Linking Semantic and Knowledge Representations in a Multi-Domain Dialogue System

Myroslava O. Dzikovska[1], James F. Allen[2], Mary D. Swift[2]
[1]Human Communication Research Centre,
2 Buccleuch Place, Edinburgh EH8 9LW, United Kingdom
m.dzikovska@ed.ac.uk
[2] Department of Computer Science, University of Rochester,
Rochester, NY 14526
{james,swift}@cs.rochester.edu

Abstract

We describe a two-layer architecture for supporting semantic interpretation and domain reasoning in dialogue systems. Building systems that support both semantic interpretation and domain reasoning in a transparent and well-integrated manner is an unresolved problem because of the diverging requirements of the semantic representations used in contextual interpretation versus the knowledge representations used in domain reasoning. We propose an architecture that provides both portability and efficiency in natural language interpretation by maintaining separate semantic and domain knowledge representations, and integrating them via an ontology mapping procedure. The ontology mapping is used to obtain representations of utterances in a form most suitable for domain reasoners, and to automatically specialize the lexicon. The use of a linguistically motivated parser for producing semantic representations for complex natural language sentences facilitates building portable semantic interpretation components as well as connections with domain reasoners. Two evaluations demonstrate the effectiveness of our approach: we show that a small number of mapping rules is sufficient for customizing the generic semantic representation to a new domain, and that our automatic lexicon specialization technique improves parser speed and accuracy.

1 Introduction

This paper presents an architecture of a language interpretation system that integrates semantic interpretation and domain reasoning tasks in a portable and efficient way. Medium-scale dialogue systems for applications such as intelligent planning assistants or tutorial dialogue involve complex reasoning and planning tasks. Building such systems requires integrating two strands of current research: semantic interpretation and its connection to domain reasoning. We define semantic interpretation as the process of determining the meaning of an utterance in context, including tasks such as as speech act identification, reference resolution, fragment interpretation and discourse context update. Domain reasoning is the task of determining an appropriate system response to

the user's actions, which requires relating natural language to non-linguistic knowledge sources such as databases, reasoners and planners.

Effective language processing in dialogue systems involves the production of representations for natural language that can be used both in semantic interpretation and in efficient domain reasoning. Sophisticated semantic representations such as UDRS [24] or MRS [15] are supported by deep grammars such as LINGO ERG [14] or XLE [37]. However, these semantic representations are limited to features that can be reliably extracted from syntax, which presents difficulties for their application in domain reasoning tasks (see Section 2.2 for discussion). As a result, practical systems using these representations have been implemented only in small domains requiring fairly simple domain knowledge, such as appointment scheduling [30]. Dialogue systems that use more complex domain reasoners tend to rely on either on domain-specific grammars, or on wider-coverage syntactic grammars with domain-specific lexicons [42, 33, 17, 28]. However, the language components in such dialogue systems are difficult to port to new domains because of their domain-specific representations. They are also not easily extended to handle complex phenomena in semantic interpretation because the knowledge representation languages used for efficient reasoning are not well-equipped to represent the features needed for contextual interpretation, for example, underspecification or representations for context-dependent entities such as pronouns.

This division reflects a general distinction between linguistic and pragmatic knowledge and domain knowledge. Linguistic and pragmatic knowledge is generally domain-independent. Many algorithms for reference resolution [10], intention recognition [8], dialogue management [32] and fragment interpretation [44] are formulated using generic notions such as plans and resources. They may need to call on domain reasoners to verify that something is a resource or a plan in the given domain, but the core of the algorithm uses general knowledge about dialogue that does not change between applications. Domain reasoners, however, are built to encode knowledge and support efficient queries and inferences within a given domain. We address the problem of how to build parsers and grammars that support the diverging representation requirements imposed by those tasks.

We propose a parsing and interpretation system architecture that combines the benefits of a wide-coverage deep parser for semantic interpretation with customization techniques to support efficient domain reasoning. The parser builds representations that support semantic interpretation in multiple domains using a lexicon linked to a domain-general language ontology (constructed with the aid of several task-oriented dialogue corpora). The domain-general semantic representations are linked with knowledge representations in specific domains via an ontology mapping technique. Maintaining a domain-general grammar and lexicon facilitates building interpretation components that are easy to port across domains, while our ontology mappings provide customized connections to the domain knowledge representation. Moreover, we develop an automatic method of specializing the lexicon based on the ontology mappings that significantly improves parsing speed and accuracy for in-domain sentences, while maintaining portability and reusability in the grammar and lexicon.

We demonstrate the effectiveness of this approach by evaluating the performance of the same wide-coverage grammar and lexicon in two different domains. We show that the parser performs similarly in both domains, and that a small number of mapping rules

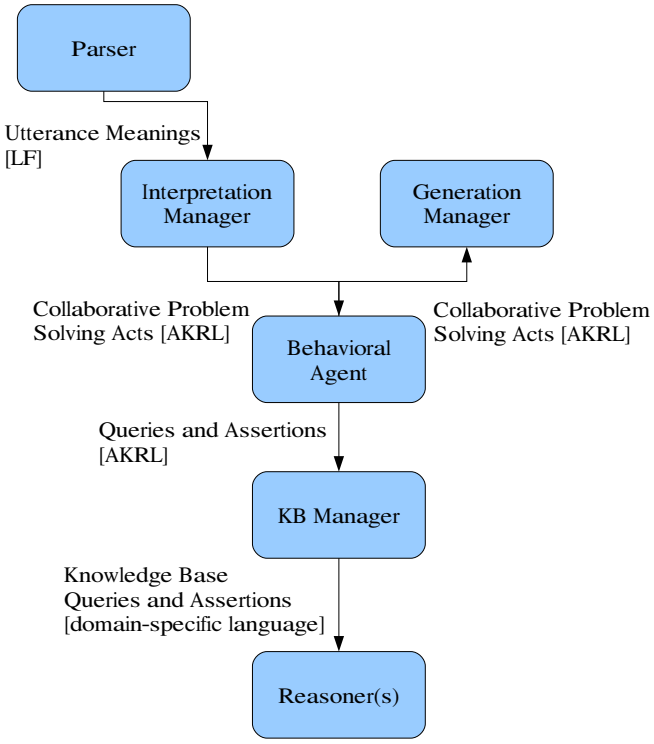

Figure 1: Knowledge representations in the TRIPS system architecture. Notation in square brackets indicates different representation languages used in the system, discussed in more detail in Section 3.

is sufficient for specialization to a new domain. We also show that lexicon specialization results in significant gains in accuracy and reductions in parsing time for in-domain utterances.

The paper is organized as follows. In Section 2 we present the general architecture of our system and show how the diverging needs of semantic interpretation and domain reasoning motivate our two-layer architecture. We describe our customization method for converting the domain-independent semantic representations into domain-customized representations for reasoning in Section 3. We discuss our lexicon specialization technique in Section 4. System evaluations are presented in Section 5, and Section 6 discusses how our method can be integrated with other parsing and interpretation approaches and concludes with an overview of related work.

2 Motivation and Background

2.1 Natural Language Understanding in a Multi-domain System

This work is grounded in our experience in building TRIPS, a task-oriented spoken dialogue system that has been deployed in a number of domains, including train routing (TRAINS [5]), emergency response planning (Pacifica [2], Monroe [47]), medication scheduling (Medadvisor [4]) and learning tasks on the web [29].

The architecture of the interpretation components in the system is shown in Figure 1. The parser is used to process the natural language input, the Interpretation Manager (IM) manages reference resolution, dialogue management and collaborative problem solving tasks, and the Behavioral Manager determines system behavior using reasoners specific to the domain and application.

This architecture separates the domain-independent and domain-specific components in the system. Domain-independent conversational expertise is localized in the IM, which does not change between domains. In contrast, we use different reasoners in different domains, choosing those which can accomplish the domain tasks with optimal speed and accuracy. So far, with this architecture we have used the KM reasoner [13], the TRIPS planning system, the LOOM description logic system [35], and reasoners using OWL ontology representations.

Most utterances contain a mixture of content, some of which is handled by the domain reasoners, but some by the IM. For instance, in the sentence *What if we went along the coast instead*, the content of the speech act (going along the coast) is domain-specific, but the utterance also contains domain-general words (*what if*, *instead*). These words indicate that the IM should form a hypothetical query about an alternative to the current problem-solving act. The strategy for dealing with such constructs is independent of the specific back-end reasoner. It is handled by a domain-independent intention recognition model within the IM [1]. Separating out domain-general semantic representations therefore enables re-use of the IM components in new domains without the need to adapt them to new knowledge representations.

2.2 Linguistic Ontology Versus a Domain Model

Existing deep parsers build semantic representations using features that can be reliably inferred from syntax, but do not address the problem of producing semantic representations for content words which can be directly used for reasoning. They represent, for example, information for reference resolution and fragment interpretation based on quantifiers and syntactic constraints on fragments [44]. For content words, however, they assume that sense distinctions are perfectly correlated with syntactic behavior. For example, in the LINGO ERG lexicon, a verb with two different meanings reflected in different subcategorization frames will be assigned two different senses; but senses with identical syntactic distributions, *e.g.*, *bank* as an institution and *bank* as a geographical object, are not differentiated.

