries.

Most modern approaches to this problem use WordNet (Miller, 1995) as the lexical resource because of the linguistic and semantic annotations that accompany the glosses. Some work encodes WordNet glosses into variants of first-order logic (FOL) (Harabagiu et al., 1999; Moldovan and Rus, 2001; Hobbs, 2008), such as Hobbs Logical Form (HLF) (Hobbs, 1985), while other work encodes them into OWL-DL (OWL Working Group, 2004; Allen et al., 2011, 2013; Orfan and Allen, 2015; Mostafazadeh and Allen, 2015). A particularly noteworthy line of work is that by Allen et al. (2013), which integrates information from a high-level ontology with glosses of semantically related clusters of words to construct inference-supporting micro-theories of concepts corresponding to these words. While these advances are significant, they are limited by the expressivity of the representations used, in comparison with the richness of natural language.

### A.2.1 Limitations of Logical Representations Used by Previous Approaches

As discussed by Schubert (2015), the choice of semantic representation is an important component of the natural language understanding problem. Because of space constraints, we will discuss only a few of the relevant issues and point the reader to (Schubert, 2015) for a more in-depth analysis of the issues at hand. The logical representation used for robust language understanding must satisfy the following requirements:

- Express the semantic content of most, if not all, possible natural language constructions;
- Have associated methods of inference;
- Have a formal interpretation.

The semantic representations used by previous approaches fall short on at least one of the above requirements. FOL struggles to express predicate modification (especially nonintersective modification), nonstandard quantifiers such as *most* or *at least 50*, and modality. Approaches that rely

on functionalizing predication and connectives as a means of allowing for arbitrary propositional attitudes ultimately fail because quantifiers cannot be functionalized; thus they cannot capture the meaning of sentences with a modally embedded quantifier such as the following (with *believes* taking scope over *every*):

*Kim believes that every galaxy harbors life.*

HLF (Hobbs, 1985) is another common choice of semantic representation. It strives to capture sentential meaning within a subset of FOL by treating all words as predicates, including negation, disjunction, quantifiers, and modifiers. But it is unable to distinguish between events and propositions and between predicate and sentence modifiers, and the formal interpretation of quantification in HLF can lead to contradiction (Schubert, 2015).

OWL-DL (OWL Working Group, 2004) was designed for knowledge engineering on specific domains and thus cannot handle many common natural language phenomena, such as predicate and sentence reification, predicate modification, self-reference, and uncertainty. There have been many efforts to allow for such phenomena, with varying degrees of success. As just one example, consider the common practice in OWL-DL of treating predicate modification as predicate intersection. For example, “whisper loudly" is represented as whisper $\sqcap \forall_{of}$-1.(loudly). *whisper* is the set of individual whispering events and $\forall_{of}$-1.(loudly) is the set of individual events that are modified by the adverb *loudly*. But according to WordNet, to whisper is to speak softly, so under an intersective interpretation of the modifiers, a loud whisper is both soft and loud. Similarly, WordNet glosses the verb *spin* as *revolve quickly*, so that under an intersective interpretation, a slow spin is both quick and slow. Analogously for nouns, a large pond or large brochure would be both large and small (*brochure* is glossed as *a small book*, and *pond* as *a small lake*). Even more difficult issues, from an OWL-DL perspective, are generalized quantifiers, uncertainty, attitudes, and reification, such as exemplified in the sentence

*When self-driving cars are properly adopted, vehicles that need humans to drive them will probably be banned, according to Tesla CEO Elon Musk.*

For a fuller discussion of issues in representations based on FOL, HLF, OWL-DL, etc., again

see (Schubert, 2015).

## A.3 Background

This section describes background material underlying our semantic parsing approach. First, we describe WordNet (Miller, 1995), our input lexical resource. Then, we describe Episodic Logic (EL), our choice of semantic representation for lexical axioms.

### A.3.1 WordNet

WordNet is a lexical knowledge base that contains glosses for words, enumerates the word senses of each word, groups synonyms into *synsets*, encodes generality/specificity relations as *hypernym/hyponyms*, and provides schematic sentence structures for each word in the form of simple *frames*. The semantic annotations accompanying the glosses help in building a robust parser by reducing the amount of inference necessary for building axioms and assisting in handling mistakes in the glosses. Also, a significant proportion of the words in WordNet glosses have been tagged with their word senses and part-of-speech in the Princeton Annotated Gloss Corpus.[1] This helps with the important but often neglected word sense disambiguation (WSD) aspect of the interpretation problem; certainly ambiguous or faulty WSD can lead to misunderstandings and faulty inferences (is *Mary had a little lamb* about ownership or dining?). We use WordNet 3.0, which at the time of writing is the most recent version that is fully available for the UNIX environment, and focus on the verbs in this paper.

### A.3.2 Episodic Logic

EL (Schubert and Hwang, 2000) was designed to be close to natural language, with the intuition that a logical representation that retains much of the expressivity of natural language will be able to

[1] http://wordnet.princeton.edu/glosstag.shtml

more fully represent the complex constructs in natural language. EL provides constructs that are not common in most FOL-based languages, such as predicate modifiers, generalized quantifiers, reification, and ways of associating episodes (events or situations) with arbitrarily complex sentences. Importantly, EL is backed by a comprehensive inference system, EPILOG, which has been shown to be competitive with FOL theorem provers despite its substantially greater expressivity (Morbini and Schubert, 2009).

EL uses infix notation for readability, with the "subject" argument preceding the predicate and any additional arguments following the predicate. For associating episodes with logical sentences, EL introduces two modal operators '**' and '*'. [$\Phi$ ** $e$] means that $\Phi$ *characterizes* (i.e. describes as a whole) episode $e$ and [$\Phi$ * $e$] means that $\Phi$ is true in (i.e. describes a piece or aspect of) episode $e$.

We show that EL overcomes some of the limitations of previous work that have been discussed using an example. Below is the EL representation for the sentence *Kim believes that every galaxy harbors life*.

```
(Kim.name believe.v
 (That (∀x (x galaxy.n)
           (x harbor.v (K life.n)))))
```

`That` and `K` are sentence and predicate reification operators, respectively and `(∀x Φ(x) Ψ(x))` is equivalent to `(∀x (Φ(x) → Ψ(x)))`.[2] For discussion of the semantic types of the operators alluded to in this section and the connection to Davidsonian event semantics and other variants of event/situation semantics, see the papers describing EL (Schubert and Hwang, 2000; Schubert, 2000).

