Deep Linguistic Processing for Spoken Dialogue Systems

James Allen
Department of Computer Science
University of Rochester
james@cs.rochester.edu

Myroslava Dzikovska
ICCS-HCRC
University of Edinburgh
mdzikovs@inf.ed.ac.uk

Mehdi Manshadi
Department of Computer Science
University of Rochester
mehdih@cs.rochester.edu

Mary Swift
Department of Computer Science
University of Rochester
swift@cs.rochester.edu

Abstract

We describe a framework for deep linguistic processing for natural language understanding in task-oriented spoken dialogue systems. The goal is to create domain-general processing techniques that can be shared across all domains and dialogue tasks, combined with domain-specific optimization based on an ontology mapping from the generic LF to the application ontology. This framework has been tested in six domains that involve tasks such as interactive planning, coordination operations, tutoring, and learning.

1 Introduction

Deep linguistic processing is essential for spoken dialogue systems designed to collaborate with users to perform collaborative tasks. We describe the TRIPS natural language understanding system, which is designed for this purpose. As we develop the system, we are constantly balancing two competing needs: (1) deep semantic accuracy: the need to produce the semantically and pragmatically deep interpretations for a specific application; and (2) portability: the need to reuse our grammar, lexicon and discourse interpretation processes across domains.

We work to accomplish portability by using a multi-level representation. The central components are all based on domain general representations, including a linguistically based detailed semantic representation (the Logical Form, or LF), illocutionary acts, and a collaborative problem-solving model. Each application then involves using a domain-specific ontology and reasoning components.

The generic LF is linked to the domain-specific representations by a set of ontology mapping rules that must be defined for each domain. Once the ontology mapping is defined, we then can automatically specialize the generic grammar to use the stronger semantic restrictions that arise from the specific domain. In this paper we mainly focus on the generic components for deep processing. The work on ontology mapping and rapid grammar adaptation is described elsewhere (Dzikovska et al. 2003; forthcoming).

2 Parsing for deep linguistic processing

The parser uses a broad coverage, domain-independent lexicon and grammar to produce the LF. The LF is a flat, unscoped representation that includes surface speech act analysis, dependency information, word senses (semantic types) with semantic roles derived from the domain-independent language ontology, tense, aspect, modality, and implicit pronouns. The LF supports fragment and ellipsis interpretation, discussed in Section 5.2

2.1 Semantic Lexicon

The content of our semantic representation comes from a domain-independent ontology linked to a domain-independent lexicon. Our syntax relies on a frame-based design in the LF ontology, a common representation in semantic lexicons (Baker et al., 1998, Kipper et al., 2000). The LF type hierarchy is influenced by argument structure, but provides a more detailed level of semantic analysis than found in most broad coverage parsers as it distinguishes senses even if the senses take the same argument structure, and may collapse lexical entries with different argument structures to the same sense. As a very simple example, the generic lexicon includes the senses for the verb *take* shown

Proceedings of the 5th Workshop on Important Unresolved Matters, pages 49–56,
Ann Arbor, June 2005. ©2005 Association for Computational Linguistics

in Figure 1. Our generic senses have been inspired by FrameNet (Baker et al., 1998).

In addition, types are augmented with semantic features derived from EuroWordNet (Vossen et al., 1997) and extended. These are used to provide selectional restrictions, similar to VerbNet (Kipper et al., 2000). The constraints are intentionally weak, excluding utterances unsuitable in most contexts (*the idea slept*) but not attempting to eliminate borderline combinations.

The generic selectional restrictions are effective in improving overall parsing accuracy, while remaining valid across multiple domains. An evaluation with an earlier version of the grammar showed that if generic selectional restrictions were removed, full sentence semantic accuracy decreased from 77.8% to 62.6% in an emergency rescue domain, and from 67.9 to 52.5% in a medical domain (using the same versions of grammar and lexicon) (Dzikovska, 2004).

The current version of our generic lexicon contains approximately 6400 entries (excluding morphological variants), and the current language ontology has 950 concepts. The lexicon can be supplemented by searching large-scale lexical resources such as WordNet (Fellbaum, 1998) and Comlex (Grisham et al., 1994). If an unknown word is encountered, an underspecified entry is generated on the fly. The entry incorporates as much information from the resource as possible, such as part of speech and syntactic frame. It is assigned an underspecified semantic classification based on correspondences between our language ontology and WordNet synsets.

