

A Practical Semantic Representation For Natural Language Parsing

by

Myroslava O. Dzikovska

Submitted in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Supervised by

Professor James F. Allen

Department of Computer Science

The College

Arts and Sciences

University of Rochester

Rochester, New York

2004

Curriculum Vitae

The author was born on November 1st 1976 in Lviv, Ukraine. She attended the University of Lviv from 1992 to 1997, and graduated with a Bachelor of Arts in Applied Mathematics and Informatics in 1997. She received International Soros Science Education Program Undergraduate Fellowship in 1995. She came to the University of Rochester in the Fall of 1997 and began her studies in computer science. She pursued her research in natural language understanding under the direction of Professor James Allen and received the Master of Science degree from the University of Rochester in 1999.

Acknowledgments

When I came to graduate school, I wanted to do science, and expected to enjoy it. I indeed greatly enjoyed my experience at the University of Rochester Computer Science department, but when I started, I didn't really expect so many bumps in the road. This thesis would not be possible without support of many people.

I thank my advisor, James Allen, for his unfailing support throughout the years. He was great in letting me do what I wanted, helping me when things were not going well, encouraging me and putting my welfare first. He has great insight, the ability to make the complex ideas easy and the research fascinating. The TRIPS research group he created is a wonderful and exciting group to work in, thanks to his leadership. As my advisor, he was incredibly patient with my many failings, and his trust in my abilities helped me overcome many difficulties. I knew I could always rely on his help and support, and I very much appreciate it.

My sincerest thanks go to my committee members, Len Schubert, Jason Eisner and Greg Carlson. They all were there for me with advice both on my research and on my career, and contributed ideas to this thesis. In my five semesters as his teaching assistant, Len listened to me, supported me and taught me a lot of what I know about artificial intelligence and natural language understanding.

My internship supervisor, Candy Sidner at Mitsubishi Electric Research Laboratories, advised and supported me both professionally and personally, and made a big difference in my life.

I would like to acknowledge my professors at Applied Mathematics and Informatics and at Mechanics-Mathematics departments at the University of Lviv. They strove to provide the excellent quality of education under less than ideal conditions, and gave me knowledge and skills essential to my research. Their dedication to teaching is an inspiration for me. Especially I want thank my undergraduate thesis supervisors, Oksana Vasylivna Kostiv and Mykola Mykolajovych Voytovych, who helped me get involved in research early on, and supported me in getting

to graduate school here. Also, the researchers from the Institute of Applied Physics at Lviv University, for providing a research topic and help for my first conference paper.

All of my work was done as part of the TRIPS group, and I thank all its past and current members for the ideas and help I had over the years, and in particular James Allen, George Ferguson, Nate Blaylock, Donna Byron, Ellen Campana, Nate Chambers, Lucian Galescu, Paul McCarty, Amanda Stent, Scott Stoness, Bob Swier, Mary Swift and Joel Tetreault. I am especially grateful to Donna Byron and Mary Swift for letting me talk, teaching me how to write well, and helping me shape my ideas in a way that could be understood by others. Thanks also to Jessica Bayliss, who read my area paper, even though it was not her area. This was the first paper in English I ever had to write, and her editing was both helpful and instructive.

I received a lot of help from the Department of Computer Science staff. Our secretaries and administrators, Elaine Herberle, Marty Guenther, Peggy Meeker, JoMarie Carpenter, Eilieen Pullara and Jill Forster helped me with paperwork, reminded me of deadlines, and otherwise worked to make sure that everything went smoothly. Our support staff, Dave Costello, Jim Roche and Liudvikas Bukys, solved all sorts of technical difficulties for me.

The undergraduate students, Bob Swier, Mike Lehr, David Easwaran and Preethum Prithviraj developed the Java tools I constantly use in my research. Thank you also to Meghan Hurley and Michael Isman who annotated some of the data I used in my evaluation.

I was blessed with many people in my personal life who helped me get here. My mother supported and encouraged me, and came here when I was sick or just needed help. In Rochester, I am very grateful to Roman and Nadia Tratch for giving me “home away from home”; to Fr. Patrick Cowles and my friends from Orthodox Christian Fellowship for good advise and encouragement; to Laura Flamand, for hugs and steadfast friendship; to Amy, Eva, Kara, Laura and Stacey for being there when I needed it the most. And, of course, to everyone at the Computer Science Department over the years, for making it a great place to be.

This material is based upon work supported by a grant from Office of Naval Research under grant number N00014-95-1-1088, a grant from Defense Advanced Research Projects Agency under grant under number F30602-98-0133, a grant from DOD/Navy under grant #N00014-01-1-1015, a grant from Defense Advanced Projects Agency’s subcontract to SRI under award number 03-000223, fellowship at University of West florida and the National Science Foundation under grant number EIA-0080124. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of these organizations.

Abstract

This thesis deals with the problem of building fast, accurate and portable parsers for natural language understanding. Our focus is a multi-domain dialogue system in which we need a deep linguistically-motivated parser to produce the representations of the input suitable for reasoning. In this dissertation, we are concerned with building parsers which have the wide coverage and portability offered by a general syntactic grammar without sacrificing parsing speed and accuracy.

Our approach relies on a domain-independent deep parser and grammar which uses selectional restrictions to control parsing speed and accuracy. We develop a feature list representation as the basis for selectional restrictions, and a formal model for using selectional restrictions in a unification based framework.

We then develop a lexicon design for multi-domain parsing and semantic interpretation. We show how the restrictions based on feature sets can be integrated with a traditional frame-based semantics, and extend our formal model to cover inheritance and defaults in the lexicon. We show that none of the existing large-scale ontologies and lexicons provide all the information necessary for parsing and semantic disambiguation, and we develop a parsing lexicon suitable for use with a wide-coverage grammar in multiple domains.

Our domain-independent lexicon provides coverage and portability over four different application domains. To customize the representations produced by the parser for domain reasoning, we designed an architecture with mappings between our domain-independent ontology and a domain model. This architecture allows us to produce semantic representations optimally suited for different application domains. In addition, we use the mappings to specialize the lexicon for improved parsing speed and accuracy, and show that our specialization method significantly improves parsing performance.

Finally, we develop a statistical model to learn selectional restrictions from corpora and show how it can be used to distinguish between acceptable and unacceptable verb-object pairs in data sets derived from our lexicon and from the World Street Journal corpus.

Table of Contents

Curriculum Vitae	ii
Acknowledgments	iii
Abstract	v
List of Tables	viii
List of Figures	xi
1 Introduction	1
1.1 Parsing and Semantic Interpretation	2
1.2 Semantic Representation Issues	6
1.3 Semantic disambiguation with selectional restrictions	8
1.4 Parsing and interpretation in a practical dialogue system	10
1.5 Lexicons and Ontologies	13
1.6 Summary	18
2 Features in a Domain Independent Ontology	20
2.1 Background	21
2.2 The TRIPS feature system	24
2.3 Using features in selectional restrictions	40
2.4 Discussion	58
2.5 Evaluation	62
2.6 Conclusions and Future Work	69

3	A domain-independent ontology and lexicon for parsing natural language	72
3.1	Motivation and Background	73
3.2	Domain-independent ontology	83
3.3	The Generic Lexicon	85
3.4	Additional ontology and lexicon design issues	88
3.5	A formal model for ontology and lexicon inheritance	98
3.6	Discussion	104
3.7	Conclusions	117
4	Using features for domain specialization	118
4.1	Background	119
4.2	Linguistic ontology versus a domain model	121
4.3	From LF representation to domain-customized KR representation	124
4.4	Implementing the transforms	142
4.5	Lexicon Specialization	149
4.6	Evaluation	155
4.7	Discussion	160
4.8	Conclusions	166
5	Statistical Methods for Learning Features	168
5.1	Background	168
5.2	Probability Model	176
5.3	Training	184
5.4	The Iterative Maximization Algorithm	189
5.5	Learning Experiments	196
5.6	Discussion and Future Work	211
5.7	Conclusions	216
6	Conclusions and Future Work	217
6.1	Future work	220
	Bibliography	223

List of Tables

2.1	Physical Object features in the TRIPS system	27
2.2	Situation features in the TRIPS system	32
2.3	Abstract object features in the TRIPS system	35
2.4	Selectional restrictions for our motivating examples	51
2.5	Selectional restrictions with conjunctions when \forall is used to create feature sets for collective NPs. For brevity, we only provide values of <i>Origin</i> feature.	53
2.6	Selectional restrictions with conjunctions when collective set formation \square is used to create feature sets for collective NPs. For brevity, we only provide values of <i>Origin</i> feature.	56
2.7	Selectional restrictions with conjunctions when collective set formation \square is used to create feature sets for verb phrase conjunctions. For brevity, we only provide values of <i>Origin</i> feature.	57
2.8	Feature types used in grammar and selectional restrictions	63
2.9	Physical object features in selectional restrictions	64
2.10	Features shared between physical objects and other types in selectional restrictions	65
2.11	Abstract object features in selectional restrictions	66
2.12	Situation features in selectional restrictions	67
2.13	Time features in selectional restrictions	68
2.14	The comparison of parsing results with and without selectional restrictions in the Medication Advisor domain	70
3.1	Large scale lexical databases.	79
3.2	The types of syntactic arguments in the templates	87

3.3	Lexicon statistics in our system	106
3.4	The semantic roles in the TRIPS system which appear only in subcategorization frames.	108
3.5	The semantic roles in the TRIPS system which can either be subcategorized arguments or adverbial modifiers.	109
3.6	Corpus Statistics	110
3.7	Coverage statistics for our Medadvisor corpus	111
3.8	Sample sentences from our test corpora	112
3.9	Examples of repairs and ungrammatical utterances in our test corpora	113
4.1	Design considerations for LF and KR ontologies.	124
4.2	Choices in representing modifiers to adverbial adjuncts. (a) In TRIPS WKB, <i>straight</i> fills a slot in the <i>path</i> frame; (b) In KM, <i>straight</i> fills a slot in the main MOVE frame; (c) In some cases, modifiers may be ignored, we use KM syntax as an example	140
4.3	The abstract operations in the transform application algorithm	145
4.4	Lexicon statistics.	157
4.5	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.	157
4.6	Example sentences picked out of the same lattice with and without specialization	158
5.1	Sample property functions for modeling feature sets. Properties can express features assigned to words, selectional restrictions, and dependencies between feature values.	178
5.2	Properties with non-zero values for the word pair $\langle \textit{send}, \textit{truck} \rangle$ annotated with features (<i>Main-type Phys-obj</i>), (<i>Mobility Movable</i>), (<i>Aspect -</i>)	182
5.3	The data sets for our learning experiments	199
5.4	The comparison of feature weights for <i>remove</i> , <i>snowstorm</i> and <i>orange</i>	201
5.5	Training sets with different amounts of seeding in the TRIPS-lex corpus. The columns show the number of nouns for which seeds were retained, the corresponding number of training data pairs, and the total percentage of data annotated.	202

- 5.6 PWSD test accuracy on the TRIPS-lex set. “1” in features column denotes models using a single *Main-type* feature, and “1+12” denotes a 12 feature models initialized from the 1-feature model. The differences in PWSD accuracy of more than 1.6 percentage points are statistically significant at the 0.05 level. The best results are highlighted in bold. Hofmann row describes the performance of the class-based Hoffman model. 203
- 5.7 Evaluation on seeded and unseeded data with the seeded-244 model. The “Unseeded” columns contain PWSD accuracy and perplexity calculated over the test pairs for which the nouns did not have seeds in the training data; the “Seeded” column is over test pairs for which the noun objects had seeds in the training data. Differences of PWSD accuracy over 2.3 percentage points are statistically significant at the 0.05 level. 207
- 5.8 PWSD test accuracy on the WSJ2 set. The table shows that adding a prior improves both PWSD test accuracy and model perplexity. The differences in PWSD accuracy of more than 2.6 percentage points are significant at 0.05 level. The best 3 and 12 feature models and the comparable class model are highlighted in bold. The difference in PWSD results for those models is not statistically significant at the 0.05 level 209
- 5.9 PWSD test accuracy on the WSJ2-ann set. The differences of more than 10 percentage points are statistically significant. 209

List of Figures

1.1	A syntactic parse of <i>I saw a bird with yellow feathers</i>	3
1.2	A possible frame representation for <i>I saw a bird with yellow feathers</i>	4
1.3	Sample dialogue in a medication advisor domain	5
2.1	Physical objects and their features in TRIPS-Monroe	32
2.2	Words classified as situations in TRIPS-Monroe	35
2.3	Abstract objects in the TRIPS-Monroe lexicon	36
2.4	Times in TRIPS-Monroe lexicon	38
2.5	The list of feature dependencies in the TRIPS feature system	40
2.6	Simple feature sets and selectional restrictions	41
2.7	A feature set for the word <i>Person</i> we will use in our examples	46
3.1	The LF representation of the sentence <i>load the oranges into the truck.</i>	75
3.2	LF type definitions for physical objects in the LF ontology	83
3.3	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.	84
3.4	Defining words in the lexicon (a)Lexicon definitions for the verb <i>load</i> in the LF::Filling sense; (b) The template used to define the syntactic pattern for <i>load the oranges into the truck</i> (c) The template used to define the syntactic pattern for <i>load the truck with oranges</i>	85
3.5	The definitions for verb <i>load</i> passed to the parser created from the lexicon definitions in Figures 3.4 and 3.3. (a) <i>load the oranges into the truck</i> (b) <i>load the truck with oranges.</i>	88

3.6	The LF definition for LF::Manufactured-object, LF::Vehicle and LF::Air-vehicle.	89
3.7	(a) A logical form for <i>Send a truck</i> (b) a simple possible LF for <i>Send a truck from Avon to Bath</i>	92
3.8	Logical forms for (a) <i>Send a truck from Avon to Bath</i> (b) <i>Send a truck from Avon straight to Bath.</i>	94
4.1	Ontology fragment for physical objects in TRIPS Pacifica domain.	121
4.2	Action ontology fragment in TRIPS Pacifica domain.	122
4.3	A sample description in the TRIPS WKB language	126
4.4	A sample description in the KM language	126
4.5	The definitions of the MOVE action in a) TRIPS WKB language b) KM language	127
4.6	A Simple LF-KR transforms to establish the correspondence between the LF type LF::Vehicle*truck and KR type TRUCK.	128
4.7	Transforms using a lexical form (a) A transform for medications in the medication adviser system (b) a transform for geopolitical regions in the Monroe system, and (c) an additional transform for handling <i>state</i> in the sense GEO-STATE	129
4.8	A sample hierarchical KR structure	130
4.9	Transform between the domain-independent form for LF::Filling and a domain-specific LOAD action	131
4.10	The transforms for (a) TRANSPORT and (b) MOVE actions in TRIPS Pacifica domain. The precondition limits the TRANSPORT transform application only to the cases when a cargo slot can be filled.	132
4.11	A transform to collect all path adverbials into a single path frame in the Trips Planner language. *1 denotes an operation of creating a new variable for the path frame.	134
4.12	The LF representation for <i>Go to Avon</i>	135
4.13	Transforming the LF::To-loc modifier (a) The transform for <i>go</i> and <i>LF::to-loc</i> adverbial together in the KM language (b) the same transform in the TRIPS-WKB language	136
4.14	Transforming the utterance <i>go to Avon</i> (a) The LF representation; (b) The main transform for LF::Move*go; (c) The applicable transform for LF::To-loc in context of LF::Motion; (d) The resulting KM representation	138

4.15	Transforming a LF with a unary modifier. (a) The LF representation for <i>a red truck</i> (b) the transform for colors; (c) the resulting KM representation	139
4.16	A logical form for <i>Go straight to Avon.</i>	140
4.17	Transform to move the external modifiers into the slot of a GEO-PATH frame attached to MOVE	141
4.18	Transform to move the external modifiers into the MOVE frame	142
4.19	Transform to ignore the direction modifiers	142
4.20	The BNF for lf-kr transforms	144
4.21	The algorithm for transforming LF into KR representation	146
4.22	(a) A logical form for <i>Send a truck to Avon</i> (b) The applicable transform	146
4.23	The KR form for <i>Send a truck to Avon</i> in TRIPS WKB language after all possible transforms were applied	147
4.24	The KR form for <i>Send a truck to Avon</i> in the KM language after all possible transforms were applied	147
4.25	The slot equivalence macro for LF::To-loc.	149
4.26	The lexicon entry for the verb <i>load</i> specialized for Pacifica domain (a) for <i>load the oranges into the truck</i> (b) for <i>load the truck with oranges</i>	151
4.27	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.	151
4.28	(a) Query for <i>Which prescriptions do I need to take</i> with no type coercion; (b) representation for <i>prescriptions</i> coerced to <i>medication</i>	154
4.29	(a) Operator to coerce prescriptions to medications. (b) Lexical entry for <i>prescription</i> with coercion feature generated from operator in (a)	154
4.30	A sample lattice for <i>that looks good</i> , misrecognized as <i>that looks gave it</i> . Numbers at nodes denote confidence scores for lexical items, numbers on arcs denote the confidence scores on transitions.	156
5.1	A simplified feature set used in our learning experiments	200
5.2	Pseudo word sense disambiguation results on TRIPS-lex dataset depending on seeding	204

5.3	Perplexity evaluation results on TRIPS-lex dataset depending on seeding	204
5.4	Coverage graph for TRIPS-lex dataset seeded-40 model. The x axis plots the percentage difference between the good and bad probabilities in test pair. As it increases, the number of pairs with that much difference in probability decreases.	206
5.5	Coverage vs. accuracy graph for TRIPS-lex dataset seeded-40 model. The x axis plots the % of pairs covered (recall), the y axis — the accuracy at that level (precision)	206
5.6	Coverage vs. accuracy graph for WSJ2 data set. The x axis plots the percentage difference between the good and bad probabilities in test pair. As it increases, the number of pairs with that much difference in probability decreases.	210
5.7	Coverage vs. accuracy graph for TRIPS-lex data set. The x axis plots the % of pairs covered (recall), the y axis — the accuracy at that level (precision)	210

1 Introduction

Computer understanding of language has many practical applications, for example, easy to use interfaces, machine translation and intelligent question answering systems. We are interested in systems that permit actual understanding of language, that is, the ability to draw inferences, make plans and execute actions based on a semantic content of an utterance. The first step of the understanding process is parsing and semantic interpretation - the translation between language, which is the representation understood by humans, and the formal knowledge representation suitable for computer processing.

