Integrating linguistic and domain knowledge for spoken dialogue systems in multiple domains

Myroslava O. Dzikovska, James F. Allen, Mary D. Swift
Computer Science Department
University of Rochester
Rochester, NY, USA, 14627
{myros, james, swift}@cs.rochester.edu

Abstract

One challenge for developing spoken dialogue systems in multiple domains is facilitating system component communication using a shared domain ontology. Since each domain comes with its own set of concepts and actions relevant to the application, adapting a system to a new domain requires customizing components to use the ontological representations required for that domain. Our research in multiple domain development has highlighted differences in the ontological needs of a general-purpose language interface and a task-specific reasoning application. Although different domain applications have their own ontologies, many aspects of spoken dialogue interaction are common across domains. In this paper, we present a new method of customizing a broad-coverage parser to different domains by maintaining two ontologies, one that is generalized for language representation, and another that is customized to the domain, and defining mappings between them. In this way we preserve the broad-coverage language components across domains as well as produce semantic representations that are optimally suited to the domain reasoners.

1 Introduction

One of the challenges in developing spoken dialogue systems for multiple domains is making the system components communicate with each other using a specific domain ontology. Each domain comes with its own set of concepts and actions that are relevant to the application, so adapting a system to a new domain means that all components must be able to interface with the ontological representations required for that domain. Researchers working with multi-component systems typically require all components to use a shared ontology (*e.g.*, [Goddeau *et al.*, 1994]), or they develop methods to map between the internal representations and a communication language which uses a shared ontology (*e.g.*, [Gurevych *et al.*, 2003]). Our research in multiple domain development has highlighted differences in the ontological needs of a general-purpose language interface and a task-specific reasoning application. While each domain comes with its own ontology, many aspects of spoken dialogue interaction are common across domains. This observation has led us to develop a new method that customizes a broad-coverage, domain-independent parser to new domains that allows us to preserve the language information that is common across domains, while addressing specialization issues unique to each domain as much as possible and at the same time keeping the development for custom domain adaptation to a minimum.

Traditionally, the most common way to adapt a dialogue system to a new domain is to map the words in the system lexicon directly to the concepts in the domain model, and write a grammar customized for the domain (see for example [Seneff, 1992]). This process yields a parser that can quickly and accurately obtain semantic representations of in-domain utterances in the form needed by the system reasoning components. However, the performance gains in accuracy and efficiency are offset by the lack of portability of the language interpretation components, which often require large portions of the grammar and lexicon to be rewritten to adapt them to new domains.

A system that has a method of preserving syntactic information is AUTOSEM [Rosé, 2000], which uses the syntactic lexicon COMLEX [Macleod *et al.*, 1994] as a source of reusable syntactic information, and manually links subcategorization frames in the lexicon to the domain-specific knowledge representation. The linking is performed directly from syntactic arguments (*e.g.*, subject, object ...) to the slots in a frame-like domain representation output by the parser and used by the reasoners. Rosé's approach speeds up the development process for developing tutoring systems in multiple domains, but does not provide a mechanism for preserving general semantic representations across domains.

Our approach introduces an intermediate layer of abstraction, a generic language ontology for the parser that is linked to the lexicon and allows us to preserve general syntactic and semantic information across domains. Thus we maintain two ontologies: domain-independent for the parser, and domain-specific for the knowledge representation. The parser uses the generic language ontology to create meaning representations of the input speech which can then be converted into the required domain representation. To integrate the linguistic and domain knowledge, we define a set of mappings between the general semantic representations produced by the parser and

the domain-specific knowledge sources localized in the back-end application. We also use the mappings to specialize the lexicon to the domain, resulting in substantial improvement in parsing speed and accuracy.

The customization method described here was developed in the process of adapting the TRIPS dialogue system [Allen *et al.*, 2001] to multiple domains, including a transportation routing system [Allen *et al.*, 1996] and a medication scheduling adviser [Ferguson *et al.*, 2002]. We assume a generic dialogue system architecture [Allen *et al.*, 2000] that includes a speech module, a parser (responsible for syntactic parsing and initial logical form generation), an interpretation manager (responsible for contextual processing and dialogue management), and a back-end application that is responsible for the general problem-solving behavior of the system.

The paper is organized as follows. First we contrast the needs of linguistic ontologies and domain ontologies, in section 2. Next, we describe the organization of our domain-independent lexicon and ontology and how the parser uses it to generate a semantic representation of the language input that is then converted to representations that can be used by the domain-specific reasoners,in section 3. In section 4, we describe how the domain-independent representations are linked with the domain-specific KR ontology via a set of transforms. In section 5, we specify how the transforms are used to specialize the generic lexicon, which improves speed and accuracy for the parsing of in-domain utterances. Finally, in section 6 we discuss issues involved in using this method to port to new domains.