[2]However, EL's quantifier syntax also allows, e.g., `(most.det x Φ(x) Ψ(x))`, which is not reducible to FOL.

## A.4 Gloss Axiomatization

In this section, we describe our approach to semantic parsing and axiomatization of WordNet entries. Our approach consists of three major steps:

1. Argument structure inference (Section A.4.1)
2. Semantic parsing of the gloss (Section A.4.2)
3. Axiom construction (Section A.4.3)

Figure A.2 shows the entire process for the previously introduced example, *slam2.v*. The argument inference step refines the WordNet sentence frames using the provided examples. Specific pronouns associated with argument position are inserted as dummy arguments into the corresponding argument positions in the gloss, and the modified gloss is semantically parsed into EL. Axiom construction replaces the dummy arguments with variables and constructs a scoped axiom relating the entry word and the semantic parse of the gloss using the characterization operator '**'. In the simple example *slam2.v*, most of the subroutines used in each step have no effect. All transformations outside the scope of the BLLIP parser are performed with hand-written rules, which were fine-tuned using a development set of 550 verb synset entries.

### A.4.1 Argument Structure Inference

We initially use the frames in WordNet to hypothesize the argument structures. For example, the frames for *quarrel1.v* are [Somebody quarrel1.v] and [Somebody quarrel1.v PP]. From this we hypothesize that *quarrel1.v* has a subject argument that is a person, no object argument, and may include a prepositional phrase adjunct.

Then we refine the frames by looking at the examples and gloss(es) available for the *synset*. The examples for *quarrel1.v*: *"We quarreled over the question as to who discovered America"*

**WordNet entry**
*slam2.v*
Tagged gloss: `(VB strike1) (RB violently1)`
Frames: `[Somebody slam2.v Something]`
`[Somebody slam2.v Somebody]`
Examples: (“slam the ball")

**4.1 Argument Structure Inference**
Refined Frames:
`[Somebody slam2.v Something]`

**4.2 Semantic Parsing**
Parse: `(Me.pro (violently1.adv (strike1.v It.pro)))`

**4.3 Axiom Construction**
Axiom:
```
(∀x1 (∀y1 (∀e [[x1 slam2.v y1] ** e]
              [[[x1 (violently1.adv (strike1.v y1))] ** e]
               and [x1 person1.n] [y1 thing12.n]])))
```

Figure A.2: Example gloss axiomatization process for WordNet entry *slam2.v*. The numbering corresponds to the subsections where these stages are discussed in detail.

and *“These two fellows are always scrapping*[3] *over something"* suggest that the subject argument can be plural and the PP can be specialized to PP-OVER. We identify the arguments and semantic types of the examples through a semantic parse, which is obtained using the method described in Section 4.2. Then we either update existing frames or introduce additional frames based on the agreement among examples and the number of available examples. We similarly obtain semantic types for arguments from glosses. For example, *paint1.v* has the gloss *“make a painting"* and the frame [Somebody -s Something]. Based on the gloss, we infer that the semantic type for the object argument is *painting*. Gloss-based argument structure inference can be done during the gloss parsing step, to avoid redundant computation.

Finally, we merge redundant frames. For example, frames that differ only in that one has *somebody* in a certain argument position where the other has *something* are merged into one frame where we simply use *something* (as a category allowing for both things and persons). Also there are rules for merging predicate complement types (Adjective/Noun & PP → Adjective/Noun/PP) and adding dative alternations to ditransitive frames [Somebody -s Somebody Something] → [Somebody -s Something to Somebody].

[3] *quarrel1.v* and *scrap2.v* are in the same synset, so they share example sentences and are interchangeable in this context.

### A.4.2 Semantic Parsing of Glosses

Sentence-level semantic parsers for EL have been developed previously, which we can use for semantic parsing of the glosses (Schubert, 2002; Schubert and Tong, 2003; Gordon and Schubert, 2010; Schubert, 2014). For the parser to be effective, some preprocessing of the glosses is necessary because glosses often omit arguments, resulting in an incomplete sentence. There are also some serious shortcomings to general semantic parsers, particularly in handling coordinators *and/or*. In this section, we describe the complete semantic parsing process of glosses and the details of each step. Throughout our semantic parsing implementation, we use the tree-to-tree transduction tool (TTT) (Purtee and Schubert, 2012) for transparent and modular tree transformations[4] and the BLLIP parser (Charniak, 2000) to get Treebank parses.

The general outline of the gloss processing steps is described below:

1. Create separate POS-tagged word sequences for distinct glosses:
   a. Label gloss $g$ with POS tags using the Princeton Annotated Gloss Corpus, backing off to the synset type in the sense key.[5]
   b. Split multigloss trees along semicolons for individual POS tagged glosses $p_1, p_2, ..., p_n$.
2. Create an easy-to-parse sentence for each gloss:
   a. Factor out coordinators, leaving the first conjunct in the gloss. Save the coordinated phrases $c_{p_i}$ for later insertion.
   b. Insert dummy arguments (*I*, *it*, *them*).
   c. Drop POS tags to create new gloss $g_i'$.
3. Syntactically parse each gloss sentence into initial LFs:

[4] We do not explicitly state where TTT is used in the algorithm since it is a general tree transformation tool, which is used throughout the algorithm whenever a tree transformation is necessary.

[5] Every word in the glosses of the Princeton Annotated Gloss Corpus is labeled with the POS tag or the sense key. The synset type distinguishes between nouns, verbs, adjectives, and adverbs.

a. Parse $g_i'$ into tree $t_i$ using the BLLIP parser.

b. Refine POS tags in $t_i$ using the Princeton Annotated Gloss Corpus.

c. Run $t_i$ through the sentence-level semantic parser to get logical form $s_i$.

4. Refine the initial LFs:

a. Reinsert coordinated phrases $c_{p_i}$ into $s_i$.

b. Introduce word senses into the logical form.

We now describe the sentence-level semantic parser, coordinator factorization, argument insertion/inference, and word sense introduction in more detail.