A core component necessary for parsing and semantic interpretation is the system lexicon. This is a data store which lists all words known to the system, and encodes their syntactic properties and the correspondences between words in the language and concepts in the computer knowledge representation. In this thesis we will be exploring the structures which efficiently represent the information needed for interpretation in the system lexicon, and the way parsing speed and semantic disambiguation accuracy can be improved with the use of semantic feature vectors and efficient integration of domain independent and domain specific information in the lexicon.

In this chapter, we start with the review of the traditional approaches for parsing and semantic interpretation, and motivate our choice of a deep syntactic parser tightly coupled with semantic interpretation rules as the basis for the interpretation component in a multi-domain dialogue system. We then discuss the semantic disambiguation problem and the application of selectional restrictions to disambiguation. Based on the requirements outlined in this chapter, in the rest of the thesis we develop a typed feature representation as a way of encoding semantic information in the lexicons and show how it can be integrated with traditional ontologies. We show that it provides us with the flexibility to integrate the available domain-specific information

into the system, and discuss how probability models for feature assignments can be learned from corpora.

1.1 Parsing and Semantic Interpretation

Parsing is a process of transforming natural language into an internal system representation, which can be trees, dependency graphs, frames or some other structural representation. Syntactic only parsing attempts to convert the natural language strings into either tree structures or dependency links representing the syntactic structure of the utterance. The syntactic structures can later be sent for a semantic interpreter for further processing. The most common syntactic parsers today are probabilistic context free grammar parsers, which combine a context free grammar with a probability model which determines the most likely parse out of a large number of syntactic trees consistent with a given utterance (see for example Charniak [1997], Collins [1999]), though there are also parser build on other formalisms such as as Minipar [Lin, 1998] or LINGO [Copestake and Flickinger, 2000].

An example syntactic parse is shown in Figure 1.1.

We are interested in syntactic parsing as the syntactic relationships in a sentence correspond to functional relationships in the underlying meaning representation. For example, in a sentence *I saw a bird with yellow feathers, a bird with yellow feathers* is the object of *saw*, which in the underlying meaning representation corresponds to the fact that the bird denoted by the phrase is an argument (sometimes called THEME or PATIENT) of a seeing action. This relationship has long been studied in linguistics, and it is well known that often there are many possible syntactic structures consistent with the same string. The correct syntactic parse is (informally) defined as the one which humans see as corresponding to the correct semantic interpretation of the utterance. It is the job of the semantic theory to select the correct parse and the corresponding interpretation from the set of all parses consistent with a sentence. This thesis is concerned with the lexical information needed to solve ambiguity problems during parsing and semantic interpretation, and we will return to the discussion of ambiguity problem in more detail in Section 1.3

At the other end of the spectrum for parsing are semantic or case-frame grammars. Semantic parsing uses very shallow syntactic analysis of a string (possibly tagging parts of speech and noun phrase chunks), and sends the sentences to the semantic interpreter to derive the semantic interpretation. The semantic interpreter can be purely frame-based, searching for a combination

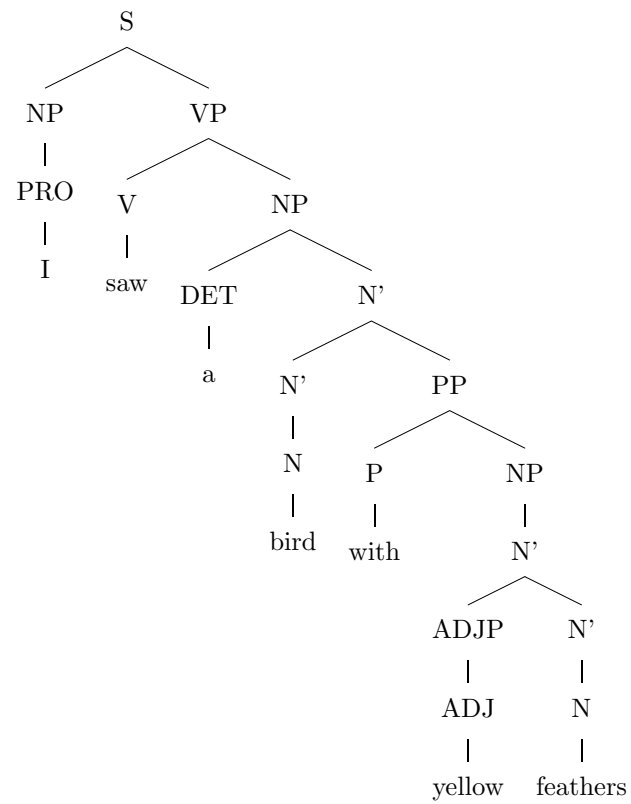


Figure 1.1: A syntactic parse of *I saw a bird with yellow feathers*

```
(seeing-action
  (actor I)
  (object (bird (color yellow))))
```

Figure 1.2: A possible frame representation for *I saw a bird with yellow feathers*

of meanings of words which together can form a coherent interpretation of an utterance, or it can use templates that utilize domain-specific syntactic knowledge to guide the search for the semantic interpretation.

Semantic grammars rely on the fact that knowledge about the domain can be represented straightforwardly as a set of frames. To handle our example above, a semantic grammar may define a frame in Figure 1.2. Note that the frame shown is a simplification of the relationships in the sentence (for example, the concept for *feathers* was left out altogether, as it is frequently the case with case frames, which focus on concepts and attributes, and may ignore other relevant relationships to simplify processing). Caseframe parsers offer a good abstraction across a range of information retrieval applications, and fast and robust language interpretation for small domains. However, the assumption about the frame structure limits the expressivity of the grammars, which do not handle well the complexities involved in quantification, negation, or many complex modifiers. Therefore they work best in systems where dialogue state is encoded as a set of contexts (see [Allen *et al.*, 2001] for detailed discussion). In a medium-scale application which requires plan- and agent-based dialogue models, more complex semantic representations are required.

To show how syntax provides important information for discourse interpretation, consider a dialogue fragment from our Medication Advisor domain, shown in Figure 1.3. Reference resolution algorithms use syntactic structure in combination with semantic information to achieve the best performance ([Tetreault, 2001], [Byron, 2002a]). More importantly, understanding of long-distance syntactic dependencies is crucial for correctly interpreting the utterances. For example, in utterance 19, *anything else* is an argument of *take* in the sense TAKE-MEDICATION, and this is determined syntactically based on the structure of a relative clause. This dependency is difficult to find during semantic search, because the phrase *anything else* is semantically unconstrained and can be attached either to *is* or to *take*. Semantic grammars try to solve this problem on the basis of the fact that the TAKE-MEDICATION sense of *take* has an unfilled parameter form MEDICATION, which is required to be filled in for consistent interpretation. However, both *need* and *take* have a large number of senses, and as the domain becomes larger, the ambiguity between

the senses becomes worse. Semantically, *need* can have multiple arguments, for example *I need John to leave now*, and we need syntactic constraints to tell that in this case it is not possible to extract an object of *need*. This information is not easy to represent in a clear and extensible way within the simple caseframe representations available to the parsers.

```

utt12    U: "Why am I taking celebrex? "
utt13    S: "You are taking celebrex to help with your arthritis. "
utt14    U: "Should I take one now? "
utt15    S: "Yes, you are taking celebrex every morning. "
utt16    U: "Is there anything else I need to take now? "
utt17    S: "Yes, you should take your Prinivil."
utt18    U: "Which one is that?"
utt19    S: "It's the yellow one. "

```

Figure 1.3: Sample dialogue in a medication advisor domain

Another important motivation for using a syntactic parser in a multi-domain dialogue system is portability. Semantic grammars are often difficult to port to new domains, because the domain-specific frame structures suitable to one domain may not be equally acceptable in another domain. For example, in our medication adviser domain the word *take* will be interpreted as an instance of TAKE-MEDICATION action. However, in another domain the meaning may be more general, referring to consuming different substances, such as *I take sugar with my coffee*. Semantic grammars rely on tight selectional restrictions to direct the search for correct interpretation, and in absence of strong semantic constraints, syntactic information is necessary to help with parsing. Since the syntax of the language does not change from domain to domain, using a general syntax-based parser offers a portable basis for a multi-domain language processing system.

The balance between domain-independent and domain-specific information in the system lexicon and grammar is a general problem for any system that attempts to link the words in the lexicon directly to the domain-specific knowledge representation. We discuss it in more detail in Chapter 4, and propose a new approach which improves portability in a system that relies on a wide coverage parser with a domain-independent knowledge representation combined with domain specialization rules.

To enable semantic processing, a syntactic grammar needs to have an associated set of

semantic rules to build the semantic representation of an utterance (called a **logical form**) in parallel with a syntactic structure. This is especially convenient in the computational grammars of language based on feature structures, especially HPSG and LFG grammars, which allow for keeping and propagating semantic structures through features as part of lexical entries.

If we attach semantic interpretation rules to syntax rules, it becomes crucially important that the syntax trees produced by the grammar align well with the intended semantic interpretations. This is the issue, for example, with Treebank grammars, which among other things do not make distinctions between arguments and adjuncts in syntactic structure. The lack of this distinction makes it considerably more difficult to obtain semantic interpretations suitable for reasoning based on simple treebank analyses.

It is worth pointing out that any system where syntactic and semantic processing are interleaved can be decomposed in a two-stage process with syntactic parsing first, followed by semantic interpretation which will resolve the attachment ambiguities. We emphasize combined syntactic and semantic processing because interleaving syntactic and semantic processing allows us to speed up the parsing process by imposing semantic constraints to limit the parser's search space and guide the search algorithm. The remainder of this thesis will develop a representation that uses selectional restrictions coming from type constraints on the semantic form arguments to speed up parsing and improve semantic disambiguation. For example, in utterance 13 (Figure 1.3, *with your arthritis* can either attach to *taking* or to *celebrex* or to *help*, all three structures will be equally syntactically valid. But the knowledge that *take* in the sense TAKE-MEDICATION does not normally accept medical conditions as arguments, and that *help* can be used with situations, should be sufficient to make the correct attachment decision.

1.2 Semantic Representation Issues

Given the reasons for interleaved syntactic and semantic parsing outlined above, we are going to consider more closely the problem of building practical parsers with the emphasis on speed, disambiguation accuracy and portability.

For a parser to be useful in a practical dialogue system, it has to build logical forms which can be utilized for reasoning in the system back-end. The lexicon has to provide the semantic information for building logical forms, as well as the information necessary for effective semantic disambiguation, that is, selecting the correct interpretation from multiple alternatives. The questions that need to be discussed first are the following:

- **What kind of semantic information is useful in parsing and semantic disambiguation, and what is its source?** Potentially, all kinds of semantic, discourse and task-related information can be useful in deciding what is the correct syntactic and corresponding semantic information. We are going to limit our approach to selectional restrictions, which are type constraints between entities and their arguments. Other constraints can be useful, but require sufficient reasoning resources, which could slow down the parsing process. We are going to show how both domain-independent and domain-specific selectional constraints can be integrated during parsing, and that using them substantially improves parsing quality.
- **How can we design a semantic type representation for efficient use during parsing** The possibilities are a symbolic model based on the available knowledge representation, or a probability model trained from a corpus. Probability models have been successful in parsing and semantic disambiguation (see for example [Charniak, 1997; Ratnaparkhi, 1998]). However, these models rely on the availability of large training corpora for a domain, which are difficult and costly to collect for small domains, especially involving spoken interfaces. We are going to explore a typed semantic feature representation. This is a symbolic representation well understood in linguistics and computer science, and there exist efficient algorithms to employ it in a unification based parser. Additionally, we are going to propose a method to learn feature representations if a corpus is available, and discuss the results of learning experiments on a small corpus.
- **How do we improve system portability and effectively integrate domain-specific and domain-independent information in the lexicon.** As noted previously, the efficiency of semantic parsers comes from using tight domain constraints, but tightening the constraints results in loss of portability. In contrast, syntactic parsers are generally portable, but slow. We are going to develop a system with a portable domain-independent lexicon with syntactic and domain-independent semantic information, develop an algorithm to customize the representations produced to specific domains. Finally, we are going to show how the use of feature representation makes it easy to transparently integrate domain-independent and domain-specific selectional restrictions, and how it can be used to speed up parsing of in-domain utterances.

In Chapter 2 we discuss typed feature sets as a domain-independent semantic representation for selectional restrictions, and develop a formal semantics for feature operations necessary for selectional restrictions. Then, in Chapter 3 we discuss a domain-independent ontology as a

basis for generating logical form, and show how typed feature representations can be integrated with a more traditional ontological representation in a computational lexicon. In Chapter 4 we discuss the integration of domain-independent and domain-dependent information via the system lexicon. Finally, in Chapter 5 we return to the statistical model to show how feature probabilities can be learned from corpora and used in word sense disambiguation tasks.

1.3 Semantic disambiguation with selectional restrictions

In our discussion of the semantic information needed for parsing, let us consider again the semantic knowledge necessary for resolving syntactic ambiguities. Consider a set of syntactically very similar examples:

- (1) a. I saw [_{NP} a bird] [_{PP} with yellow feathers]
 b. I saw [_{NP} a bird] [_{PP} with binoculars]

In both of those sentences the *with* PP can attach either to the noun phrase *a bird*, or to the main verb *saw*. Syntactically, either attachment is equally valid, but in absence of strong bias to the contrary, people interpret *with yellow feathers* as attaching to *a bird*, and *with binoculars* as attaching to *saw*. In both cases we will also interpret *saw* as a past tense of the verb *see*, and not a present tense of the verb *to saw*. These judgments are made on the basis of semantic information, namely, that binoculars is a kind of instrument useful for *seeing*, and *feathers* is a natural covering which cannot be an instrument in a seeing action but can be associated with living beings.

The notion that words restrict their arguments is commonly referred to as selectional restrictions. Selectional restrictions were first introduced by Katz and Fodor [1964], who proposed that selectional constraints apply as a filter after all transformation grammar rules. The original formulation of selectional restrictions theory was based on the idea of a hierarchical lexicon where different word senses are distinguished by different “semantic markers”, and each word sense specified the semantic markers to which it could be applicable. For example, a word *ball* would have among its senses one distinguished by the semantic marker (*Physical Object*) and another with a semantic marker (*Assembly*). The semantic markers were assumed to be hierarchical, and (*Assembly*) would be in a tree under (*Social Activity*). Then, the adjective *Red* will be associated with a semantic marker (*Color*), and marked as selecting for (*Physical Object*). Based on that definition, *a red ball* will select a sense of *ball* marked with a (*Physical Object*) marker.

Katz and Fodor did not explore the nature and the source of the semantic markers beyond the claim that they correspond to some semantic concepts. The guideline they established was that a new semantic marker needs to be added if it will allow the semantic theory to distinguish between a significant number of different word meanings. There is a tradeoff between the improvement in distinguishing various sentence meanings with adding a new marker, and the increase in the complexity of a semantic system caused by this addition. Katz and Fodor suggested that at some point adding more semantic markers would no longer bring additional benefits compared with the complexity of maintaining the system, and this is exactly the point which would be the optimal state of any semantic theory.

Various implementations of selectional restrictions have been used widely in small-domain natural language systems for purposes of semantic disambiguation. One of the main issues with the implementation is what exactly are the semantic markers, and who provides them, and this is the issue that was left deliberately open in the original selectional restrictions theory. In a small domain, it is easy to provide a complete description of all entities possible in the domain and relationships between them. For example, we can define a SEE action with three arguments: a PERSON, a OBJECT and a TOOL. The interpretation of a sentence consists of finding the predicate SEE which corresponds to the word *saw*, and filling in the argument positions. In the lexicon, *I* will be declared of type PERSON, *bird* of type ANIMAL, which is a kind of OBJECT, and *binoculars* of type TOOL, which will allow them to fill in the corresponding slots and complete the sentence interpretation.

While it may be very easy to provide a comprehensive list of word types and relationships between them in a small application domain, the task becomes increasingly more complex as the size of the domain and the coverage of the system grows. This will be one of the main themes of our work, providing an ontology and a lexicon suitable for parsing in a knowledge-based system for intermediate and large size domains. We will describe the more specific requirements on the lexicon in Section 1.5, and use it as the motivation for our approach.

So far, we have discussed encoding selectional restrictions with the help of semantic types associated with words. This representation is especially suitable for knowledge based systems, which are connected with ontologies describing the domain model. However, the strict notion of selectional restrictions cannot account for all possible uses of language. While cars in general cannot smile, it is possible to imagine a world of animated movies where they have mouths and can indeed be smiling. In addition, restrictions can be violated because of metonymy, metaphor and other creative uses of language. Wilks proposed that the restrictions are not strict, but are rather preferences in the language[Wilks, 1975]. He developed the system that used heuristics

to relax selectional restrictions if constraints were impossible to satisfy, scoring parses using the number of constraint violations encountered in each parse. Unfortunately selecting the proper preferences manually is a too complex task for a reasonably large lexicon, so it has not been used as widely in the existing NLP systems.