2 Linguistic ontology versus a domain model

In the process of developing our dialogue system in multiple domains, it has become clear that the language and domain-specific knowledge representation have differing needs. As an example of a domain-specific ontology, consider our Pacifica transportation domain. In this domain, the user is given the task to evacuate people and possibly other cargo before an approaching storm. A fragment of an ontology for physical objects used by the planner for this domain is shown in Figure 1. The top-level distinction is made between fixed objects such as geographical-locations, and movable objects such as vehicles and commodities, which are 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, TRANSPORT and LOAD. MOVE is the action when a vehicle is moved between 2 points, without any loading actions involved. TRANSPORT is the action of transporting a cargo between the two points. If the vehicle is not specified explicitly, a suitable vehicle needs to be chosen, and actions necessary to load it planned; LOAD is the action of putting a cargo into a vehicle. Pragmatically, TRANSPORT is further subdivided into a JUST-TRANSPORT action, when there's already a vehicle at a place where the cargo is located, and MOVE-THEN-TRANSPORT, where a vehicle needs to be moved to that location first. Ontology definitions representing those actions is shown in Figure 2.

Although this ontology is optimally suited for planning and

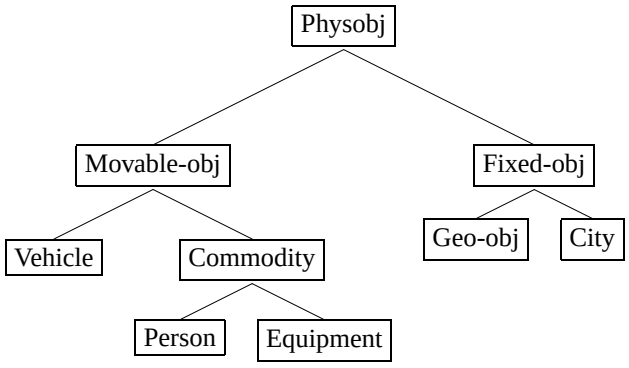

Figure 1: Ontology fragment for physical objects in TRIPS Pacifica domain.

reasoning in the Pacifica domain, it is not the best choice for the language ontology, especially in a multi-domain system. In the ontology for physical objects, making the difference between MOVABLE-OBJ and FIXED-OBJ a top-level distinction is counterintuitive in our medication adviser domain. Having a COMMODITY category is also not a good choice for a general ontology. In a medication adviser domain, this category is not relevant to the task, and people should be classified as living beings. Arguably, this could be solved by introducing multiple inheritance and making PERSON a child of both COMMODITY and LIVING-BEING. However, this does not solve the problem completely, because it is difficult to classify many physical objects consistently as cargos. For example, whether a door is a subtype of COMMODITY may be dependent on a specific application.

Similarly, the action ontology has a set of associated problems. We need to have the type restrictions to limit the parser search space and improve disambiguation accuracy. Yet the type restrictions tailored for the Pacifica domain are not well suited to other domains. For example, the sentence *I moved to a different place* clearly does not fall within the Pacifica representations of either MOVE or TRANSPORT. We could add a new class to the ontology to represent these actions, but this would mean adding another sense to the word *move*, increasing the ambiguity in the lexicon and making maintenance more difficult. The very specific categories, JUST-TRANSPORT and MOVE-THEN-TRANSPORT, cannot be distinguished in the lexicon at all, and they just add confusion to trying to determine which are the correct ontological categories for different words in the lexicon.

From a parsing perspective, a good classification for movement verbs would have a general MOTION type which covers all instances of objects being moved, whether they are cargo or vehicles, with a subtype for transportation covering the verbs where transportation of cargo is clearly implied, *e.g.*, *transport*, *ship*. For physical object classification, we still want to retain the information on whether some objects (such as mountains) are inherently fixed or not, but it should not be included as the topmost division in the hierarchy.

Another point of tension between the needs of parsing and reasoning components is the argument structure. From the point of view of planner, it would be most convenient to have

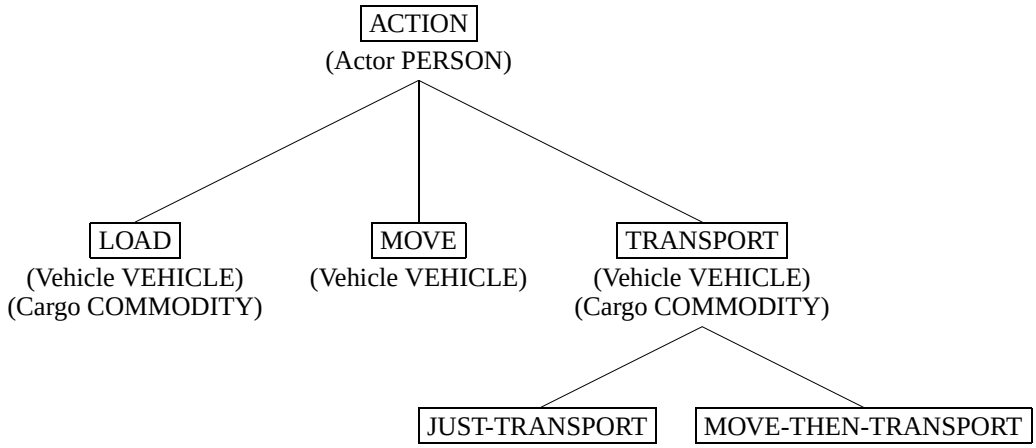

Figure 2: Action ontology fragment in TRIPS Pacifica domain.

the definition of MOVE and TRANSPORT which have a single :PATH slot, where the :PATH slot has a complex value with :SOURCE, :DESTINATION and :VIA slots. This was the original implementation in the TRAINS-95 system, but it was abandoned because its divergence from linguistic structure made it difficult to implement in practice. For example, the values filling :PATH slots can come from a variety of adverbial modifiers, *e.g.*, *to, toward* for :DESTINATION, *through*, *via*, *by* for :VIA, and occasionally from a direct object, such as *leave Avon for Bath at 5pm*. It proved difficult to handle this kind of variation in the grammar without introducing domain-specific rules which were not suitable in other domains. A representation more suitable for a parser is a MOVE type with TO-LOC and FROM-LOC arguments which can be filled either by the subcategorized complements or by PP-modifiers as they are encountered.