2.2 Grammar

The grammar is context-free, augmented with feature structures and feature unification, motivated from X-bar theory, drawing on principles from GPSG (e.g., head and foot features) and HPSG. A detailed description of an early non-lexicalized version of the formalism is in (Allen, 1995). Like HPSG, our grammar is strongly lexicalized, with the lexical features defining arguments and complement structures for head words. Unlike HPSG, however, the features are not typed and rather than multiple inheritance, the parser supports a set of orthogonal single inheritance hierarchies to capture different syntactic and semantic properties. Structural variants such as passives, dative shifts, gerunds, and so on are captured in the context-free rule base. The grammar has broad coverage of spoken English, supporting a wide range of conversational constructs. It also directly encodes conventional conversational acts, including standard surface speech acts such as inform, request and question, as well as acknowledgments, acceptances, rejections, apologies, greetings, corrections, and other speech acts common in conversation.

CONSUME	*Take an aspirin*
MOVE	*Take it to the store*
ACQUIRE	*Take a picture*
SELECT	*I'll take that one*
COMPATIBLE WITH	*The projector takes 100 volts*
TAKE-TIME	*It took three hours*

Figure 1: Some generic senses of take in lexicon

To support having both a broad domain-general grammar and the ability to produce deep domain-specific semantic representations, the semantic knowledge is captured in three distinct layers (Figure 2), which are compiled together before parsing to create efficient domain-specific interpretation. The first level is primarily encoded in the grammar, and defines an interpretation of the utterance in terms of generic grammatical relations. The second is encoded in the lexicon and defines an interpretation in terms of a generic language-based ontology and generic roles. The third is encoded by a set of ontology-mapping rules that are defined for each domain, and defines an interpretation in terms of the target application ontology. While these levels are defined separately, the parser can produce all three levels simultaneously, and exploit domain-specific semantic restrictions to simultaneously improve semantic accuracy and parsing efficiency. In this paper we focus on the middle level, the generic LF.

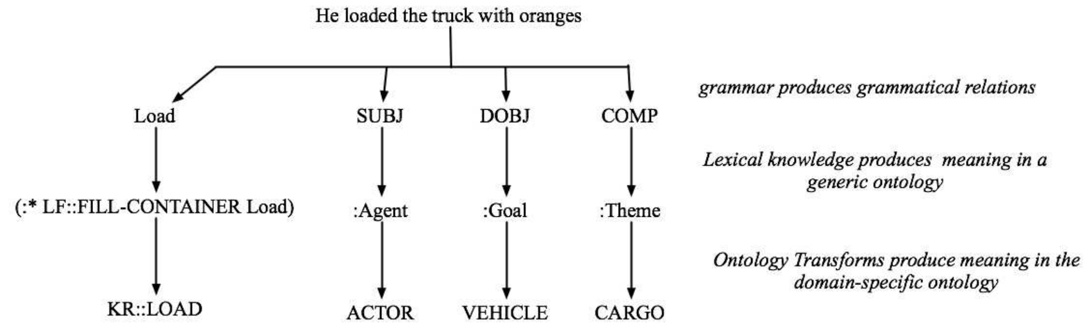

Figure 2: The Levels of Representation computed by the Parser

The rules in the grammar are weighted, and weights are combined, similar to how probabilities are computed in a PCFG. The weights, however, are not strictly probabilities (e.g., it is possible to have weights greater than 1); rather, they encode structural preferences. The parser operates in a best-first manner and as long as weights never exceed 1.0, is guaranteed to find the highest weighted parse first. If weights are allowed to exceed 1.0, then the parser becomes more "depth-first" and it is possible to "garden-path" and find globally suboptimal solutions first, although eventually all interpretations can still be found.

The grammar used in all our applications uses these hand-tuned rule weights, which have proven to work relatively well across domains. We do not use a statistical parser based on a trained corpus because in most dialogue-system projects, sufficient amounts of training data are not available and would be too time consuming to collect. In the one domain in which we have a reasonable amount of training data (about 9300 utterances), we experimented with a PCFG using trained probabilities with the Collins algorithm, but were not able to improve on the hand-tuned preferences in overall performance (Elsner et al., 2005).

Figure 3 summarizes some of the most important preferences encoded in our rule weights. Because we are dealing with speech, which is often ungrammatical and fragmented, the grammar includes "robust" rules (e.g., allowing dropped determiners) that would not be found in a grammar of written English.