Such syntactically based word senses are not best suited for efficient domain reasoning. The meaning of concepts based on their syntactic behavior is not always clear, as syntactic and semantic structures are not perfectly correlated in practice. Additionally, reasoning in many knowledge representations such as LOOM [35] or KM [13] is incomplete in various ways, and queries that closely follow a natural language formulation may be inefficient [20].

Our approach is instead to establish a domain-specific mapping between the word senses selected by the parser and the concepts in the domain model. This task is easier in many respects, because many words have only a small number of senses within the context of a given domain. However, the mapping is not straightforward because domain-specific representations often do not line up well with linguistic structure.

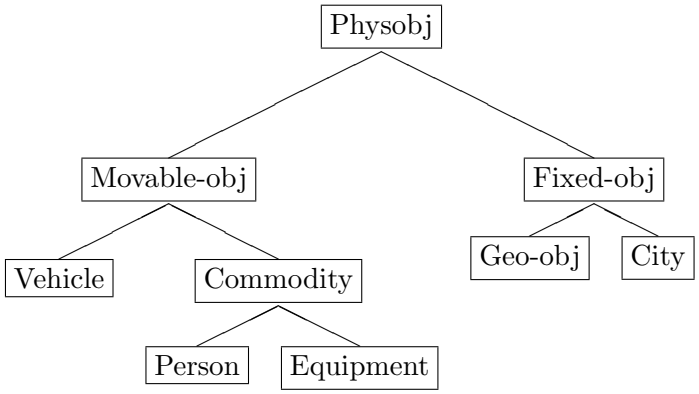

Figure 2: Ontology fragment for physical objects in TRIPS Pacifica domain.

As an example of a domain-specific ontology, consider our Pacifica transportation domain. In this domain, the user is given the task of developing a plan to evacuate a fictional island before a storm. A fragment of the ontology for physical objects used by the planner in Pacifica is shown in Figure 2. The top-level distinction is made between fixed objects such as geographical locations, and movable objects such as commodities suitable for transportation. People are classified as a kind of commodity because they are transported as cargo in the Pacifica scenario.

The planner knows 3 main actions: MOVE, LOAD and TRANSPORT. MOVE is the action when a vehicle is moved, LOAD is the action of loading a vehicle with cargo, and TRANSPORT is the action of transporting cargo, which may involve choosing and loading a suitable vehicle. The MOVE and TRANSPORT actions both take a path argument, which is a complex-valued description of type GEO-PATH with slots representing the origin, the destination and optionally interim points on the path. Ontology definitions representing those actions are shown in Figure 3.

Although this ontology is optimally suited for planning and reasoning in the Pacifica domain, it is not the best choice for a semantic representation, especially in a multi-domain system. Design choices for the Pacifica domain such as making a top-level distinction between MOVABLE-OBJ and FIXED-OBJ and classifying people as COMMODITY are counterintuitive in domains that are not specialized for transportation, such as medication scheduling and computer purchasing. In addition, type restrictions tailored for Pacifica are not well suited to other domains. For example, the sentence *I moved to a different city* does not satisfy the type constraints of either MOVE or TRANSPORT, because in Pacifica only vehicles can move objects between locations.

KR concept types and their slots in the domain ontology are also not well-suited to straightforward alignment with linguistic structure. The values of complex *:path* slots can come from a variety of expressions for path phrases, *e.g.*, *to*, *toward* for *:destination*; *through*, *via*, *by* for *:via*; and occasionally from subcategorized complements, as in *Leave Avon for Bath at 5pm.* It proved difficult to handle this kind of variation in the grammar without introducing domain-specific rules that were not applicable in other domains. A

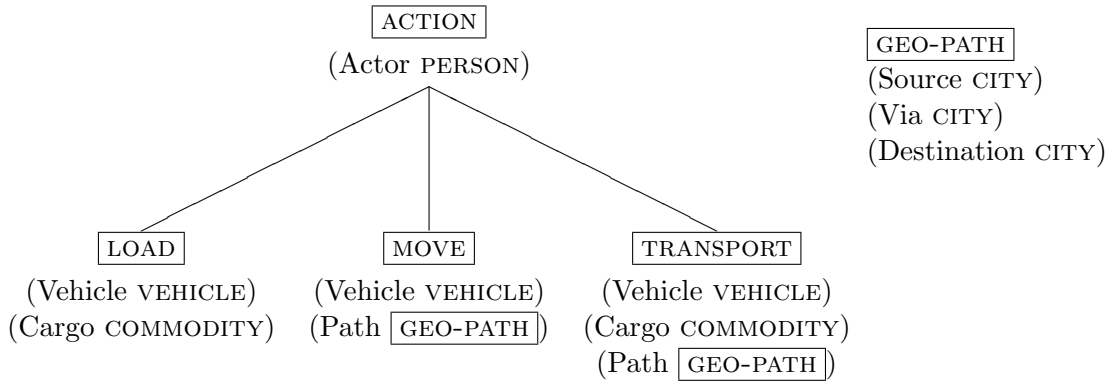

Figure 3: Action ontology fragment in TRIPS Pacifica domain. The MOVE and TRANSPORT arguments have a slot *Path* which is filled with a frame of type GEO-PATH.

representation more suitable for a general-purpose grammar is a general MOTION type with :To-Loc and :From-Loc arguments, which can be filled by either subcategorized complements or by adjuncts as needed.

To balance the needs of domain-specific and domain-general representations, we separate the ontology used in parsing and semantic interpretation (the LF ontology) from the ontology used in reasoning (the KR ontology). The design considerations for the LF and KR ontologies are summarized in Table 1.

In the LF ontology we go a step beyond purely syntax-based word classes by introducing word senses that depend both on syntactic distinctions and on semantic properties that we do not expect to change between domains. For example, all motion verbs are classified under a general LF::Motion type, which covers various instances of objects being moved and includes subtypes for more specific motion events such as transformation, transfer or filling a container. They also differentiate between objects based on a small set of general semantic features such as general type (physical object, abstract object or action), origin (living or non-living) etc. These features are based on a domain-independent EuroWordNet feature set [51] which we modified to better suit parsing purposes [23].

The LF ontology contains linguistically motivated semantic types and associated roles for events, objects and properties. The semantic roles can be straightforwardly linked to syntax with simple templates specified in the lexical entries (see Section 4.1). In this way, the semantic representation derived from the LF ontology abstracts away from specific surface realizations, so that syntactic alternations such as *Load the truck with oranges* and *Load the oranges into the truck* have identical semantic interpretations.

The hierarchical structure of semantic types in the LF ontology provides a stepping-stone for linking semantic and domain representations, by defining semantically motivated word senses which can be disambiguated based on a combination of syntactic and semantic features. This simplifies the connection with domain reasoners by providing

LF Ontology	KR Ontologies
As general as possible, broad coverage of concepts	Domain-specific concepts and organization
Relatively flat structure, linguistically motivated concepts and sense distinctions	Deeper structure, with fine-grained distinctions between concepts as relevant to domain
Simple representations with semantic roles easily linked to syntactic structure	Concepts and slots organized for efficient reasoning without regard for linguistic structure

Table 1: Design considerations for LF and KR ontologies.

a useful level of abstraction over syntactic representations, as discussed in Section 4.1. See [22, 19] for detailed discussion of our LF ontology design and content.

3 From Generic LF Representation to Customized KR Representation

The transformation from LF representation to KR representation consists of two steps: mapping the ontological concepts in the LF (LF types) into the domain-specific ontological concepts in the KR (KR classes); then converting the domain-independent LF-representation syntax into the syntax required by the KR.

In this section, we discuss only the ontology mapping, the most difficult part of the process. Our transform engine converts the LF syntax into AKRL, an abstract syntax using the KR ontology concepts. It is then relatively simple to convert between AKRL and the domain-specific KR syntax. We have implemented transforms from AKRL to representations in KM, which is a frame-based language, and LOOM [35], a description logic language. For brevity, our examples omit this part of the transformation, presenting the results of transforming the LF expressions directly into the target KR language syntax.

Mapping between the ontologies requires solving two problems: mapping the concepts and aligning the arguments. Given an LF type from the semantic representation, we first have to determine the KR type to which it corresponds. Once the concept type correspondence has been established, we need to align the semantic roles of the LF type with the appropriate slots in the KR representation. As discussed in Section 2.2, this alignment may not be straightforward. We therefore develop a system of transforms to account for the different possible alignments between the semantic and domain representations.

3.1 LF and KR Representations

Our semantic representation is a flat unscoped neo-Davidsonian representation, using event arguments and semantic roles. It is similar to QLF [6] and Minimal Recursion Semantics [15] in that it uses identifiers to link the (non-recursive) terms together.

```
(F e123 (:* LF::Fill-container load) :Agent pro1 :Theme v1 :Goal v2)
(IMPRO pro1 LF::Person :context-rel *YOU*)
(THE v1 (SET-OF (:* LF::Fruit orange)))
(THE v2 (:* LF::Vehicle truck))
```

Figure 4: LF representation of the sentence *Load the oranges into the truck.*

```
(define-type LF::Motion
 :sem (Situation (Aspect Dynamic))
 :arguments (:Theme (Phys-obj (Mobility Movable)))
            (:Source (Phys-obj))
            (:Goal (Phys-obj))

(define-type LF::Fill-container
 :parent LF::Motion
 :sem (Situation (Cause Agentive))
 :arguments (:Agent (Phys-obj (Intentional +)))
            (:Goal (Phys-obj (Container +))))
```

Figure 5: LF type definitions for LF::Motion and LF::Filling. In the lexicon, feature vectors from LF arguments are used to generate selectional restrictions based on mappings between subcategorization frames and LF arguments.