Given these issues, we decided to separate the ontology used in parsing (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.

The LF ontology is designed to be as general as possible and cover a wide range of concepts that are needed to generate semantic representations of the language input for use in a variety of practical domains. Accordingly, the LF ontology has a relatively flat structure and makes only linguistically motivated sense distinctions. For example, the LF ontology distinguishes between LF_CONSUMPTION and LF_REMOVING senses of *take*, since these two senses can be distinguished with different syntactic and semantic patterns. The LF_CONSUMPTION sense of *take* requires a consumable substance as an object and often occurs with a temporal modifier on the *take* event, as in *I take an aspirin at bedtime*, while the LF_REMOVING sense of *take* requires a moveable entity as object and often occurs with a path modifier on the event, as in *The truck took the cargo to the station*. The semantic representations we generate from the LF ontology are as close to the original linguistic structure as possible, so there is straightforward mapping between lexical entries and their ontological representation.

The KR ontology, on the other hand, is designed specifically for a given domain, so the concepts in the ontology are organized in ways that are best suited for domain-specific reasoning. The KR ontology makes fine-grained distinctions between concepts as relevant to the domain, so its hierarchical structure is deeper than that of the LF ontology. And because the KR concepts are organized to facilitate reasoning in the task domain, their representation may be inconsistent with how concepts are expressed linguistically, as is the case with the MOVE and TRANSPORT concepts described above.

In the remainder of the paper, we describe our method of integrating the information in the domain-independent linguistic ontology with domain-specific KR ontologies to maximize both system portability and efficient parsing.

3 From language to general semantic representation

The LF ontology is close in structure to linguistic form, so it can be easily mapped to natural language and used in multiple domains. It classifies entities (i.e., objects, events or properties) primarily in terms of argument structure. Every LF type declares a set of linguistically motivated thematic arguments, a structure inspired by FRAMENET [Johnson and Fillmore, 2000], but which covers a number of areas where FRAMENET is incomplete, such as words related to planning and problem solving. We use the LF ontology in conjunction with a generic grammar covering a wide range of syntactic structures and requiring minimal changes between domains. For example, adapting the parser from the transportation to the medical domain required adding LF types for medical terms (our generic hierarchy was incomplete in this area) and corresponding vocabulary entries, but we did not need to change the grammar or existing lexical entries, and we continue to use the same lexicon in both domains.

The LF types in the LF ontology are organized in a single-inheritance hierarchy. We implement multiple inheritance via semantic feature vectors associated with each LF type. Each LF type has an associated feature set type, which classifies entities as physical objects, abstract objects, propositions, situations or times. The feature type is associated with a set of features that encode basic meaning components used in

LF Ontology	KR Ontology
As general as possible, broad coverage of concepts	Domain-specific concepts and organization
Relatively flat structure and linguistically motivated sense distinctions	Deeper structure, with fine-grained distinctions between concepts as relevant to domain
Simple representations that are close to linguistic form (argument structure)	Roles organized for efficient reasoning w/o regard for linguistic structure

Table 1: Design considerations for LF and KR ontologies.

semantic restrictions, such as form, origin and function for physical objects. For example, Figure 3 contains LF type definitions used for various types of vehicles (LF_Vehicle), food items (LF_Food) and people (LF_Person). The feature vector for LF_Vehicle, denoted by :sem, encodes the fact that it is a physical entity, which has the form of an object (as opposed to a formless substance), is man-made *(origin artifact)*, functions as a vehicle *(object-function vehicle)*, and is a mobile object *(mobility movable)*.

```
(define-type LF_Vehicle
 :parent LF_Manufactured-object
 :sem (phys-obj (form object) (origin artifact)
                (object-function vehicle)
                (mobility movable)))

(define-type LF_Food
 :parent LF_Physical-object
 :sem (phys-obj (origin natural) (object-function comestible)
                (mobility movable)))

(define-type LF_Person
 :parent LF_Physical-object
 :sem (phys-obj (origin human) (intentional +)
                (form object) (mobility movable)))
```

Figure 3: LF type definitions for some objects in the LF ontology.

While the distinctions we encode with feature vectors can be represented in a multiple inheritance hierarchy, a feature-based representation makes it easy to implement an efficient type-matching algorithm based on [Miller and Schubert, 1988]. More importantly, using feature vectors allows us to easily modify the semantic information associated with a lexical entry, a property utilized during the customization process described below. More detailed description of our feature set can be found in [Dzikovska *et al.*, to appear].

The semantic features encode basic meaning components that we use in selectional restrictions on verb arguments. To illustrate, consider Figure 4, which shows definitions for a certain class of words that specify filling events that involve containers. Intuitively, LF_Filling defines a motion event in which an intentional being (AGENT) loads a movable object (THEME) into another physical object that can serve as a container (GOAL). The restriction on the agent argument specifies that the entity filling this role must be an intentional being.