3 The Logical Form Language

The logical form language captures a domain-independent semantic representation of the utterance. As shown later in this paper, it can be seen as a variant of MRS (Copestake et al., 2006) but is expressed in a frame-like notation rather than predicate calculus. In addition, it has a relatively simple method of computing possible quantifier scoping, drawing from the approaches by (Hobbs & Shieber, 1987) and (Alshawi, 1990).

A logical form is set of terms that can be viewed as a rooted graph with each term being a node identified by a unique ID (the variable). There are three types of terms. The first corresponds to generalized quantifiers, and is on the form (<quant> <id> <type> <modifiers>*). As a simple example, the NP *Every dog* would be captured by the term (Every d1 DOG). The second type of term is the propositional term, which is represented in a neo-Davidsonian representation (e.g., Parsons, 1990) using reified events and properties. It has the form (F <id> <type> <arguments>*). The propositional terms produced from *Every dog hates a cat* would be (F h1 HATE :Experiencer d1 :Theme c1). The third type of term is the speech act, which has the same form as propositional terms except for the initial indicator SA identifying it as a performed speech act. The speech act for Every dog hates a cat would be (SA sa1 INFORM :content h1). Putting this all together, we get the following (condensed) LF representation from the parser for *Every large dog hates a cat* (shown in graphical

Prefer
- Interpretations without gaps to those with gaps
- Subcategorized interpretations over adjuncts
- Right attachment of PPs and adverbials
- Fully specified constituents over those with dropped or "implicit" arguments
- Adjectival modification over noun-noun modification
- Standard rules over "robust" rules

Figure 3: Some Key Preferences used in Parsing

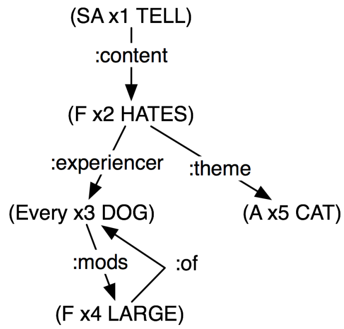

Figure 4: The LF in graphical form

form in Figure 4).

(SA x1 TELL :content x2)
(F x2 HATE :experience x3 :theme x5)
(Every x3 DOG :mods (x4))
(F x4 LARGE :of x3)
(A x5 CAT)

4 Comparison of LF and MRS

Minimal Recursion Semantics (MRS) (Copestake et al. 2006) is a semantic formalism which has been widely adopted in the last several years. This has motivated some research on how this formalism compares to some traditional semantic formalisms. For example, Fuchss et al. (2004) formally show that the translation from MRS to Dominance Constraints is feasible. We have also found that MRS is very similar to LF in its descriptive power. In fact, we can convert every LF to an equivalent MRS structure with a simple algorithm.

First, consider the sentence *Every dog hates a cat*. Figure 5 shows the LF and MRS representations for this sentence.

	h1 : Every(x, h2, h3)
(SA :content v1)	h4: Dog(x)
(F v1 Hate :Experiencer x :Theme y)	h5: A(y, h6, h7)
	h8: Cat(y)
(Every x Dog)	h9: hate(x, y)
(A y Cat)	{h0 =q h9, h2 =q h4, h6 =q h8}

Figure 5: The LF (left) and MRS (right) representations for the sentence "Every dog hates a cat."

The first step toward converting LF to MRS is to express LF terms as n-ary relationships. For example we express the LF term *(F v1 Hate :Experiencer x :Theme y)* as *Hate(x, y)*. For quantifier terms, we break the LF term into two relations: one for the quantifier itself and one for the restriction. For example *(Every x Dog)* is converted to *Every(x) and Dog(x)*.

There is a small change in the conversion procedure when the sentence contains some modifiers. Consider the modifier *large* in the sentence *Every large dog hates a cat.* In the LF, we bring the modifier in the term which defines the semantic head, using a *:MODS* slot. In the MRS, however, modifiers are separate EPs labeled with same handle as the head's. To cover this, for each LF term T which has a (*:MODS* v_k) slot, and the LF term T1 which defines the variable v_k, we assign the same handle to both T and T1. For example for the terms *(F x Dog :MODS v2)* and *(F v2 Large :OF x)*, we assign the same handle to both *Dog(x)* and *Large(x)*. Similar approach applies when the modifier itself is a scopal term, such as in the sentence *Every cat in a room sleeps*. Figure 7 shows LF and MRS representations for this sentence. Figure 8, summarizes all these steps as an algorithm which takes a LF representation as the input and generates its equivalent MRS.