A (simplified) example representation for *Load the truck with oranges* is shown in Figure 4. The representation is a set of terms in the form

`(<specifier> <variable> <type> <argument>*)`

where the specifier can be `F` for predicates produced by verbs and adjectives, a quantifier for noun phrases, and `IMPRO` for implicit arguments, such as subjects of imperatives. Note that in this paper we omit tense, aspect, speech act and some referential information from our examples for simplicity.

The LF form in Figure 4 identifies the sense of the main verb *load* as an instance of concept LF::Fill-container.[1] It identifies *oranges* as a :Theme argument of the filling action, that is, the object being moved, and *truck* as a :Goal of the filling action. Since the sentence is an imperative, the parser also infers an implicit pronoun as its subject, corresponding to the :Agent role. A special notation is used to denote word types: they are in the form `(:* LF::lf-type lex-form)` where the `lf-type` is the word sense from the LF ontology, *e.g.*, LF::Fill-container, and `lex-form` is the base form of the word. The primary reason for including both semantic type and form is to balance generality and computability in our domain-independent lexicon [22]. Domain applications can use the sense information encoded in the LF type, but may also use the precise word form in reasoning.

[1]The first version of the LF ontology was based on a pre-release version of FrameNet [27] where LF::Filling was declared the appropriate type for this sense of *load*.

```
(AND (TYPE e1 LOAD) (actor e1 YOU123)
     (cargo e1 oranges2) (vehicle e1 truck3))
     (The oranges2 (type oranges2 ORANGE))
     (The truck3 (type truck3 TRUCK))
```

Figure 6: A sample description in the TRIPS WKB language representing *Load the truck with oranges.*

The corresponding LF ontology entries are shown in Figure 5. The full details of these representations are beyond the scope of this paper. In brief, the LF types are a domain-independent representation inspired by FrameNet [27], with a simpler set of semantic arguments, and augmented with typed feature vectors [22, 23]. We use vectors of semantic features as a flexible semantic representation for selectional restrictions which can be easily extended with new features, a property crucial for the lexicon specialization algorithm discussed in Section 4. We discuss how the LF representations are obtained from natural language input in Section 4.

An example of a KR representation used for reasoning is given in Figure 6. Apart from changing the representation syntax, the main goal in LF to KR mapping is to find correspondences between predicates and arguments in LF and KR representations. Examples in this section are based on mappings between LF representations and KR representations consistent with the Pacifica ontology from Section 2.2. However, they represent the general cases we found necessary to map between the ontologies in different domains supported by the TRIPS system.

3.2 Determining the KR Type of an Expression

The correspondence between the LF and KR ontologies is established through mappings that link each relevant LF type to a corresponding KR class. For example, the transform in Figure 7(a) states that (:* LF::Vehicle truck) corresponds to the TRUCK class in the domain ontology.

The transforms utilize the hierarchical structure of the LF ontology to map larger classes of words. For example, if we declare a transform LF::Transport → TRANSPORT, then all children of type LF::Transport must map into the TRANSPORT predicate on the domain side, unless a more specific transform applies. This reduces the effort involved in linking between the ontologies. For example, in the Pacifica domain a variety of movement verbs that can be interpreted as a transport action are covered by a single transform.

The mapping may become more complicated if words with the same LF type need to be split between multiple KR classes. For example, all medication names are assigned the same type, LF::Medication, in the LF ontology, but in the medication scheduling KR representation those are further subdivided into different domain-specific drug categories. Such complex alignments are supported in our transform system through several additional mechanisms: using lexical forms of words in transforms, exceptional rules for specific subtypes of a more general LF, and restrictions on transform application.

Using lexical forms in the transforms is possible because many ontologies are designed

(a) (define-lf-to-kr-transform truck-transform
:typetransform ((:* LF::Vehicle truck) → TRUCK)
(b) (define-lf-to-kr-transform medication-transform
:typetransform ((:* LF::Medication ?lf-form) → ?LF-FORM)

Figure 7: Transforms to establish the correspondence between an LF and a KR type (a) a simple transform based on a specific type (b) a more complex transform utilizing a lexical form.

to be human-readable and have consistent naming schemes for KR types. An example of a transform using a lexical form is given in Figure 7(b). In this case, for any item of LF type LF::Medication, its lexical form will be used as its KR type. This transform utilizes the property of the KR ontology in our Medadvisor domain where medication names serve directly as type names in the ontology. Thus, (:* LF::Medication Aspirin) will be transformed into the KR type ASPIRIN, and (:* LF::Medication Zoloft) into the KR type ZOLOFT. Our implementation also supports an easy extension to obtain a KR type from the lexical form by calling a function from within a transform. This can be used, for example, to add a prefix to a lexical form, obtaining, for example MEDICINE_ZOLOFT as a type name.

Using the lexical form in transforms is easiest if the KR ontology is in effect a domain-specific extension of the LF ontology - for example, all medications are still subtypes of a single KR type corresponding to a concept "medication". In that case the transforms can be applied straightforwardly, though consistency checking is necessary to verify that the type name generated by the transform rule is appropriate. The system for consistency checks and dealing with exceptional cases is presented in [19].

It is possible, however, that the LF type and lexical form are not enough to determine the KR type. For example, in our basic electricity and electronics domain [20] the word *in* corresponds to COMPONENT-OF in the phrase *The bulb in 1*, and to CONTAINS in the phrase *The bulb in the closed path.* If this is the case, two transforms can be written specifying alternative translations for *in.* Then, type restrictions from the domain reasoner can be used to disambiguate between the possible KR types.

In section 4 we present a lexicon specialization algorithm that supports choosing the appropriate KR type in ambiguous cases during parsing by automatically propagating selectional restrictions directly into the lexicon. When the parse is complete, the appropriate KR types for all the words can be read off directly from the lexical entries. An alternative that we used in one of our systems is to apply all possible combinations of transforms to a given LF form, and then decide on the correct translation based on a combination of type restrictions and discourse constraints. This is more appropriate for the domains where there are many entities with types not known until the reference resolution is complete (for example, if there are many onscreen objects identified only with labels such as *a*, *b*, *c*).

Other features that can be utilized in determining a KR type of an expression include the presence of specific LF arguments in the LF form, and exceptional rules limiting the application of transforms to certain LF subtypes. See [19] for details.

```
(define-lf-to-kr-transform load-transform-trips
   :typevar ?vv
   :typetransform ( LF::Fill-container → LOAD)
   :argtransforms (((:Agent ?a) → (actor ?vv ?a))
                   ((:Theme ?t) → (cargo ?vv ?t))
                   ((:Goal ?g) → (container ?vv ?g))))
```

Figure 8: Transform between the domain-independent form for LF::Filling and a domain-specific LOAD action which transforms the LF representation in Figure 4 into the KR representation in Figure 6.

3.3 Transforming Semantic Arguments

Once the correspondence between LF types and KR concepts is established, the semantic arguments from the LF representation have to be aligned with KR slots. The following cases have to be considered in the alignment: establishing direct one-to-one correspondence; cases where a semantic role may map to different KR slots depending on type constraints; and cases where several semantic roles need to be collected in a more complex structure.

The simplest case is where there is a one-to-one mapping between semantic roles and slots in the corresponding KR type. An example of such a transform for LOAD actions in the Pacifica domain is shown in Figure 8. This maps any LF representation of type LF::Fill-container into an instance of the LOAD concept. This transform specifies that the *actor* slot of LOAD will be filled with the value of the :Agent argument of LF::Fill-container (`?a`), the *cargo* slot with value of :Theme (`?t`) , and the *container* slot with the value of :Goal (`?g`).

A KR ontology may make finer distinctions between arguments than would be expressed in a domain-general language ontology. For example, the general LF::Move type describes a motion event with a :Theme argument which is the object moved. This covers cases like *Send a truck to Avon* and *Send the people to Avon*. However, the Pacifica ontology distinguishes explicitly between cargo and vehicles. In such cases, multiple transforms need to be specified to cover all possible alignments, and type restrictions in the KR ontology are used to disambiguate between them.

Transforms that link instances of LF::Motion with TRANSPORT and MOVE KR classes are shown in Figure 9. The transform in (a) applies to *Send the people to Avon*, while the transform in (b) is applicable to *Send the truck to Avon*, because the KR type *truck* is a subtype of VEHICLE, and the KR type *people* is a subtype of CARGO in Pacifica ontology.