In the LF ontology, many similar words can be mapped to the same LF type, because, as noted in section 2, we only make the distinctions in the LF ontology that are linguistically motivated and as independent of any given domain as possible. For example, *cram*, *fill*, *pack* and *load* are all linked to LF_Filling. It is difficult to further subdivide them in a way that will be consistent across domains, therefore the LF ontology does not make any further distinctions.[1] In the Pacifica domain, it is in fact sufficient to interpret all of them as instances of a LOAD action. However, these words obviously differ in meaning, and in a different domain we may need to make more specific distinctions between them. To retain those distinctions, word senses are treated as leaves of the semantic hierarchy. The complete LF type of a word is written as as LF-parent*LF-form, where the LF-parent is the type defined in the LF ontology (for example, LF_Filling for *load*), and the LF-form is the canonical form associated with the word (for example, LF_Filling*load for all morphological variations of a verb *load*).

For every word sense in the lexicon, we specify syntactic features (such as agreement, morphology, etc.), LF type, and the subcategorization frame and syntax-semantics mappings.

To illustrate, consider the verb *load* in the sense *to fill the container*. The lexicon entry for *load* linked to LF_Filling is shown in Figure 5. It contains two possible mappings between the syntax and the LF: one in which the THEME is realized as direct object, corresponding to *Load the oranges into the truck*, and another in which the THEME is realized as prepositional complement, corresponding to *Load the truck with oranges*. In this figure, SUBJ denotes the restriction on the subject of the verb, DOBJ the restriction on the direct object, and COMP3 the restriction on the prepositional complement. SEM denotes the selectional restrictions propagated from the definition in Figure 4 based on role correspondences. These restrictions are used as follows: Given the restriction on the THEME argument in Figure 4, only objects marked as *(mobility movable)* are accepted as a direct object or prepositional *with* complement of *load*. Finally, the SUBJ-MAP, DOBJ-MAP and COMP3-MAP denote the semantic arguments into which these syntactic arguments will be mapped.

The parser produces a flattened and unscoped logical form using reified events [Davidson, 1967]. A simplified representation showing the semantic content of *Load the oranges into the truck* is shown in Figure 6. The representation has the form:

(⟨QUANTIFIER⟩ ⟨VARIABLE⟩ ⟨TYPE⟩ ⟨RESTRICTION⟩)

[1] This is consistent with the FrameNet interpretation, where all of these words are instances of the Filling frame.

```
(a)
 (LF LF_Filling*load)
 (SUBJ (NP (SEM (phys-obj (intentional +)))))
      (SUBJ-MAP AGENT)
 (DOBJ (NP (SEM (phys-obj (mobility movable)))))
      (DOBJ-MAP THEME)
 (COMP3 (PP (ptype into) (SEM (phys-obj (container +))))))
      (COMP3-MAP GOAL)
(b)
(LF LF_Filling*load)
 (SUBJ (NP (sem (phys-obj (intentional +)))))
      (SUBJ-MAP AGENT)
 (DOBJ (NP (SEM (phys-obj (container +)))))
      (DOBJ-MAP GOAL)
 (COMP3 (PP (ptype with) (SEM (phys-obj (mobility movable))))))
      (COMP3-MAP THEME)
```

Figure 5: Lexicon definitions for the verb *load* used in (a) *load the oranges into the truck* (b) *load the truck with oranges*.

```
(define-type LF_Motion
 :sem (situation (aspect dynamic))
 :arguments
   (THEME (phys-obj (mobility movable)))
   (SOURCE (phys-obj))
   (GOAL (phys-obj))

(define-type LF_Filling
 :parent LF_Motion
 :sem (situation (cause agentive))
 :arguments
   (AGENT (phys-obj (intentional +)))
   (GOAL (phys-obj (container +))))
```

Figure 4: 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.

where ⟨QUANTIFIER⟩ is either a generalized quantifier coming from a noun phrase specifier, or a quantifier denoting the function of the variable, *e.g.*, "SPEECHACT" for speech acts, "F" for forms derived from verb and adverbial predicates, "IMPRO" for implicit pronouns, such as the implicit subject of an imperative. Note that for simplicity, the representation shown there is a simplified version that omits some of the reference and discourse information and focuses on the semantic content of an utterance.

4 From general semantic representation to domain-specific KR

To produce domain-specific KR representations from the generic LF representations, we developed a method to customize parser output. Our method transforms the intermediate semantic representation produced by the parser (the LF representation) into the specific KR representation used by the reasoners. The transformation from LF representation to KR representation needs to accomplish two things: it has to convert the domain-independent LF-representation syntax into the syntax required by the KR, and it has to map the general ontological concepts in the LF (LF types) into the domain-specific ontological concepts in the KR (KR types). We illustrate our method by showing how the LF representations can be transformed into two different knowledge representation languages: the knowledge representation currently used by the TRIPS planner (TRIPS Planner Language), and the KM knowledge representation language [Clark and Porter, 1999].

```
(SPEECHACT sa1 SA_REQUEST :content e123)
(F e123 LF_Filling*load :agent pro1 :theme v1 :goal v2)
(IMPRO pro1 LF_Person :context-rel +YOU+)
(THE v1 LF_FOOD*orange)
(THE v2 LF_VEHICLE*truck)
```

Figure 6: The LF representation of the sentence *load the oranges into the truck.*

First we illustrate how the LF representation syntax can be converted into the syntax required by either the Trips Planner Language or the KM language. For simplicity, these examples assume that both of these languages use a definition of the LOAD action with ACTOR, CARGO and VEHICLE slots as specified in the Pacifica ontology (described in section 2).