Another option for encoding selectional restrictions as preferences is a lexicalized probability model. We can learn from corpora a model that predicts the probability of attachments between verbs and PP objects, which will assign a high probability to *see, with binoculars* and a low probability to *see, with feathers*. Moreover, the model may cluster words so that *binocular, telescope, microscope* will all belong to the same cluster, and the probability between *see* and the cluster will be high. Such models in effect encode semantic information in the probability distribution. For example, if *binoculars* and *telescope* belong to the same cluster, this means that they are likely similar to each other in meaning and use, and belong to the same semantic category.

Lexicalized probability models have been very successful in tasks such as information retrieval and machine translation. However, they require large corpora for training, which may not be available in medium sized domains, and are expensive to obtain, particularly if speech data collection is required. Moreover, the representation they produce are generally difficult to interpret for humans. This is a problem for knowledge-based systems which, in order to reason and execute commands, need a precise symbolic representation of the meaning of their language input. Therefore, it is worthwhile to develop symbolic selectional restrictions encoded in a domain-independent ontology which can be used to rapidly develop applications in novel domains. We demonstrate how the selectional restrictions improve parsing and disambiguation in two of our domains where the sufficient training data for a probability model were not available at the time of the evaluation. Then, in chapter 5, we will also discuss a statistical model based on the typed feature lists similar to the symbolic representations we use in our ontology, providing a way to possibly transform selectional restrictions into selectional preferences based on probabilities.

1.4 Parsing and interpretation in a practical dialogue system

Many of the issues we are concerned with in parsing and interpretation are empirically grounded in our research in conversational user interfaces. The testbed for our research is the TRIPS

dialogue system developed at the University of Rochester [Allen *et al.*, 1996]. This is a multi-domain dialogue system that sets a goal to use unrestricted speech as an input for applications that require planning and reasoning. Over the years, we implemented dialogue tasks in multiple domains, and we also collected corpora of human-human dialogues for these tasks. These are

- **TRAINS.** The TRAINS domain is a train routing domain where users are asked to create schedules for train routing that satisfy time restrictions based on distances and congestion on the traffic routes. There is a spoken corpus of task-oriented dialogues collected as part of the effort [Heeman and Allen, 1995]. In the corpus, the tasks are more complex, requiring other actions connected with routing (e.g. delays occasioned by the fact that if we need to transport orange juice by a specified train, the juice needs to be made from oranges before the train can depart).
- **Pacifica** The Pacifica domain is a small fictional island where the user needs to make a plan to evacuate people off the island before the approaching storm using trucks and helicopters. The vehicles have different speeds and can carry different number of people. We do not have a corresponding corpus of human-human dialogues, but we have a corpus of utterances users said to the system with the speech recognizer output, which we used in our evaluation.
- **Monroe** In the Monroe domain, users are presented with a realistic map of a Monroe county and asked to coordinate an emergency response team with police, ambulances and repair crews. There is a corpus of human-human dialogues [Stent, 2000], and the implementation of a full dialogue system is currently under development
- **Medadvisor** In the Medication Advisor (Medadvisor) system [Ferguson *et al.*, 2002], the users ask the system about the medications they are taking and possible interactions. There's an implemented prototype system, and a small corpus of Wizard of Oz dialogues collected with human subjects
- **CALO-EPA** In the CALO-EPA domain, the system acts as a personal secretary responsible for purchasing decisions. A prototype system and data collection are currently under development.

Developing a multi-domain spoken dialogue system presents significant challenges. We have already outlined many of the issues, and here we present the summary of our goals as they relate to developing conversational interfaces in a multi-domain system. The primary goal of the entire

system is to use natural language and dialogue as a natural user interface in reasonably complex domains. We showed earlier (Figure 1.3) an example from the Medadvisor domain. In this and other domains people use a wide variety of language expressions to describe their needs and goals or to ask questions. In building the system, the goal is to accept unrestricted input, which means that the system needs to be able to process concepts expressed in any way syntactically, and deal with a variety of complex expressions.

The system needs not only to recognize complex syntactic constructions, it needs to correctly interpret them in all domains. Several of those domains are developed in parallel, and we need to re-use as much of syntactic and semantic information between them as possible, so that improvements made in one domain translate to improvements across the entire system. Under the circumstances, a syntactic parser is a more attractive choice, since clearly the syntax of English does not change between domains, and it should help considerably in making the parser and grammar re-usable across domains.

The system has to interact with users in real time, so both fast response times and correct interpretations are extremely important. This necessitates the use of semantic information to guide parsing search and disambiguation, but raises the issue of how the semantic information is related to the domain-independent syntactic information. We mentioned before that many selectional restrictions are domain specific, moreover, the same words tend to have different sense frequencies in different domains: the most frequent sense of *take* is MOVE in TRAINS, Pacifica and Monroe domains, CONSUME in the Medadvisor domain, and SELECT (as in *I will take this system*) in the CALO-EPA domain. We would like to use this domain-specific information in disambiguation, but this raises the issue of integrating it with the domain-independent grammar and lexicon in the system.

Speech input introduces a set of additional complications, in particular, due to disfluencies and speech recognition errors. Speech repairs and incomplete sentences are very common in spoken language, which make the interpretation task considerably more difficult. In the earlier versions of the system (TRAINS, Pacifica) we alleviated the problem by introducing the push-to-talk interface, which requires users to hold the button while they speak. In our experience, it considerably reduces the number of incomplete and incorrect sentences in the system, however, it limits the naturalness of the dialogue. In this thesis, we are not going to be concerned with speech repairs, concentrating mainly on the problems outlined above for processing full utterances. However, we hope that using selectional restrictions can be a useful way to improve speed and efficiency of syntax-based methods for dealing with repairs such as developed by Core and Schubert [1999].

Errors made by speech recognition make the existing problems with speech even more difficult. Often the first hypothesis output by the speech recognizer is not the best possible one (the sentence accuracy rate in our Pacifica domain is about 50%). Therefore, the system needs to be robust to speech recognition errors. There are two possible approaches to the problem: instead of taking a single hypothesis from a speech recognizer, we can take a lattice input which encodes multiple possible hypotheses, or we can use clarification questions if we didn't understand part of the utterance. Using lattice input offers a chance to use stronger syntactic and semantic constraints from the parser to select a better interpretation from a set of n-best recognition hypotheses, but at the cost of efficiency, because the parser needs to deal with multiple utterances every time. Using clarifications offer a chance of recovery if parts of a sentence were misrecognized, but requires the ability to interpret a sentence which may contain out-of-domain words.

We are going to evaluate our system on the lattice inputs as a way to show that it can help select the correct sentences out of the large lattices, with selectional restrictions providing a way to do it efficiently. We do not implement special robust interpretation strategies beyond allowing fragments as complete utterances, but we believe that the domain-independent parsing system we developed offers a good basis for robust dialogue management by allowing us to produce interpretations of all sentences, including those that contain words not in the domain.

1.5 Lexicons and Ontologies

Let us now consider in more detail the design of lexicons and ontologies for practical NLP systems, and the implementation of selectional restrictions. A NLP lexicon designed for parsing and semantic interpretation needs to contain the following information for each word

- Syntactic features. These are the features that govern the syntactic parse and determine which words can be combined into grammatical strings. The examples are tense and agreement for verbs, agreement, mass and count distinctions for nouns, predicate versus attributive distinctions for adjectives.
- Subcategorization frame. This is an especially important syntactic feature which determines the phrase structures in which the word can occur. The availability of subcategorization frames is key to the interface of syntax and semantics in parsing, because they correspond to argument structures in the semantic interpretation. For example, the word *see* may be associated with two frames: $NP _ NP$ and $NP _ NP PP$, which corresponds

to two possible syntactic configurations: $[_{NP}I] \text{ saw } [_{NP} a \text{ bird}]$ and $[_{NP}I] \text{ saw } [_{NP} a \text{ bird}] [_{PP} \text{ with binoculars}]$.¹

- Semantic type. This is the concept (or a complex expression) that the parser will use as a meaning representation for the word. In the simplest case, the entry will contain information that *see* means SEE1, where SEE1 is understood to be a logical concept known to the reasoning system, denoting an action of perceiving something by sight. There may be additional senses specified - for example, *see* may be associated with UNDERSTAND, for sentences like *I see that he knows it*.
- Syntax-semantic linking. This is the mapping between the syntactic arguments in the subcategorization frame and the semantic expression. For example, if the verb *see* is associated with the semantic predicate SEE1 and $NP _ NP$ frame, then the linking can say that the first (subject) NP will be mapped to the first argument of the SEE, and the second NP will correspond to the second argument, so that *John saw a bird* corresponds to SEE1(JOHN1,BIRD1)

In a practical NLP system syntactic features and subcategorization frames can be obtained from a general-purpose syntactic lexicon. Determining and listing all relevant syntactic features is in itself a difficult problem, but we are going to concentrate on the semantic parts of the lexicon, namely, the source for the semantic types, and the syntax-semantic linking. A typical source of semantic information is an ontology, which is a classification of semantic concepts that exist in the world and are known to the processing system. The proper ways to classify the types were studied extensively in philosophy and knowledge representation [Sowa, 1999], but at present there is no general consensus on best possible classification.

Application specific ontologies are often developed to describe the specific application domains. They are most commonly used as a source of semantic types in the existing system lexicons (see for example the TINA system [Seneff, 1992] or the SmartKom project [Chang *et al.*, 2002]). The small ontologies can be integrated with grammars for speech recognition, improving the accuracy of rescoring for small domain applications, as described, for example, in Gurevych *et al.*[2003]. Using application-specific ontologies is subject to portability problems similar to those of semantic grammars; thus, a domain-independent ontology would be preferable in a multi-domain system. In Chapter 3, we discuss several existing large scale ontologies which can be useful for NLP, describe their limitations with respect to the information needed for

¹We are disregarding the fact that *with binoculars* is an adjunct in this case, for purposes of keeping our examples uniform

lexicon development, and describe the domain-independent ontology geared for parsing which we developed as part of our thesis research.

In addition to the lexicon requirements outlined above, to use selectional restrictions for disambiguation purposes we need to put restrictions on the types of arguments for every word. For example, for the seeing action, we need the information in the lexicon which states that the subject NP must be a person or an animal. Most application specific ontologies contain type restrictions, so if the first argument of SEE is restricted in the application to be a PERSON, we can use this information as a selectional restriction on the subject based on the correspondences between syntactic and semantic arguments.

In order to reason about selectional restrictions, we need to have information about type compatibility, to be able to answer questions like “Is entity X a subtype of type Y”. The subtype relations impose a hierarchical structure on the classes of objects in the world, and can be represented as a hierarchy where the parent node is a superclass that is partitioned in a number of smaller child subclasses.

In a subtype hierarchy, we need to be concerned with both conceptual economy and efficiency. The former usually requires introduction of multiple inheritance. We need to be able to express that, for example, all trucks are vehicles, and all trucks are also containers. This is necessary to account for the fact that a truck can participate in sentences like *The truck goes to Avon*, patterning with other vehicle-denoting nouns, *car*, *bicycle*, and also in constructions like *The truck is full*, patterning with container words *box*, *glass*. These are two separate types, because both **The bicycle is full* and **The glass goes to Avon* are infelicitous, but *truck* must be a subtype of both. An alternative would be declaring two different subtypes, CONTAINER_VEHICLE and NON_CONTAINER_VEHICLE. However, this results in a rapid increase in the number of concepts in the ontology, and correspondingly more difficulty in selecting the correct concept for each word and maintaining the consistency of the entire hierarchy.

In contrast, when multiple inheritance is introduced into a type hierarchy, efficiency becomes a serious consideration. The primary operation for checking selectional restrictions is Least Lower Bound (LLB), finding an object of the type consistent with both of the types in the constraint. In a single inheritance hierarchy this is a very efficient operation, requiring constant time and linear storage space [Schubert *et al.*, 1987]. However, for arbitrary hierarchies the problem is more complex and is coNP hard in the worst case (inheritance hierarchy with exclusive partitioning) [Schubert, 1979]. While most practical hierarchies do not fall in that class, in many cases the complexity of the algorithm is polynomial in the size of the hierarchy, and the design of efficient algorithms for computing type unification remains problematic. Kiefer [1999] proposes

a fast LLB algorithm that pre-compiles all possible unifications to achieve efficiency, but pre-compiling all possible unification requires additional space, and makes it more difficult to add new concepts “on the fly”, which is one of the projects planned in our future work. Instead, we consider multiple inheritance formalized as feature lists as a lexical semantic representation for selectional restrictions.

A special case of a decomposable multiple inheritance hierarchy is a set of “overlapping” single inheritance hierarchies where each concept is classified along several different dimensions. For example, physical objects can be classified according to their function (vehicle/building/tool etc.), and according to their spatial properties(enclosure/hole/solid object). Schubert *et al.*[1987] show that constant time/linear space algorithm can be used for the case where the concepts are leaves of several overlapping single inheritance hierarchies and each concept belongs to only a small number of hierarchies, and our representation satisfies this restriction. We can use a feature list to represent the concept, where features correspond to different dimensions, and values are the types of the concept in the hierarchy which corresponds to that dimension. In that case, we can reason about unification in space and time linear in the number of features, or, alternatively, we can interpret it as multiple inheritance with concepts corresponding to different valid feature assignments, and have a constant time unification algorithm (at the expense of exponential increase in space, which may still be tolerable for small feature sets).

Feature lists have other attractive properties, because feature representations are well understood and widely used in syntactic parsing, and having explicit dimensions can make the task of selecting the correct semantic type for the word easier, as discussed in Section 2.4, and offer flexibility and extensibility we utilize in our domain specialization algorithm in Chapter 4. In Chapter 2 we discuss a way to represent multiple inheritance with typed semantic feature lists, a generalization of feature lists which offer a more efficient reasoning representation than unrestricted multiple inheritance, and use it to implement selectional restrictions. Then, in Chapter 3, we return to the question of using a large scale ontology as a source of information for parsing. We develop an ontology for practical NLP which integrates a traditional type hierarchy with feature sets, and show how a large-coverage grammar and lexicon can be utilized together with it to provide a domain-independent parser for practical dialogue systems.

1.5.1 Implementing Selectional Restrictions

Another issue that needs to be considered is how the selectional restrictions can be used in the system and integrated with parsing. This question is tightly connected with the general way in

which syntactic and semantic interpretation are related in a parser. In a caseframe based system, checking selectional restrictions is the core of the parsing process, since it directs the search for acceptable slot fillers. In a syntactic context free grammar parser, selectional restrictions have to be implemented as filters or additional constraints. In a feature-based unification grammar, it is possible to integrate checking selectional restrictions as a part of general parsing process, as was done, for example, in the CORE language engine [Alshawi, 1992].

Although the integration of selectional restrictions with parsing is clearly possible, selectional restrictions have not been actively used in large coverage parsers. For example, the HPSG Lingo [Copestake and Flickinger, 2000] grammar, used for the VerbMobil project [Kay *et al.*, 1994], does not use selectional restrictions directly. Instead, a separate scheme for sortal constraints in disambiguation [Stede *et al.*, 1998] is implemented. While in most cases the precise implementation scheme does not make a difference, it turns out that for certain examples the specific operation used to check selectional restrictions (subtype check vs. unification) leads to different results. In chapter 2 we describe formal selectional restrictions operations and the effect they have on parsing and lexical semantics for certain underspecified items, such as pronouns.

Many large-coverage syntactic parsers avoid using selectional restrictions because, generally speaking, effective selectional restrictions that constrain the inputs for the best possible interpretation are domain-dependent. We believe however that useful weak selectional restrictions can nevertheless be used to guide the search and disambiguation in a practical dialogue system. In Chapters 2 and 3, we develop a system to express selectional restrictions through typed feature sets with features that describe only the most general meaning components, and thus limit the complexity of defining selectional restrictions on most items. The resulting domain-independent restrictions are not strong enough to provide the best possible disambiguation, but are sufficient to obtain reasonable parsing speed for out of domain utterances. Then in Chapter 4 we describe a semi-automatic procedure that allows to augment the domain-independent restrictions with domain specific information for each domain, allowing for optimal speed and disambiguation accuracy for in-domain utterances.

The feature set we develop provides a good representation for selectional restrictions in multiple domains, as evidenced by the evaluation of our system coverage in Section 3.6.2. However, we recognize that it is difficult to develop a set which will cover all nuances of language use, and in general lexical preferences, rather than hard restrictions, are a more general model. Therefore, in Chapter 5 we develop a statistical model of selectional restrictions which could be used in a lexicalized PCFG. Because typed feature lists have advantages for system building described in previous chapters, we develop a statistical model based on the feature list model we use in our

symbolic lexicon. We describe our experiments with learning selectional preferences using this model, and outline the ways in which it can be integrated, or used in parallel with our symbolic restrictions during parsing.

1.6 Summary

In the previous section, we outlined the parts of our thesis concerned with implementation of selectional restrictions. Speaking generally, in this thesis we develop a typed feature representation and a domain-independent ontology and lexicon geared for efficient parsing and disambiguation in practical NLP systems.

In Chapter 2 we present a detailed motivation for using feature sets as a basis for selectional restrictions, and describe a feature set we developed for use in the TRIPS multi-domain parser. We then formalize our typed feature list model based on Carpenter’s theory of typed feature structures [Carpenter, 1992], and use our formal model to show the effect of two different implementations of selectional restrictions on handling issues which arise when conjunction, pronouns and generic terms are introduced in parsing and interpretation. To our knowledge, these problems have not been studied before, possibly because they arise in the constructions rarely encountered in small domain systems. We believe that our model provides a useful framework for making implementation choices in a practical system. Finally, we discuss the general issues in designing our feature set and outline general principles for extending and further developing feature sets to express selectional restrictions.

In Chapter 3 we describe the motivation and design of a domain-independent ontology and lexicon for parsing, which integrate a traditional symbolic representation with typed feature sets as a basis for selectional restrictions. We provide a formal foundation for default feature inheritance and value dependencies between features in our lexicon, and discuss the implementation of selectional restrictions in our grammar.