**Sentence Level Semantic Parser**

The sentence-level semantic (EL) parser we use is modeled after the partial interpreter used by the KNEXT system (Van Durme et al., 2009; Gordon and Schubert, 2010). First, the parser applies corrective and disambiguating transformations to raw Treebank trees. For example, these correct certain systematic prepositional phrase (PP) attachment errors, distinguish copular *be* from other forms, assimilate verb particles into the verb, particularize SBAR constituents to relative clauses, adverbials, or clausal nominals, insert traces for dislocated constituents, etc. Second, the parser uses about 100 rules to compositionally interpret Treebank parses into initial interpretations. Finally, coreference resolution, quantifier, coordinator, and tense scoping, temporal deindexing, (non-embedded) Skolemization, equality reduction, conjunction splitting and other canonicalization operations are applied to refine the logical form.

**Argument Insertion and Inference**

WordNet glosses (and glosses in general) only include arguments when necessary to specify some semantic type for the argument. Figure A.3 displays example glosses from WordNet that demonstrate this treatment of arguments. Both the subject and object arguments in the gloss of *slam2.v* are

| *Argument position* | *English text* | *EL atom* |
|---|---|---|
| subject | I/my/myself | Me.pro |
| direct object | it | It.pro |
| indirect object | them | They.pro |

Table A.1: Mappings between dummy argument position, text, and EL atoms.

*slam2.v* – *subject* strike *object* violently
*paint1.v* – *subject* make a painting

Figure A.3: Example glosses demonstrating the treatment of arguments in glosses. Underlined words are arguments and italicized arguments indicate where an argument should exist, but does not in the gloss.

omitted, and the subject is omitted from the gloss of *paint1.v*, while the object in the gloss is included.

We make arguments explicit and unify their treatment in order to improve Treebank and semantic parses and simplify the axiom construction step, described in Section A.4.3. Figure A.4 shows unified versions of the glosses that appear in Figure A.3, *slam2.v* and *paint1.v*. In this unified treatment, all arguments are represented by argument position-specific dummy arguments. Table A.1 lists the dummy arguments and their relation to the argument position and EL. Dummy arguments are inserted into the POS tagged gloss $p_i$ based on the inferred argument structure from Section 4.1 and the insertions are achieved through pattern matching of the POS tags.

Finally, some glosses contain references to the subject using the terms *one*, *one's*, or *oneself* (e.g. *sprawl1.v* : *sit or lie with one's limbs spread out*). These are mapped to *I*, *my*, and *myself*, respectively to correctly corefer with the dummy subject argument *I*.

*slam2.v* – *I'll* strike *it* violently
*paint1.v* – *I'll* make *it*; (*it* : a painting)

Figure A.4: Unified versions of WordNet glosses from Figure A.3.

### Coordinator Factorization

Treebank and semantic parsers are prone to errors for coordinated phrases, often mistaking them for appositives, or vice-versa. To minimize such errors, we developed a method of factorizing coordinated phrases. The conjuncts can usually be identified by syntactic and semantic relatedness. This can be seen in the WordNet gloss for *edit1.v*: *prepare for publication or presentation by correcting, revising, or adapting*. We use linguistic phrase types as a proxy for syntactic and semantic relatedness. That is, we identify coordinated groups of verb phrases, noun phrases, adjectival phrases, and prepositional phrases. These phrase groups are pulled out of the sentence, and only the first phrase in the group is left in the sentence.

The phrase groups are identified using a set of rules that were fine-tuned with reference to the development set of verb synsets. The rules tend to handle common modifications, such as adjectives in noun phrases. For ambiguous cases, such as prepositional modification, factorization is not performed.

The phrase groups are passed through a modified sentence-level semantic parser (stopping short of the coordinator scoping step), and embedded back into the gloss logical form before the coordinator scoping step in the semantic parsing of the gloss. The place of insertion is identified by matching the first phrase in the phrase group with a phrase in the logical form.

### Word Sense Introduction

Word sense introduction is assisted by the hand-tagged word senses in WordNet. All words that are not hand-tagged with a word sense are given the lowest numbered word sense with a frame matching the context of its use in the gloss. Generally, the lower numbered word senses in WordNet are the most relevant senses of the word.

### A.4.3 Axiom Construction

Finally, we take the results from Sections 4.1 and 4.2 and construct the axiom. Dummy arguments in the parsed gloss are correlated with the arguments in the frame using the mapping in Table A.1. We replace the arguments with variables, introduce logical formulas asserting the semantic types (from the argument structure in Section 4.1), and construct an axiom asserting that the truth of the entry word with the proper argument structure (without semantic types) implies the truth of the semantic parse of the gloss and semantic types of the arguments. Before axiom construction, the example from Figure A.2, *slam2.v*, has the following refined frame and semantic parse of the gloss from Sections 4.1 and 4.2, respectively:

```
[Somebody slam2.v Something]
[Me.pro (violently1.adv (strike1.v It.pro))]
```

After we replace the arguments and create formulas asserting the semantic types, we have:

```
[x1 slam2.v y1]
[x1 (violently1.adv (strike1.v y1))]
[x1 person1.n], [y1 thing12.n]
```

Finally, we construct an axiom of form `(∀x Φ(x) Ψ(x))` (equivalent to `(∀x (Φ(x) → Ψ(x)))`) and using the modal *characterization* operator **:

```
(∀x1,y1,e
  [[x1 slam2.v y1] ** e]
  [[[x1 (violently1.adv
          (strike1.v y1))] ** e]
   and [x1 person1.n] [y1 thing12.n]])
```

We can easily generate converse axioms as well, such as that if a person strikes something violently, then it is probably the case that he or she slams it (in the *slam2.v* sense). EL allows us to express a degree of uncertainty in the formulation of the converse, and this is appropriate to the extent that lexical glosses cannot be expected to provide complete, “airtight" definitions, but rather

just the most important semantic content. However, in this paper we limit ourselves to discussion of the “forward" version of gloss-derived axioms.