Consider the example sentence *Load the oranges into the truck*. The target forms that we want to sent to the Planner for both KR languages are given in Figure 8: (a) is the representation in the TRIPS Planner Language, and (b) is the representation in the KM language. Our task is to transform the LF representation for this sentence (shown in Figure 6) into these target representations. The focus of this first example is how we obtain the different syntactic representations required by the different KR languages. However, note that the first step in the transformation process is for reference resolution to replace all variables in definite descriptions in the

```
(a) (define-lf-to-kr-transform load-transform-trips
      :pattern ((?spec ?ev LF_Filling :agent ?a :theme ?t :goal ?g)
                -> ((TYPE ?ev LOAD) (actor ?ev ?a) (cargo ?ev ?t) (container ?ev ?g))))

(b) (define-lf-to-kr-transform load-transform-km
      :pattern ((?spec ?ev LF_Filling :agent ?a :theme ?t :goal ?g)
                -> (?ev is (a LOAD with (actor ?a) (cargo ?t) (container ?g)))))
```

Figure 7: Transforms between the domain-independent form and domain-dependent forms (a) in TRIPS planner language (b) in the KM language

```
(a)
(AND (TYPE e1 LOAD) (ACTOR e1 YOU123)
     (CARGO e1 oranges2) (VEHICLE e1 truck3))
(b)
e12 is (a LOAD with
          (actor YOUR123)
          (CARGO oranges2)
          (VEHICLE truck3))
```

Figure 8: The KR representation for *Load the oranges into the truck* in (a) the TRIPS planner language (b) the KM language.

LF representation with constants denoting the entities they refer to in the domain. In this example, v1 is replaced by ORANGES2 and v2 is replaced by TRUCK3. The details of this substitution process are beyond the scope of this paper.

In order to transform the domain-independent LF representation in Figure 6 into the domain-specific KR representations in Figure 8, we define mapping rules that specify the pattern that the LF must match, and the corresponding KR representation. The relevant transform for our example is shown in Figure 7. It will map all LF representations using LF type LF_Filling into instances of the LOAD concept. This transform specifies that the ACTOR slot of LOAD will be filled with the variable from the AGENT argument of LF_Filling (`?a`), the CARGO slot with variable from the THEME argument (`?t`) , and the CONTAINER slot with the variable from the GOAL argument (`?g`).

In this example, there is a direct correspondence between the arguments of the LF type and the slots of a KR type. A more complex case involves transforming a single LF form into a set of frames or predicates in the target KR language. For example, our LF representations treat all path adverbials (LF_To-loc, LF_From-loc, etc.) as separate components, which we have found to be the most convenient domain-independent implementation for capturing their syntactic variability. However, for a transportation domain, it is more convenient to collect all adverbials into a single path frame, because path descriptions are essential for routing. The transform shown in Figure 10 maps all path adverbials that modify an LF_Motion event into a single path frame in the KM language. The result of the application is a set of frames (or a frame with another embedded frame) shown in Figure 9.

The transforms use the hierarchical properties of the LF ontology. LF_Motion is a high-level LF ontology concept that covers a large class of motion words (*motion*, *move*, *transport*, *walk*, etc.) In the example above, the system will apply the path transform to any descendants of LF_Motion unless a more specific transform exists.

```
e45 is (a MOVE
          with (actor YOU123)
          (path (a PATH with (source Pittsford)
                             (destination Rochester))))
```

Figure 9: The representation of a MOVE event as a MOVE frame with an embedded PATH frame in the KM language.

Conversely, we can define transforms that use adverbial modifiers to fill slots in a single frame. For example, we can define a transform that converts *Go straight to Pittsford*, where *straight* is analyzed as modifying *to* in the LF representation and creates a MOVE frame with the slot value (Manner DIRECT), if this is the form supported by the system reasoners.

In our examples so far, all arguments are optional. For example, in Figure 10, we have LF_From-loc, LF_To-loc and LF_Via as semantic arguments for a MOVE event. These are not always overtly expressed in language, and if not all of them are present in the LF representation, the transform still applies. However, there are some cases in which an argument must be present for a transform to apply. In such cases, we specify preconditions on the required arguments. For example, the TRIPS Pacifica ontology distinguishes between MOVE, a request to move vehicles without any additional planning required, and TRANSPORT, a request to move cargos, as described in section 2. With this representation, it only makes sense to interpret motion verbs as instances of TRANSPORT if there is an identifiable cargo present. We achieve this with the transform in Figure 11, which specifies that the THEME semantic argument that fills the CARGO slot is required for the transform to apply.