In Chapter 4 we show how the flexibility of our typed feature set representation permits easy integration domain-independent and domain-dependent restrictions in the lexicon. We develop a formal system to map the domain-independent semantic representations produced by our parser into domain-specific representations customized for reasoning, and use the mappings to automatically propagate selectional restrictions to the lexicon. We show how the inclusion of domain-specific restrictions can help the parser to improve speed and disambiguation accuracy over a set of speech recognition lattices.

Finally, in Chapter 5 we describe a statistical model for feature based selectional preferences and describe experiments on learning semantic features from corpus data.

The contributions of our thesis are the following

- Developing a typed feature list representation for selectional restrictions. This includes a domain-independent feature set we use in four different domains, and a formal semantics for feature sets in selectional restrictions. Our model supports handling conjunction through collective sets, something that to our knowledge has not been described previously.
- Designing a domain-independent ontology for natural language parsing, and a lexicon linked to this ontology; integrating all syntactic and semantic information necessary for parsing and semantic disambiguation, which is not achieved by any of the existing large scale lexicons for natural language processing; exploring the issues and tradeoffs involved in building lexicons for parsing and semantic interpretation in multi-domain systems.
- A semi-automatic method to specialize the representations produced by the domain-independent parser to various application domains; extending the architecture to automatically specialize grammar and lexicon for the domain to improve parsing speed and accuracy.
- A statistical model for learning selectional restrictions from corpora based on semantic feature lists.

2 Features in a Domain Independent Ontology

In the introduction, we formulated our basic approach to building multiple-domain dialogue systems interfacing with reasoning components. One of the major issues that arise with this approach is the appropriate representation of semantic information needed for parsing and disambiguation, and especially for selectional restrictions. In this chapter, we propose the use of typed semantic feature lists for selectional restrictions, and show how it can be an easy and convenient way to represent selectional restrictions for parsing. We describe semantic features used in our domain-independent lexicon, and a formal set of operations that need to be implemented to operate on feature values.

The goal of this chapter is two-fold: to introduce a feature system we developed for use in a practical dialogue system, and to provide formal semantics for operations on features. There are several practical questions which need to be answered with respect to using feature sets as a lexical semantic representation:

- What are the reasonable features and values?
- What does it mean for a feature set to satisfy a selectional restriction?
- What are some general principles by which a feature set for a practical dialogue system can be constructed, and what are the advantages and limitations of the feature representation?

This chapter attempts to provide answers to these questions. We begin with a review in Section 2.1 of the way feature sets have been used to describe various linguistic phenomena, as a motivation for using feature sets in selectional restrictions. We discuss the feature set we developed for the TRIPS system in Section 2.2. Section 2.3 deals with the question of what it means to satisfy selectional restrictions. We introduce some motivating examples, and, given

the problems we need to solve, in Section 2.3.2 we develop formal semantics for feature sets by adapting the standard definitions of typed feature lists and structures from Carpenter [1992] and Copestake [1992]. We provide a formal definition of selectional restrictions in this system, and observe that there are two different models of selectional restriction satisfaction, based on either the unification operation, or a subtype operation, and they have different ramifications on the acceptability of certain constructions. Finally, in Section 2.4 we return to a more general discussion of the use of feature sets as a basis for selectional restrictions, and provide a general assessment of how feature sets need to be constructed for purposes of parsing lexicons.

2.1 Background

In Section 1.5 we discussed ontologies as a source of semantic information for selectional restrictions. A typical NLP lexicon is set up so that every word corresponds to a type, and may place restrictions on the types of its arguments. The types have to be organized in a multiple hierarchy, for example, we showed why a word *truck* must have both VEHICLE and CONTAINER as parents. We discussed the issues with using multiple inheritance in a practical system, which include increased difficulty in maintaining the ontology and the complexity of the algorithm for finding a least lower bound of two concepts. The latter is especially important for us, because this is a very frequent operation during parsing, as we have to check selectional restrictions multiple times. We propose to use typed semantic feature sets without multiple inheritance between the types as a way to deal with these issues.

The idea behind using a typed feature set representation is that instead of treating the semantic type of a word as a single concept with multiple parents, *e.g.*, having *truck* be a child of both VEHICLE and CONTAINER, we characterize the entity along several different dimensions - for example, saying that *truck* has feature *Mobility* set to *Self-moving* and feature *Container* set to *+*, written as (*Mobility Self-moving*) (*Container +*). Then the adjective *full* can require that its argument should be (*Container +*), while the verb *go* will require its subject have (*Function Vehicle*). In our approach, every feature can have a set of hierarchical values, but only single inheritance is allowed between the values. Under those conditions, an efficient algorithm can be developed to match selectional restrictions, with time linear in the number of features. If we expect a number of features involved in each selectional restriction to be small (1 or 2), then we can count on constant time on average for checking selectional restrictions in the system. We will also argue that using a feature set representation can offer additional flexibility and ease of use in lexicon acquisition.

Feature systems are a frequently used tool in computational linguistics. Computational grammars such as HPSG [Pollard and Sag, 1994] and GPSG [Gazdar *et al.*, 1985] use feature Attribute-Value structures to represent syntactic information, and dependencies between different semantic features. Features have also been used in computational phonology [Kenstowicz, 1993]. Typed feature representations have well understood theoretical properties, and we will be using and extending existing formal definitions for our semantic feature sets.

Many linguistic theories have made the observation that word meanings can have identifiable components that can be used to explain various syntactic phenomena. Even purely syntactic theories often use properties like *+animate* to denote plausible subjects of verbs, and *+location* to denote acceptable complements of a like *put*. These features are semantic in nature, because they refer to the meaning of a word or a phrase which is the argument of a given verb. Here, we will discuss explicitly two examples when features were used to explain aspectual phenomena for verbs and novel meanings of adjectives, and which will be relevant to the construction of our feature set.

Several theories of aspect used features to explain the behavior of verbs with respect to tense and temporal adverbials. For example, *He reached the top* is more felicitous (without prior context) than *He was reaching the top*. This can be modeled with a set of (binary) aspectual features associated with the word *reach*. Different sets were proposed: Atomic and Conseq ([Moens and Steedman, 1988]), Telic and Persistent ([Thomas and Pulman, 1999]), Dynamic, Durative and Telic ([Dorr and Olsen, 1997]). In all cases, the set of features assigned to a verb determines whether it can participate in certain constructions, such as progressive, and whether it can be modified by durational adverbials. Our feature set includes features based on the work of Moens and Steedman, and we will return to this discussion later in this chapter.

Another example of the use of semantic features is the ACQUILEX [Copestake *et al.*, 1993] lexicon. Generative lexicon qualia structures [Pustejovsky, 1995] are a model of noun and adjective semantics that accounts for novel meanings and word combinations of adjectives and nouns. Each noun is associated with four features: Formal, Constitutive, Telic and Agentive. The roles define different aspects of meaning of a noun: Formal role specifies physical properties such as orientation and shape, Constitutive describes the material and component elements, Telic describes purpose and function, and Agentive the factors involved in the origin of the object. The values of the qualia roles are used to systematically predict the meaning alternations for nouns that arise from combinations with adjectives and different verbs. Copestake *et al.* [Copestake *et al.*, 1993] developed a computation lexicon, ACQUILEX, which uses typed feature structures to computationally represent qualia roles and implement generative semantics in a computational

system, and describes experiments of learning parts of qualia structures from dictionaries. We used a simplified system based on qualia features to describe properties of nouns in our system.

Since different linguistic theories already use features as a means of expressing linguistic generalizations, it makes sense to use the same representation in a practical system. In fact, in a practical lexicon using features is especially convenient, because now we can combine insights from multiple linguistic theories within a unified feature set to model selectional restrictions. At the same time, features can be used together with a regular class based inheritance hierarchy, as was done in two computational lexicons: VerbNet and EuroWordNet.

EuroWordNet is a multi-lingual database which comprises hierarchical lexicons in many European languages[Vossen, 1997]. The lexicons for each language are organized into a WordNet[Miller, 1995] subsumption hierarchy. We will discuss the details of that representation later in Chapter 3, but what is important for our purposes here is that the standard hierarchical structure in EuroWordNet is combined with a language-independent semantic feature representation. Each word is linked to an Inter Lingual Record (ILL) which contains a set of features corresponding to basic word characteristics. The features were selected based on the analysis of several different domains and insights of existing lexical semantic theories. The ILL structures are used to link words across languages and represent the basic meaning components that characterize concepts. Therefore, EuroWordNet provides a good starting point for a language- and domain-independent semantic representation. In the next section, we are going to describe the EuroWordNet feature set as a basis for our parsing feature set, and show the modifications that need to be made for purposes of a practical parsing system.

The VerbNet lexicon is a verb lexicon which combines syntactic and semantic information for verbs[Kipper *et al.*, 2000]. The semantic information in VerbNet includes selectional restrictions represented using semantic features. The VerbNet feature set is a multiple inheritance [Schuler, 2002] with 36 feature values currently possible. The feature set is originally based on the EuroWordNet feature system, but not all EuroWordNet features are included, and some new values were added.

Clearly, the EuroWordNet feature set can be used as a basis for a semantic lexicon including selectional restrictions, but, since their feature set was not developed with this explicit goal in mind, changed need to be made to develop a parsing lexicon using features. The analysis of features we found useful in parsing, and the feature set we developed for that, is one of the major goals of this chapter. We will discuss our feature set in detail, and then in Section 2.4.3 compare it with the independently developed VerbNet feature set. We will find that some similar changes were made, but there are also differences. One of the big reasons for that is that

VerbNet is a verb only lexicon: it uses features to express selectional restrictions on verbs, but not on adjectives and adverbs. It also does not address the issue of the source of its features, because building a complete lexicon annotated with semantic features is outside of the scope of that project.

The aim of our system is to create a complete lexicon for natural language processing, with a feature set sufficient to describe the restrictions necessary for parsing and semantic disambiguation. For that purpose, we not only provide a complete feature set, but also a formal semantics for operations needed to provide selectional restrictions in a computational lexicon. We will also provide statistics about the usage of different features in our system, and discuss guidelines for developing and further extending our feature system.

2.2 The TRIPS feature system

So far, we have talked about using feature lists as a basis for a lexical semantic representation. One additional refinement needs to be made at this point. It is clear from the existing theories that different words can be associated with different feature sets, which are in general mutually incompatible. For example, Constitutive Qualia role does not make much sense for verbs, because they denote actions that cannot have material or weight, nor is it useful to characterize many abstract nouns, for example, *happiness*, *idea*. Similarly, aspectual features such as *atomic/extended* do not apply to nouns such as *box* or *person*. This leads to a notion of **typed feature lists**. A typed feature list consists of a type and a list of features. The feature set type assigned to the object determines which features can be set on it: for example, the type *Physical object* will license defining qualia features, whereas the type *Situation* will license aspectual features. In the rest of this chapter, we assume that we are using typed feature lists in all our representations.

2.2.1 EuroWordNet features as a basis for a parsing lexicon

To develop our feature system, we started with the EuroWordNet feature set [Vossen *et al.*, 1997]. We annotated our lexicon with EuroWordNet features, and analyzed their usability for a parsing lexicon. This resulted in a number of changes and additions to make the set more suitable for parsing, and resulted in a new TRIPS feature set used in our domain-independent lexicon.

The EuroWordNet top ontology distinguishes three main classes of objects: 1stOrder entities, which are the objects present in the physical world, 2ndOrder entities, which represent properties, situations and relations which cannot be seen as independent physical things, and 3rdOrder entities, which represent propositions which can be true or false, e.g. *idea*, *information*, *theory*.

The 1stOrder entities in EuroWordNet have a set of five features associated with them: Form, Origin, Function, Composition and Group, based on the insights of the generative lexicon as well as other lexical semantics theories. The feature values can generally be seen as independent of each other: for example, *person* will have (*Origin Human*) (*Form Object*) while *box* will be associated with (*Origin Artifact*) (*Form Object*). This is a desirable property in a feature set, because if many feature values are mutually dependent, ensuring consistency during lexicon acquisition becomes more difficult.

Although the 1stOrder feature set from EuroWordNet covers most basic properties of objects, it does not have sufficient coverage of spatial properties for our purposes. Physical objects inherently exist in space, and can be seen in different spatial relationships, and we found them very important in several of our domains, especially Monroe [Dzikovska and Byron, 2000]. There is a significant body of work in psycholinguistic literature which describes human spatial processing and why certain phrases, such as *in the hole* or *in the box*, are felicitous, while others, such as *in the button*, sound strange. To develop features for representing spatial properties of objects, we analyzed the work of Herskovits [1986] on spatial predicates, and used it as a basis for our extension discussed in the next section.

The 2ndOrder entities in EuroWordNet have 2 associated features: SituationType, which defines the aspectual properties, and SituationComponent, which defines components relevant to the situation. This set is different in structure from the 1stOrder entities in that the feature values in EuroWordNet 1stOrder are generally orthogonal: something classified as (*Origin Human*) cannot be (*Origin Artifact*) at the same time. In contrast, a 2ndOrder entity assigned (*SituationProperty Location*) can also be assigned (*SituationProperty Manner*). This is undesirable for parsing. Now, if something is marked as (*SituationProperty Location*), we cannot automatically assume that it is not (*SituationProperty Manner*), and the unification algorithm becomes linear in the number of the possible feature values, because now the distinctness of SituationProperty values cannot be used during matching. Therefore, we modified the 2ndOrder set to produce features with mutually exclusive values better suitable for parsing and expressing selectional restrictions.

Another problem we encountered when attempting to use EuroWordNet 2ndOrder features in the TRIPS lexicon is that many entities classified as 2ndOrder objects do not share similar

features. For example, for most verbs and nominalizations, we can specify a *Cause*, which can either be an agent, such as for words *make*, *build*, *calculation*, or it may be a stimulus, such as for words *smell*, *see*. However, for many relations such as *red*, *electrical*, *good* no relevant cause can be identified. In contrast, words like *good* can be classified depending on whether they have degrees of intensity, a concept not relevant to most situations. In addition, times are a highly specialized subclass of words which can be classified neither with situation features nor with other abstract object features, and which have special referential properties, discussed later in this Section.

Given these distinctions, we split the 2ndOrder entities into three subclasses: situations, which include states and events; references to times, and other abstract objects and properties. This 3-way split is helpful in developing relevant feature sets for each of the types, as we discuss in detail in the next section. We kept the 3rdOrder objects as a separate class without change. In the next section, we present a detailed description of our feature system and describe the changes that need to be made to develop a feature set useful for expressing selectional restrictions in parsing.

2.2.2 TRIPS feature values

The EuroWordNet terms 1stOrder, 2ndOrder and 3rdOrder are not very descriptive, therefore used our own names instead to make the representation more readable. The EuroWordNet type 1stOrder denotes physical objects, and in our representation corresponds to feature type *Phys-obj*. As discussed earlier, 2ndOrder was split into 3 subclasses: actions and events (*Situation*), times (*Time*) and other abstract objects and relations (*Abstr-obj*). 3rdOrder category, which corresponds to entities which can be true or false, is called *Proposition* in our set.

Physical Object Features

Consistent with the EuroWordNet 1stOrder definition, the objects assigned *Phys-obj* feature list are the physical objects present in the real world, which can be touched, changed, moved or felt in their impact on the physical things - *truck*, *city*, *electricity*. The words assigned *Phys-obj* type are typically concrete nouns. The features defined for physical objects are *Form*, *Origin*, *Mobility*, *Spatial-abstraction*, *Intentional*, *Container* and *Information*. The feature values are summarized in Table 2.1.

We designed the feature set for physical objects to reflect the most common properties which can be easily determined, and useful in selectional restrictions. The basic qualia related

Feature	Values	Comment
Form	Substance Liquid, Solid, Gas Object Solid-object, Hole, Enclosure, Geographical-object	Substance shape can be can be changed at will to suit the container, e.g. <i>water</i> or <i>metal</i> Objects the real world have their own shape which cannot be randomly changed
Origin	Natural Living Human, Plant, Animal, Creature Natural-non-living Artifact	Exist in nature or as natural creatures of human imagination, e.g., <i>trees</i> , <i>unicorns</i> . Objects and substances made by man
Group	+/-	Plus set on entities denoting collections of objects, e.g., <i>team</i> , <i>furniture</i>
Mobility	Fixed Movable Self-moving, Non-self-moving	Fixed objects are cities and similar large objects which typically cannot change place easily. Movable objects are classified depending on whether they have means to self-locomote or need to be transported.
Spatial-abstraction	Point, Region, Line, Strip	The way people tend to conceptualize physical objects in space
Container	+/-	+ for objects which can hold collections of other objects and substances
Intentional	+/-	+ for entities which can form intentions to act
Information	- Information-content Data, Mental-Construct	For objects which carry associated information which can be summarized or extracted, such as <i>maps</i> and <i>schedules</i> .

Table 2.1: Physical Object features in the TRIPS system

features from EuroWordNet — *Form* and *Origin* — are clearly useful in expressing selectional restrictions. For example, *Form* determines the set of the actions that can be taken on a physical object. For example, only liquids can be poured, while, generally speaking, only objects can be repaired, because, generally speaking, breakage implies some problem with the shape, and substances which do not have a specific form cannot have a problem with it. Similarly, *Origin* is tied to the capabilities of an entity - only living things can be born or die, and only artifacts can be manufactured. Originally, we also included two other qualia-related features, *Function* and *Composition*, in our feature set. However, in the later analysis (discussed in Section 2.5) we found that very few values of the *Function* feature were useful in our selectional restrictions, and we decided not to include it into our current feature set until a better analysis of its usefulness can be made. For the *Composition* feature, we found it useful to keep group versus individual distinction, but we found it difficult to say which things were *Part* in a domain-independent fashion, so we simplified the feature into a binary *Group* feature.