Finally, a KR representation may use more complex structures than the LF representation, requiring that LF arguments be combined into new KR frames. For example, in the Pacifica domain paths are represented as complex slots of type GEO-PATH while in the LF representation all path adverbials are represented as separate semantic arguments (see Section 2.2). The LF representation of *Leave Abyss for Delta* contains *Abyss* as a :From-loc argument, and *Delta* as a :To-loc argument. Thus, we need a transform which combines those LF arguments into a frame of type GEO-PATH as a filler of the

```
(a) (define-lf-to-kr-transform transport-transform
      :typevar ?vv :preconditions ((:obligatory :Theme))
      :typetransform (LF::Motion → TRANSPORT)
      :argtransforms (((:Agent ?a) → (actor ?vv ?a))
                      ((:Theme ?t) → (cargo ?vv ?t))
                      ((:Instrument ?vh) → (vehicle ?vv ?vh))))
(b) (define-lf-to-kr-transform move-transform
      :typetransform (LF::Motion → MOVE)
      :argtransforms (((:Agent ?a) → (actor ?vv ?a))
                      ((:Theme ?t) → (vehicle ?vv ?t)))
```

Figure 9: The transforms for (a) TRANSPORT and (b) MOVE actions in the TRIPS Pacifica domain. The precondition limits the TRANSPORT transform application only to the cases when the cargo slot can be filled.

```
(define-lf-to-kr-transform path-transform-trips
   :typevar ?vv
   :abstract t
   :typetransform (LF::Motion → ?)
   :argtransforms ((NIL → (path ?vv ?pv))
                   ((:From-loc ?fl) → (source ?pv ?fl))
                   ((:To-loc ?tl) → (destination ?pv ?tl))
                   ((:Via ?va) → (mid-point ?pv ?va))))
```

Figure 10: A transform to collect all path adverbials into a single path frame in the TRIPS Planner language. NIL in an argument denotes the operation of creating a new frame.

path slot. The LF-KR transforms support an operator to instantiate new classes to cover this case.

An example transform instantiating a complex-valued slot is shown in Figure 10. It contains an argument mapping with *NIL* on the left-hand side, interpreted as a request to instantiate a new frame. During transform application an instance of GEO-PATH object will be created to fill the *path* slot, and values from the :To-loc and :From-loc arguments will be used to fill its slots.[2] This transform generalizes over converting path arguments of verbs of type LF::Motion, regardless of whether they are converted into MOVE or TRANSPORT, and complements the transforms in Figure 9.

This discussion glosses over additional difficulties introduced by certain adjuncts, especially modifiers such as *straight* in *Go straight to Avon.* Depending on the target KR representation, this sentence can be handled in a variety of ways. In the Pacifica ontology, *straight* can be seen as a slot filler in the GEO-PATH frame; if an ontology does not use a separate frame for path representations, *straight* may be defined as a slot

[2]The type of the new class is not specified in the transform explicitly, but taken from the type restriction on the *path* slot of MOVE and TRANSPORT in the Pacifica ontology.

filler on the MOVE frame; finally, in simpler domains it may be that the interpretation chooses to ignore the modifiers, and drop them completely from the representation. Our transform system supports these choices in possible mappings. A detailed description, along with the algorithm to implement the transform, is provided in [19].

4 Lexicon Specialization

The previous section discussed the use of transforms to connect LF and KR ontologies. In this section, we reuse the transforms in a pre-processing stage to specialize the lexicon. By integrating the domain-specific semantic information into the lexicon and grammar, we increase parsing speed and improve semantic disambiguation accuracy.

4.1 The Domain-Independent Lexicon

Lexicon entries in TRIPS encode the linking between syntactic structure and semantic representation. The rules in the TRIPS grammar compute a set of grammatical functions based on syntactic structure. Let us consider verbs, which are the heads of S structures. There are potentially four grammatical roles in a basic S structure: the subject of the verb (SUBJ), the direct object (DOBJ), an indirect object (IOBJ), and other non-NP complements (COMP). The lexicon entries then define the connection between the grammatical roles and the LF arguments. Each entry identifies an LF type for the verb, and defines a mapping from the grammatical roles to the semantic arguments used in the LF. Type constraints from the LF ontology are propagated into the lexical entries for use as selectional restrictions (note that these are necessarily weak since they are domain-general).

As an example, one entry for the verb *load* would specify the information in Figure 11(a). The LF type indicates that an S structure headed by this verb will produce an LF form with type (:* LF::Fill-container load), that the SUBJ fills the :Agent role (if it is a physical object with semantic feature *Intentional* set), that the DOBJ fills the :Theme role (if its *Mobility* feature is set to *Movable*), and the COMP, if it is a prepositional phrase with preposition *into*, fills the :Goal role (if its *Container* feature is *+*). [3]

This entry licenses the sentence *He loaded the cargo into the truck*, but not *Load the ideas into the truck*, since its direct object violates selectional restrictions. It also would not work for *He loaded the truck with oranges*, because of the syntactic and semantic restrictions. The latter usage is covered by a second entry for *load* (Figure 11(b)). Together these entries cover the syntactic variation known as the spray/load alternation [34] for *load*.

As this example illustrates, argument structure alternations are handled in the lexicon with templates that map semantic roles to grammatical positions. Common alternations such as the passive (*He loaded the truck with oranges / The truck was loaded with oranges (by him)*) and the dative (*She gave him the book / She gave the book to him*) are handled in the grammar; the passive rules apply to verbs with a semantic role that is linked to the direct object position (transitive verbs), and the dative alternation

[3] The selectional restrictions are obtained automatically from the LF definition shown in Figure 5 based on the role mappings specified in the lexical entry.

(a) LF Type: (:* LF::Fill-container load)
Preference: 1.0
Role mappings:

SUBJ	NP	:Agent	*(Phys-obj (intentional +))*
DOBJ	NP	:Theme	*(Phys-obj (mobility movable))*
COMP	(PP (ptype into))	:Goal	*(Phys-obj (container +))*

(b) LF Type: (:* LF::Fill-container load)
Preference: 1.0
Role mappings:

SUBJ	NP	:Agent	*(Phys-obj (intentional +))*
DOBJ	NP	:Goal	*(Phys-obj (container +))*
COMP	(PP (ptype with))	:Theme	*(Phys-obj (mobility movable))*

Figure 11: Lexical information for the verb *load* in the TRIPS lexicon (a) as in *Load the oranges into the truck* (b) as in *Load the truck with oranges.*

rules apply to verbs with a semantic role that is linked to the indirect object position (ditransitive verbs).

The grammar employs a scoring model for parse selection. Each lexical entry contains a numerical preference value used to rank alternative senses that satisfy all syntactic and semantic constraints. Grammar rules are also assigned preferences. The scoring model is informed by development in multiple domains, and performs comparably to corpus-based scoring in dialogue domains where corpus data is limited because it is costly and time-consuming to collect [18].

The parse selection model is also used to guide the parser's search. The parser uses a n-best beam-search algorithm, and chooses the order in which the constituents are added to the agenda based on their preference score. In our case, both entries have the same preference value, so they will be treated as equally likely during parsing. In the next section, we discuss how preferences are used in specialization to focus the search on the word senses specific to the domain.

The lexical entries map syntactic structure to the more abstract semantic structure. The ontology mapping transforms add a third level of interpretation, mapping the semantic arguments onto roles specific to the particular domain, and adding domain-specific selectional restrictions, as described in the next section. Figure 12 shows the whole process, adding the KR transform shown earlier in Figure 8. Note that Figure 12 shows the sources of information only, not a sequence of interpretations. In fact, most of those representations are computed simultaneously by the parser as it builds the constituents (the mappings between LF and KR arguments are computed during lexicon specialization but not applied until interpretation starts).

The question arises as to whether all these levels are necessary. What advantage is there, for instance, in having the intermediate domain-independent semantic representation? Could we not just produce mappings directly from the grammatical roles to the KR ontology? The answer is the key to our claim of portability. If a developer had to produce mappings directly from the grammatical relations to the KR ontology, adapt-

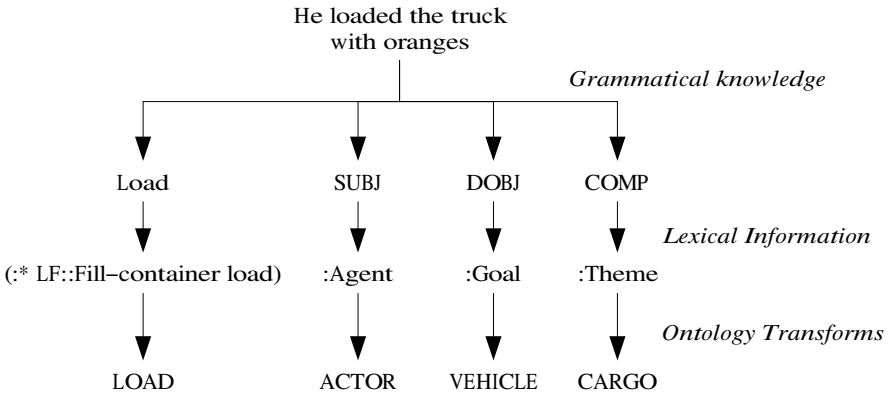

Figure 12: Sources of information in the TRIPS semantic interpretation.

ing the system to a new domain would be a significant programming effort, involving modification of a large number of lexicon entries. For example, one would need to define separate mappings from each of the syntactic configurations possible for *load* to the KR representation, and even more mappings to cover linguistic variation in path expressions (discussed in Section 2.2). The intermediate LF representation allows us to deal with all the domain-independent complications just once and then to reuse this work in every new domain. There are considerably fewer LF types than lexical entries, so producing the mapping rules for a new domain is a more manageable task. In addition, since the LF types are organized in an inheritance hierarchy, we can often state a mapping rule that applies to multiple LF types. The reduction in work when adapting to a new domain is substantial. We provide empirical evidence to this effect in the evaluation in Section 5.