To make transform writing more efficient, we use the lexical (orthographic) form of words in transforms. For example, from the point of view of the LF ontology, medication names have similar distributions across syntactic contexts, and therefore are represented as leaves under the LF_DRUG type, *e.g.*, LF_DRUG*prozac, LF_DRUG*aspirin. However, the KR ontology in the medication adviser domain makes pragmatic distinctions between medications (*e.g.*, prescription vs. over-the-counter), but uses medication names as leaf

```
(define-lf-to-kr-transform path-transform-km
   :pattern ((?spec ?ev LF_Motion :lf_from-loc ?fl :lf_to-loc ?tl :lf_via ?va)
            -> ((?ev is (a MOVE with (path *1)))
                (*1 is (a PATH with (source ?fl) (destination ?tl) (mid-point ?va))))
```

Figure 10: A transform to collect all path adverbials into a single path frame in the KM language. *1 denotes an operation of creating a new variable for the path frame.

```
(define-lf-to-kr-transform transport-transform-trips
   :preconditions ((:obligatory theme))
   :pattern ((?spec ?ev LF_Motion :agent ?a :theme ?t :instrument ?vh)
            -> ((TYPE ?ev LOAD) (actor ?ev ?a) (cargo ?ev ?t) (vehicle ?ev ?vh))))
```

Figure 11: The transform for TRANSPORT actions in TRIPS Pacifica domain. The precondition limits the transform application only to the cases when a cargo slot can be filled.

```
(define-lf-to-kr-transform drug-transform-medadvisor
   :pattern ((?spec ?en (:* LF_DRUG ?lf-form))
            -> ((TYPE ?en ?lf-form))))
   :defaults ((?lf-form substance))
```

Figure 12: A transform for medications in the medication adviser system, using the TRIPS planner language.

types in the KR hierarchy. We can write a single transform for all LF_DRUG children, shown in Figure 12, that converts the LF type to the KR type based on the lexical form specified in the entry. Note that since the transform allows a variable (?lf-form) to determine the resulting KR type, it also requires the user to specify a default value that will be assigned if the class with the name derived from ?lf-form cannot be found in the KR ontology.[2]

For instance, assume a KR ontology in which the medication ASPIRIN is defined, but the medication ZOLOFT is not defined. Then given the transform in Figure 12 and the logical form in Figure 13(a), we obtain an entity of KR type ASPIRIN, since ASPIRIN is defined in the KR ontology. However, since the concept ZOLOFT is not defined in the KR ontology, we obtain an entity of KR type SUBSTANCE, as shown in Figure 13(b), because SUBSTANCE was specified as the default value in the transform.

(a) (The v1 LF_DRUG*Aspirin) =>
    (TYPE v1 ASPIRIN)
(b) (The v2 LF_DRUG*Zoloft) =>
    (TYPE v2 substance)

Figure 13: The logical forms and the resulting KR forms after applying the transform in Figure 12 (a) when the type ASPIRIN is defined in the KR ontology (b) when the type ZOLOFT is not defined in the KR ontology.

5 Specializing the lexicon

In addition to using the transforms described above to convert LF representations to KR representations, we also use them 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 improves semantic disambiguation accuracy.

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 old definition but contains a new feature in the semantic vector, *kr-type*, with the value of the KR ontology class specified in the transform. Thus, we obtain a (possibly larger) specialized set of lexical entries that specify the KR class to which they belong. We then propagate type information from the LF representation into the syntactic arguments, which creates tighter selectional restrictions in the lexicon. Finally, we increase the preference values for the senses for which mappings were found, so domain-specific entries will be tried first during parsing.

We illustrate the lexicon specialization process with the verb *load*. Given the definitions of *load* and LF_Filling in Figure 4, and the transform definitions in Figure 7, the algorithm to generate the lexical entry for the verb *load* proceeds as follows:

- Fetch the definition of *LF_Filling* and the semantic feature vectors for it and its arguments;
- Determine the applicable transform, in this case `load-transform`;
- Add *KR-type load* to the semantic feature vector of *load*, as specified by the transform;
- Lexicon queries the ontology about selectional restrictions on arguments, and determines that the element that fills the *CARGO* slot needs *kr-type COMMODITY*;
- Add *KR-type COMMODITY* to the semantic feature vector of the THEME argument;

[2] Using the lexical form directly is the simplest possible case, since it assumes that the KR will use the lexical forms directly as concepts. An easy extension is determining the KR type based on some operator applied to a lexical form, for example, concatenating it with a pre-defined prefix used in the KR ontology, *e.g.*, MED-ASPIRIN, MED-TYLENOL, etc.

(a)
(LF LF_Filling*load)
(SUBJ (NP (SEM (phys-obj (intentional +) **(kr-type Person) (origin human) (form solid-object)**)))
(SUBJ-MAP AGENT)
(DOBJ (NP (SEM (phys-obj (mobility movable) (kr-type commodity)))))
(DOBJ-MAP THEME)
(COMP3 (PP (ptype into) (SEM (phys-obj (container +) **(kr-type Vehicle) (origin artifact) (form object)**))))
(COMP3-MAP GOAL)
(b)
(LF LF_Filling*load)
(SUBJ (NP (sem (phys-obj (intentional +) **(kr-type Person) (origin human) (form solid-object)**)))
(SUBJ-MAP AGENT)
(DOBJ (NP (SEM (phys-obj (container +) **(kr-type Vehicle) (origin artifact) (form object)**))))
(DOBJ-MAP GOAL)
(COMP3 (PP (ptype with) (SEM (phys-obj (mobility movable) **(kr-type Commodity)**))))
(COMP3-MAP THEME)

Figure 14: The lexicon entry for the verb *load* specialized for Pacifica domain (a) for *load the oranges into the truck* (b) for *load the truck with oranges*

- Apply similar transforms to the rest of the arguments.

