

Customizing meaning: building domain-specific semantic representations from a generic lexicon

Myroslava O. Dzikovska, Mary D. Swift and James F. Allen

Computer Science Department

University of Rochester

Rochester, NY, USA, 14627

{myros,swift,james}@cs.rochester.edu

April 2003

Abstract. Language input to practical dialogue systems must be transformed into a semantic representation that is customized for use by the back-end domain reasoners. At the same time, we want to keep front-end system components as domain independent as possible for easy portability across multiple domains. We propose a transparent way to achieve domain specificity from a broad-coverage domain-independent parser. We maintain a domain-independent ontology and define a set of mappings from it into a domain-specific knowledge representation. We use the mappings to customize the semantic representations output by the parser for the reasoners, and to specialize the lexicon to the domain, which improves parsing speed and accuracy. This method facilitates our approach to instances of semantic type coercion common in our domains by combining lexical representations with domain-specific constraints on interpretation.

Keywords: parsing, semantic interpretation, domain specialization

1. Introduction

Most practical dialogue systems are developed for specific domains to maximize performance efficiency. Back-end system components use a knowledge representation tailored to application needs, and the language input must be converted into that representation. This is traditionally achieved by linking the lexical definitions directly to the concepts in the domain-specific ontology. This linking is also commonly used to bring in domain-specific selectional restrictions to increase parsing efficiency. In this approach, adapting the system to a new domain requires relinking the lexicon to the new ontology.

We propose an alternative method that is an easy, transparent way to achieve domain specificity from a broad-coverage deep parser. In our approach, we maintain two ontologies: domain-independent for the parser and domain-specific for the knowledge representation, and we define a set of mappings between domain-specific knowledge sources and the semantic representations generated by the parser. Our method allows us to easily obtain domain-specific semantic representations with-



© 2003 Kluwer Academic Publishers. Printed in the Netherlands.

out modifying the lexicon or grammar. We also use the mappings to specialize the lexicon to the domain, resulting in substantial improvement in parsing speed and accuracy. In this paper, we describe our customization method and illustrate how it facilitates our approach to semantic type coercion by combining lexical representations with domain-specific constraints on interpretation.

The customization method described here was developed in the process of adapting the TRIPS dialogue system (Allen et al., 2001) to several different domains, including a transportation routing system (Allen et al., 1996) and a medication scheduling system (Ferguson et al., 2002). We assume a dialogue system architecture (Allen et al., 2000) that includes a speech module, a parser, an interpretation manager (responsible for contextual processing and dialogue management), and a back-end application responsible for the general problem-solving behavior of the system. The system architecture is shown in Figure 1. Our philosophy is to keep all components as domain-independent as possible for easy portability, and to develop tools to facilitate component customization to different domains. In particular, our goal is to develop a parser and grammar that can handle language input from different application domains, but still retain speed and efficiency, and produce output suitable for domain-specific reasoning done by the behavioral agent and domain reasoners.