We extended the original EuroWordNet feature set to cover several other important physical object properties we observed in our domains. These are spatial properties, mobility, intentionality and associated information content. The complete list of physical object features in our feature set is enumerated below.

- Form and Spatial-abstraction.

The *Form* feature describes the composition of a physical entity, either a substance or an object. Substances are further subdivided into *Liquid*, *Solid* and *Gas*. We already discussed how these values of *Form* taken from EuroWordNet can be used in selectional restrictions in the system. In addition, we further extended the value set to take into account spatial composition of objects in the world.

Spatial properties are intrinsic for physical objects, and language which describes spatial relationships between objects forms a large part of our everyday vocabulary. We used the work of Herskovits [1986] on spatial predicates as a basis for our extensions. Two of the key ideas of this work is that the way we use spatial prepositions depends on several factors, among them the object composition, and how it can be conceptualized in space. Roughly speaking, object composition determines three-dimensional properties of objects, on which the use of prepositions such as *in* or *on* is based. The conceptualization in space determines larger scale properties of objects: we can imagine objects moving along the streets or rivers, but not as easily along the cities or computers, because normally streets and rivers are visualized as lines in space, while cities are imagined as points or regions,

and relatively small objects like computers are visualized only as points.¹

Based on this idea, we made two extensions to the original physical object feature set. First, we subdivided the (*Form Object*) value into four subtypes to reflect different possible compositions:

solid-object - an object composed of solid substance parts, like a pen or a shelf. Things can be placed *on* solid objects, but not *in* them without changing the object, i.e., generally speaking, we can put something on the shelf, but not in the shelf unless it is a hole or some other structural change.

hole in space, a bounded area surrounded by other objects. Generally speaking, things can be *in* holes, but not *on* them unless some additional inference is assumed, for example, that the object is larger than the hole and covers it completely.

enclosure - a spatial area surrounded by some physical shell, like a box, we often talk about things being *inside* the enclosures

geographical-object - a composed object that occupies some area on the map and can be a collection of smaller objects. With the exception of street names, we cannot generally talk about things being *on* the geographical objects.

We also added a special *Spatial-abstraction* feature with values which correspond to different possible visualizations of objects: *spatial-point*, set for objects that can be represented as abstract spatial points, *line*, *strip*, which is a line of bounded width, and *region*, which is a bounded planar area.

Our system for spatial features is a simplification of a considerably richer system found in Herskovits's work, but we found it sufficient to express useful selectional restrictions in our Monroe domain, which is based on a realistic city map and includes a rich set of locational descriptions [Dzikovska and Byron, 2000].

- Origin

This feature describes the origin of the physical object, which can be *natural* or made by man (*Artifact*). The natural things are subdivided into *Living* and *Non-living*, and living things are partitioned in *plants*, *humans*, *animals* and *creatures* (imaginary living entities such as unicorns). This feature works in selectional restrictions for verbs like *die*, which

¹The possible visualizations are a matter of degree, and there are special context in which even smaller objects can be imagined as regions. However, we felt that it was valuable to provide typical assignments common in the domains, assuming that feature relaxation algorithms can be implemented to relax these constraints as necessary.

can only be applied to living beings, and *make* or *create*, which are generally applicable to artifacts only.²

- Group

This is a binary feature which is set to “+” if the object is a collection of smaller objects uniform in some way, e.g. a team, that is composed of human members. It is similar to the EuroWordNet *Composition* feature, but we simplified it to a binary feature because we could not keep a consistent annotation with the *Part* value of the *Composition* feature.

- Mobility

The *Mobility* feature reflects the fact that motion verbs, similarly to spatial predicates, are a very important class of verbs in human language. We subdivided the physical objects into *fixed* and *movable*. The special subtype of movable objects are *self-moving* objects, which are objects that do not require any external sources to move them, such as people or vehicles. This feature is somewhat domain dependent, since in the right setting almost any object can be conceived as movable. However, in several of our domains there has been a noticeable distinction between the objects that can be moved and those that can not, and we believe that reasonable defaults can be provided, e.g., that in most cases cities are not going to be mobile objects. The idea of including this feature is similar to including the *Mobile* and *Located* distinction present in the domain-independent sortal constraints of CORE language engine [Alshawi, 1992], but our definition of mobility is more extensive, while theirs included only people and animals.

- Intentional

Intentional is a feature which allows us to generalize the fact that not only humans, but complex entities such as corporations and other organizations can have intentions and desires and act on them. We need to keep (*Origin Human*) separate for actions which can only be done truly by humans, such as smiling or crying. But other actions can be ascribed to entire organizations composed of people - we talk about governments deciding on rules, or companies buying and owning stock. In our system, we say that these actions require an agent which is an intentional entity, and correspondingly mark those entities with (*Intentional +*).

²There are obvious exceptions, such as God creating earth and men, but we still found that these distinctions are useful within the domains of practical dialogue dealing with problem-solving and question answering. The same reasoning applies to many other features in our set.

- Container

The *Container* feature is set for objects that can contain other objects and can correspondingly be used in “add to” and “remove from” operations, e.g. trucks, boxes and other containers. Generally speaking, all containers will have (*Form Enclosure*), however, there may be large enclosures (e.g., a tent) which cannot be regarded as containers in usual sense. In addition, containment is a general property that generalizes to abstract objects: plans, schedules or charts can be seen as generalized contained objects to which we can add elements. Therefore, we decided to keep (*Container +*) and (*Form Enclosure*) as separate features.

- Information

We added the *Information* feature to our set because there exist many objects that have relevant “information content” which can be summarized, generalized or re-told, and these include both physical and abstract objects. For example, one can understand the map, summarize a book or revise a chart. Such objects are assigned a feature value (*Information Information-content*), which can have subtypes *Data* or *Mental-Construct*. The *Data* value is set on objects which represent the abstract data and information representations - *map*. *Mental-construct* is sent on the objects with information content related to events or actions - *plan*, *problem*, *issue*, *explanation*, and can only be set for abstract objects or situations.

An example set of words classified as physical objects in the TRIPS lexicon, with features, is shown in figure 2.1.

Situation Features

The objects are assigned feature lists of type *Situation* if they correspond to actions or states that have temporal and aspectual properties, and can be caused, or have location and time arguments, *move*, *accept*, *see*, *meeting*, *dissatisfied*. The words of type *Situation* are usually realized as verbs, event nouns and nominalizations, or deverbal adjectives. The features assigned to *Situation* are *Aspect*, *Time-Span*, *Cause* and *Trajectory*. The feature values are summarized in Table 2.2.

1. Aspect and Time-span

The aspect feature impacts the time adverbials that can modify the verb, the time constructs it can be involved in, and can be selected for by other predicates. We adopted

Person - (Form Solid-Object) (Spatial-abstraction Spatial-point)
 (Origin Human) (Mobility Movable) (Intentional +)
 (Container -)

Truck - (Form Enclosure) (Spatial-abstraction Spatial-point)
 (Origin Artifact) (Mobility Movable)
 (Container +) (Intentional -)

City - (Form Geographical-object)
 (Spatial-abstraction (OR Spatial-point Spatial-region))
 (Origin Artifact)
 (Mobility Fixed) (Container -) (Intentional -)

Figure 2.1: Physical objects and their features in TRIPS-Monroe

Feature	Values	Comment
Aspect	Static Indiv-level, Stage-level Dynamic Bounded, Unbounded	Differentiates between states and events, and between telic(<i>Bounded</i>) and atelic events.
Time-span	Atomic, Extended	Differentiates between achievements and accomplishments.
Cause	Force Phenomenal, Agentive Mental Stimulating	Determines whether the action or state is caused by a natural force, intentional agent, an external stimulus, or a mental construct.
Trajectory	+/-	Set to <i>+</i> for objects that allow PATH arguments.

Table 2.2: Situation features in the TRIPS system

some ideas of Moens and Steedman [1988] for our theory of aspect. In Moens and Steedman's theory, the propositions conveyed by English can be subdivided into events and states. Events can be further subdivided into 4 types: point, culmination, process and culminated process.

An important property of the approach is that the division of events into four classes can be characterized by two features: *conseq*, which determines whether there is a consequent state inherent in the action, and *Atomic*, which determines whether the event can be seen as an instantaneous atomic occurrence, or an extended event which can be subdivided into parts. Points are atomic events which do not have an important consequent state - *e.g.*, *hiccup*, *wink*; culminations are atomic events where the consequent state is important - *e.g.*, *recognize*, *reach*, culminated processes are extended events where the consequent state is important, *e.g.*, *building a house*, and processes are extended events where the consequent state is not important in context, *e.g.*, *running*, *walking*.

The EuroWordNet SituationType feature distinguishes between Bounded and Unbounded events, which corresponds to events with or without the consequent state. We therefore used that feature (renaming it to Aspect), and added another feature, Time-span, to distinguish between atomic and extended events.

- Aspect
 - Dynamic. Dynamic situations involve some sort of change in the world, and correspond to events in Moens and Steedman theory. Their subtypes are *bounded* situations, which have a definite culmination point and result in a state that is pragmatically important (*+conseq* in Moens and Steedman), and *unbounded* situations, where the process is more important than the culmination point (*-conseq* in Moens and Steedman).
 - Static

Static situations correspond to real world states. EuroWordNet subdivides those in properties and relations. However, this classification is orthogonal to static/dynamic distinction (see also [Alonge *et al.*, 2000]), thus, we removed the subdivision. Instead, we classify states as

 - * *Stage-level* predicates - states which can be transient and change with time. Verbs marked as *Stage-level* can participate in progressive constructions, *e.g.*, *I am thinking that ...*

* *Individual-level* - states that characterize the individual and are unlikely to change with time. Verbs marked as *Individual-level* typically cannot participate in the progressive constructions, e.g., **I am knowing that ...*

- Time-span

This feature classifies situations as *atomic*, if the preparatory process is not important, or extended otherwise. Atomic situations cannot be modified by time interval adverbials unless we do some type coercion, but they can be easily modified by AT-TIME adverbials. In Moens and Steedman’s theory, atomic/extended distinction did not apply to states. In our system, we set all states to have the time values *extended* by definition. Whether the extended situations can be modified by “at-time” adverbials without type coercion depends on the Aspect value.

Note that the feature values described above are assigned to words as typical values associated with the word. In the process of parsing, this assignment may change depending on aspectual coercion as described in [Moens and Steedman, 1988].

2. Cause

The cause feature determines the basic cause behind a situation and its relationship with the verb. These can be: *force*, which is subdivided into *agentive*, where the situation is brought about by an intentional entity, and *phenomenal*, where the situation is changed by a natural force, e.g., weather phenomena; *stimulating*, when the situation is caused by the impact of the outside world on a passive agent (experiencer), and *mental*, where the situation reflects a state of mental attitudes of a being, e.g. love or belief.

In the EuroWordNet ontology, *Agentive* and *Phenomenal* are two distinct SituationCause values, without a common parent. In practice, however, many events can have either natural or intentional causes - a destruction of a city, for example, can be caused either by a hurricane, which is a natural force, or by an army, which is an intentional force. Therefore, we introduced a common parent *cause* in our feature system to denote cases like this.

3. Trajectory

Path modifiers are a large subclass of domain-independent adverbials, and they show up especially frequently in our domains where transportation is involved. Therefore, we added *Trajectory* feature as an indicator for verbs which can have path arguments. It is set to “+” on the verbs that can be modified by trajectory adverbials.

Some example feature assignments for verbs in the lexicon are given in figure 2.2

Move - (Aspect Dynamic) (Cause Agentive)
 (Time-Span Extended) (Trajectory +)
 Accept - (Aspect Bounded) (Cause Agentive) (Trajectory -)
 (Time-Span Atomic)
 See - (Aspect State) (Cause Stimulating) (Trajectory -)
 (Time-Span Extended)

Figure 2.2: Words classified as situations in TRIPS-Monroe

Abstract object features

Abstract objects properties of objects that are not related to events or states and cannot have location or time arguments - *e.g., red, electrical, good*. The words with *Abstr-obj* type are usually realized as abstract nouns, adjectives or adverbs. The features associated with abstract objects are *Gradability, Measure, Scale* and *Information* and *Intentional*. Features for abstract objects are summarized in Table 2.3.

Feature	Values	Comment
Gradability	+/-	For objects that can have grades.
Scale	Weight, Volume, Area, Length, Color, Speed	For abstract entities referring to values on various scales
Measure	-, Point, Value, Interval, Unit	The abstraction on the scale
Container	+/-	+ for objects which can hold collections of other objects and substances
Intentional	+/-	+ for entities which can form intentions to act
Information	- Information-content Data, Mental- Construct	For objects which carry associated information which can be summarized or extracted, such as <i>maps</i> and <i>schedules</i> .

Table 2.3: Abstract object features in the TRIPS system

1. Gradability This is a binary feature that stems from the notion of gradability for adjectives. Some adjectives, such as *good*, *pretty* are gradable [Rusiecki, 1985]. They can come in different intensities, and can participate in comparative and superlative constructions, so it is possible to say *prettier* or *very good*. Other adjectives, such as *electrical* or *additional*, do not have degrees and expressions such as **very additional* do not make much sense, and so they are marked as (*Gradable -*).
2. Measure-function and Scale

A large number of abstract objects are actually measure functions over some abstract domain - pounds and kilograms measure weight, seconds and minutes measure time, in a way, one can say that color is a function over an abstract domain with values corresponding to a set of possible colors in the world. This intuition is captured with measure and scale features. The scale feature reflects the abstract scales that can be measuring things, and can currently take values such as *distance*, *weight*, *volume*, *area*, *color*, *speed*, and *quantity*. Units of measure form a large but closed set of words (barring physical advances). Currently we only provided the scale values for the most common physical units, classifying all other units of measure as *Other-scale*, but we intend to extend this set as necessary.

The *Measure-function* is set to reflect the different ways concepts can be related to a scale. The basic distinction is between a *Value* on a scale and a *Term* which can take values. For example, *5 meters* is a value of a term *length*, and *red* is a value of the term *Color*. *Units* are separated out as a special subtype of values, because they exhibit special behavior in measure phrases.

Examples of abstract objects in the lexicon are in figure 2.3.

```

Electrical - (Gradability -) (Scale -) (Measure-function -) (Information -)
Centimeter - (Scale Length) (Measure-function Value) (Gradability -)
              (Information -)
Good -       (Information -) (Intentional -) (Gradability +)
              (Scale -) (Measure-function -)

```

Figure 2.3: Abstract objects in the TRIPS-Monroe lexicon

Times

times are a special class of objects which participate in many syntactic patterns different from other words, and have special semantic and referential properties. Normally times cannot be referred to by pronouns such as *it* and *that*. For example, *on it* cannot possibly be interpreted as a reference to time as in *on Monday*, while it can clearly be a location, as in “on the book”. Therefore objects denoting times, including time periods such as days or minutes, or words referring to times such as *deadline* are assigned a special feature set of type *Time*.

The *Time* set has two functions associated with it: *Time-function* and *Scale*. The former corresponds to the way a given time can function in an utterance, and the latter encodes whether a word refers to a period or a point.

1. *Time-function*. This feature reflects different kinds of temporal entities a word can denote.

These are the following:

- *Time-of-day* is set for words and phrases locating a specific point or interval during the day - which can be more specifically defined within a second, minute or hour. It is subdivided into
 - *Clock-time* - an explicit clock time, *e.g.*, 5pm, midnight. Normally, preposition *at* can only be used with explicit times, and explicit times cannot be used with *during* and other duration adverbials.
 - *Day-part* - an expression that identifies part of the day, *e.g.*, evening, afternoon. These are used with *in* or *during* prepositions, and can participate in constructions with day names, such as *Monday afternoon*.
- *Time-of-year* values refer to the entities identifying points on the calendar - these are *Day-of-week*, *e.g.*, *Monday*, *Month-name*, *e.g.*, *July*, *Date*, *e.g.*, *21st of May* and *Year-name*, *e.g.*, *1993*. All of these can be used with *on* or *in* prepositions, and the feature values are used to determine allowable combinations in dates, such as *Monday, June 21st*.
- *Time-interval* - values used to identify the words that refer to time intervals, with a special subclass of *Time-units*, which are units that can participate into frequency expressions, such as *every 5 minutes*, and in the duration specifications such as *for 5 hours*.
- *Frequency* - value used to identify frequency phrases, such as *twice*.

2. *Time-Scale*. Scale is a feature which determines whether a given word can serve as a point or as an interval in time. For example, *5pm* is clearly a point, and something can happen *at 5 pm*, but not **during 5pm*. In contrast, *3 hours* is an interval, so things can happen *for 3 hours*, but not *by 3 hours*.

It is important to note that (*Time-scale Interval*) and (*Time-function Time-interval*) overlap. However, they are distinct. For example, *July* is a time interval, though its primary function is *Month-name*. This is important distinction, because function determines how something can function in date formation, and finer distinctions between temporal prepositions, but *Time-scale* matters as a general category for many verbs such as *take 5 hours*, and we found it convenient to have it as a separate classification.

Figure 2.4 shows examples of feature assignments in our lexicon.

```
July - (Time-Scale Interval) (Time-function Month-name)
Hour - (Time-scale Interval) (Time-function Time-unit)
Noon - (Time-scale Point) (Time-function Clock-time)
```

Figure 2.4: Times in TRIPS-Monroe lexicon

Common features

There is a small set of features that can be associated with entities of any type. They were originally introduced above as defined for physical objects.