This process creates a new definition of *load*, shown in Figure 14, that has 2 entries corresponding to the same two senses described in section 3, but with stricter selectional restrictions on the arguments as the result of domain specialization. Since this specialized lexical entry for *load* conforms to KR requirements, now suitable objects or prepositional complements of *load* must be not only movable, as specified in the LF ontology, but also identified as belonging to the class COMMODITY in the domain. Comparable transforms apply to the relevant nouns, so *oranges*, *people* and other cargos will have a *kr-type* value that is a subtype of COMMODITY inserted in their semantic feature vectors.

The entries are further specialized with the use of feature inference rules. Feature inference is a general mechanism in the system lexicon used to express dependencies between feature values. For example, we have a feature rule associated with our feature system declaring that if something is marked as a human *(origin human)*, it is also a solid object *(form solid-object)*, in (Figure 15(a)). During the specialization process we add the rules that declare dependencies between the values of *kr-type* features and the values of domain-independent features. For example, in our Pacifica domain we have a rule declaring that if something is marked as *(kr-type Person)*, then it must also have the domain-independent feature value *(origin human)*, shown in Figure 15(b). When that value is added to the feature vector in the subject restriction in the entry for *load*, it will trigger the domain-independent rule in 15(a), which causes *(form solid-object)* to be added to the feature set. The new features added as the result of specialization and feature inference are highlighted with bold in Figure 14.

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. It can then participate in any constructions with specialized entries. This makes the system more robust for out of domain utterances, because it allows us to find a parse and respond more intelligently even when out of domain words are present. At the same time, using feature specialization rules for domain-specific entries allows us further restrict search space even for unspecialized entries through tighter restrictions on argument slots. For example, the domain-independent entry for *load* allows any containers to fill the GOAL argument of the loading action. The specialized entry restricts the GOAL argument to vehicles. Thus, the verb phrase *load the dispenser* will be accepted only if there is no other interpretation available, because while *dispenser* does not have a specialized entry in Pacifica, it is not marked with the domain-independent value *(object-function vehicle)* and therefore does not satisfy the restriction.

(a) (origin human) => (form solid-object)

(b) (kr-type Person) => (origin human)

Figure 15: Some feature inference rules in the TRIPS lexicon. (a) The domain-independent inference rule in the generic lexicon; (b) The domain-dependent inference rule defined in the Pacifica domain.

Lexicon specialization considerably speeds up the parsing process. We conducted an evaluation comparing parsing speed and accuracy on two sets of 50-best speech lattices produced by our speech recognizer: 34 sentences in the medical domain and 200 sentences in the Pacifica domain. Table 2 provides some statistics on the lexicons used in these domains. The results presented in Table 3 show that lexicon specialization considerably increases parsing speed and improves disambiguation accuracy. The times represent the average parsing time per lattice, and the errors are the number of cases in which the parser selected the incorrect word sequence out of the alternatives in the lattice.[3]

[3]For the purpose of this evaluation, we considered correct

	Generic	Pacifica	Medical
# of senses	1947	2028	1954
# of KR classes	-	228	182
# of mappings	-	113	95

Table 2: Lexicon statistics.

	Pacifica	Medical
# of sentences	200	34
Time with KR (sec)	4.35 (870)	2.5 (84)
Time with no KR (sec)	9.7(1944)	4.3 (146)
Errors with KR	24%(47)	24% (8)
Errors with no KR	32% (65)	47% (16)

Table 3: Average parsing time per lattice in seconds and sentence error rate for our specialized grammar compared to our generic grammar. Numbers in parentheses denote total time and error counts for the test set.

The amount of work involved in domain customization is relatively small. The generic lexicon and grammar stay essentially the same across domains, and a KR ontology must be defined for the use of back-end reasoners anyway. We need to write the transforms to connect the LF and KR ontologies, but as their number is small compared to the total number of sense entries in the lexicon and the number of words needed in every domain, this represents an improvement over handcrafting custom lexicons for every domain.

6 Discussion

In previous work, we have relied on the built-in ontology support in the TRIPS system to provide ontology definitions, subtype unification and other information necessary in the customization process [Dzikovska *et al.*, to appear]. The built-in ontology facilitates efficient processing, but it limits the expressiveness of the language to the forms supported in the TRIPS knowledge representation (first-order logic and frames with simple slots). Using an external ontology such as KM allows us to utilize the richer representations that support reasoning, but comes at a performance cost of having to call the external ontology during parsing, which can potentially be expensive. However, with caching the results of the previous calls, or pre-compiling some of the information, we believe it should not significantly impair system performance.

In lexicon specialization, there is a tradeoff between the strength of the selectional restrictions and the lexicon size. The number of distinct lexical entries may increase in the specialized lexicon because there is no one-to-one correspondence between the LF and KR ontologies, so several transforms may apply to the same LF type depending on the semantic arguments that are filled. A new entry is created for every transform that applies, and during parsing the selectional restrictions propagated into the entries will effectively select the correct definitions. Table 2 shows that the number of lexicon entries increased from 1947 to 2048 (5%) in our Pacifica domain, and from 1947 to 1954 (0.3%) in our Medication Advisor domain.