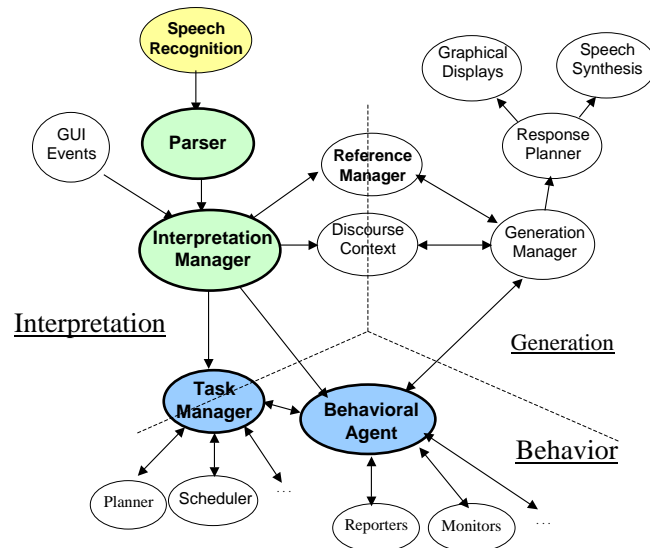


Figure 1. The architecture of the TRIPS dialogue system

2. Background

The most common approach for building robust and customizable parses for different domains is the use of semantic grammars. In a semantic grammar, the lexicon entries are linked to frames in a domain-specific representation. During parsing, lexical items are matched with frame slots they can fill, and unfilled slots can be queried from context. Probabilities of matching the words with frame slots can be trained on a corpus to achieve the best accuracy. For example, the TINA parser (Seneff, 1992) in the multi-domain dialogue system GALAXY (Goddeau et al., 1994) provides a mechanism for semi-automatically learning a probabilistic model from a training corpus. Similarly, parsers for information extraction use probabilistic models trained on text corpora to learn subcategorization frames and selectional restrictions in a given domain (*e.g.*, (Utsuro and Matsumoto, 1997)).

Semantic grammars offer a nice abstraction across a range of information retrieval applications, and robust language interpretation for small domains. They work best for the tasks which can be encoded by a small set of variables, with dialogue state encoded as a set of contexts. In a conversational interface where intention recognition is critical, and requires plan- and agent-based dialogue models, more detailed semantic representations are required, such as those that can be obtained with a broad-coverage deep grammar (Allen et al., 2001).

Linguistically motivated deep parsers have been used in dialogue systems, most notably the LINGO (Copestake and Flickinger, 2000) grammar used for VERBMOBIL project (Kay et al., 1994). However, such grammars have efficiency and accuracy problems caused by ambiguity. Syntactic constraints alone are not sufficient to disambiguate word meanings or structure, and when methods for handling speech fragments, such as bottom-up chart parsing, error correction rules and lattice-based inputs are added, they seriously aggravate already existing efficiency and accuracy problems of the current parsers.

When a large enough training corpus is available, speed and accuracy can be improved by adjusting probabilities of grammar rules to best reflect the corpus. However collecting training corpora is considerably more difficult and expensive for speech applications than for text applications, since it requires recording and transcribing live speech. When suitable training corpora are not available, a common solution is to use selectional restrictions to limit the search space or to filter out results that could not be disambiguated by the parser. This approach is employed, for example, in the GEMINI system (Dowding et al., 1993). Selectional restrictions encoded in the system lexicon consider-

ably speed up parsing and improve disambiguation accuracy in parsing of in-domain sentences.

However, another issue that needs to be addressed in porting dialogue systems between domains is linking the lexical entries to the domain semantic representation. TINA lexical entries specify the frames to which the words are linked, and GALAXY requires that all system components use a shared ontology. Therefore, for new domains the system lexicon needs to be re-linked to the new ontology. Similarly, GEMINI encodes the selectional restrictions in the system lexicon, which need to be changed for each new domain.

The AUTOSEM (Rosé, 2000) system makes re-linking easier by separating the re-usable syntactic information from the links to the domain ontology. It uses COMLEX (Macleod et al., 1994) as a source of reusable syntactic information. The subcategorization frames in the lexicon are manually linked 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. Rosé’s approach speeds up the process of developing tutoring systems in multiple domains. Similarly, McDonald (McDonald, 1996) maps the output of a partial parser to the semantic representation for information extraction to improve parsing speed and accuracy.

While AUTOSEM re-uses syntactic information across domains, it does not provide a way to re-use common semantic properties of words. In our approach, we introduce an intermediate layer of abstraction: a generic ontology for the parser (the **LF Ontology**) that is linked to the lexicon and preserved across domains. In this way, we preserve basic semantic features associated with lexical entries (*e.g.*, whether a word represents an event or an object) as well as some general selectional restrictions that do not change across our domains (*e.g.*, the verb *cancel* takes an action or event as an object argument). The parser uses this ontology to supply meaning representations of the input speech to the interpretation manager, which handles contextual processing and dialogue management and interfaces with the back-end application, as shown in Figure 1. The domain-specific ontology used for reasoning (the **KR ontology**) is localized in the back-end application. We then customize the communication between the parser/interpretation manager and the back-end application via a set of mappings between the LF and KR ontologies, as described in section 4.