- *Intentional* - set for physical or abstract objects that can have intentions and act on them, *e.g.*, people, which are physical entities, and corporations, which are abstract entities which nevertheless can have intentions
- *Container* - set for physical and abstract objects that can contain other objects and can correspondingly be used in “add to” and “remove from” operations. Physical containers are entities like boxes and trucks. But we can also see notions like *plans*, *schedules* or *recipes* as containers to which we can add steps and appointments, licensed by (*Container +*) feature.
- *Information* - in addition to physical representations which contain abstract content, such as *maps*, abstract objects can be datasets that can be summarized, *e.g.*, *data*, *information*, which are associated with (*Information Data*) value. Information that can be summarized

may also be relevant to actions and events, corresponding to (*Information Mental-construct*) feature, e.g., *plan, problem, issue, explanation*.

2.2.3 Feature Dependencies

We noted previously that the feature value assignments in the system are generally independent, so that assigning (*Origin Artifact*) does not restrict the values that can be assigned to *Mobility* or *Form*, for example. This is a very useful property in a feature set. During lexicon acquisition, if feature value assignments are interdependent, extra work is required to ensure that feature sets assigned to all words are consistent. In addition, independent feature values can be acquired/learned from different sources without worrying how setting them will impact the rest of the set.

However, it is very difficult to find a set where all feature values are completely independent. For example, if we would like have (*Origin Living*) in the lexicon, because obviously words like *die* or *born* can apply to living beings only. However, living beings obviously cannot be substances, so (*Form Gas*) should not be permitted as a feature assignment in this case. This is supported in our system with a feature inference rule mechanism which allows the developer to say that a feature value for one feature implies that some other features need to have pre-set values.

As a design principle for our feature set, we tried to keep the feature dependencies to a minimum. The complete list of current feature inference rules is shown in figure 2.5.

The rules are used two ways in our system: they are used to check the consistency of a feature set, and to infer feature values if none are specified. For example, if the word *person* is assigned a feature set with features (*Phys-obj (Origin Human) (Form Solid-object)*), the rule is used to check that *Solid-object* is a subtype of *Object* and is therefore acceptable in combination with (*Origin Human*). On the other hand, if for a word like *Plant* we specified only (*Phys-obj (Origin Plant)*) in the lexicon, the rule would be used to add (*form object*) to that definition, instead of the usual default which is an unrestricted value. We discuss the way feature inference is applied, along with default feature values, in Chapter 3 as part of the general discussion of the TRIPS lexicon organization. Furthermore, a general feature dependency mechanism is very useful in domain specialization. We will return to it in Chapter 4, where we will show how feature dependencies can be used to specialize entries and easily integrate domain-independent and domain-specific information in the lexicon.

(Origin Living)	:implies	(Phys-obj (Intentional +) (Form Object))
(Origin Plant)	:implies	(Phys-obj (Intentional -))
(Origin Human)	:implies	(Phys-obj (Intentional +) (Form Solid-object))
(Origin Animal)	:implies	(Phys-obj (Form Solid-object))
(Aspect Static)	:implies	(Situation (Time-span Extended))

Figure 2.5: The list of feature dependencies in the TRIPS feature system

2.3 Using features in selectional restrictions

2.3.1 Motivating Examples

So far, we have introduced an intuitive notion of features and feature lists, and selectional restrictions as requirements on feature sets of words that need to be combined. To ensure that these operations are sound, we need to give theoretical definitions of operations that create and compare feature sets. While most of the time intuitive notions of feature restrictions work well, there are some more complex cases when we need to have a precise definition of what it means to satisfy a selectional restriction to make sure that they are treated correctly. In our discussion, we will focus on three main problems: the use of underconstrained nouns such as *organism*, pronouns, and conjunctions.

In our discussion, we will consider a set of definitions in Figure 2.6. Intuitively, they define features for physical objects *boy*, *dog*, *house*, *organism* and *idea*, and three actions, *meet*, *see* and *smile*. The restriction on *smile* is that only humans can smile, and *meet* is an action that it involves humans or animals meeting other humans and animals. These definitions are the simplified definitions of words from our lexicon, with feature sets restricted to only a couple of features for exposition purposes.

The main issue that needs to be addressed is what does it mean to satisfy a selectional restriction. Consider the following examples:

- (2) a. The boy smiled
 b. *The idea smiled
 c. ?The organism smiled

In most cases, it is reasonably clear whether selectional restrictions are satisfied: (2a) is good, because only humans or animals can smile, and a person is clearly a human; (2b) is clearly bad,

Boy: (Phys-obj (Origin Human) (Form Solid-object))
 Dog: (Phys-obj (Origin Animal) (Form Solid-object))
 House: (Phys-obj (Origin Artifact) (Form Enclosure))
 Organism: (Phys-obj (Origin Living) (Form Solid-object))
 Idea: (Abstr-obj (Information Information-content))
 Smile: (Situation (Aspect Unbounded) (Time-span Atomic)
 (Subject (Phys-obj (Origin Human))))
 Meet: (Situation (Aspect Unbounded) (Time-span Atomic)
 (Subject (Phys-obj (Origin (or Human Animal))))
 (Object (Phys-obj (Origin (or Human Animal))))
 See: (Situation (Aspect Static) (Time-span Extended)
 (Subject (Phys-obj (Origin (or Human Animal))))
 (Object (Phys-obj))
 Break: (Situation (Aspect Dynamic) (Time-span Atomic)
 (Subject (Phys-obj (Form Object))
 (Object (Phys-obj (Form Object)))

Figure 2.6: Simple feature sets and selectional restrictions

because the abstract object *idea* is in not compatible with (*Origin human*) or (*Origin Animal*). But what about (2c)? People are organisms, but not all organisms are people. There are two possible answers to this question. One possibility is to interpret a selectional restriction in (2c) as giving us extra information: something about which we knew before only that it was a living thing, we now discover through the language that it is more than that, it is a human or an animal. This means we are being permissive - as long as we find that the objects and the restriction could potentially be compatible, the restriction is satisfied. It is particularly helpful for reference resolution - now we can view pronouns as entities with underconstrained feature sets, but in the process of parsing this set is specialized with additional information coming from selectional restrictions.

A different choice is to say that the example (2c) is unacceptable, it should be handled as an exceptional occurrence in a conversation, and not allowed in the regular speech. So if we are talking about smiling organisms, than something is wrong with our interpretation, and we may need to check our assumption. This approach is particularly useful in situations where recognition errors are possible, for example, a dialogue system connected to a speech recognizer which produces multiple recognition hypotheses for each word. In this case, our best strategy may be to choose a different speech recognition hypothesis that uses a more specific words.

These two approaches correspond to two different operations on the feature lists - unification and the subtype relationship. In the next section, we are going to discuss the formal definitions and the impact the choice of the operation has on implementing two major parts of the system: pronoun resolution and conjunctions.

A similar issue arises with handling pronouns in the system, because they also have underconstrained types. In a practical parser, where checking restrictions is automatic for all verb-object combinations, we need to assign feature sets to all linguistically meaningful objects, including pronouns like *it* or *she*. Clearly the selectional restrictions need to be checked in that case, because *It smiled* is not a good sentence given our restriction that only people can smile. It turns out that it is easier to handle pronouns when unification is used to check selectional restrictions, and we will return to this issue in more detail in the next section.

Another question that needs to be answered is how conjunction interacts with selectional restrictions. The problem arises when we conjoin two entities with disparate feature sets, for example *a house and a boy*. The feature value of *Origin* is (*Origin Artifact*) for *house*, and (*Origin Human*) for *boy*. The question is, what is the resulting feature value for *Origin* on the noun phrase *The house and the boy*? The answer makes a difference in the following examples:

- (3) a. I saw a house and a boy
 b. *I met a house and a boy
 c. I met a dog and a boy
 d. ?I met an organism and a boy

In the example (3a), *house* is (*Origin Artifact*), while *boy* is (*Origin Human*). I am going to argue that this conjunction is viable because *see* does not restrict the *Origin* value of its objects. In contrast, example (3b) is bad because *meet* requires its objects to be either (*Origin Human*) or (*Origin Animal*), and *house* is (*Origin Artifact*). At the same time, example (3c) satisfies the restriction on *meet*, even though *dog* is (*Origin Animal*) and *boy* is (*Origin Human*), because *meet* allows either. The interpretation of the example (3d) depends on the decisions we are making for the other underconstrained type example, (2c). If it is acceptable, then (3d) must be considered acceptable, and vice versa.

Intuitively, we want to make a system for selectional restrictions checking, under which a feature value assigned to the conjunction should be such that fails to satisfy the restriction if one of the conjunction elements does not satisfy it, but passes the selectional restriction if every conjoined NP satisfies it, even if the values on those NPs.

A similar issue arises when two selectional restrictions are conjoined, for example, if two verb phrases have a common subject. We are going to look at the following set of examples:

- (4) a. *A dog saw a house and smiled
 b. A boy saw a house and smiled
 c. *A dog and a boy saw a house and smiled

We need to make sure that the first sentence is ruled out because *dog* satisfies the condition on *meet*, but not *smile*. The second example is acceptable, because *boy* satisfies restrictions on both verbs, even if they are not compatible. Finally, in the example (4c) the collective reading must be ruled out because part of the conjunctive noun phrase, *a dog*, does not satisfy the restriction on *smile*, which is part of the conjunctive verb phrase.

As it turns out, the solution to those problems is not trivial. For that purpose, we are going to introduce a special set-making operation, and show how it interacts with our unification operations.

2.3.2 Formal feature operations

So far, we have discussed an intuitive definition of feature lists, and feature values that can be used in natural language processing. In order to apply the notion in the practical system, we need to define a set of operations which we can use to give formal semantics to notions of selectional restrictions. In this section we define a formalized notion of typed feature sets in terms of assignment functions, and formally define the operations needed for implementation of selectional restrictions. In the next section, we discuss in more detail the applicability of those operations and design considerations that arise in the process.

Operations on feature values

Carpenter [1992] and Copestake [1992] give the generalized definition of subsumption and unification on typed feature structures with complex values. In our case, the feature structures contain only simple values and the definitions can be simplified as follows.

Definition 2.1 (Inheritance Hierarchy) *An inheritance hierarchy is a pointed finite partial order $\langle \mathbf{Val}, \sqsubseteq \rangle$*

In this definition, \mathbf{Val} denotes a finite set of values. For example, in the hierarchy for feature *Origin* shown in Table 2.1, the \mathbf{Val} consists of values $\langle \textit{Natural}, \textit{Living}, \textit{Human}, \textit{Plant}, \textit{Creature}, \textit{Animal}, \textit{Non-living}, \textit{Artifact} \rangle$. The hierarchy is ordered by the subtype relation \sqsubseteq , where a less specific value is said to be a subtype of a more specific value.³ For example, $\textit{Human} \sqsubseteq \textit{Living} \sqsubseteq \textit{Natural}$. By definition of partial ordering, \sqsubseteq is reflexive, anti-symmetric and transitive: $x \sqsubseteq x$; $x \sqsubseteq y$ and $y \sqsubseteq x$ iff $x = y$; if $x \sqsubseteq y$ and $y \sqsubseteq z$, then $x \sqsubseteq z$. In our hierarchy it means that every concept by definition is a subtype of itself, $\textit{Human} \sqsubseteq \textit{Human}$; that there are no cycles: if *Natural* is a supertype *Human*, then *Human* is not a supertype of *Natural*; and that a parent is a supertype of all its descendants, that is, given that $\textit{Human} \sqsubseteq \textit{Living}$, $\textit{Plant} \sqsubseteq \textit{Living}$, and $\textit{Living} \sqsubseteq \textit{Natural}$, we get that both $\textit{Human} \sqsubseteq \textit{Natural}$ and $\textit{Plant} \sqsubseteq \textit{Natural}$.

³There is disagreement over notation in current literature. Carpenter[1992] uses the term “a more general value subsumes a more specific value”, denoting it by \sqsubseteq symbol, while Copestake [1993] says that “a more specific value subsumes a more general value”, also denoted by the \sqsubseteq symbol. We chose to follow the latter notation, with \sqsubseteq denoting “more specific than” relationship, and replacing the term “subsumes” with less ambiguous “is subtype of” and “is supertype of”

The pointedness requirement means that there is a unique element \top such that for any $v \in \mathbf{Val}$, $v \sqsubseteq \top$.⁴ In our system, we automatically add a root value to every feature value hierarchy. For example, we will add *Any-origin* as a root of the *Origin* hierarchy in Table 2.1. In our notations, we will assume that if a feature value is not specified, then it defaults to the root value \top .

Definition 2.2 (Value Unification) *The unification of feature values $v_1, v_2 \in \mathbf{Val}$ $v_1 \sqcap v_2$ is their highest common bound under the subtype relation, that is, a value v such that $v \sqsubseteq v_1$, $v \sqsubseteq v_2$ and for any v' such that $v' \sqsubseteq v_1$ and $v' \sqsubseteq v_2$, $v' \sqsubseteq v$.*

The unification of two features is their greatest lower bound, that is, the highest common subtype. In our feature hierarchy, *Natural* \sqcap *Human* = *Human*. By definition, unification is symmetrical: $v_1 \sqcap v_2 = v_2 \sqcap v_1$.

Given our value set for the *Origin* feature, the unification is not complete - for example, *Plant* and *Animal* do not have a common subtype. To ensure that the result of a unification operation is always defined, we introduce a special type, \perp , which is the result of the unification in such cases, and write that *Plant* \sqcap *Animal* = \perp .⁵

In practical terms, \perp represents a unification failure result. Now we can use unification operation for selectional restrictions - if the result of the unification is \perp , the restriction does not apply, so if the restriction required *Plant*, then something marked as *Animal* won't satisfy it. We also define that for any value v , $\perp \sqcap v = \perp$. This ensures that once the application of a selectional restriction failed, it cannot be reversed in the future. The application of subsumption and unification to selectional restrictions is discussed in detail in section 2.3.3.

Definition 2.3 (Value Meet) *The meet of feature values v_1 and v_2 $v_1 \sqcup v_2$ is their least upper bound under the subtype relation, that is, a value v such that $v_1 \sqsubseteq v$ and $v_2 \sqsubseteq v$ and for every v' such that $v_1 \sqsubseteq v'$ and $v_2 \sqsubseteq v'$, $v \sqsubseteq v'$.*

The meet of two feature values is the the lowest value which is a set supertype of both. Thus, for *Plant* and *Animal*, *Plant* \sqcup *Animal* = *Living*. We will return to using meets later, in the discussion of conjunction.

⁴This makes the an upper semilattice, where every pair of elements has a least upper bound. We are using the terminology from Carpenter [1992] in this case.

⁵This in effect makes our hierarchy a lattice, because every pair of elements has a greatest lower bound.

Operations on feature sets

Now that we have definitions for operations on single feature values, we can define operations on the typed feature sets.

Definition 2.4 (Typed Feature Sets) *A typed feature set is a tuple $F = \langle T, Q \rangle$ where T is the atomic set type, and Q is a function $Q : \mathbf{Feat} \rightarrow \mathbf{Val}$, where \mathbf{Feat} is a set of all possible feature names, and \mathbf{Val} is the feature value set.*

In the examples in this section, we will use a feature set for the word *person*, defined in Figure 2.1, and repeated here in Figure 2.7. We will write it as $FP = \langle TP, QP \rangle$. Under our definition, $TP = \text{Phys-obj}$, $QP(\text{Origin}) = \text{Human}$, $QP(\text{Form}) = \text{Solid-object}$, etc We adopt a notational convention that for any feature not shown explicitly in the feature set, the assumed value is the root value, $Q(f) = \top$.

$TP = \text{Phys-obj} (\text{Form Solid-Object}) (\text{Spatial-abstraction Spatial-point}) (\text{Origin Human})$
(Mobility Movable) (Intentional +) (Container -)

Figure 2.7: A feature set for the word *Person* we will use in our examples

Our definition of typed feature sets is a weaker case of the definition given by Copestake. In her definition, feature values themselves can be typed feature structures, which necessitates introduction of the notion of a path through a feature structure, which is a trace between the root of the structure and an atomic node. In our case, features can only have simple values, therefore all path have length 1, and the notion of a path is redundant

We can now define subtype and meet operations between feature sets based on the operations defined for individual values.

Definition 2.5 (Typed Feature Set Subsumption) *A feature set $F_1 = \langle T_1, Q_1 \rangle$ is a subtype of $F_2 = \langle T_2, Q_2 \rangle$, $F_1 \sqsubseteq F_2$ iff $T_1 \sqsubseteq T_2$ and for every $f \in \mathbf{Feat}$, $Q_1(f) \sqsubseteq Q_2(f)$.*

For example, our set TP is a subtype of a feature set $\langle T_2, Q_2 \rangle = (\text{Phys-obj} (\text{Origin Living}))$, because $TP = \text{Phys-obj} \sqsubseteq \text{Phys-obj } T_2$, $TP(\text{Origin}) = \text{Human} \sqsubseteq Q_2(\text{Origin}) = \text{Living}$, and for all other features, $TP(\text{Form}) \sqsubseteq \top = Q_2(\text{Form})$, $TP(\text{Intentional}) \sqsubseteq \top = Q_2(\text{Intentional})$ etc

Our definition is equivalent to Copestake's definition of subsumption. On the surface, there are two differences. First is that Copestake uses paths, but we already discussed that this is

not an issue: saying $Q(f)$ in our feature structures is a value of a 1-node path which ends at an atomic feature f , and there are no other paths possible in our structures. The other difference is that in Copestake's definition the function Q is a partial function, so it may not be defined on some values, so the subtype relationship holds as long as it holds for all features defined on Q_2 . By our definition Q is a total function, and if we do not specify a feature value explicitly when we write up a feature set, it is assumed to be \top . In practice, this is equivalent to saying that the feature is "not present". This is illustrated by the example in previous paragraph: since for any value v , $v \sqsubseteq \top$, it does not matter what value is assigned to features *Form* in TP , it will always be a subtype of \top . The same reasoning holds for all other definitions presented in this section.

We can now define unification and meet for feature sets similarly to the way we defined them for single values.