## A.5 Experiments

We conducted two experiments to demonstrate the efficacy of our approach for semantic parsing and the usefulness of the resulting axioms for inference.[6]

### A.5.1 Semantic Parsing Evaluation

We constructed a gold standard set of axioms by selecting 50 random WordNet synsets that were not used during development. Gold standard axioms for these synsets were written by the first author, then refined in collaboration between the two authors.[7] The 50 synsets resulted in 52 axioms and 2,764 triples in the gold standard. The results in Table A.2 show the system performance using both *EL-smatch* and full axiom metrics. In the full axiom metric, the precision measures the number of axioms that are completely correct, and the recall measures the number of axioms generated (which can vary due to merged glosses and multiple frames). The *EL-smatch* score of 0.83 shows that the axioms are generally good, even when not completely correct. Generating completely correct axioms is difficult because there are multiple non-trivial subproblems, such as prepositional attachment and word sense disambiguation. *EL-smatch* displays a more fine-grained measure of our system performance than the full axiom metric.

[6] One reviewer suggested comparing our axioms with ontologies linked to WordNet, such as SUMO (Niles and Pease, 2001) and DOLCE (Gangemi et al., 2002), or with the hypernym hierarchy of WordNet. Such an experiment was performed by Allen et al. (2013), which showed that WordNet glosses contain information that is not found in the structural relations of WordNet. A similar experiment by us is unlikely to shed additional light on the topic.

[7] Due to time constraints, this evaluation was performed on a gold standard developed primarily by only one annotator. We hope to remedy this in future work including an analysis of interannotator agreement.

| Measure | Precision | Recall | F1 |
|---|---|---|---|
| *EL-smatch* | 0.85 | 0.82 | 0.83 |
| Full Axiom | 0.29 | - | - |

Table A.2: Performance against gold standard parses of 50 synsets.

### A.5.2 Inference Evaluation

To our knowledge, no earlier work evaluates inference in a manner that captures the details of semantically rich lexical axioms. Therefore, in order to compare our results to previous work, we evaluate a stripped-down version of our inference mechanism on a manually created verb entailment dataset (Weisman et al., 2012). This dataset contains 812 directed verb pairs, $v1 \rightarrow v2$, which are annotated 'yes' if the annotator could think of plausible contexts under which the entailment from $v1$ to $v2$ holds. For example, *identify* entails *recognize* in some contexts, does not entail *describe* is any context. Though the dataset is not rich, many previous systems (Mostafazadeh and Allen, 2015; Weisman et al., 2012; Chklovski and Pantel, 2004) have evaluated on this dataset, establishing it as a basic standard of comparison. In order to fit our axioms to this dataset, we remove semantic roles (verb arguments and adjuncts) from our axioms. Also, since the dataset has no word senses, the inferences begin with every synset that contains a sense of the starting word, and the final predicted entailments suppress sense distinctions. When generating inferences, we find verbs in the consequent of the axiom that are not modified by a negation or negating adverb (e.g., *nearly, almost,* etc.). Such inferences are chained up to three times, or until an abstract word is reached (e.g., *be, go,* etc.), which glosses do not sufficiently describe. This blacklist contains 24 abstract words.

Table A.3 shows the results on this dataset. *TRIPS* is an approach by Mostafazadeh & Allen (2015), which constructs axioms from WordNet using the TRIPS parser and represents its axioms in OWL-DL, *Supervised* is a supervised learning approach by Weisman et al. (2012), *VerbOcean* classifies entailments according to the strength relation of the VerbOcean knowledege-base (Chklovski and Pantel, 2004), and *Random* is a method that randomly classifies the pair with probability equal to the distribution in the testset (27.7%). The performance of our system is com-

| Method | Precision | Recall | F1 |
|---|---|---|---|
| Our Approach | 0.43 | 0.53 | 0.48 |
| *TRIPS* | 0.50 | 0.45 | 0.47 |
| *Supervised* | 0.40 | 0.71 | 0.51 |
| *VerbOcean* | 0.33 | 0.15 | 0.20 |
| *Random* | 0.28 | 0.29 | 0.28 |

Table A.3: Performance against gold standard parses of 50 synsets.

petitive with state-of-the-art systems *TRIPS* and *Supervised* on this task. Our system performance splits the performance of *TRIPS* and *Supervised* in all three measures.

The inference capabilities of our axioms exceed what is evaluated by this testset. Because of space constraints, an example of a more expressive inference using extracted axioms is included in supplementary material[8].

### A.5.3 Error Analysis

In the semantic parsing evaluation, most of the parsing errors arose from a failure in the sentence parser or preprocessing directly preceding the sentence parser. That is, 17 out of the 52 axioms had errors arising from the sentence parser. These errors arose from either linguistic patterns that we did not encounter in our development set or in complex sentences (e.g. *take a walk for one's health or to aid digestion, as after a meal*). Many of these can be avoided in the future by increasing the development set. Fortunately, the semantic parser uses keywords to mark ambiguous attachments or phrases, so that in many cases, axioms that are not fully parsed can be identified and ignored, rather than using an incorrectly parsed axiom.

WSD and incorrect scoping of semantic types are also major sources of errors. The challenge of WSD was minimized by the subset of hand-tagged word senses in WordNet. We may be able to reduce such errors in the future by merging together redundant or overly specific word senses. Incorrect scoping of semantic types is particularly problematic when the semantic type is specified

[8] http://www.cs.rochester.edu/u/gkim21/papers/high-fidelity-lex-supplementary.pdf

in the gloss itself, as the type constraint needed to move across scopes. Our system performed well on coordinator scoping. We correctly scoped 23 of the 27 instances of coordinators in the dataset. Coordinators are generally a great source of error in parsers and this result is evidence of the effectiveness of our coordinator handling mechanism. In all four instances, the disjunctions were extracted from the gloss correctly, but were not reintroduced into the axiom. As such, this error did not make these axioms incorrect, rather incomplete.

## A.6 Future Work and Conclusions

There are many attractive directions for future work. The scope of this project can be broadened to include nouns, adjectives, and adverbs, as required for any system that actually tackles the natural language understanding problem. There are also many ways to refine and deepen the gloss interpretation process. The parses may be improved by looking through the *hypernym* graph and borrowing results from parses of parents (generalizations) of words. We can also incorporate techniques from Allen et al. (2011; 2013) and Mostafazadeh & Allen (2015) to integrate results from related sets of glosses. The high-level TRIPS ontology could be used to improve robustness in the face of inconsistencies in WordNet and interpretation errors. Also, more sophisticated WSD techniques, such as those from the SENSEVAL-3 task on WSD (Litkowski, 2004), could be used to improve semantic precision, and argument coherence could be improved using techniques from Mostafazadeh & Allen (Mostafazadeh and Allen, 2015). Another possible avenue is concurrent use of information from multiple dictionaries, such as Wiktionary, VerbNet, and WordNet, to construct more complete and reliable axioms, in particular with respect to argument structure and types.