There is a small change in the conversion procedure when the sentence contains some modifiers. Consider the modifier *large* in the sentence *Every large dog hates a cat.* In the LF, we bring the modifier in the term which defines the semantic head, using a *:MODS* slot. In the MRS, however, modifiers are separate EPs labeled with same handle as the head's. To cover this, for each LF term T which has a (*:MODS* v_k) slot, and the LF term T1 which defines the variable v_k, we assign the same handle to both T and T1. For example for the terms *(F x Dog :MODS v2)* and *(F v2 Large :OF x)*, we assign the same handle to both *Dog(x)* and *Large(x)*. Similar approach applies when the modifier itself is a scopal term, such as in the sentence *Every cat in a room sleeps*. Figure 7 shows LF and MRS representations for this sentence. Figure 8, summarizes all these steps as an algorithm which takes a LF representation as the input and generates its equivalent MRS.

The next step is to bring handles into the repre-

Step 1:	Step 2:	
Hate(x,y)	h1: Hate(x, y)	h0 =
Every(x), Dog(x)	h2: Every(x, h6, h7)	h6 =
	h3: Dog(x)	
A(y), Cat(y)	h4: A(y, h8, h9)	h8 =
	h5: Cat(y)	

Figure 6: The steps of converting the LF for "Every cat hates a cat" to its MRS representation

sentation. First, we assign a different handle to each term. Then, for each quantifier term such as *Every(x)*, we add two handles as the arguments of the relation: one for the restriction and one for the body as in *h2: Every(x, h6, h7)*. Finally, we add the handle constraints to the MRS. We have two types of handle constraint. The first type comes from the restriction of each quantifier. We add a qeq relationship between the restriction handle argument of the quantifier term and the handle of the actual restriction term. The second type of constraint is the qeq relationship which defines the top handle of the MRS. The speech act term in every LF refers to a formula term as content (*:content* slot), which is actually the heart of the LF. We build a qeq relationship between h0 (the top handle) and the handle of this formula term. Figure 6 shows the effect of applying these steps to the above example.

(SA :content v1)	*h1 : Every(x, h2, h3)*
(F v1 Sleep :Theme x)	*h4: Cat(x)*
(Every x Cat :MODS v2)	*h4: In(x,y)*
(F v2 In :OF x :VAL y)	*h5: A(y, h6, h7)*
(A y Room)	*h8: Room(y)*
	h9: Sleep(x)
	{h0 =q h9, h2 =q h4, h6 =q h8}

Figure 7: The LF and MRS representations for the sentence "Every cat in a room sleeps."

Another interesting issue about these two formalisms is that the effect of applying the simple scoping algorithms referred in section 3 to generate all possible interpretations of a LF is the same as applying MRS axioms and handle constraints to generate all scope-resolved MRSs. For instance, the example in (Copestake et al. 2006), *Every nephew of some famous politician saw a pony* has the same 5 interpretations using either approach.

As the last point here, we need to mention that the algorithm in Figure 8 does not consider fixed-scopal terms such as scopal adverbials or negation. However, we believe that the framework itself is able to support these types of scopal term and with a small modification, the scoping algorithm will work well in assigning different possible interpretations. We leave the full discussion about these details as well as the detailed proof of the other claims we made here to another paper.

5 Generic Discourse Interpretation

With a generic semantic representation, we can then define generic discourse processing capabilities that can be used in any application. All of these methods have a corresponding capability at the domain-specific level for an application, but we will not discuss this further here. We also do not discuss the support for language generation which uses the same discourse context.

There are three core discourse interpretation capabilities that the system provides: reference resolution, ellipsis processing, and speech act interpretation. All our different dialog systems use the same discourse processing, whether the task involves collaborative problem solving, learning from instruction or automated tutoring.