Our method of separating domain-specific and domain-independent ontologies has a number of advantages. First, it allows developers to write mappings in semantic terms at a higher level of abstraction, so there is no need to address the details of the grammar and subcat-

egorization frames such as those used in COMLEX. Developers can instead use descriptive labels for semantic arguments, such as AGENT, THEME, etc. Second, it allows developers to take advantage of the hierarchical structure of the domain-independent ontology and write mappings that cover large classes of words (see example in section 4). Third, the mappings are used to convert the generic representation into the particular form utilized by the back-end application, either a frame-like structure or a predicate logic representation, without changing the grammar rules, as described in (Dzikovska et al., 2002). Finally, the lexicon is specialized to the domain via the mappings, which both improves parsing speed and accuracy and provides a transparent way to map lexical forms to domain-specific meanings.

3. Domain-independent representation

3.1. THE LF ONTOLOGY

Entries in the generic lexicon are linked to the LF ontology, a domain-independent ontology for the parser. The LF ontology is kept as general as possible so it can be used across multiple domains. The LF ontology consists of a set of representations (LF types) that classify entities corresponding to (classes of) lexical items in terms of argument structure and selectional restrictions, with a hierarchical structure inspired by FRAMENET (Johnson and Fillmore, 2000). Every LF type declares a set of thematic arguments with selectional restrictions. The LF ontology is used in conjunction with a unification-based grammar that covers a wide range of syntactic structures.

The LF types are organized in a single-inheritance hierarchy. We implement multiple inheritance via semantic feature vectors associated with each LF type. The features correspond to basic meaning components and are based on the EuroWordNet (Vossen, 1997) feature system with some additional features we have found useful across domains. While the same distinctions 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 semantic feature vectors allows us to easily augment semantic information associated with a lexical entry during the customization process and the semantic coercion operations described below. Our choice of the LF types and semantic features included in the LF ontology is linguistically motivated, an approach similar to Lascarides and Copestake (Lascarides and Copestake, 1998), who encode the knowledge “appropriate as a

locus for linguistically-relevant generalizations” in their lexical semantic hierarchy.

The semantic features are organized into five basic clusters: *ft_phys-obj* which are the objects that can be seen and felt in real world, *ft_situation*, which include states and events, *ft_time*, which are references to times, and *ft_abstr-obj*, which are other abstract objects and properties, e.g., *weight*; and *ft_proposition*, which denote entities which can be true or false, denied or confirmed, e.g. ideas, plans. Some of the features associated with physical objects, situations and times are shown in Figure 2

For physical objects, we mostly kept the features defined in EuroWordNet. These are *form*, which differentiates solid objects from substances, *origin*, which differentiates natural (living and non-living) things from artifacts, and *object-function*, which classifies the objects by the main function they perform in real world. We additionally defined feature *mobility* to handle the distinctions between the objects that are inherently fixed, e.g., cities and oceans, and objects that can be easily moved, e.g., people and trucks. Feature *spatial-abstraction* is intended for handling spatial properties of objects (Dzikovska and Byron, 2000), to capture the distinctions between objects such as roads, which are best visualized as lines or strips, and cities, which can be visualized as either points on the map or larger regions with defined area and borders.

We use the features *aspect* and *time-span* to represent event structure based on Moens and Steedman (1987). The *cause* feature can be *Force*, either an intentional agent or a natural phenomenon, *Stimulating*, set for verbs which tend to have an experiencer role, such as *see*, *smell*, and *Mental*, which is some internal non-agentive process, such as *love* or *hate*. We also added the *trajectory* feature to differentiate the situation types that can be easily modified by trajectory adverbials such as FROM-LOC and TO-LOC.

For times, the *function* feature is designed to describe the most common functions of temporal expressions we observed in our corpora.

Our feature dimensions are generally orthogonal, but there are dependencies between some feature values. For example, if something is marked as a human being, it is also a solid object. We express these dependencies in feature inference rules, such as the one shown in Figure 3. When a feature vector is processed in the system, if the origin feature is set to (*origin human*), the form feature, if not already specified, will be automatically filled with (*form solid-object*). This mechanism is also used for specializing feature vectors to the domain as described in section 4.2.