In spite of the increased number of lexical entries, we believe that our approach offers significant benefits with respect to portability. It is difficult to quantify the effort of porting a system to a new domain, but as a first approximation, we list the tasks that are involved:

- **Define a domain model and the KR ontology.** This comes for free in a dialogue system, since the reasoning components need it in any case.
- **Define lexical entries for previously unseen words and add new type to the LF ontology as needed.** The number of words that need to be added diminishes as the lexicon and the LF ontology grow. We are also working on automatic methods to obtain new words from sources such as WordNet and FrameNet.
- **Modify the grammar as needed to cover new syntactic constructs.** We already have a wide-coverage grammar that can deal with a variety of constructs such as conjunction, long-distance dependencies, relative clauses and a variety of other phenomena common in speech. So the development of the grammar is a one-time investment that requires diminishing amount of work as more domains are added.
- **Define the LF-KR transforms.** This is the most time-consuming part of the process. However, the size of the transform set is proportional in size to the size of the domain, which is considerably smaller than the set of all lexical entries in the domain, and convenience features such as using the LF hierarchical structure and the lexical forms help speed up the process considerably. In addition, the separation between the linguistic and domain ontologies allows developers to write mappings in semantic terms, using descriptive labels assigned to semantic arguments such as AGENT, THEME, etc., at a level of abstraction that avoids addressing the details of the grammar and subcategorization frames such as those in COMLEX. They can also take advantage of the hierarchical structure in the domain-independent ontology and write mappings that cover large classes of words.
- **Define feature inference rules.** This is not required, but it is very helpful in improving parsing speed, especially when parsing speech inputs where out of domain words can be present. In the specialization process, we actively utilize the hierarchical structure of the KR ontology - a rule defined for a parent is inherited by all its children, so the rules need only be defined for a relatively small set of concepts.

Note that the first three items in the list above (defining a domain grammar, 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

choices in which a different pronoun, article or tense form were substituted, *e.g., can I tell my doctor* and *could I tell my doctor* were considered equivalent. However, equally grammatical substitutions of a different word sense, *e.g., drive the people* vs. *get the people* were counted as errors.

in the new domain, we can re-use the old lexicon and define LF-KR transforms and inference rules, the number of which is an order of magnitude smaller than the number of lexical items in the lexicon.

Another advantage of using a domain-independent semantinc representation that is later mapped to a domain-specific form is that it makes it easier to design domain-independent reference resolution and discourse management components. Many current ontologies do not have explicit support for encoding information needed for reference and dialogue processing, such as distinguishing between definite and indefinite descriptions. In our system, we use the LF representations generated by the parser to encode this information, which guarantees that the reference module can get it encoded consistently regardless of the specific language used by the back-end. During reference resolution, these descriptions are replaced with references to constants or sets known to the back-end reasoners, thus insulating them from the details necessary for reference resolution and discourse processing.

7 Conclusion

Our development of spoken dialogue systems in multiple domains has highlighted the need for a method of adapting a broad-coverage parser to different domain applications that allows us to preserve general language information common across domains and also produce interpretations customized to the needs of each domain. We have developed the method described here to meet this need. By maintaining two ontologies, one that is generalized for language representation, and another that is customized to the domain, and defining mappings between them, we can preserve our broad-coverage language components across domains as well as produce semantic representations that are optimally suited to the domain reasoners.

8 Acknowledgments

This material is based upon work supported by the Office of Naval Research under grant number N00014-01-1-1015 and the Defense Advanced Research Projects Agency under grant number F30602-98-2-0133. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of ONR or DARPA.

References

[Allen *et al.*, 1996] 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.

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

[Allen *et al.*, 2001] 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.

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

[Davidson, 1967] Donald Davidson. The logical form of action sentences. In Nicholas Rescher, editor, *The Logic of Decision and Action*, pages 81–95. University of Pittsburgh Press, Pittsburgh, 1967. Republished in Donald Davidson, *Essays on Actions and Events*, Oxford University Press, Oxford, 1980.

[Dzikovska *et al.*, to appear] 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.

[Ferguson *et al.*, 2002] G.M. Ferguson, J.F. Allen, N.J. Blaylock, D.K. Byron, N.W. Chambers, M.O. Dzikovska, L. Galescu, X. Shen, R.S. Swier, and M.D. Swift. The medication advisor project: Preliminary report. Technical Report 766, Computer Science Dept., University of Rochester, May 2002.

[Goddeau *et al.*, 1994] D. Goddeau, E. Brill, J. Glass, C. Pao, M. Phillips, J. Polifroni, S. Seneff, and V. Zue. Galaxy: A human-language interface to online travel information. In *Proc. ICSLP '94*, pages 707–710, Yokohama, Japan, September 1994. URL http://www.sls.lcs.mit.edu/ps/SLSps/icslp94/galaxy.ps.

[Gurevych *et al.*, 2003] Iryna Gurevych, Stefan Merten, and Robert Porzel. Automatic creation of interface specifications from ontologies. In *Proceedings of the HLT-NAACL Workshop on The Software Engineering and Architecture of Language Technology Systems (SEALTS)*, May 2003.

[Johnson and Fillmore, 2000] 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.

[Macleod *et al.*, 1994] 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.

[Miller and Schubert, 1988] Stephanie A. Miller and Lenhart K. Schubert. Using specialists to accelerate general reasoning. In Tom M. Smith, Reid G.; Mitchell, editor, *Proceedings of the 7th National Conference on Artificial Intelligence*, pages 161–165, St. Paul, MN, August 1988. Morgan Kaufmann.

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

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