4.2 Specializing the Lexicon

To specialize the lexicon, we pre-process every lexical entry by determining all possible transforms that apply to its LF type. For each transform that applies, a new sense definition is created that is identical to the first definition except that a new semantic feature called *Kr-type* is added, with the value extracted from the ontology restrictions specified in the transform. Given the situation shown in Figure 12, where the transform in Figure 8 was applied to an entry, we use the definition of LOAD in the KR ontology (shown in Figure 3) to find that the filler of the *Vehicle* role must be of type VEHICLE. Thus the semantic feature *(Kr-type* VEHICLE*)* is added to the DOBJ selectional restrictions. Likewise, the restriction *(Kr-type* COMMODITY*)* is added to the semantic feature vector for the COMP.

In addition to adding the *Kr-type* feature, we use feature inference rules that allow the parser to derive additional semantic restrictions. In Pacifica, we defined a rule that allows the parser to infer *(Origin Human)*[4] from *(Kr-type* PERSON*)*, and another that allows the system to infer *(Form Solid-Object)* from *(Origin Human)*.[5] After the feature

[4] *Origin* is a domain-independent feature with values *human,animal,plant,non-living,artifact*

[5] Feature inference is a well founded mechanism used throughout the LF ontology to also describe

LF Type: (:* LF::Fill-container load)
Preference: 1.03
Role mappings:

SUBJ	NP	:Agent	*(Phys-obj (intentional +)* **(Kr-type** PERSON**) (Origin Human) (Form Solid-Object)***)*
DOBJ	NP	:Theme	*(Phys-obj (mobility movable)* **(Kr-type** COMMODITY**) (Origin Artifact) (Form Solid-Object)***)*
COMP	(PP (ptype into))	:Goal	*(Phys-obj (container +)* **(Kr-type** VEHICLE**)***)*

Figure 13: Lexical information for the specialized verb *load*, corresponding to the non-specialized entry in Figure 11(a). The changes made by the specialization algorithm are highlighted in bold.

inference rules are completed, we have a revised lexical entry with tighter semantic features based on the domain, as shown in Figure 13.[6]

The algorithm for computing transforms is shown in Figure 14. In addition to tightening selectional restrictions, we increase the preference value of specialized entries by a pre-determined factor, to ensure that they are tried first during parsing. We used the factor of 1.03 in our experiments, adjusted based on parsing speed and accuracy results on a development dataset.

Once the new lexical entries have been generated we have a domain-specialized parser, without having to add any additional mechanisms to the parser except for the generic capability to query subtype relations in the KR ontology when it is checking the *Kr-type* feature. We also have the option of either retaining the original entry in addition to the specialized entries, or removing it, depending on how tightly we expect the language to fit the domain model.

Our lexicon specialization process is designed to easily integrate the specialized and non-specialized entries. If no specialization is found for a lexical entry, it remains in the lexicon, though with a lower preference, with the assumption that *Kr-type* is assigned an undefined value *KR-root*, which will satisfy any *Kr-type* restriction. When the parser analyzes a sentence, it will try the specialized senses first because their preference is higher, but will use non-specialized entries as necessary to obtain a full parse. Thus, the search focuses on words specific to the domain, but words from the domain-independent language interpretation components are intermixed freely with domain-specialized words.

dependencies between domain-independent feature values [19].

[6]There will also be a similar specialized entry obtained from the entry in Figure 11(b).

Given a lexical entry E:

1. Determine applicable KR transforms based on the LF type of E.
2. For each transform, make a copy of the entry and perform the following:
   (a) Add the feature *(Kr-type* T_{kr}*)* where T_{KR} is the KR type in the mapping
   (b) Query the KR ontology for type restrictions on the roles of T_{KR}.
   (c) Add the KR restrictions to the selectional restrictions on the appropriate arguments
   (d) Apply the feature inference rules to new changed semantic feature vector to add entailed features.
3. Increase the preference of the specialized sense by a pre-defined factor

Figure 14: An algorithm for specializing lexical entries.

5 Evaluation

5.1 Portability

Overall system portability is influenced by two main factors: the coverage of a domain-general grammar in new domains, and the effort involved in customizing the system to a new domain.

The detailed speed and accuracy evaluation presented in the next section shows that the generic version of the grammar performs similarly well on two evaluation domains. Previous evaluations of our generic (unspecialized) grammar on completely new domains [18] demonstrated that the grammar has sentence-level recall (i.e. can find a complete parse) for 63% of sentences in human-human task-oriented dialogue, with 69% precision (i.e. can find a parse with fully complete and correct semantic representation). This provides a good starting point for porting a dialogue system to a new domain. Many of the failures are due to gaps in lexical coverage (our domain-general lexicon is relatively small, as seen in Table 2). However, we have recently developed methods for rapidly expanding lexical coverage with the aid of existing wide-coverage lexical resources [50, 16], which help alleviate coverage problems. The results in the next section also show that our specialization mechanisms help improve parsing precision.

The lexicon statistics presented in Table 2 also show that the amount of work involved in domain customization is relatively small. The number of mappings that need to be written is an order of magnitude smaller than the number of lexical entries, and is proportionate to the size of the domain as defined by the number of classes. It is also about 8 times smaller than the number of specialized word senses that result from the mapping, which demonstrates that mapping from the LF ontology rather than lexical entries directly provides useful generalizations. In an earlier version of a similar evaluation, we also demonstrated that speed and accuracy improvements for specialization persisted even after a domain-general grammar was extended to improve coverage [19].

Domain	Lex. entries	KR classes	Mappings	Specialized entries
Generic	2457	-	-	-
Medadvisor	2575	182	95	665
Pacifica	2656	210	98	720

Table 2: Lexicon and domain statistics for our evaluation domains.

Table 3 lists the steps involved in porting the system to a new domain. Note that the first three items in the table (defining a domain ontology, defining new words and defining new grammar rules) need to be done in any system which is being employed in a new domain. The advantage of our approach is that instead of trying to change and adapt the lexical entries and the rules from the previous domain to suit the language in the new domain, we can re-use the existing lexicon and grammar. Developing a wide-coverage lexicon and grammar is a time-consuming task, but it is a one-time investment that requires diminishing amounts of work as new domains are added, as demonstrated in the case study discussed in this section.

In this approach, a domain reasoner needs to provide an ontology with a hierarchy of possible domain type and relations (true in all the domains we have encountered so far). To derive maximal benefit in terms of improving parsing speed and accuracy, the domain reasoner should also provide type restrictions on all arguments. Depending on the domain, this may require some additional work from the developers, but it is also a common feature of ontologies implemented in knowledge-based systems. Finally, it would be helpful (though not essential) to have a domain representation that is mostly compositional - see Section 6 for further discussion.

Writing LF-KR transforms is the most time-consuming part of the domain specialization process. However, the size of the transform set is proportional to the number of entities and relations in the KR ontology, which is considerably smaller than the set of all lexical entries in the domain. In addition, the generalizations provided by the LF ontology, including its hierarchical structure and the encoding of semantic concepts and roles independently of surface syntactic form, considerably reduce the amount of effort necessary in transform writing.

As a case study, we implemented a language interpreter in a new domain, a basic electricity and electronics tutor [53]. The task domain was a lab teaching the student to measure current in a circuit, supported by an existing domain reasoner implemented using the LOOM knowledge representation [35]. Domain statistics are presented in Table 4.

This was a challenging test for our system, because the parser had to cope both with a new subject domain (electricity) and a new genre (tutoring rather than collaborative planning). This is reflected in the relatively high number of lexical entries which needed to be added. These included both domain terms not encountered previously (*e.g.*, *multimeter, terminal*) and domain-general terms used in tutoring but not frequent in our previous problem-solving domains (*e.g.*, *learn, represent, signify*). Around 20 of these new entries were new subcategorization frames that we had not previously encountered in our corpora for words already defined in the lexicon. This is reflected in the small number of changes required to the LF ontology to cover the new word types. We also

Step	Comment
Define a domain model and the KR ontology.	Required for any dialogue system, part of building the reasoning components
Define lexical entries and LF types as needed for previously unseen words	Domain-specific terminology and any words not encountered in previous domains.
Modify the grammar with new domain-independent rules as needed to cover new syntactic constructs.	Effort diminishes with time as the coverage becomes more complete
Define the LF-KR transforms.	The main required step to link LF and KR ontologies.
Define feature inference rules.	Optional, but helpful in improving parsing speed and accuracy

Table 3: Steps to porting our interpretation architecture to a new domain.

Domain Concepts	Domain Relations	Transforms	Lexical entries added	LF types added	Grammar rules added
59	68	22	75	8	1

Table 4: Domain statistics for Beetle domain customization.

had to add a grammar rule for relative clauses acting as adverbials.

The corpus collected for the domain contained fairly complex sentences such as *What component should I connect to the battery using wire 1.* There was also a large number of pronouns and definite descriptions referring to objects in the simulation environment. We therefore implemented a reference resolution algorithm using features from our semantic representation.[7] It took less than 1 month's effort by a developer familiar with the grammar and LF ontology to add the words missing from the lexicon, write a simple reference resolution module, add a set of transforms to map the semantic representations to the domain ontology, and implement a converter between AKRL and LOOM syntax.