We argued that the semantic representations used in previous approaches to extracting lexical axioms from dictionaries are insufficient for achieving a natural language understanding system. We presented an approach to extracting lexical axioms of verbs from WordNet into EL, an expressive semantic representation that overcomes the shortcomings of the representations used in the past. We also presented a generalized *smatch* scoring metric, *EL-smatch*, which we used to evaluate our

system. The evaluation shows that our approach constructs precise verb axioms from WordNet. Furthermore, we demonstrate that the generated axioms perform competitively against the state of the art in a verb entailment task. We aim to apply these axioms to more comprehensive language understanding tasks and commonsense reasoning tests when we have sufficient coverage of the lexicon.

# Appendix B Traditional Aristotelian Syllogisms in ULF

## B.1 Scoping and Polarity Derivations

Before proving the syllogisms we go through the scoping and polarity derivations of the propositions used in the syllogisms.

**Proposition i.** "Every $X$ is a $Y$"

1. ((every.d $X$) (be.v (= (a.d $Y$)))) — ULF
2. (every.d $x$: ($x$ $X$) ($x$ (be.v (= (a.d $Y$))))) — Scope every.d
3. (every.d $x$: ($x$ $X$)<br>(a.d $y$: ($y$ $Y$) ($x$ (be.v (= $y$))))) — Scope a.d
4. (every.d$_{(-,+)}$ $x$: ($x$ $X$)<br>(a.d$_{(+,+)}$ $y$: ($y$ $Y$)<br>($x$ (be.v (= $y$))))) — Lexical monotonicity
5. (every.d $x$: ($x$ $X$)$_{-}$<br>(a.d $y$: ($y$ $Y$)$_{+}$<br>($x$ (be.v (= $y$)))$_{+}$)$_{+}$) — Local entail. context
6. (every.d $x$: ($x$ $X$)$^{-}$<br>(a.d $y$: ($y$ $Y$)$^{+}$<br>($x$ (be.v (= $y$)))$^{+}$)$^{+}$) — Global polarity

**Proposition ii.** "Some $X$ is a $Y$"

1. ((some.d $X$) (be.v (= (a.d $Y$)))) — ULF

2. (some.d $x$: ($x$ $X$) Scoping
   (a.d $y$: ($y$ $Y$) ($x$ (be.v (= $y$)))))
3. (some.d$_{(+,+)}$ $x$: ($x$ $X$) Lexical monotonicity
   (a.d$_{(+,+)}$ $y$: ($y$ $Y$)
   ($x$ (be.v (= $y$)))))
4. (some.d $x$: ($x$ $X$)$_{+}$ Local entail. context
   (a.d $y$: ($y$ $Y$)$_{+}$
   ($x$ (be.v (= $y$)))$_{+}$)$_{+}$)
5. (some.d $x$: ($x$ $X$)$^{+}$ Global polarity
   (a.d $y$: ($y$ $Y$)$^{+}$
   ($x$ (be.v (= $y$)))$^{+}$)$^{+}$)

**Proposition iii.** "No $X$ is a $Y$"

1. ((no.d $X$) (be.v (= (a.d $Y$)))) ULF
2. (no.d $x$: ($x$ $X$) Scoping
   (a.d $y$: ($y$ $Y$) ($x$ (be.v (= $y$)))))
3. (no.d$_{(-,-)}$ $x$: ($x$ $X$) Lexical monotonicity
   (a.d$_{(+,+)}$ $y$: ($y$ $Y$)
   ($x$ (be.v (= $y$)))))
4. (no.d $x$: ($x$ $X$)$_{-}$ Local entail. context
   (a.d $y$: ($y$ $Y$)$_{+}$
   ($x$ (be.v (= $y$)))$_{+}$)$_{-}$)
5. (no.d $x$: ($x$ $X$)$^{-}$ Global polarity
   (a.d $y$: ($y$ $Y$)$^{-}$
   ($x$ (be.v (= $y$)))$^{-}$)$^{-}$)

**Proposition iv.** "Not every $X$ is a $Y$"

1. (not (every.d $X$) (be.v (= (a.d $Y$)))) ULF
2. (not (every.d $x$: ($x$ $X$) Scoping
   (a.d $y$: ($y$ $Y$)
   ($x$ (be.v (= $y$))))))
3. (not$_{(-)}$ (every.d$_{(-,+)}$ $x$: ($x$ $X$) Lexical monotonicity
   (a.d$_{(+,+)}$ $y$: ($y$ $Y$)
   ($x$ (be.v (= $y$))))))
4. (not (every.d $x$: ($x$ $X$)$_{-}$ Local entail. context
   (a.d $y$: ($y$ $Y$)$_{+}$
   ($x$ (be.v (= $y$)))$_{+}$)$_{+}$)$_{-}$)

5. (not (every.d $x$: ($x$ $X$)$^{+}$ Global polarity
(a.d $y$: ($y$ $Y$)$^{-}$
($x$ (be.v (= $y$)))$^{-}$)$^{-}$)$^{-}$)

## B.2 Deriving the Syllogisms

**Syllogism 1 (Barbara).** "Every $M$ is a $P$" and "Every $S$ is a $M$" entail "Every $S$ is a $P$".

**Proof**

1. ((every.d $M$) (be.v (= (a.d $P$)))) Assumption
2. ((every.d $S$) (be.v (= (a.d $M$)))) Assumption
3. ((every.d $M$)$^{-}$ (be.v (= (a.d $P$)))) Polarity marking, 1.
4. ((every.d $S$) (be.v (= (a.d $P$)))) UMI, 2.,3.

**Syllogism 2 (Darii).** "Every $M$ is a $P$" and "Some $S$ is a $M$" entail "Some $S$ is a $P$".

**Proof**

1. ((every.d $M$) (be.v (= (a.d $P$)))) Assumption
2. ((some.d $S$) (be.v (= (a.d $M$)))) Assumption
3. ((some.d $S$) (be.v (= (a.d $M$)$^{+}$))) Polarity marking, 2.
4. ((some.d $S$) (be.v (= (a.d $P$)))) UMI, 1.,3.