Definition 2.6 (Typed Feature Set Unification) *The unification of typed feature sets*

$F_1 = \langle T_1, Q_1 \rangle$ and $F_2 = \langle T_2, Q_2 \rangle$ $F_1 \sqcap F_2$ is

(1) a tuple $F = \langle T, Q \rangle$ where $T = T_1 \sqcap T_2$, $Q(f) = Q_1(f) \sqcap Q_2(f)$ iff $T_1 \sqcap T_2 \neq \perp$ and for all $f \in \text{Feat}$ $Q_1(f) \sqcap Q_2(f) \neq \perp$

(2) \perp otherwise.

It is easy to see that this definition is equivalent to saying that $F_1 \sqcap F_2 = F$ iff $F \sqsubseteq F_1$, $F \sqsubseteq F_2$ and for any F' , $F' \sqsubseteq F_1$ and $F' \sqsubseteq F_2$ implies $F' \sqsubseteq F$, which is a general definition of unification. By definition, feature unification is a symmetric and reflexive relation, similarly to the unification of the feature values.

For example, if we have a set $F_2 = (\text{Phys-obj } (\text{Origin Living}))$, then $F_2 \sqcap FP = FP$. This is because if $F_1 \sqcap F_2 = \perp$ we are going to say that the unification because $\text{Phys-obj} \sqcap \text{Phys-obj}$, $\text{Living} \sqcap \text{Human} = \text{Human} = Q_2(\text{Origin})$, and for all other features, $Q_1(\text{Form}) = \top \sqcap Q_2(\text{Form}) = Q_2(\text{Form})$, etc. In contrast, for $F_3 = (\text{Phys-obj } (\text{Origin Artifact}))$, $F_3 \sqcap FP = \perp$, because $\text{Living} \sqcap \text{Artifact} = \perp$. What this means in practice is that if the selectional restriction requests (Origin Living) , the feature set for *person* will pass it, but the feature set for *truck* will fail, because the latter has an *Origin* value which does not unify with *Living*.

Similarly to the feature values, we are also going to define a meet on typed feature sets.

Definition 2.7 (Typed Feature Set Meet) *The meet of typed feature sets* $F_1 = \langle T_1, Q_1 \rangle$

and $F_2 = \langle T_2, Q_2 \rangle$ $F_1 \sqcup F_2$ is a tuple $F = \langle T, Q \rangle$ where $T = T_1 \sqcup T_2$, $Q(f) = Q_1(f) \sqcup Q_2(f)$ for all $f \in \text{Feat}$.

Again, this definition is equivalent to the general definition of meet as an operation so that if $F = F_1 \sqcup F_2$, then $F_1 \sqsubseteq F$, $F_2 \sqsubseteq F$ and for any F' , $F_1 \sqsubseteq F'$ and $F_2 \sqsubseteq F'$ implies $F \sqsubseteq F'$.

From the three operations defined in this section, we will use subtype and unification directly in selectional restrictions. We will use the meet operation later in Chapter 3 when we define a formal semantics for feature dependency rules.

Set feature values

So far we have discussed the formal semantics of feature sets assuming that every feature has an atomic value. We now proceed to extend the definition to the notion of disjunctive values, formalized as sets. In the selectional restrictions, we sometimes need to represent a notion that something is either a plant or an animal, but not a human or any other concept compatible with *Living*. If we only allow atomic feature values, we cannot express it with our feature set. One option is to complete the hierarchy with nodes that correspond to all possible disjunctions, e.g. *Animal_or_plant*, *Animal_or_human*, etc., but this will result in addition of a large number of extra nodes, making the representation less transparent. Therefore, we need to define a formal semantics for feature values as sets of possibilities.

To do that, we introduce a notion of disjunctive sets, which are sets of pairwise incompatible values which represent uncertainty as to the actual feature value.

Definition 2.8 (Disjunctive Sets) *A disjunctive set \mathbf{vset} of feature values is a set $\{v | v \in \mathbf{Val}\}$ such that for any $v', v'' \in \mathbf{vset}$ if $v' \neq v''$, then $v' \not\sqsubseteq v''$.*

In disjunctive sets, all the values are guaranteed to be distinct - for example $\{human, artifact\}$ satisfies the definition of a disjunctive set, because an object can be either a human being, or an artifact, but not both. In contrast, $\{living, human\}$ is not a disjunctive set, because anything that is human is also living. Disjunctive sets allow us to represent uncertainty about the type of the object, and by enforcing incompatibility between the types in a set. We can now extend the definitions of subsumption, unification and meet to disjunctive sets as follows.

Definition 2.9 (Disjunctive set subsumption) *Given disjunctive sets σ and τ , $\sigma \sqsubseteq \tau$ iff for any $v \in \sigma$ there is $v' \in \tau$ such that $v \sqsubseteq v'$.*

On the base level, if we assume that a simple type *Natural* is a singleton disjunctive set, this definition says that *Natural* is more specific than $\{Natural, Artifact\}$, so we can count that

if something is (a subtype of) *Natural*, it will be consistent with a restriction that it should be either *Natural* or *Artifact*. Similarly, $\{Human, Natural\text{-non-living}\} \sqsubseteq \{Natural\}$ because $Human \sqsubseteq Natural$ and $Natural\text{-non-living} \sqsubseteq Natural$. In a more complex example, if we know that something is $\{Plant, Artifact\}$, that is, either a plant or an artifact, then we can conclude that it will satisfy the restriction $\{Natural, Artifact\}$, because if it is *Plant*, it is a subtype of *Natural*, and if it is *Artifact*, then obviously it is a subtype of *Artifact*. In practice, the restrictions don't usually get that complex, but it is useful to have the formal definition to ensure soundness of reasoning.

We will define the meet of the disjunctive sets first, because unification is easier to define in terms of meets.

Definition 2.10 (Disjunctive set meet) *Given disjunctive sets σ and τ , $\sigma \sqcup \tau = \{\gamma \in \sigma \cup \tau \mid \text{for any } \gamma' \in \sigma \cup \tau, \gamma' \neq \gamma \Rightarrow \gamma \not\sqsubseteq \gamma'\}$* ⁶

Intuitively, if we are computing a meet of two simple values, we walk up the hierarchy to find their common parent. If we are computing a meet of two sets, we can take the union of two sets (which provides a more general disjunctive set). But the union may result in a non-disjunctive set if an element in one set is a subtype of an element of the other set, so we convert the union into a disjunctive set by getting rid of the more specific type in every such pair. For example, $\{Natural, Artifact\} \sqcup \{Natural\} = \{Natural, Artifact\}$, because neither $Natural \sqsubseteq Artifact$ nor $Artifact \sqsubseteq Natural$. $\{human, Natural\text{-non-living}\} \sqcup \{Natural\} = \{Natural\}$, because both $Human \sqsubseteq Natural$ and $Natural\text{-non-living} \sqsubseteq natural$, so they are removed to keep the union as a disjunctive set. Finally, $\{Plant, Artifact\} \sqcup \{Natural, Artifact\} = \{Natural, Artifact\}$, because $Plant \sqsubseteq Natural$.

Unification is then defined as a meet of all lower bounds for two sets:

Definition 2.11 *Given disjunctive sets σ and τ , $\sigma \sqcap \tau = \sqcup\{\gamma \mid \gamma \sqsubseteq \sigma \text{ and } \gamma \sqsubseteq \tau\}$*

Intuitively, if we want to unify two sets, we need to find a set of elements which have parents in both sets, and this set must be made as general as possible. Then $\{Natural, Artifact\} \sqcap \{Natural\} = \{Natural\}$, because it is the smallest set which can be consistent with both being a natural thing and being either of natural or of artificial origin. $\{Human, Natural\text{-non-living}\} \sqcap \{Natural\} = \{Human, Natural\text{-non-living}\}$, because both $Human \sqsubseteq Natural$

⁶Throughout this thesis, we will be using the standard set notation symbols: \cup for union, \cap for intersection, \subset for a subset relationship.

and $Natural\text{-}non\text{-}living \sqsubseteq Natural$, and there are no other elements such that $\gamma \sqsubseteq Natural$ and either $\gamma \sqsubseteq Natural\text{-}non\text{-}living$ or $\gamma \sqsubseteq Human$. Finally, $\{Plant, Artifact\} \sqcap \{Natural, Artifact\} = \{Plant, Artifact\}$, for similar reasons.

Carpenter [1992] shows that given a subtype hierarchy, it can be completed to a hierarchy equivalent under disjunctive set formation. However, this results in the explosion of the number of nodes - to complete our hierarchy for the *Origin* feature, we would need to add 10 more values corresponding to all possible disjunctive subsets to the hierarchy. We chose to keep the disjunction operation in place in our system, to be used as necessary in restrictions.

The main purpose of introducing the disjunctive value sets is the capability to express restrictions like “a subject of *walk* must be either a human or an animal”. To simplify our notation, we are going to denote by $v_1 \vee v_2$ the operation of disjunctive set formation from two feature values. Based on the definitions of the disjunctive types, We have that

- $v_1 \vee v_2 = \{v_1\}$ iff $v_2 \sqsubseteq v_1$;
- $v_1 \vee v_2 = \{v_2\}$ iff $v_1 \sqsubseteq v_2$;
- $v_1 \vee v_2 = \{v_1, v_2\}$ otherwise (if v_1 and v_2 are incompatible)
- $v_1 \sqsubseteq (v_2 \vee v_3)$ iff $v_1 \sqsubseteq v_2$ or $v_1 \sqsubseteq v_3$
- $(v_1 \vee v_2) \sqsubseteq v_3$ iff both $v_1 \sqsubseteq v_3$ and $v_2 \sqsubseteq v_3$

These definitions can be extended straightforwardly to disjuncts with multiple variables.

2.3.3 Feature operations in selectional restrictions

Having defined the formal semantics for feature operations, we can now discuss its application to selectional restrictions on feature sets. As we discussed in the section , the first question to be asked is what it means to satisfy a selectional restriction. With the operations we defined two alternative definitions are possible:

Definition 2.12 (Weak selectional restrictions) *A feature set F satisfies a selectional restriction R if and only if $F \sqcap R \neq \perp$*

Definition 2.13 (Strong selectional restrictions) *A feature set F satisfies a selectional restriction R if and only if $F \sqsubseteq R$*

It should be noted that if F satisfies a selectional restriction R under the strong definition, then it is also the case that F satisfies the weak definition, because $F \sqsubseteq R \Rightarrow F \sqcap R = F \neq \perp$. If F does not satisfy a weak selectional restriction, then it also does not satisfy a strong restriction, because $F \not\sqsubseteq R$ implies $F \sqcap R = \perp$.

Let us now consider how these definitions apply to our motivating examples (2). Table 2.4 summarizes the relevant information. The $F \sqcap R$ column corresponds to the weak restriction, and $F \sqsubseteq R$ column to the strong restriction. As can be seen from the table, The difference between strong and weak restrictions shows in treating the borderline example 2c. Obviously, the good example (2a) is acceptable for both weak and strong versions of the restrictions, and the bad example (2b) is rejected by both weak and strong versions. But in the case of (2c), with weak selectional restrictions it is considered acceptable, because $F \sqcap R$ is defined, but this example is rejected in case of strong selectional restrictions, because $living \not\sqsubseteq human$, hence, $F \not\sqsubseteq R$.

Example	R	F	$F \sqcap R$	$F \sqsubseteq R$
(2a) The boy smiled	<i>Phys-obj</i> (<i>Origin Human</i>)	<i>Phys-obj</i> (<i>Origin Human</i>)	<i>Phys-obj</i> (<i>Origin Human</i>)	yes
(2b) *The idea smiled	<i>Phys-obj</i> (<i>Origin Human</i>)	<i>Abstr-obj</i>	\perp	no
(2c) ?The organism smiled	<i>Phys-obj</i> (<i>Origin Human</i>)	<i>Phys-obj</i> (<i>Origin Living</i>)	<i>Phys-obj</i> (<i>Origin Human</i>)	no

Table 2.4: Selectional restrictions for our motivating examples

As discussed in section 2.3.3, the decision whether such borderline examples are acceptable in the system corresponds to a pragmatic difference whether we want to be permissive and see the restrictions as giving us additional information about the semantics of the object with underdetermined features, or whether we want to treat the restrictions as requiring the most specific information possible, and rejecting these examples in favor of trying different interpretations. From the point of view of unification grammars the weak version of the restrictions is the most natural, because it corresponds directly to the unification operation and can therefore be built into the grammar easily. In the TRIPS system parser, we chose to implement the weak selectional restriction mechanism. However, it appears that most reasoning systems which include type constraints use subsumption, and, therefore, a stronger version of selectional restrictions. The semantics proposed above gives us the formal framework to describe and discuss such implementation decisions. It can also be used to guide the system building decisions, as

well as to build constraint-relaxation algorithms: in a system that allows automatically relaxing constraints, it makes sense to start with the strong version of the restrictions, then relax it as needed to the weak restrictions.

An important place where the processing of selectional restrictions makes a difference is the semantics assigned to pronouns. Consider two examples:

- (5) a. A boy saw it
 b. *It smiled
 c. I broke it

We want to ensure that *it* is assigned a feature set such that Example (5a) is acceptable to the system, while Example (5b) is rejected. It is very easy with weak selectional restrictions: we associate *it* with a feature set

$FIT = (Phys-obj (Origin \{Animal, Plant, Natural-non-living, Artifact\}))$.⁷ Then it will unify with every possible feature set except for the one that states (*Origin Human*), and thus satisfy any selectional restriction which does not require (*Origin Human*). It is more complicated, however, with strong restrictions: it is easy to check, for example, that $FIT \not\sqsubseteq (Phys-obj (Form Object))$ because $FIT(Form) = \top \not\sqsubseteq Object$. Thus, *it* won't be accepted as an object for a verb *broke* in Example (5c).

A formalization presented here does not offer an explicit way to resolve this difficulty with strong selectional restrictions. If they are used in a practical system, since the class of affected items is quite small (pronouns and a few special nouns such as *thing*), an exception in the procedure can be implemented to allow the restrictions to fail in this case. The full formalization of strong restrictions can be done by introducing variable types which can range over a set of values and be compatible with all of them. We did not further explore this issue because we made a choice in favor of weak restrictions in the TRIPS system, but we present the issues here since they would affect the implementation decisions that need to be made in developing any practical systems with selectional restrictions.

Let us now discuss selectional restrictions in combination with conjunction. Let's start with strong selectional restrictions, which present an easier case. Given noun phrases with feature sets $F_1 = \langle T_1, Q_1 \rangle$ and $F_2 = \langle T_2, Q_2 \rangle$, assign to the conjunction the feature set $F = \langle T_1 \vee T_2, Q_1 \vee Q_2 \rangle$ where $Q_1 \vee Q_2$ is a function Q such that $Q(f) = Q_1(f) \vee Q_2(f)$.

⁷It is a little bit more complex, because *it* should be able to also refer to abstract objects, but we will ignore this complication for a moment

Example	R	F1	F2	F1∨F2
(3a) I saw a house and a boy	$Phys-obj$	$Phys-obj$ $(Origin\ Artifact)$	$Phys-obj$ $(Origin\ Human)$	$Phys-obj$ $(Origin$ $(Human\ \vee\ Artifact))$
(3b) *I met a house and a boy	$Phys-obj$ $(Origin$ $(Human\ \vee\ Animal))$	$Phys-obj$ $(Origin\ Artifact)$	$Phys-obj$ $(Origin\ Human)$	$Phys-obj$ $(Origin$ $(Human\ \vee\ Artifact))$
(3c) I met a dog and a boy	$Phys-obj$ $(Origin$ $(Human\ \vee\ Animal))$	$Phys-obj$ $(Origin\ Animal)$	$Phys-obj$ $(Origin\ Human)$	$Phys-obj$ $(Origin$ $(Human\ \vee\ Animal))$
(3d) ?I met an organism and a boy	$Phys-obj$ $(Origin$ $(Human\ \vee\ Animal))$	$Phys-obj$ $(Origin\ Living)$	$Phys-obj$ $(Origin\ Human)$	$Phys-obj$ $(Origin\ Living)$

Table 2.5: Selectional restrictions with conjunctions when \vee is used to create feature sets for collective NPs. For brevity, we only provide values of *Origin* feature.

Table 2.5 summarizes the relevant examples and features assigned to conjunctive NPs in that case. For brevity, we only show the values of feature *Origin*, because this is the only feature *R* restricts. The strong selectional restrictions are satisfied iff $F1 \vee F2 \sqsubseteq R$. Our prediction is that good cases (3a) and (3c) should be accepted by both strong and weak selectional restrictions, (3b) should be rejected, and the borderline Example (3d) should be acceptable under weak restrictions and rejected under strong restrictions.

Indeed, it is easy to see that examples (3a) and (3c) satisfy strong selectional restrictions. Example (3b) does not satisfy strong selectional restrictions, because $Artifact \not\sqsubseteq Human$ and $Artifact \not\sqsubseteq Animal$, and hence $(Artifact \vee Human) \not\sqsubseteq (Artifact \vee Animal)$. Similarly, Example (3d) does not satisfy the strong restriction because neither $Living \sqsubseteq Human$ nor $Living \sqsubseteq Animal$. This is fully consistent with our idea of strong restrictions as accepting only cases in which the object clearly satisfies the restriction.

In our verb phrase conjunction examples in (4), the situation is more problematic. If we declare the selectional restrictions of the conjoined phrase are disjunctions of selectional restrictions from component phrases, we have a problem. In the example (4c), *A dog and a boy saw a house and smiled*, the restriction on the conjoined VP will then be $(Origin\ (Living \vee Animal))$, which is obviously satisfied by *a dog and a boy* even under strong restrictions. We will return to this problem later in the section, after we define a notion of collective sets.