FT_Phys-obj	FT_Situation
Form	Aspect
<i>Substance</i>	<i>Static</i>
<i>solid, liquid, gas</i>	<i>indiv-level, stage-level</i>
<i>Object</i>	<i>Dynamic</i>
<i>solid-object, hole,</i>	<i>Bounded, unbounded</i>
<i>geo-object, enclosure</i>	Time-span
Origin	<i>atomic, extended</i>
<i>Natural</i>	Cause
<i>living</i>	<i>Force</i>
<i>human, plant</i>	<i>agentive, phenomenal</i>
<i>animal, creature</i>	<i>Stimulating</i>
<i>non-living</i>	<i>Mental</i>
<i>artifact</i>	Trajectory (+/-)
Object-function	FT_time
<i>covering, comestible, etc.</i>	Function
Mobility	<i>Time-of-day</i>
<i>fixed</i>	<i>clock-time, day-part</i>
<i>movable</i>	<i>Time-of-year</i>
<i>self-moving</i>	<i>frequency</i>
<i>non-self-moving</i>	<i>time-interval</i>
Information (+/-)	<i>time-unit</i>
Intentional (+/-)	
Container (+/-)	
Spatial abstraction	
<i>line, strip, point, region</i>	

Figure 2. Some feature set dimensions in the TRIPS domain-independent lexicon

(origin human) => (form solid-object) (intentional +)

Figure 3. An example feature inference rule in the TRIPS lexicon

3.2. THE DOMAIN INDEPENDENT LEXICON

Word senses are treated as leaves of the semantic hierarchy. The following information is specified for every word sense in the lexicon:

- Syntactic features such as agreement, morphology, etc.;
- LF type;
- The semantic feature vector (mostly inherited from LF type definition)
- The subcategorization frame and syntax-semantics mappings.

To illustrate, consider the definition for the verb *take* in the sense *to consume substances*, as in *take aspirin*. The LF type definition for the *consume* sense of *take* is shown in Figure 4. It specifies a generic semantic feature vector associated with the type and selectional restrictions on its arguments. Intuitively, LF_Consume defines a dynamic event in which some physical object (AGENT) consumes some substance. The lexicon entry for *take* (shown in Figure 5) is linked to the LF type definition by mapping the syntactic roles to the semantic argument labels in the LF type. The selectional restrictions specified in the LF type arguments are propagated into the lexicon. When trying to create a verb phrase, the parser checks that the semantic feature vector specified in the argument restriction matches the semantic feature vector associated with the noun phrase. Thus, only noun phrases marked as substances in their feature vectors (*form substance*) are accepted as direct objects of verb *take* in the *consume* sense.

The parser uses the LF types to construct a general semantic representation (the base logical form) of the input language, which is a flattened and unscoped logical form using reified events (Davidson, 1967). A base logical form for *take aspirin* is shown in Figure 6. The representation is a conjunction 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 and speech act information from our examples for simplicity.

4. Constructing domain-specific representations

4.1. CUSTOMIZING PARSER OUTPUT

To produce domain-specific KR representations from the base logical form, we developed a method to customize parser output. The current

```
(define-type LF_CONSUME
:semfeatures (Situation (aspect dynamic) (cause force))
:arguments (AGENT (Phys-obj (form object)))
            (THEME (Phys-obj (form substance))))
```

```
(define-type LF_DRUG
:semfeatures (Phys-obj (form substance)))
```

Figure 4. LF type definitions for LF_CONSUME and LF_DRUG.

```
(take
:lf LF_CONSUME*take
:semfeatures (Situation (aspect dynamic) (cause force)))
:subject (NP (Role AGENT)
            (Restriction (Phys-obj (form object))))
:object (NP (Role THEME)
            (Restriction (Phys-obj (form substance))))

(aspirin
:lf LF_DRUG*aspirin
:semfeatures (Phys-obj (form substance) (origin artifact)))
```

Figure 5. Lexicon entries for *consume* sense of *take* and for *aspirin*.

system supports two knowledge representation formalisms often used by reasoners: a frame-like formalism where types have named slots, and a representation that has predicates with positional arguments. For this paper, we assume a frame representation used by the reasoners.