**Syllogism 3 (Celarent).** "No $M$ is a $P$" and "Every $S$ is a $M$" entail "No $S$ is a $P$".

**Proof**

1. ((no.d $M$) (be.v (= (a.d $P$)))) Assumption
2. ((every.d $S$) (be.v (= (a.d $M$)))) Assumption
3. ((no.d $M$)$^{-}$ (be.v (= (a.d $P$)))) Polarity marking, 1.
4. ((no.d $S$) (be.v (= (a.d $P$)))) UMI, 2.,3.

**Syllogism 4 (Ferio).** "No $M$ is a $P$" and "Some $S$ is a $M$" entail "Not every $S$ is a $P$".

Using the logical interpretation of Sánchez-Valencia's Negation Rule twice (5.1), we see that the syllogism is true iff "Every $S$ is a $P$" and "Some $S$ is a $M$" entail "Some $M$ is a $P$". We prove this as follows.

**Proof**

| | |
|---|---|
| 1. ((every.d $S$) (be.v (= (a.d $P$)))) | Assumption |
| 2. ((some $S$) (be.v (= (a.d $M$)))) | Assumption |
| 3. ((some $S$)$^{+}$ (be.v (= (a.d $M$)))) | Polarity marking, 2. |
| 4. ((some $P$) (be.v (= (a.d $M$)))) | UMI, 1.,3. |
| 5. ((some $M$) (be.v (= (a.d $P$)))) | Conversion, 4. |

This could alternatively be proved by contradiction without the use of the equivalent of the Negation Rule, where "Not every $S$ is a $P$" becomes negated to "every $S$ is a $P$", after which, applying UMI and Conversion leads to a contradiction.

# Appendix C Transition Parser Appendices

## C.1 Full ULF Alignment Details

The ULF-English alignment system takes into account the similarity of the English word to the ULF atom without the type extension, the similarity of the type extension with the POS tag, and the relative distance of the word and symbol in question.

Given a sentence $s = w_{1:n}$, which is tokenized, $t_{1:n}$, lemmatized, $l_{1:n}$, and POS tagged, $p_{1:n}$, a set of symbols that are never aligned $S_u$, and a list of ULF atoms $a_{1:m}$, which can be broken up into the base stems, $b_{1:m}$, and suffix extensions, $e_{1:m}$, in order of appearance in the formula (i.e. DFS preorder traversal), the word/atom similarity is defined using the following formulas.

$$\begin{aligned}\text{Sim}(w, a) = &\max(\text{Olap}(t, b), \text{Olap}(l, b)) \\ &+ 0.5 * (\text{Olap}(p, e) + (1 - |\text{RL}(w, n) - \text{RL}(a, m)|))\end{aligned}$$

where token overlap, Olap, is defined as

$$\text{Olap}(x, y) = \frac{2 * |\text{MaxSharedSubstr}(x, y)|}{|x| + |y|}$$

and relative location RL is defined as

$$\text{RL}(x, n) = \frac{\text{IndexOf}(x)}{n}$$

Next, in order of $\text{Sim}(w, a)$, we consider each word-atom pair, $(w_i, a_i)$, $1 \leq i \leq n$ until $\text{Sim}(w, a) <$

MinSim, where MinSim is set to $1.0$, based on cursory results. We further disregard any alignments that include an atom which shouldn't be aligned ($a_i$ s.t. $a_i \in S_u$). We assume that spans of words align to connected subgraphs, so we cannot accept all word-atom pairs. An word-atom pair, $(w_i, a_i)$, is accepted into the set of token alignments, $A_t$, if and only if the following conditions are met:

1. $w_i$ has no alignments or $a_i$ is connected to an atom, $a'$, that is already aligned to $w_i$.
2. $a_i$ is not in any other alignment or $w_i$ is adjacent to another, $w'$ which is already aligned to $a_i$.

The token-level (word-atom) alignment, $A_t$, is then converted to connected (span-subgraph) alignment, $A$. This is done with the following algorithm:

1. For every atom $a_i$ in one of the aligned pairs of $A_t$, merge all of the words aligned to $a_i$ into a single span, $s_i$. During the initial alignment, we ensured that these words would form a span.
2. Merge all overlapping spans into single spans and collect the set of atoms that are aligned to each of these spans into a subgraph.[1] These collected subgraphs will be connected because we ensured that for any word the nodes that it is aligned to forms a connected subgraph.

## C.2 RoBERTa Handling Details

Except for RoBERTa, all other embeddings are fetched from their corresponding learned embedding lookup tables. RoBERTa uses OpenAI GPT-2 tokenizer for the input sequence and segments words into subwords prior to generating embeddings, which means one input word may correspond to multiple hidden states of RoBERTa. In order to accurately use these hidden states to represent each word, we apply an average pooling function to the outputs of RoBERTa according to the alignments between the original and GPT-2 tokenized sequences.

[1] This can be done in $\mathcal{O}(n \log n)$ time by sorting the spans, then doing a single pass of merging overlapping elements.

## C.3 Full Tables

Tables of the full set of raw results and parameters are presented in this section. Table C.1 shows the ablations on the model without decoding constraints. This is the basis of Figure 4.5 in the main text. Table C.2 shows the performance change with the lexicon constraint and Table C.3 shows the performance change with the composition constraint. These tables are the basis of Table 4.2 in the main text. Our experiments with the lexicon constraint were more extensive since the type constraint takes considerably longer to run due to requiring a larger beam size and more computational overhead. Table C.5 presents all of the model parameters in our experiments.