This case study shows the potential of our method for rapid development of dialogue systems in new domains. Even though some words and a grammar rule needed to be added, once the correct definitions were in place we immediately got correct domain-independent and domain-specific interpretations for the utterances involved. Moreover, we continued to observe improvements from sharing the grammar between the domains. During our specialization experiment, we held the grammar without change. At the end, we conducted a coverage evaluation on parsing human-human dialogues in the BEE domain, recording 50% sentence-level accuracy (details to be reported elsewhere). Immediately after concluding the evaluation, we brought in the changes in the grammar done in the interim for the benefit of the computer purchasing domain, and conducted the evaluation again. The coverage increased to 54% without any additional work involved, just through sharing the grammar development between the domains. We plan

[7]Note that LOOM does not support the representation features needed for reference resolution, so having an intermediate semantic representation was a requirement for system implementation.

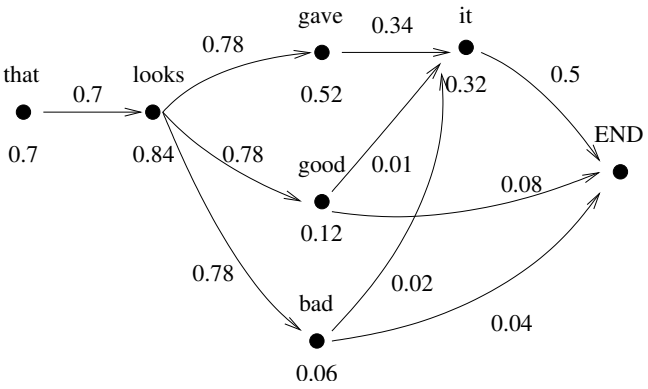

Figure 15: A sample lattice for *that looks good*, where the best speech hypothesis is *that looks gave it*. Numbers at nodes denote confidence scores for lexical items, numbers on arcs denote the confidence scores on transitions.

to conduct a more detailed evaluation of sharing between the domains and portability improvements in future work.

5.2 Speed and Accuracy

To evaluate the effect of our lexicon specialization algorithm on parsing speed and semantic representation accuracy, we compare the behavior of generic and specialized versions of our lexicon on speech lattices in two application domains. We chose lattices for our evaluation because they offer a way to utilize the power of syntactic parsing to make the system more robust. Speech recognition in a speaker-independent system is a difficult task, and depending on our domain, the 1-best sentence accuracy of our speech models could be as low as 50%. A common approach to this problem is to have the recognizer return a speech lattice, which is a weighted graph representing a set of recognition hypotheses.

A part of a lattice for the sentence *That looks good*, taken from an actual system run, is shown in Figure 15. The highest scoring sequence is *That looks gave it*, which is not syntactically correct. Using the syntactic information, the parser should be able to determine that *That looks good* is a better interpretation. To do that, it needs to be able to select between a large number of possible word sequences, which seriously complicates the parsing task.

The evaluation was conducted on the same domain-general grammar and lexicon and two different domains: transportation (Pacifica) and medication scheduling (Medadvisor). We collected lattices produced by our speech recognizer during regular system runs based on 50-best recognition hypotheses to use as test data.

As a measure of speed we report average parse times per lattice. As measures of representation accuracy, we use word error rate and concept error rate. Word error rate is the measure of how well the parser can pick out the exact words the user said from the lattices. However, word error rate does not fully reflect the impact of specialization. For example, the parser has no grounds to choose between different verb forms (can/could), or between different determiners (the/this/that). The specialization can only affect

Domain	Sents	Words per sent. (stdev)	Time-generic	Time-KR	Constits-generic	Constits-KR
Medadvisor	33	6.03 (2.11)	10.06	4.04**	28243	15951 **
Pacifica	189	4.24 (3.09)	13.67	5.15**	153966	95760 **

Table 5: Average parsing time per lattice in seconds, and total number of constituents built by the specialized grammar compared to the generic grammar. Statistically significant differences are indicated with **. Tests were done on a 1.6Hz Pentium 4 Linux workstation with 512M of memory running Allegro Common Lisp 7.0.

Domain	Word count	WER	Concept count	CER
Generic-Medadvisor	199	0.17	200	0.17
KR-Medadvisor	199	0.18	200	0.12**
Generic-Pacifica	1041	0.15	812	0.19
KR-Pacifica	1041	0.13	812	0.16**

Table 6: Semantic accuracy evaluation for the specialized grammar compared to the generic grammar. Measures reported are word error rate(WER) and concept error rate (CER). Statistically significant differences between generic and specialized versions are indicated with **.

choices of content words, on which the domain can place semantic constraints. Therefore we measure the error rate on concepts, using the same formula as for word error rate, but on sequences of LF types corresponding to content words.

The results presented in Table 5 show that lexicon specialization considerably increases parsing speed and improves disambiguation accuracy. The times represent the average parsing time per lattice, and differences are significant at $p < 0.0001$ (paired 2-tailed t-test, $t(33) = 4.74$, $t(189) = 8.86$). Note that each lattice in effect corresponds to 50 different utterances to be parsed. Some of the parse time is due to garbage collection, which could be reduced by improving the unification algorithm [31]. However, there is also a significant reduction in the number of constituents built which shows that the overall parsing efficiency is significantly improved with this method.

Table 6 presents results on representation accuracy. The differences in word error rates are not statistically significant, though there is a trend for improvement in the Pacifica domain (Pacifica: $p < 0.08$, Medadvisor: $p < 0.77$). However, the improvements in content word error rates are significant in both domains (Pacifica: $p < 0.02$, Medadvisor: $p < 0.05$).[8]

The reason why overall word error rate differences are not statistically significant is due to the treatment of non-content words in our scoring function. During lexicon specialization we increased the preference values associated with content words related to the domain (based on defined transforms), but also of non-content functional words such as determiners and prepositions to avoid unnecessary deletions. However, this

[8]We used a paired 2-tailed t-test to compare the distributions of edit distances to evaluate significance, as suggested in [48].

Words uttered	Words selected by generic version	Words selected with KR
How long will that take	How will that wait	How long will that take
Let's use the helicopter instead	Legs use the helicopter instead	Let's use the helicopter instead
Meanwhile use the other truck to get the people from Exodus to Delta	NO SPANNING PARSE	Meanwhile use the other truck to drive the people from Exodus to Delta

Table 7: Examples of words picked from the same lattice with and without specialization.

resulted in a number of extra insertions; for example, discourse adverbials such as *now* were inserted if present in the lattice. In addition, different tense and pronoun forms were often substituted based on the lattice, because domain information is not helpful in deciding on the appropriate word form to choose.

The improvements in concept error rates in both domains show that the algorithm helps build better semantic representations of utterance content. Finding better scoring functions to address non-content words together with content words is planned as future work.

The improvement in concept accuracy comes from two sources. The tighter selectional restrictions limit the search space and help to correctly disambiguate according to domain knowledge. The increased preference values for specialized entries ensure that these are tried first during parsing, and that constituents with more specialized entries are preferred over those which contain out-of-domain words. Both of these strategies also help the parser find an interpretation for in-domain utterances faster, resulting in significant speed increases.

To illustrate how specialization helps to pick better word sequences from lattices, Table 7 lists several example sentences from the evaluation, listing the words that were said, and the sentence that was selected with and without specialization. In the first two examples the non-specialized version returns questionable interpretations, which nevertheless are difficult to exclude on a domain-independent basis.[9] In Pacifica, however, the reasoners know nothing about manner of waiting, but we can answer questions about the duration of actions. Domain constraints also specify that only people are suitable actors for the class USE. This is reflected in the existence of a corresponding LF to KR mapping and thus specialized lexicon entries with a higher preference, leading the parser to choose the correct interpretation.

The last example in the table comes from a longer utterance, and a correspondingly large lattice with a very large number of possible paths. The non-specialized version ran over the limit of chart size specified in the parser and gave up before a spanning parse could be found. The specialized version actually found a plausible interpretation, even if it substituted the more specific word *drive* for *get*.

Overall, this evaluation demonstrates that our customization technique promotes parser portability between domains, while providing significant speed and accuracy im-

[9]Manner modifiers are possible for *wait*, as in *I will wait patiently*, and the corresponding question is strange for pragmatic rather than syntactic or semantic reasons.

provements when parsing in-domain utterances through propagation of domain-specific restrictions into the lexicon.

6 Discussion

In this paper we argue that maintaining separate semantic and knowledge representations linked via an ontology mapping procedure helps alleviate the tension between the needs of semantic interpretation and domain reasoning components, and, in particular, helps us build a more portable parser for semantic interpretation, while providing speed and accuracy improvements via domain specialization.

We presented evidence of parser portability in Section 5.1. Further evidence comes from the use of our parser and grammar across domains. In addition to the four collaborative problem-solving domains from the TRIPS system (see Section 2), the parser has been ported to two different tutoring domains: teaching basic electricity and electronics and tutoring symbolic differentiation. The collaborative problem-solving domains use the same set of language interpretation components for parsing and discourse interpretation, including reference resolution, intended speech act recognition, and ellipsis/ fragment interpretation. During discourse interpretation, calls are made to the domain-specific reasoners to evaluate various hypotheses, but all the algorithms are driven from the domain-general representations and the interpretation components do not require any modification when switching domains. All the required changes for interfacing with a new set of domain reasoners are captured in the ontology transforms. Furthermore, our most recent systems also share a domain-general surface generation capability that is driven by automatically reversing the ontology transforms [12].