We use LF-to-frame transforms to convert from base logical form into a frame representation. These transforms specify the KR frame that the LF type maps to, the mappings between LF arguments and KR slots, and additional functions that can be applied to arguments during the transform process. These transforms can be simple and name the slot into which the value is placed, as in Figure 7, or more elaborate

```
(F V51016 LF_CONSUME*take
   :THEME V51380 :AGENT V51537)
(IMPRO V51537 :CONTEXT-REL +YOU+)
(A V51380 LF_DRUG*aspirin)
```

Figure 6. LF representation of *take aspirin*.

```

(a) (LF-to-frame-transform takemed-transform
      :pattern (LF_CONSUME TAKEMED)
      :arguments (AGENT :ACTOR)
                 (THEME :MEDICATION))

      (LF-to-frame-transform drug-transform
      :pattern (LF_DRUG :LF-FORM
                     (lf-form-default MEDICINAL-SUBSTANCE))

(b) (define-class TAKEMED
      :isa ACTION
      :slots (:ACTOR PERSON)
             (:MEDICATION MEDICINAL-SUBSTANCE))

      (define-class ASPIRIN
      :isa MEDICINAL-SUBSTANCE)

(c) (TAKEMED v123)
      (:ACTOR v123 +USER+) (:MEDICATION v123 v456)
      (ASPIRIN v456)

```

Figure 7. LF-to-frame-transform. (a) Transforms for LF_CONSUME and LF_DRUG types; (b) Definition of KR class TAKEMED that the transform maps into, and the class ASPIRIN, into which LF_DRUG*aspirin will be mapped; (c) The KR frame that results from applying the transform in (a) to the consume event representation in Figure 6.

and specify an operator expression that is applied to the value, which we use, for example, for semantic type coercion, described in section 5.

The Interpretation Manager takes the parser's base logical form, determines the most specific transform consistent with it, and uses the transform to convert the base representation into the domain knowledge representation.

To illustrate, consider the base logical form for *take aspirin* in Figure 6. The applicable transform is `takemed-transform` (Figure 7a), and when the Interpretation Manager applies it, the frame shown in Figure 7c is the result. The details of the process and the way transforms are used to adapt the base logical form to different possible formalisms used by reasoners are described in (Dzikovska et al., 2002). Note that this single transform covers a class of words. For example, the same transform also covers *have* used in the *consume* sense, as in *have an aspirin every day*.

Transforms can also utilize the lexical form of a word. For example, medication names are all grouped as leaves under the `LF_Drug` type in our ontology, but the transform shown in Figure 7(a) for medicines uses the lexical form of the item transformed to determine the correct KR class name for the mapping. In the figure, the `:LF-FORM` keyword indicates that when an entity of type `LF_DRUG` is transformed, the lexical form of the item will be taken as the class name. For example, when `LF_DRUG*aspirin` undergoes this transform, it will be converted to an instance of KR class `ASPIRIN`. If a class with the corresponding name does not exist, then the default class supplied in `:lf-form-default` argument will be used.

4.2. LEXICON SPECIALIZATION

We use the transforms described above in a post-processing stage to customize the generic parser output for the reasoners. We also use them in a pre-processing stage to specialize the lexicon, which speeds up parsing and improves semantic disambiguation accuracy by integrating the domain-specific semantic information into the lexicon and grammar.

We pre-process every entry in the lexicon by determining all possible transforms that apply to its LF type. For each transform, we create a new sense definition identical to the existing generic definition plus a new feature *kr-type* in its semantic vector. The value of *kr-type* is the KR ontology class that results from applying this transform to the entry. Thus, we obtain a (possibly larger) set of entries which specify the KR class to which they belong. We then propagate type information into the syntactic arguments, making tighter selectional restrictions in the lexicon. This allows us to control the parser search space better and obtain greater parsing speed and accuracy.