| **Ablation** | SEMBLEU | | EL-SMATCH | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | F1 | | Precision | | Recall | |
| | Dev | Test | Dev | Test | Dev | Test | Dev | Test |
| Full | $46.4 \pm 1.4$ | $\mathbf{47.4} \pm 1.3$ | $58.4 \pm 0.7$ | $\mathbf{59.8} \pm 1.0$ | $59.1 \pm 1.1$ | $\mathbf{60.7} \pm 1.5$ | $57.8 \pm 0.5$ | $\mathbf{59.0} \pm 0.7$ |
| -RoBERTa | $45.5 \pm 2.4$ | $47.2 \pm 1.7$ | $58.3 \pm 1.4$ | $59.3 \pm 1.0$ | $59.1 \pm 1.6$ | $60.5 \pm 1.1$ | $57.5 \pm 1.2$ | $58.3 \pm 0.9$ |
| -CharCNN | $46.4 \pm 1.0$ | $46.9 \pm 0.7$ | $58.8 \pm 0.8$ | $59.3 \pm 0.4$ | $59.4 \pm 1.3$ | $60.1 \pm 0.5$ | $58.1 \pm 0.6$ | $58.5 \pm 0.5$ |
| -$e_f(C)$ Feats | $47.0 \pm 1.2$ | $46.6 \pm 1.2$ | $58.6 \pm 0.5$ | $58.8 \pm 1.1$ | $60.4 \pm 1.2$ | $60.2 \pm 1.1$ | $56.9 \pm 0.4$ | $57.4 \pm 1.2$ |
| -POS | $43.8 \pm 1.7$ | $45.1 \pm 1.2$ | $56.9 \pm 1.1$ | $58.3 \pm 1.1$ | $56.8 \pm 1.0$ | $58.7 \pm 1.1$ | $56.9 \pm 1.2$ | $57.9 \pm 1.2$ |
| -GloVe | $43.2 \pm 1.8$ | $44.3 \pm 1.2$ | $56.6 \pm 1.0$ | $57.1 \pm 0.9$ | $56.9 \pm 2.7$ | $58.3 \pm 2.2$ | $56.4 \pm 1.7$ | $56.1 \pm 2.2$ |

Table C.1: Ablation results without decoding constraints, mean and standard deviation of 5 runs.

| **Ablation** | SEMBLEU | | EL-SMATCH | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | F1 | | Precision | | Recall | |
| | Dev | Test | Dev | Test | Dev | Test | Dev | Test |
| Full | $47.3 \pm 0.6$ | $46.2 \pm 0.3$ | $56.3 \pm 0.7$ | $57.5 \pm 0.8$ | $60.2 \pm 0.5$ | $61.5 \pm 1.2$ | $52.9 \pm 0.9$ | $54.1 \pm 1.5$ |
| $\Delta\bar{x}$ | | -1.2 | | -2.3 | | +0.8 | | -4.9 |
| -RoBERTa | $46.6 \pm 1.3$ | $46.9 \pm 0.6$ | $56.1 \pm 0.6$ | $57.8 \pm 0.4$ | $60.0 \pm 0.7$ | $60.5 \pm 0.9$ | $52.6 \pm 0.6$ | $55.3 \pm 0.5$ |
| -CharCNN | $45.8 \pm 2.3$ | $45.5 \pm 2.5$ | $56.1 \pm 1.4$ | $56.9 \pm 1.1$ | $59.3 \pm 2.4$ | $59.6 \pm 1.8$ | $53.3 \pm 1.1$ | $54.5 \pm 1.5$ |
| -$e_f(C)$ Feats | $45.9 \pm 1.5$ | $45.6 \pm 0.9$ | $56.5 \pm 0.6$ | $57.0 \pm 0.5$ | $62.0 \pm 0.8$ | $61.4 \pm 0.6$ | $52.0 \pm 1.1$ | $53.3 \pm 0.5$ |
| -POS | $44.1 \pm 2.0$ | $44.5 \pm 0.9$ | $55.3 \pm 0.2$ | $56.6 \pm 0.7$ | $58.5 \pm 2.2$ | $60.4 \pm 0.8$ | $52.6 \pm 2.3$ | $53.2 \pm 1.4$ |
| -GloVe | $46.1 \pm 1.1$ | $45.4 \pm 1.4$ | $55.9 \pm 0.9$ | $57.0 \pm 0.6$ | $59.5 \pm 1.5$ | $60.3 \pm 0.8$ | $52.7 \pm 1.0$ | $54.0 \pm 0.7$ |

Table C.2: Ablation results with the lexicon constraint, mean and standard deviation of 5 runs. $\Delta\bar{x}$ is the difference in the mean score between the test set results of the model with the lexicon constraint and without, i.e. Table C.1. We only list this for the full model, but the pattern of higher precision but lower scores on other metrics generally holds for the other variants as well.

| **Ablation** | SEMBLEU | | EL-SMATCH | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | F1 | | Precision | | Recall | |
| | Dev | Test | Dev | Test | Dev | Test | Dev | Test |
| Full | 38.3 ± 2.3 | 40.0 ± 1.4 | 54.2 ± 1.2 | 55.8 ± 1.2 | 57.6 ± 1.0 | 59.1 ± 1.2 | 51.1 ± 1.5 | 52.8 ± 1.4 |
| $\Delta\bar{x}$ | | -7.4 | | -4.0 | | -1.6 | | -6.2 |

Table C.3: Ablation results with the type composition constraint, mean and standard deviation of 5 runs. $\Delta\bar{x}$ is the difference in the mean score between the test set results of the model with the type constraint and without, i.e. Table C.1. We only ran the full model for this test because this constraint takes much longer to run.

| **Model** | Fragments/Sentence | |
|---|---|---|
| | $\alpha$ | $\tau$ |
| Full | 1.4 | 2.9 |
| -CharCNN | 1.1 | 3.5 |
| -$e_f(C)$ Feats | 1.4 | 3.9 |
| -POS | 1.5 | 3.2 |
| $\bar{x}$ | 1.4 | 3.4 |

Table C.4: Fragments per sentence on the test set decoding results for a subset of the ablated lexicon-constrained models (Table C.2). $\alpha$ is the original model and $\tau$ is with the type composition constraint.

| | |
|---|---|
| **GloVe.840B.300d embeddings** | |
| dim | 300 |
| **RoBERTa embeddings** | |
| source | RoBERTa-Base |
| dim | 768 |
| **POS tag embeddings** | |
| dim | 100 |
| **Lemma embeddings** | |
| dim | 100 |
| **CharCNN** | |
| num_filters | 100 |
| ngram_filter_sizes | [3] |
| **Action embeddings** | |
| dim | 100 |
| **Transition system feature embeddings** | |
| dim | 25 |
| **Word encoder** | |
| hidden_size | 256 |
| num_layers | 3 |
| **Symbol encoder** | |
| hidden_size | 128 |
| num_layers | 2 |
| **Action decoder** | |
| hidden_size | 256 |
| num_layers | 2 |
| **MLP decoder** | |
| hidden_size | 256 |
| activation_function | ReLU |
| num_layers | 1 |
| **Optimizer** | |
| type | ADAM |
| learning_rate | 0.001 |
| max_grad_norm | 5.0 |
| dropout | 0.33 |
| num_epochs | 25 |
| **Beam size** | |
| without type composition filtering | 3 |
| with type composition filtering | 10 |
| **Vocabulary** | |
| word_encoder_vocab_ size | 9200 |
| symbol_encoder_vocab_ size | 7300 |
| **Batch size** | 32 |

Table C.5: Default model parameters.