The two tutoring domains use a different interpretation system currently under development, because existing models of collaborative problem-solving dialogue do not cover many dialogue phenomena common in tutoring [21]. However, these two domains already share the same reference resolution module utilizing our semantic representations, and the dialogue managers (implemented using the TrindiKit [32]) will share substantial classes of update rules based on our LF representations, using linguistic features to detect indirect speech acts, hedging and other phenomena common in tutorial dialogue.

Recent advances in developing wide-coverage lexical semantic databases provide opportunities to improve system portability further. When our system was originally developed, wide-coverage lexicons with word classes and semantic roles suitable for parsing were not yet available. We used a pre-release version of FrameNet [27] as the basis for our LF ontology, but needed to modify it to make parsing and semantic disambiguation possible [22]. Since then, semantic role labeling parsers have been developed [11, 25] that output semantic frames with sense distinctions at a level similar to our LF ontology. Thus, a similar system for mapping between domain-general and domain-specific representations can help leverage their output in connection with domain reasoners.

This paper describes the issues which need to be addressed in the mappings between semantic and knowledge representations: aligning the ontologies at class level, and then transforming semantic arguments between the aligned classes. The latter requires considering cases where the semantic arguments can be aligned straightforwardly by direct

mapping, or may need to be combined into more complex structures, as discussed in Section 3.3.

The ontology mapping approach assumes that the KR ontology into which the semantic representations are being mapped employs compositional representations. If a KR representation contains many composite concepts such as *LitLightBulb, NonLitLightbulb, OpenCircuit, ClosedCircuit*, then the ontology mapping may be less effective, because many individual rules may have to be written to build up each composite concept from multiple concepts in the LF representation. Recently we have added some general abilities to support such cases in the mappings by defining customized functions to combine concepts. Composite concepts, however, present their own challenges to the maintenance of KR representations because of the large number of different classes. Thus it is reasonable to assume that for ontologies in medium scale domains, compositionality will also be maintained in the domain ontology.

One opportunity for improvement is finding ways to generate automatically domain-general to domain-specific mappings. A possible approach would be to use ontology and schema matching techniques developed in the semantic web community [46]. The string-based and taxonomy-based techniques would likely to be the most suitable to leverage the hierarchical structure of the domain-general ontology, as well as the string similarity between concept names.

In future we also plan to explore the use of our architecture with other deep grammars. The information in our base lexicon is not strongly dependent on a specific syntactic theory. As long as a grammar is able to identify syntactic functions such as subject and direct object, it should be possible to augment the semantic representation output by the grammar with semantic classes and roles from our ontology. Such a combination could be beneficial both for our interpreter, by providing another fast and reliable grammar to use, and for wide-coverage syntactic grammars, by providing a framework for linking semantic and domain-specific representations.

The underlying assumption in this work, supported by practical experience, is that semantic representations and domain representations have different requirements both in form (i.e. syntactic devices to express underspecification, undefined and fragmentary interpretations etc.) and in content (i.e. semantic hierarchies and semantic roles suitable for interpretation or reasoning). Building more NL-like knowledge representations would help address issues of form. However, reasoning in dialogue systems requires detailed domain models, including axioms specifying relationships between objects and actions. These are more difficult to build in a large-scale domain-independent fashion. Thus, our approach provides a workable alternative for connecting semantic and knowledge representations in practical systems.

6.1 Related Work

Currently, the most common approach for building interpretation modules for dialogue systems is using semantic grammars that directly link the lexicon entries to frames in a domain-specific representation, augmented with corpus-based methods to learn such mappings in small domains [45, 41]. However, assumptions about frame structure limits the expressivity of the grammars, which do not fully handle the complexities involved in quantification, negation, or many complex modifiers. Therefore they work best in sys-

tems where the dialogue state is encoded as a set of fixed contexts (see [3] for a detailed discussion). Thus, linguistically motivated parsers have been used for medium scale domains, including LINGO[14] for appointment scheduling in VerbMobil [30], CARMEL [43] for tutoring qualitative physics and OpenCCG [7] for parsing mathematical proofs in DIALOG [52].

LINGO ERG provides sophisticated models of underspecification in scoping [15], as well as for interpreting fragments in context [44]. It was used in building a dialogue system in the appointment scheduling domain [30]. This domain, however, involves only a very simple model of domain knowledge (the only thing which is necessary to know is which times are available, and which times are busy). Implementing the same methods in other domains with more complex knowledge representations remains an open problem, in part due to the difficulty of building a domain model capable of reasoning with the domain-independent representations output by LINGO.

CARMEL uses a syntactic parser based on the COMLEX syntactic lexicon [36] to provide wide syntactic coverage. However, it requires re-linking lexical entries in each domain to domain-specific knowledge representations, and does not provide general semantic representations for quantifiers and referring expressions. Thus it suffers from the same problems as domain-specific grammars where the limitations on expressivity in knowledge representation limit its usability for semantic interpretation.

The DIALOG system uses a deep parser for parsing mathematical discourse which outputs semantic role labels based on tectogrammatical relations, but without marked word senses. The system separates the utterance content used in dialogue processing and in reasoning, though we have not been able to find a detailed description of the algorithm used. Our system offers a clear architecture for this separation, because the domain-general representations are clearly defined through types in the LF ontology and linked to the KR ontology via LF-KR transforms.

The idea of using a multi-level representation to separate linguistic knowledge from domain application knowledge has been developed previously in the PHLIQA1 question-answering system [38, 9]. PHLIQA1 initially constructs a semantic deep structure for a natural language query using ambiguous predicate names corresponding to lexemes. This semantic structure is then transformed into an intermediate representation consistent with the "world model" of the system, independent of both the input language and the back-end database structure, which is then used to compute the database query.

Our approach shares the idea of using multiple levels of representation, but is different in that it relies on a domain-independent ontology of semantic types, disambiguated based on domain-independent syntactic and semantic features. In our system, we assume that the domain model may still contain distinctions that cannot be disambiguated within a domain-independent model, which is why our approach focuses on the transforms as mappings between the domain-independent and domain-specific ontologies. Additionally, we assume that it generally makes sense to disambiguate domain-independent types through a combination of syntactic and semantic features, hence the reason to connect the domain-independent ontology types directly to the lexical entries.

FrameNet has been used as an intermediate semantic representation in question answering [39] and information extraction [49]. In the information extraction evaluation, extraction patterns were formulated using FrameNet frames rather than individual words. The FrameNet-based system performed worse than the fully customized system,

but could be written much faster. This points to the potential usefulness of this approach in bootstrapping system development: the intermediate representation can be used to quickly start system development, and implement semantic interpretation modules based on domain-independent algorithms, while the KR mapping can be tuned for maximum precision in further development, possibly through the use of corpus-based methods.

Palmer et al. propose that different levels of sense granularities are necessary for different applications [40]. They identify sense distinctions at a medium granularity above the purely syntax-based senses as the appropriate level for an information extraction task. This is consistent with our approach of defining an intermediate level ontology as a starting point to connect language interpretation with reasoning tasks.

A rich ontology with word senses at different levels of granularity is now being developed based on corpus annotations [26]. The availability of such an ontology, and development of word sense disambiguation methods with the appropriate word senses, would provide another resource for intermediate-level information useful in connection to reasoning.

7 Conclusions

In this paper, we described a two-layer architecture for building multi-domain parsers and interpretation components. We separate domain-general and domain-specific information by maintaining separate ontologies: a domain-general LF ontology for language representation, and domain- and application-specific KR ontologies. The ontologies are linked with the help of LF-KR transforms. The architecture allows us to utilize a portable linguistically motivated parser capable of interpreting complex natural language sentences and producing semantic representations for interpretation. Our specialization technique allows us to link the semantic representation with domain knowledge representations simply by defining a small number of transforms between the representations. This approach allows us to utilize semantic representations to build portable interpretation modules, while linking them to domain knowledge representations in domain reasoners. We demonstrated that this approach helps maintain a parser and grammar that are portable between domains, while our lexicon specialization algorithm brings speed and accuracy improvements in comparison to the generic parser.

Acknowledgments

We thank Matthew Stone, Gwendolyn Campbell and Colin Matheson and the anonymous reviewers for comments and discussion in preparing this paper. This material is based on work supported by grants from ONR #N000149910165, NSF #IIS-0328811, DARPA #NBCHD030010 via subcontract to SRI #03-000223 and NSF #E1A-0080124.

References

[1] James Allen, Nate Blaylock, and George Ferguson. A problem solving model for collaborative agents. In Maria Gini, Toru Ishida, Cristiano Castelfranchi, and

W. Lewis Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, pages 774–781. ACM Press, July 2002.

[2] James Allen, Donna Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu, and Amanda Stent. An architecture for a generic dialogue shell. *Natural Language Engineering, Cambridge University Press*, 6(3):1–16, 2000.

[3] James Allen, Donna Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu, and Amanda Stent. Towards conversational human-computer interaction. *AI Magazine*, 22(4):27–38, 2001.

[4] James F. Allen, Nathanael Blaylock, Donna Byron, Nathanael Chambers, Myroslava Dzikovska, George Ferguson, and Mary Swift. Chester: Towards a personal medical advisor. *Journal of Biomedical Informatics*, 39(5):500–513, 2006.