When specializing the lexicon to the medical domain, given the definition of the verb *take* and `LF_Consume` in Figure 4, and the definitions in Figure 7, the system determines the applicable LF-to-frame-transform, `takemed-transform`, and adds (*kr-type takemed*) to the feature vector of *take*. Next, it applies the argument mappings from the transform. For example, the mappings specify that the LF argument `THEME` maps to KR slot `:medication`, and therefore should be restricted to medicinal substances. Since `THEME` is realized as a direct object of *take*, (*kr-type medicinal-substance*) is added to the semantic vector in the object restriction. Similar transforms are applied to the rest of the arguments.

As a result, a new definition of *take* with stricter selectional restrictions is added to the lexicon, and suitable objects of *take* must not only be substances, but also identified as medicines. Similarly, the

definition of *aspirin* is specialized using the **drug-transform** rule in Figure 7a, and will be mapped to ASPIRIN, which is a subclass of KR type MEDICINAL-SUBSTANCE. The new definition is shown in Figure 8, with added features shown in bold.

```
(take
:lf LF_CONSUME*take
:semfeatures (Situation (aspect dynamic) (cause agentive)))
:subject (NP (Role Agent)
  (Restriction (Phys-obj ((intentional +)
    (origin human) (form solid-object)
    (kr-type Person))))))
:object (NP (Role Theme)
  (Restriction (Phys-obj (form substance)
    (kr-type Medicinal-Substance))))))
```

Figure 8. The specialized entry for *take* in the CONSUME sense. The features changed or added during specialization are shown in bold.

In the process of specialization, the parser uses a feature inference mechanism described in section 3.1 as follows. For the KR specialization process we add rules that declare dependencies between the values of *kr-type* feature and the values of domain independent features. Two sample rules for PERSON and MEDICINAL-SUBSTANCE values are shown in Figure 9. These feature rules, together with the domain-independent rule from 3, are used to further specialize the feature vectors in the lexical entry. In this case, this results in changing the generic (*origin living*) restriction on the agent of *take* to the more specific (*origin human*) value. The domain-specific feature inference is particularly useful in the way we handle semantic type coercion, described in section 5.

4.3. EVALUATION

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 transportation domain.

```
(kr-type Person) => (phys-obj (origin human))
(kr-type Medicinal-substance) => (phys-obj (form substance))
```

Figure 9. Feature inference rules used to derive feature vectors for kr-type PERSON and kr-type SUBSTANCE in the medadvisor domain

In Table I describes the lexicon and ontologies used in these domains. The results presented in Table II 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.¹

A part of a lattice for the sentence *that looks good* is shown in Figure 10. The highest scoring sequence is *that looks gave it*, but it 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 improvement comes from two sources. The tighter selectional restrictions limit the search space and help to correctly disambiguate according to our domain knowledge. In addition, our parser has preference values associated with different senses, and correspondingly with constituents that use those senses. We increase the preference values for specialized entries, so they are tried first during parsing, which helps the parser find an interpretation for in-domain utterances faster.²

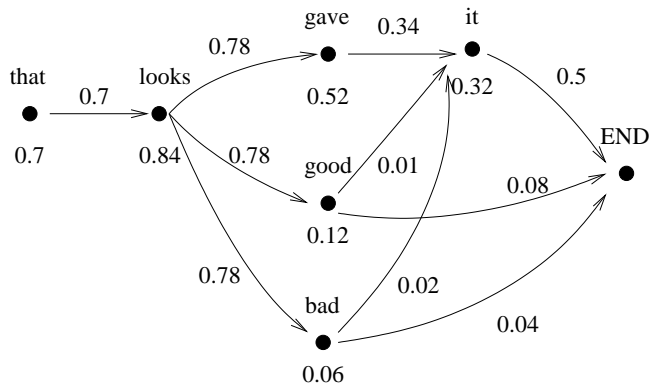


Figure 10. A sample lattice for *that looks good*, misrecognized as *that looks gave it*. Numbers at nodes denote confidence scores for lexical items, numbers on arcs denote the confidence scores on transitions.

The amount of work involved in domain customization is relatively small. The 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

¹ Choices in which a different pronoun, article or tense form were substituted, e.g., *can/could I tell my doctor* were considered equivalent, but grammatical substitutions of a different word sense, e.g., *drive/get the people* were counted as errors.

² Unspecialized entries have a lower preference, so parses for out of domain utterances can be found if no domain-specific interpretation exists.