5.1 Reference Resolution

Our domain-independent representation supports reference resolution in two ways. First, the quantifiers and dependency structure extracted from the sentence allow for implementing reference resolution algorithms based on extracted syntactic features. The system uses different strategies for re-

Input: LF, list of all logical form terms,
Output: MRS structure
1. Initialize MRS to <h0, EP={}, C={}>
2. Find the SPEECHACT term T in the form (SA … :content V_i) and the formula term T1 which defines variable V_i. Define a new handle h_j and assign it to T1; then add $h_0 =_q h_j$ to C
3. While LF is not empty, remove a term T from LF
    3.1. If T is a formula term in the form
        (F V_i $Rel_1(\ldots)$ $Rel_2(\ldots)$ … $Rel_n(\ldots)$:MODS(u_1) :MODS(u_2) … MODS(u_m))
        3.1.1. If T has not already been assigned a handle h_j, define a new handle h_j and assign it to T
        3.1.2. Add h_j:$Rel_1(\ldots)$, h_j:$Rel_2(\ldots)$, … to EP
        3.1.3. For each variable u_i, find the formula term T_i which defines u_i and assign h_j to T_i
    3.2. If T is a quantifier term T in the form (Q V_i $Rel_1(\ldots)$ … $Rel_n(\ldots)$:MODS(u_1) … MODS(u_m))
        3.2.1. Define new handles h_j, h_i, h_l and h_m
        3.2.2. Add h_j: Q(v_i, h_i, h_l), h_m:$Rel_1(\ldots)$, h_m:$Rel_2(\ldots)$, …. to EP and $h_i =_q h_m$ to C
        3.2.3. For each variable u_i, find the formula term T_i which defines u_i and assign h_m to T_i

Figure 8: The LF-MRS conversion algorithm

solving each type of referring expression along the lines described in (Byron, 2002).

Second, domain-independent semantic information helps greatly in resolving pronouns and definite descriptions. The general capability provided for resolving referring expressions is to search through the discourse history for the most recent entity that matches the semantic requirements, where recency within an utterance may be reordered to reflect focusing heuristics (Tetreault, 2001). For definite descriptions, the semantic information required is explicit in the lexicon. For pronouns, the parser can often compute semantic features from verb argument restrictions. For instance, the pronoun *it* carries little semantic information by itself, but in the utterance *Eat it* we know we are looking for an edible object. This simple technique performs well in practice.

Because of the knowledge in the lexicon for role nouns such as *author*, we can also handle simple bridging reference. Consider the discourse fragment *That book came from the library. The author* .... The semantic representation of *the author* includes its implicit argument, e.g., (The x1 AUTHOR :of b1). Furthermore, the term b1 has the semantic feature INFO-CONTENT, which includes objects that “contain” information such as books, articles, songs, etc.., which allows the pronoun to correctly resolve via bridging to the book in the previous utterance.

5.2 Ellipsis

The parser produces a representation of fragmentary utterances similar to (Schlangen and Lascarides, 2003). The main difference is that instead of using a single underspecified *unknown_rel* predicate to resolve in discourse context, we use a speech act term as the underspecified relation, differentiating between a number of common relations such as acknowledgments, politeness expressions, noun phrases and underspecified predicates (PP, ADJP and VP fragments). The representations of the underspecified predicates also include an IMPRO in place of the unspecified argument.

We currently handle only a few key cases of ellipsis. The first is question/answer pairs. By retaining the logical form of the question in the discourse history, it is relatively easy to reconstruct the full content of short answers (e.g., in *Who ate the pizza? John?* the answer maps to the representation that John ate the pizza). In addition, we handle common follow-up questions (e.g., *Did John buy a book? How about a magazine?)* by performing a semantic closeness matching of the fragment into the previous utterance and substituting the most similar terms. The resulting term can then be used to update the context. This process is similar to the resolution process in (Schlangen and Lascarides, 2003), though the syntactic parallelism constraint is not checked. It could also be easily extended to cover other fragment types, as the grammar provides all the necessary information.

5.3 Speech Act Interpretation

The presence of domain-independent semantic classes allows us to encode a large set of these common conversational pattern independently of the application task and domain. These include rules to handle short answers to questions, acknowledgements and common politeness expressions, as well as common inferences such as interpreting *I need to do X* as *please do X*.

Given our focus on problem solving domains, we are generally interested in identifying more than just the illocutionary force of an utterance. For instance, in a domain for planning how to evacuate people off an island, the utterance *Can we remove the people by helicopter?* is not only ambiguous between being a true Y-N question or a suggestion of a course of action, but at the problem solving level it might intended to (1) introduce a new goal, (2) elaborate or extend the solution to the current problem, or (3) suggest a modification to an existing solution (e.g., moving them by truck). One can only choose between these readings using domain specific reasoning about the current task. The point here is that the interpretation rules are still generic across all domains and expressed using the generic LF, yet the interpretations produced are evaluated using domain-specific reasoning. This interleaving of generic interpretation and domain-specific reasoning is enabled by our ontology mappings.