[5] James F. Allen, Bradford W. Miller, Eric K. Ringger, and Teresa Sikorski. A robust system for natural spoken dialogue. In *Proceedings of the 1996 Annual Meeting of the Association for Computational Linguistics (ACL'96)*, 1996.

[6] Hiyan Alshawi, David M. Carter, Björn Gambäck, Stephen G. Pulman, and Manny Rayner. Transfer through quasi logical form: A new approach to machine translation. Technical Report T91020, SICS, Stockholm, Sweden, 1991.

[7] Jason Baldridge. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. PhD thesis, University of Edinburgh, 2002.

[8] Nate Blaylock and James Allen. A collaborative problem-solving model of dialogue. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, 2005.

[9] Harry Bunt. *Mass terms and model-theoretic semantics*, chapter 7. Cambridge University Press, 1983.

[10] Donna K. Byron. *Resolving Pronominal Reference to Abstract Entities*. PhD thesis, University of Rochester, 2002.

[11] Xavier Carreras and Lluís Màrquez. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL-2005*, 2005.

[12] Nathanael Chambers. Real-time stochastic language generation for dialogue systems. In *Proceedings of European Workshop for Natural Language Generation*, Aberdeen, Scotland, 2005.

[13] Peter Clark and Bruce Porter. *KM (1.4): Users Manual*. http://www.cs.utexas.edu/users/mfkb/km, 1999.

[14] Ann Copestake and Dan Flickinger. An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, Athens, Greece, 2000.

[15] Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(4):281–332, 2005.

[16] Benoit Crabbé, Myroslava O. Dzikovska, William de Beaumont, and Mary D. Swift. Increasing coverage of a domain independentdialogue lexicon with VerbNet. In *Proceedings of the Thurd International Workshop on Scalable Natural Language Understanding (ScaNaLU 2006)*, New York City, 2006.

[17] John Dowding, Jean Mark Gawron, Doug Appelt, John Bear, Lynn Cherny, Robert Moore, and Douglas Moran. GEMINI: A natural language system for spoken-language understanding. In *Proceedings of the 31st Meeting of the ACL*, July 05 1994. Comment: 8 pages, postscript.

[18] Myroslava Dzikovska, Mary Swift, James Allen, and William de Beaumont. Generic parsing for multi-domain semantic interpretation. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT-05)*, 2005.

[19] Myroslava O. Dzikovska. *A Practical Semantic Representation For Natural Language Parsing*. PhD thesis, University of Rochester, 2004.

[20] Myroslava O. Dzikovska, Charles B. Callaway, and Elaine Farrow. Interpretation and generation in a knowledge-based tutorial system. In *Proceedings of EACL-06 workshop on knowledge and reasoning for language processing*, Trento, Italy, April 2006.

[21] Myroslava O. Dzikovska, Charles B. Callaway, Matthew Stone, and Johanna D. Moore. Understanding student input for tutorial dialogue in procedural domains. In *Proceedings of The 10th Workshop on the Semantics and Pragmatics of Dialogue (brandial 2006)*, 2006.

[22] Myroslava O. Dzikovska, Mary D. Swift, and James F. Allen. Building a computational lexicon and ontology with FrameNet. In *LREC workshop on Building Lexical Resources from Semantically Annotated Corpora*, Lisbon, Portugal, May 2004.

[23] Myroslava O. Dzikovska, Mary D. Swift, and James F. Allen. Customizing meaning: building domain-specific semantic representations from a generic lexicon. In Harry Bunt, editor, *Computing Meaning, Volume 3*, Studies in Linguistics and Philosophy. Kluwer Academic Publishers, to appear.

[24] Annette Frank and Uwe Reyle. Principled based semantics for HPSG. In *Proceedings of EACL-95*. Association for Computational Linguistics, 1995.

[25] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 2002.

[26] Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 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, June 2006. Association for Computational Linguistics.

[27] Christopher Johnson and Charles J Fillmore. The FrameNet tagset for frame-semantic and syntactic coding of predicate-argument structure. In *Proceedings ANLP-NAACL 2000*, Seattle, WA, 2000.

[28] Pamela Jordan, Maxim Makatchev, Umarani Pappuswamy, Kurt VanLehn, and Patricia Albacete. A natural language tutorial dialogue system for physics. In *Proceedings of the 19th International FLAIRS conference*, 2006.

[29] Hyuckchul Jung, James Allen, Nathanael Chambers, Lucian Galescu, Mary Swift, and William Taysom. One-shot procedure learning from instruction and observation. In *Proceedings of FLAIRS-06*, Melbourne, FL, 2006.

[30] Martin Kay, Jean Mark Gawron, and Peter Norvig. *Verbmobil: A Translation System for Face-To-Face Dialog*. CSLI Press, Stanford, California, 1994.

[31] Bernd Kiefer, Hans-Ulrich Krieger, John Carroll, and Rob Malouf. A bag of useful techniques for efficient and robust parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 473–480, 1999.

[32] Staffan Larsson and David Traum. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 6(3–4):323–340, 2000.

[33] Oliver Lemon, Anne Bracy, Alexander Gruenstein, and Stanley Peters. The WITAS multi-modal dialogue system I. In *Proceedings of EuroSpeech 2001*, 2001.

[34] Beth C. Levin. *English Verb Classes and Alternations: a Preliminary Investigation*. University of Chicago Press, Chicago, IL, 1993.

[35] Robert MacGregor and Raymond Bates. The LOOM knowledge representation language. In *Proceedings of the Knowledge-Based Systems Workshop*, 1987. Held in St. Louis, Missouri, April 21-23, 1987. Also available as ISI reprint series report, RS-87-188, USC/Information Sciences Institute, Marina del Rey, CA.

[36] Catherine Macleod, Ralph Grishman, and Adam Meyers. Creating a common syntactic dictionary of English. In *SNLR: International Workshop on Sharable Natural Language Resources*, Nara, August 1994.

[37] John Maxwell and Ron Kaplan. An efficient parser for lfg. In *Proceedings of the First LFG Conference*, 1996.

[38] Piet Medema, Wim Bronnenberg, Harry Bunt, Jan Landsbergen, Remko Scha, Wijnand Schoenmakers, and Eric van Utteren. Phliqa1: Multilevel semantics in question answering. *American Journal of Computational Linguistics*, microfiche 32, 1975.

[39] Srini Narayanan and Sanda Harabagiu. Question answering based on semantic structures. In *Proceedings of International Conference on Computational Linguistics (COLING 2004)*, Geneva, Switzerland, 2004.

[40] Martha Palmer, Olga Babko-Malaya, , and Hoa Dang. Different sense granularities for different applications. In *Proceedings of the 2nd Workshop on Scalable Natural Language Understanding Systems*, Boston, MA, 2004.

[41] Manny Rayner, Beth Ann Hockey, and John Dowding. Grammar specialization meets language modelling. In *Proceedings of the 7th International Conference on Spoken Language Processing*, Denver, CO, 2002.

[42] Charles Rich, Candace L. Sidner, and Neal Lesh. COLLAGEN: Applying collaborative discourse theory to human-computer interaction. *AI Magazine*, 22(4):15–26, 2001.

[43] Carolyn Rosé. A framework for robust semantic interpretation. In *Proceedings 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, 2000.

[44] David Schlangen and Alex Lascarides. The interpretation of non-sentential utterances in dialogue. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*, Sapporo, Japan, July 2003.

[45] Stephanie Seneff. TINA: A natural language system for spoken language applications. *Computational Linguistics*, 18(1):61–86, March 1992.

[46] Pavel Shvaiko and Jérôme Euzenat. A survey of schema-based matching approaches. *J. Data Semantics IV*, 3730:146–171, 2005.

[47] Amanda J. Stent. The Monroe corpus. Technical Report 728/TN 99-2, The University of Rochester, Computer Science Department, 2000.

[48] Helmer Strik, Catia Cucchiarini, and Judith M. Kessens. Comparing the performance of two CSRs: How to determine the significance level of the differences. In *Proceedings of Eurospeech 2001*, volume 3, pages 2091–2094, Aalborg, Denmark, 2001.

[49] Mihai Surdeanu, Sanda M. Harabagiu, John Williams, and Paul Aarseth. Using predicate-argument structures for information extraction. In *Proceedings of ACL-03*, pages 8–15, 2003.

[50] Mary Swift. Towards automatic verb acquisition from VerbNet for spoken dialog processing. In *Proceedings of the Interdisciplinary Workshop on the Identification and Representation of Verb Features and Verb Classes*, Saarbrucken, Germany, February 2005.

[51] Piek Vossen. EuroWordNet: a multilingual database for information retrieval. In *Proceedings of the Delos workshop on Cross-language Information Retrieval*, March 1997.

[52] Magdalena Wolska and Ivana Kruijff-Korbayova. Issues in the interpretation of input in mathematical dialogs. In Denys Duchier, editor, *Prospects and advances in the syntax/semantics interface. Lorraine-Saarland Workshop Series proceedings.*, pages 45–50, Nancy, France, 2003.

[53] Claus Zinn, Johanna D. Moore, and Mark G. Core. Intelligent information presentation for tutoring systems. In O. Stock and M. Zancanaro, editors, *Intelligent Information Presentation*, Series: Text, Speech and Language Technology. Kluwer Academic Publishers, 2005.