Table I. Some lexicon statistics in our system

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

Table II. 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.

	Transportation	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)

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 (see Table I), this represents an improvement over hand-crafting custom lexicons for every domain.

5. Domain-specific coercion

Certain semantic type coercions are frequent in our domains. For example, in our medical adviser domain, the word *prescription* frequently appears in contexts that require a word for (a type of) medication, as in (1) *Which prescriptions do I need to take?* Intuitively, this is understood to mean (2) *Which medications specified by my prescriptions do I need to take?* We have adopted a practical approach to such coercions by applying a domain-specific operator to the mismatched argument to produce an entity of the coerced type. While this is a restricted approach compared to, *e.g.*, (Pustejovsky, 1995), (Lascarides and Copestake, 1998), we adopt it as a transparent method of handling our domain-specific coercions in the most efficient way for our system.

The first problem is for the parser to recognize (1) as a valid utterance in spite of the semantic type mismatch, since *prescription* is

not semantically typed as a consumable substance, which is required for an argument of *take* in its *consume* sense. An approach frequently taken by robust parsers (*e.g.*, (Rosé, 2000)) is to relax the constraints in case of a type mismatch. However, if we need to construct a semantic representation for this utterance that is suitable for use by the back-end reasoners, the problem is deeper than just finding a parse tree. If the literal meaning of *prescriptions* is used during the interpretation process, the query asking for prescription objects consumed by the user (Figure 11) would be sent to the medication knowledge base, eliciting an empty result, since *prescriptions* are not recognized in the knowledge base as consumable objects.³

- (a) ASK ?y
 (SET-OF ?y ?x (PRESCRIPTION ?x))
 (TAKEMED v123)
 (:ACTOR v123 +USER+) (:MEDICATION v123 ?x)
- (b) (SET-OF ?y ?x
 (PRESCRIBED-MEDICATION (PRESCRIPTION ?x)))

Figure 11. (a) Query for *Which prescriptions do I need to take* with no type coercion; (b) representation for *prescriptions* coerced to *medication*.

A correct interpretation needs a semantic representation for (1) that resembles the semantic representation for (2). For this, additional information must be interpolated - in particular, the relation that holds between prescriptions and medications. We accomplish this in our framework with a library of coercion rules. To handle the semantic type coercion of *prescription* to *medication*, we define the following coercion rule, which uses a *prescribed-medication* operator (known to the knowledge base) to declare that prescriptions can be coerced into medications, in Figure 12a.

During the lexicon specialization process, information from coercion rules is propagated into the lexical entries that have domain-specific mappings. When the lexicon is specialized for the medical domain, the lexical entry for *prescription* has a nonempty coercion feature (Figure 12b).

³ Arguably, reasoning about prescriptions as consumables could be implemented in the knowledge base. However, in our system the context information needed to resolve some instances of coercion is localized in the intention recognition module, and using operators handled by the intention recognition is a way to take it into account. Such implementation differences between components with different reasoning capabilities provide another justification for the need to specialize parser output for different application back-ends.

6. Conclusion

Our method of parser customization allows us to maintain a domain-independent lexicon and grammar for improved domain coverage and portability, and at the same time provides a straightforward mechanism for constructing custom semantic representations that are optimally suited for specific domain reasoners. Our lexicon specialization process improves parsing speed and accuracy by using custom domain knowledge to boost domain-specific word senses, tighten selectional restrictions on arguments and reduce search space during parsing. We also use the domain specific information to handle semantic type coercions common in our domains in a computationally efficient manner.

Acknowledgements

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, and 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, J., D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent: 2000, 'An Architecture for a Generic Dialogue Shell'. *NLENG: Natural Language Engineering, Cambridge University Press* **6**(3), 1–16.
- Allen, J., D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent: 2001, 'Towards Conversational Human-Computer Interaction'. *AI Magazine* **22**(4), 27–38.
- Allen, J. F., B. W. Miller, E. K. Ringger, and T. Sikorski: 1996, 'A Robust System for Natural Spoken Dialogue'. In: *Proceedings of the 1996 Annual Meeting of the Association for Computational Linguistics (ACL'96)*.
- Byron, D. K.: 2002, 'Resolving Pronominal Reference to Abstract Entities'. Ph.D. thesis, University of Rochester.
- Copestake, A. and D. Flickinger: 2000, '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.
- Davidson, D.: 1967, 'The Logical Form of Action Sentences'. In: N. Rescher (ed.): *The Logic of Decision and Action*. Pittsburgh: University of Pittsburgh Press, pp. 81–95.