Similarly, in tutoring domains students often phrase their answers as check questions. In an answer to the question *Which components are in a closed path*, the student may say *Is the bulb in 3 in a closed path?* The domain-independent representation is used to identify the surface form of this utterance as a yes-no question. The dialogue manager then formulates two hypotheses: that this is a hedged answer, or a real question. If a domain-specific tutoring component confirms the former hypothesis, the dialogue manager will proceed with verifying answer correctness and carrying on remediation as necessary. Otherwise (such as for *Is the bulb in 5 connected to a battery* in the same context), the utterance is a question that can be answered by querying the domain reasoner.

5.4 A Note on Generic Capabilities

A key point is that these generic discourse interpretation capabilities are enabled because of the detailed generic semantic interpretation produced by the parser. If the parser produced a more shallow representation, then the discourse interpretation techniques would be significantly degraded. On the other hand, if we developed a new representation for each domain, then we would have to rebuild all the discourse processing for the domain.

6 Evaluation

Our evaluation is aimed at assessing two main features of the grammar and lexicon: portability and accuracy. We use two main evaluation criteria: full sentence accuracy, that takes into account both syntactic and semantic accuracy of the system, and sense tagging accuracy, to demonstrate that the word senses included in the system can be distinguished with a combination of syntactic and domain-independent semantic information.

As a measure of the breadth of grammatical coverage of our system, we have evaluated our coverage on the CSLI LKB (Linguistic Knowledge Building) test suite (Copestake, 1999). The test suite contains approximately 1350 sentences, of which about 400 are ungrammatical. We use a full-sentence accuracy measure to evaluate our coverage, since this is the most meaningful measure in terms of what we require as parser output in our applications. For a sentence representation to be counted as correct by this measure, both the syntactic structure and the semantic representation must be correct, which includes the correct assignment of word senses, dependency relations among terms, and speech act type. Our current coverage for the diverse grammatical phenomena in the corpus is 64% full-sentence accuracy.

We also report the number of spanning parses found, because in our system there are cases in which the syntactic parse is correct, but an incorrect word sense may have been assigned, since we disambiguate senses using not only syntactic structure but also semantic features as selectional restrictions on arguments. For example, in *The manager interviewed Browne after working*, the parser assigns *working* the sense LF::FUNCTION, used with non-agentive subjects, instead of the correct sense for agentive subjects, LF::WORKING. For the grammatical utterances in the test suite, our parser found spanning parses for 80%.

While the ungrammatical sentences in the set are an important tool for constraining grammar output, our grammar is designed to find a reasonable interpretation for natural speech, which often is less than perfect. For example, we have low preference grammar rules that allow dropped subjects, missing determiners, and wrong subject verb agreement. In addition, utterances are often fragmentary, so even those without spanning parses may be considered correct. Our grammar allows all major constituents (NP, VP, ADJP, ADVP) as valid utterances. As a result, our system produces spanning parses for 46% of the “ungrammatical” utterances. We have not yet done a detailed error analysis.

As a measure of system portability to new domains, we have evaluated our system coverage on the ATIS (Airline Travel Information System) speech corpus, which we have never used before. For this evaluation, the proper names (cities, airports, airline companies) in the ATIS corpus were added to our lexicon, but no other development work was performed. We parsed 116 randomly selected test sentences and hand-checked the results using our full-sentence accuracy measure. Our baseline coverage of these utterances is 53% full-sentence semantic accuracy. Of the 55 utterances that were not completely correct, we found spanning parses for 36% (20). Reasons that spanning parses were marked as wrong include incorrect word senses (e.g., for *stop* in *I would like it to have a stop in Phoenix*) or PP-attachment. Reasons that no spanning parse was found include missing senses for existing words (e.g., *serve* as in *Does that flight serve dinner*).

7 Discussion

We presented a deep parser and semantic interpreter for use in dialogue systems. An important question to ask is how it compares to other existing formalisms. At present there is no easy way to make such comparison. One possible criterion is grammatical coverage. Looking at the grammar coverage/accuracy on the TSNLP suite that was used to evaluate the LINGO ERG grammar, our grammar demonstrates 80% coverage (number of spanning parses). The reported figure for LINGO ERG coverage of CSLI is 77% (Oepen, 1999), but this number has undoubtedly improved in the 9-year development period. For example, the current reported coverage figures on spoken dialogue corpora are close to 90% (Oepen et al., 2002).