## C.4 Parse Examples

Figure C.1 shows six parse examples of our parser and the GS parser in reference to the gold standard. Generally, we find that our parser does much better on node generation for nodes that correspond to an input word. For example, the GS parser on example 1 uses *(plur *s)* for the word "speech" and *iron.n* for the words "silver" and "silence". This isn't to say that our parser doesn't make mistakes. But the mistakes are not as open-ended. For example, our parser mistakenly annotates "silver" as a noun in example 1 when in fact it should be an adjective (compared against "golden"). The GS parser seems to pick the closest word in its vocabulary, which is generated from the training set and closed. This leads to strange annotations like *iron.n* for the word "silence". If there is nothing close available, then it can derail the entire parse. In example 4, the GS parser is unable to find a node label for the word "device" which derails the parse to generate *(mod-n (mod-n man.n) (mod-n man.n iron.n) mod-n mod-n)* for the text span "device is attached firmly to the ceiling".

This isn't to say that the GS parser always performs worse than our parser. When it comes to words that are elided (*{you}.pro* in example 4), nodes generated from multiple words (*had_better.aux-s* in example 3), or logical symbols unassociated with a particular word (*multi-sent* in example 6), the GS parser consistently performs better than our parser. Our parser has no special mechanism for these handling these cases and prefers to avoid generating node labels without an anchoring word.

A common mistake by our parser seems to be nested reifiers, which is not possible in the EL type system (e.g. *(to (ka come.v))* in example 5 and *(to (ka (show.v ..)))* in example 6). Other common mistakes that could be fixed by type coherence enforcement is mistakenly shifting a term into a modifier (e.g. *(adv-a (to ...))* in example 6). In the EL type system only predicates can be shifted into modifiers.

1. *"Speech is silver but silence is golden."*
   **Gold:** (((k speech.n) ((pres be.v) silver.a)) but.cc ((k silence.n) ((pres be.v) golden.a)))
   **Ours:** (((k speech.n) ((pres be.v) silver.n)) (k silence.n) ((pres be.v) golden.a))
   **GS:** (((k (plur *s)) ((pres be.v) (= (k iron.n)))) but.cc ((k iron.n) ((pres be.v) =)))
2. *"You neglected to tell me to buy bread."*
   **Gold:** (you.pro ((past neglect.v) (to (tell.v me.pro (to (buy.v (k bread.n)))))))
   **Ours:** (you.pro ((past neglect.v) (adv-e (to (tell.v me.pro (to (buy.v (k bread.n)))))))
   **GS:** (you.pro ((past fail.v) (to (tell.v me.pro {ref}.pro))))
3. *"You'd better knuckle down to work."*
   **Gold:** (you.pro ((pres had_better.aux-s) (knuckle.v down.adv-a (to work.v))))
   **Ours:** (you.pro ((pres would.aux-s) (knuckle.v down.a (adv-a (to.p work.v)))))
   **GS:** (you.pro ((pres had_better.aux-s) (go.v (to.p-arg (k work.n)) (adv-a (to.p (ka work.v))))))
4. *"Make sure that the device is attached firmly to the ceiling."*
   **Gold:** ({you}.pro ((pres make.v) sure.a
   (that ((the.d device.n)
   ((pres (pasv attach.v)) firmly.adv-a (to.p-arg (the.d ceiling.n)))))))
   **Ours:** ( ((pres make.v) sure.a that.pro (tht
   ((the.d device.n) ((pres be.v) (k (n+preds attach.v (to.p-arg ceiling.n))))))))
   **GS:** (({you}.pro ((pres make.v) (sure.a
   (that (the.d (mod-n (mod-n man.n) (mod-n man.n iron.n) mod-n mod-n)))))) !)
5. *"Can't I persuade you to come?"*
   **Gold:** (((pres can.aux-v) not i.pro (persuade.v you.pro (to come.v)) ?)
   **Ours:** (sub ((pres can.aux-v) not i.pro (persuade.v you.pro (to (ka come.v)) ?)))
   **GS:** (((pres can.aux-v) not i.pro ( come.v (to come.v) you.pro)) ?)
6. *"Look carefully. I'm going to show you how it's done."*
   **Gold:** (multi-sent (({you}.pro ((pres look.v) carefully.adv-a)) !)
   (i.pro ((pres be-going-to.aux-v)
   (show.v you.pro (ans-to (sub how.pq (it.pro ((pres (pasv do.v)) *h))))))))
   **Ours:** ( ((pres look.v) carefully.adv-a)
   (tht (i.pro ((pres be.v) (go.v
   (adv-a (to (ka (show.v you.pro (sub how.pq (it.pro ((pres be.v) do.n ))))))))))))
   **GS:** (multi-sent (({you}.pro ((pres be.v) you.pro fine.a)) !)
   (i.pro ((pres be.aux-v) (go.v (to (do.v you.pro *h))))))

Figure C.1: Several parse examples comparing behavior of our parser with the stronger baseline, the GS parser (Cai and Lam, 2020). Errors are written in red. If something from the gold parse is omitted, a red highlighted block marks the location.

ProQuest Number: 28861049

INFORMATION TO ALL USERS
The quality and completeness of this reproduction is dependent on the quality
and completeness of the copy made available to ProQuest.



Distributed by ProQuest LLC (2022).
Copyright of the Dissertation is held by the Author unless otherwise noted.

This work may be used in accordance with the terms of the Creative Commons license
or other rights statement, as indicated in the copyright statement or in the metadata
associated with this work. Unless otherwise specified in the copyright statement
or the metadata, all rights are reserved by the copyright holder.

This work is protected against unauthorized copying under Title 17,
United States Code and other applicable copyright laws.

Microform Edition where available © ProQuest LLC. No reproduction or digitization
of the Microform Edition is authorized without permission of ProQuest LLC.

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346 USA