- Dowding, J., J. M. Gawron, D. Appelt, J. Bear, L. Cherny, R. Moore, and D. Moran: 1993, 'GEMINI: A natural language system for spoken-language understanding'. pp. 54–61.
- Dzikovska, M., J. F. Allen, and M. D. Swift: 2002, 'Finding the balance between generic and domain-specific knowledge: a parser customization strategy'. In: *Proceedings of LREC 2002 Workshop on Customizing Knowledge for NLP applications*.
- Dzikovska, M. O. and D. K. Byron: 2000, 'When is a union really an intersection? Problems resolving reference to locations in a dialogue system'. In: *Proceedings of the GOTALOG'2000*. Gothenburg.
- Ferguson, G., J. Allen, N. Blaylock, D. Byron, N. Chambers, M. Dzikovska, L. Galescu, X. Shen, R. Swier, and M. Swift: 2002, 'The Medication Advisor Project: Preliminary Report'. Technical Report 766, Computer Science Dept., University of Rochester.
- Goddeau, D., E. Brill, J. Glass, C. Pao, M. Phillips, J. Polifroni, S. Seneff, and V. Zue: 1994, 'Galaxy: A Human-Language Interface to On-line Travel Information'. In: *Proc. ICSLP '94*. Yokohama, Japan, pp. 707–710.
- Johnson, C. and C. J. Fillmore: 2000, 'The FrameNet tagset for frame-semantic and syntactic coding of predicate-argument structure'. In: *Proceedings ANLP-NAACL 2000*. Seattle, WA.
- Kay, M., J. M. Gawron, and P. Norvig: 1994, *VerbMobil: A Translation System for Face-To-Face Dialog*. Stanford, California: CSLI Press.
- Lascarides, A. and A. Copestake: 1998, 'Pragmatics and Word Meaning'. *Journal of Linguistics* **34**(2), 387–414.
- Macleod, C., R. Grishman, and A. Meyers: 1994, 'Creating a Common Syntactic Dictionary of English'. In: *SNLR: International Workshop on Sharable Natural Language Resources*.
- McDonald, D. D.: 1996, 'The interplay of syntactic and semantic node labels in partial parsing'. In: H. Bunt and M. Tomita (eds.): *Recent Advances in Parsing Technology*. Kluwer Academic Publishers, pp. 295–323.
- Miller, S. A. and L. K. Schubert: 1988, 'Using Specialists to Accelerate General Reasoning'. In: T. M. Smith and R. G. Mitchell (eds.): *Proceedings of the 7th National Conference on Artificial Intelligence*. pp. 161–165.
- Moens, M. and M. Steedman: 1987, 'Temporal Ontology in Natural Language'. In: *Proceedings of the 25th Annual Conference of the Association for Computational Linguistics*. pp. 1–7, Association for Computational Linguistics.
- Pustejovsky, J.: 1995, *The Generative Lexicon*. Cambridge, Massachusetts: The MIT Press.
- Rosé, C.: 2000, 'A Framework for Robust Semantic Interpretation'. In: *Proceedings 1st Meeting of the North American Chapter of the Association for Computational Linguistics*.
- Seneff, S.: 1992, 'TINA: A Natural Language System for Spoken Language Applications'. *Computational Linguistics* **18**(1), 61–86.
- Utsuro, T. and Y. Matsumoto: 1997, 'Learning Probabilistic Subcategorization Preference by Identifying Case Dependencies and Optimal Noun Class Generalization Level'. In: *Proceedings of 5th ANLP Conference*.
- Vossen, P.: 1997, 'EuroWordNet: a multilingual database for information retrieval'. In: *Proceedings of the Delos workshop on Cross-language Information Retrieval*.