However, the grammar coverage alone is not a satisfactory measure for a deep NLP system for use in practical applications, because the logical forms and therefore the capabilities of deep NLP systems differ significantly. A major distinguishing feature of our system is that the logical form it outputs uses semantically motivated word senses. LINGO ERG, in contrast, contains only syntactically motivated word senses. For example, the words *end* and *finish* are not related in any obvious way. This reflects a difference in underlying philosophy. LINGO ERG aims for linguistic precision, and as can be seen from our experiments, requiring the parser to select correct domain-independent word senses lowers accuracy.

Our system, however, is built with the goal of easy portability within the context of dialogue systems. The availability of word senses simplifies the design of domain-independent interpretation components, such as reference resolution and speech act interpretation components that use domain-independent syntactic and semantic information to encode conventional interpretation rules.

If the LINGO ERG grammar were to be put in a dialogue system that requires domain interpretation and reasoning, an additional lexical interpretation module would have to be developed to perform word sense disambiguation as well as interpretation, something that has not yet been done.

Acknowledgments

We thank 3 reviewers for helpful comments. This work was supported by NSF IIS-0328811, DARPA NBCHD30010 via subcontract to SRI #03-000223 and ONR N00014051004-3 and –8.

References

H. Alshawi. 1990. Resolving Quasi Logical Forms. *Computational Linguistics* 16(3):133-144.

W. Baker, C. Fillmore and J. B. Lowe. 1998. The Berkeley FrameNet Project. *COLING-ACL'98*, Montréal.

D. Byron. 2002. Resolving Pronominal Reference to Abstract Entities. *ACL-02*, Philadelphia.

A. Copestake. 1999. *The (New) LKB System*. CSLI.

A. Copestake, D. Flickinger, C. Pollard and I. Sag. 2006. Minimal Recursion Semantics: An Introduction. *Research on Language and Computation*, 3(4):281-332.

M. Dzikovska. 2004. *A Practical Semantic Representation for Natural Language Parsing*. Ph.D. Thesis, University of Rochester.

M. Dzikovska, J. Allen and M. Swift. Forthcoming. Linking Semantic and Knowledge Representations in a Multi-domain Dialogue System. *Journal of Logic and Computation*.

M. Dzikovska, J. Allen and M. Swift. 2003. Integrating Linguistic and Domain Knowledge for Spoken Dialogue Systems in Multiple Domains. *Workshop on Knowledge and Reasoning in Practical Dialogue Systems, IJCAI-2003,* Acapulco.

M. Elsner, M. Swift, J. Allen and D. Gildea. 2005. Online Statistics for a Unification-based Dialogue Parser. *IWPT05*, Vancouver.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

R. Fuchss, A. Koller, J. Niehren, S. Thater. 2004. Minimal Recursion Semantics as Dominance Constraints. *ACL-04*, Barcelona.

R. Grisham, C. Macleod and A. Meyers. 1994. Comlex Syntax: Building a Computational Lexicon. *COLING 94*, Kyoto.

J. Hobbs and S. Shieber. 1987. An Algorithm for Generating Quantifier Scopings. *Computational Linguistics* 13(1-2):47-63.

K. Kipper, H. T. Dang and M. Palmer. 2000. Class-based Construction of a Verb Lexicon. *AAAI-2000*.

S. Oepen, D. Flickinger, K. Toutanova and C. Manning. 2002. Lingo Redwoods: A Rich and Dynamic Treebank for HPSG. *First Workshop on Treebanks and Linguistic Theories* (TLT2002).

S. Oepen (1999). [incr tsdb()] User Manual. www.delph-in.net/itsdb/publications/manual.ps.gz.

T. Parsons. 1990. *Events in the Semantics of English. A Study in Subatomic Semantics*. MIT Press.

D. Schlangen and A. Lascarides 2003. The Interpretation of Non-Sentential Utterances in Dialogue. *SIGDIAL-03*, Sapporo.

J. Tetreault. 2001. A Corpus-Based Evaluation of Centering and Pronoun Resolution. *Computational Linguistics*. 27(4):507-520.

Vossen, P. (1997) EuroWordNet: A Multilingual Database for Information Retrieval. In *Proc. of the Delos workshop on Cross-language Information Retrieval*.