Another difficulty with conjunction arises for weak restrictions. It is easy to check that examples (3a,3c) are judged correctly - as we observed before, if something satisfies a strong

restriction, it will also satisfy a weak restriction. Example (3d), is also judged acceptable, as we predicted for weak restrictions. The problem arises in the case of (3b). We have that $(Animal \vee Artifact) \sqcap (Animal \vee Human) = Animal \neq \perp$, thus, this example is judged acceptable under weak restrictions, which is not correct.

Clearly, some special handling is necessary to deal with selectional restrictions if conjunction is involved. A similar problem was encountered by Alshawi [1992](p175). He proposes as a solution “union terms”. A union term is a set such that if a sortal constraint is applied to it, it must be applied to every member. However, the notion was not formalized in the system. In this work, we extend Copestake’s and Carpenter’s definitions with a notion of a collective set, and use it to formalize selectional restrictions for conjunctions.

Definition 2.14 (Collective Sets) *Collective sets are a class of sets with values $\{v | v \in \bigvee \mathbf{Val}\}$ with the following property with respect to subtype operations:*

If C_1 and C_2 are collective sets then $C_1 \sqsubseteq C_2$ iff for every $v_1 \in C_1$ and every $v_2 \in C_2$, $v_1 \sqsubseteq v_2$.

We will denote the collective sets by square brackets $[\]$, for example, $[Human, Artifact]$. It is easy to see that \sqsubseteq induces a partial hierarchical order between collective sets. The hierarchy is pointed, with the unique top level element $[\top]$. The leaves of the hierarchy are two types of collective sets. One is singleton sets, which consist of a single non-disjunctive value which itself is a leaf of a value hierarchy, for example $[Human]$. Another type of “leaf” set in the collective set hierarchy is a special set we will call an incompatible collective set.

Definition 2.15 (Incompatible Collective Sets). *We will call a collective set $C = [v_1, \dots, v_n]$ **incompatible** if $v_{11} \sqcap v_{12} \dots \sqcap v_{1k} = \perp$.*

It is easy to see that for any incompatible collective set C , there is no other set $C' \neq C$ such that $C' \sqsubseteq C$. By definition of collective set subsumption, every element of C' will have to be a subtype of every element of C . At the same time, by definition of incompatible collective sets, the elements of C have no common subtype. Thus, it cannot be possible to find a set $C' \sqsubseteq C$.

An example incompatible set is $C = [Human, Artifact]$, because $Human \sqcap Artifact = \perp$. It is useful to know when a collective set is incompatible, because it simplifies finding unification between sets, as we will see when we discuss examples of selectional restrictions later in this section.

Note that if a collective set is incompatible, it does not prevent it from entering subtype relationships. For example, $[Human, Artifact] \sqsubseteq [\top]$, because $Human \sqsubseteq \top$ and $Artifact \sqsubseteq$

⊥. Similarly, $[Human, Animal]$ is incompatible, but $[Human, Animal] \sqsubseteq [Living, Natural]$ because $Human \sqsubseteq Living$, $Animal \sqsubseteq Living$, $Human \sqsubseteq Natural$, $Animal \sqsubseteq Natural$. It is only the reverse that is true - no other collective set can be a subtype of an incompatible collective set.

Our last example also points out at one more property of collective sets - unlike disjunctive sets, they can contain members which subsume other members. For example $\{Living, Natural\}$ is not a valid disjunctive set, because $Living \sqsubseteq Natural$. But it is a valid collective set under our definition. This is an important property of collective sets, and examples later in this section will show how it helps us to handle conjunction examples which are not possible to handle correctly with disjunctive sets.

We can also use the standard definition of unification as greatest upper bound under subsumption for collective sets.

Definition 2.16 (Unification of collective sets) *Given two collective sets C_1 and C_2 the unification $C_1 \sqcap C_2$ is a set C such that $C \sqsubseteq C_1$, $C \sqsubseteq C_2$ and for any C' , if $C' \sqsubseteq C_1$ and $C' \sqsubseteq C_2$, then $C' \sqsubseteq C$.*

We gave a general definition of unification for collective sets, but let us consider how it can be constructed. It is easy to see that if $C_1 \sqsubseteq C_2$, then $C_1 \sqcap C_2 = C_1$. Now assume that $C_1 \not\sqsubseteq C_2$ and $C_2 \not\sqsubseteq C_1$.

- C_1 or C_2 is incompatible. Then, $C_1 \sqcap C_2 = \perp$. This is easy to see, because by definition of incompatible set, if C_1 is incompatible, we can find no other set $C \sqsubseteq C_1$, thus the unification fails unless $C_1 \sqsubseteq C_2$.
- $C_1 = [v_{11}, \dots, v_{1k}]$, $C_2 = [v_{21}, \dots, v_{2l}]$, and $v_{11} \sqcap \dots \sqcap v_{1l} = v_1$, $v_{11} \sqcap \dots \sqcap v_{1l} = v_2$. Then, $C_1 \sqcap C_2 = [v_1 \sqcap v_2]$. Again, it is easy to see why this is the case given that the unification of C_1 and C_2 must contain the types which are common between all elements C_1 and C_2 .

Table 2.6 provides a summary of relevant feature sets when collective sets are used for conjoined noun phrases.

It is straightforward to see that Example (3a') satisfies both strong selectional restrictions with collective sets, as does Example (3a''), because $Human \sqsubseteq (Human \vee Animal)$ and $Animal \sqsubseteq (Human \vee Animal)$. It immediately follows that these sets also satisfy weak selectional restrictions.

Example	R	$F = [F1\ F2]$	$F \sqsubseteq R$	$F \sqcap R$
(3a') I saw a house and a boy	<i>Phys-obj</i>	<i>Phys-obj (Origin [Human Artifact])</i>	yes	<i>Phys-obj (Origin [Human Artifact])</i>
(3b') *I met a house and a boy	<i>Phys-obj (Origin (Human \vee Animal))</i>	<i>Phys-obj (Origin [Human Artifact])</i>	no	\perp
(3c') I met a dog and a boy	<i>Phys-obj (Origin (Human \vee Animal))</i>	<i>Phys-obj (Origin [Human Animal])</i>	yes	<i>Phys-obj (Origin [Human Animal])</i>
(3d') ?I met an organism and a boy	<i>Phys-obj (Origin (Human \vee Animal))</i>	<i>Phys-obj (Origin [Living Human])</i>	no	<i>Phys-obj (Origin Human)</i>

Table 2.6: Selectional restrictions with conjunctions when collective set formation \sqcap is used to create feature sets for collective NPs. For brevity, we only provide values of *Origin* feature.

In the example (3b'), $F \not\sqsubseteq R$ because *Artifact* $\not\sqsubseteq$ $(Human \vee Animal)$. Thus, the strong restriction is not satisfied, and the example is correctly ruled out. For it to be ruled out under a weak restriction, we need to show that $[Animal, Artifact] \sqcap [(Human \vee Animal)] = \perp$. Notice that the set $F = [Animal, Artifact]$ is incompatible. From that, and from the fact that neither $F \sqsubseteq R$ nor $R \sqsubseteq F$, it follows immediately that $F \sqcap R = \perp$, and the example is correctly ruled out.

In Example (3d'), $[Living, Human] \not\sqsubseteq [(Animal \vee Human)]$, because *Living* $\not\sqsubseteq$ $(Animal \vee Human)$, so under strong restrictions it is judged unacceptable. In contrast, we can compute $[Living, Human] \sqcap [(Animal \vee Human)]$. According to our construction procedure, we compute $v1 = Living \sqcap Human = Human$ for F , and $v2 = (Animal \vee Human)$ for R . Then, $F \sqcap R = [v1 \sqcap v2] = [Human]$, and the weak restriction is satisfied.

Thus, we demonstrated that using collective sets in selectional restrictions allow us to handle both weak and strong cases properly and make correct predictions for all our examples. We are using the property of collective sets we pointed out earlier: that in a collective set we can have separate elements which are subtypes of each other. So during unification a disjunctive set $\{Living, Human\}$ is collapsed into $\{Living\}$, which is what causes the incorrect example (3c') to be accepted for weak restrictions. In a collective set, they are kept separated, as $[Living, Human]$, and this is what enables us to handle the example correctly.

Now we can come back to our problem with restrictions on conjoined verb phrases, assuming they are done with the collection operation. Table 2.7 summarizes the relevant information.

Example	R	F	$F \sqsubseteq R$	$F \sqcap R$
(4a) *A dog saw a house and smiled	$Phys\text{-}obj \ (Origin \ [((Human \vee Animal), Human)])$	$Phys\text{-}obj \ (Origin \ Animal)$	no	\perp
(4b) A boy saw a house and smiled	$Phys\text{-}obj \ (Origin \ [((Human \vee Animal), Human)])$	$Phys\text{-}obj \ (Origin \ Human)$	yes	$Phys\text{-}obj \ (Origin \ Human)$
(4c) *A dog and boy saw a house and smiled	$Phys\text{-}obj \ (Origin \ [((Human \vee Animal), Human)])$	$Phys\text{-}obj \ (Origin \ [Human, Animal])$	no	\perp

Table 2.7: Selectional restrictions with conjunctions when collective set formation \sqcap is used to create feature sets for verb phrase conjunctions. For brevity, we only provide values of *Origin* feature.

We need to verify that (4a) and (4c) are ruled out, while (4b) is accepted for both strong and weak selectional restrictions. First, consider (4b). It is accepted by strong selectional restrictions because $Human \sqsubseteq Human$ and $Human \sqsubseteq (Animal \vee Human)$. Since the strong selectional restriction is satisfied, the weak is satisfied, too.

Now consider strong restrictions for examples (4a,4c). Example (4a) is ruled out because $Animal \not\sqsubseteq Human$; (4c) is rejected because we need that $(Animal \vee Human) \sqsubseteq [Human, Human \vee Animal]$, which is not true because $(Animal \vee Human) \not\sqsubseteq Human$.

In case of weak restrictions, the situation is very similar. For (4a), $Animal \sqcap [Human, Human \vee Animal]$ fails because the only set $\delta \sqsubseteq [Animal]$ is $[Animal]$, and $Animal \not\sqsubseteq Human$. For (4c), $[Animal, Human] \sqcap [Human, Human \vee Animal] = \perp$, because $[Animal, Human]$ is incompatible, and $[Animal, Human] \not\sqsubseteq [Human, Human \vee Animal]$.

Thus, collective sets can be used to properly handle all kinds of selectional restrictions, including our complex case when both the subject and the verb phrase are conjunctions. We also demonstrated that a weaker implementation, using disjunctive sets, is not capable of handling all relevant cases properly. When we developed our system, we also evaluated the possibility of using the meet operation to represent feature values in conjunctions. We do not present the results here, but it is fairly straightforward to check that it encounters problems similar to using disjunctive sets, so it is not suitable for implementing selectional restrictions with conjunction.

There is an obvious compromise involved in practical implementation of selectional restrictions. Subtype, unification and meet are straightforward operations typically implemented in unification parsers. Collective set unification is somewhat more difficult because it requires special support and implementation of special collective feature structures. Our current implementation uses disjunctive sets, even though we are aware of the cases it does not handle correctly. We are planning the implementation of collective set operations as part of future system development.

2.4 Discussion

We have described the feature set we devised for our system, and a formal semantic that underlies implementation of selectional restrictions based on typed feature set. We now return to a more general discussion, to create a set of guidelines for developing practical feature sets, and describe their advantages and disadvantages compared to using ontological types in selectional restrictions.

2.4.1 Efficiency Issues

Our first motivation for using feature sets was efficiency. Unification and subtype operations on our feature sets are linear in the number of features, but it relies on several assumptions. The first assumption is that we have a simple type system, without multiple inheritance between the types. This is a big difference between our system and a system like ACQUILEX. In ACQUILEX, it is possible to declare many different types of feature structures with multiple inheritance between them. For example, we can declare an artifact type characterized by a defined TELIC structure, and a physical type, characterized by having a physical form, and then declare the art_phys type, characterized by both TELIC and FORM structures inherited from the two parents. This clearly enhances the expressiveness of the system, but at the cost of efficiency in matching. The LINGO system, which uses similar feature structures in its semantic representations, requires a significant amount of pre-compilation of its type system and possible unifications to operate efficiently [Kiefer *et al.*, 1999]. Part of the idea of this work was to find a middle ground between simple predicates and exceedingly complex feature structures for use in a practical system, and, in our experience, the simple feature structures with just 4 types and a small number of feature co-occurrence restrictions are adequate for most of our needs.

Another assumption we are making in our system is the form of the restrictions. In the worst case, the algorithm for checking a selectional restriction is linear in the number of features in the system set. Since all features come from single inheritance hierarchies, we can check unification for two feature values in constant time using Schubert’s subtype algorithm [Schubert, 1979] implemented in our system. Thus, if we need to match two complete feature sets, the time will be linear in the number of features. However, in practice most restrictions can be checked in constant time because in most cases, a selectional restriction uses no more than one or two features, and these are the only values that need to be checked. In fact, the restrictions on multiple features correspond to conjunctive restrictions in a multiple inheritance hierarchy - for example, looking for something that is an artifact and that is also liquid in form.⁸

2.4.2 Designing a feature set

The feature sets with simple values give us an edge in efficiency, but another important point to consider is the expressiveness of our representation. In most existing systems, selectional restrictions are stated as restrictions on object types - so the object of *drive* must be of type *VEHICLE*. In our system, we propose to state selectional restrictions in terms of features only. But features do not fully describe the concepts - there’s more to a notion of *car* than a movable artifact with object shape which can function as a vehicle. Stating restrictions in terms of object types can provide finer granularity. For example, in a system that allows placing restrictions on full types, if we are talking about a verb *treat*, we could say that its object must be of type *MEDICAL-CONDITION*, provided that the type is defined in the ontology. If we are defining a restriction in terms of our feature system, we can only say in feature sets that the object of *treat* must be (*Situation (Cause Stimulating)*), that is, it is restricted to the events and states caused by external factors, which is a wider class than *MEDICAL-CONDITION*, so the restriction is more permissive.

Thus, expressing permission in terms of types has a finer grain and therefore is more expressive. However, it carries in itself a danger of over-specializing. In this case, if we restrict objects of *treat* to be medical conditions, we potentially exclude symptoms like *itching* or *tiredness*, which, by themselves, are just body sensations which are not necessarily medical conditions. This can be expressed through disjunction, but finding all relevant types to include becomes an

⁸Or they correspond to a complete semi-lattice in a conjunctive hierarchy. In a complete semi-lattice the matching can be done in constant time, but at the expense of introducing nodes for all possible feature combinations, with corresponding penalty in space, or the need to pre-compile to determine all combinations relevant in the system at a particular stage in development.

increasingly difficult task as the ontology grows, and caused us significant difficulty when we tried to switch from a simpler Pacifica domain to a more complex Monroe domain. The advantage of using feature list is that, in effect, the feature lists provide a guide for determining the restriction - one needs to select the general type of the entity involved, and specify the relevant properties defined by features, which is the smaller set to choose from than the entire ontology for language, which can have hundreds and thousands of concepts. The disadvantage is the loss of precision. Note, however, that since most selectional restrictions are domain-dependent, absolute precision is not always possible. In chapter 4, we describe the system that combines domain-independent restrictions expressed by features with domain-specific restrictions expressed by full types to achieve the better speed and accuracy in parsing and disambiguation.

A way to control the granularity of restrictions expressed through features is by limiting or expanding the feature set in the system. In the extreme, all verbs have their own idiosyncratic restrictions - *e.g.*, we could conceive of a *treatable* feature for a verb *treat*. Whether we choose features or full types as restrictions in our implementation, we need to approximate at some point. The level of approximation with features may be higher than with full types, but in a domain-independent application we are better off under- than over-constraining the inputs, and specifics are handled later with domain specialization. The goal in our lexicon is to provide sufficient features to distinguish the ambiguity whenever possible between the role assignments and between the specific senses, but not to exclude every impossible combination of words.

In selecting the reasonable feature sets, we established the following guidelines:

- The features should describe linguistic generalizations which pertain to large classes of words. For example, *Form*, *Origin* and *Function* features correspond to qualia elements which have been long proposed as impacting the behavior of noun and adjective phrases for physical objects. In contrast, a *Treatable* feature would be relevant to only a very small class of verbs
- A feature value should apply to a large class of words. For example, there is a large variety of things that can be assigned (*Form Gas*), and also a large class of words which can be associated with (*Form Object*). For a *Treatable* feature, the class of words to which it would pertain is relatively small, denoting illnesses and certain physical sensations.
- A feature value should have disambiguation power over several different classes of words. For example, (*Form Substance*) will disambiguate *take* and *have* in the CONSUME sense, as in *I take aspirin every morning*, with *take* in a general REMOVE sense, as in *I took the*

book away. In addition, it can help differentiate CONTAIN sense of *hold*, as in *This can holds 5 gallons of water* from a more general GRASP sense, as in *She held the child in her arms*.

In our feature set, most features satisfy at least two of those criteria. Originally, in addition to the features described earlier, our set also contained the feature *Object-Function*, corresponding to EuroWordNet function feature. After the data analysis we discovered that we have not used its values in selectional restrictions. Therefore, we chose to remove it from our feature set.

We would like to emphasize one additional advantage to using features in the lexical semantic representation - extensibility. We discussed several linguistic phenomena covered by our feature set: spatial properties of objects used for selectional restrictions of spatial prepositions, aspectual properties of verbs, gradability of abstract objects, form and origin of physical objects used as a restriction of some verbs. There are other areas not currently covered by our feature system, for example better classification of abstract objects, and there are also finer distinctions that can be made in the areas of such as spatial properties of objects and situations. Using a feature representation enables incremental refinement of our system, when new features can be added later to account for additional semantically-grounded linguistic phenomena. As long as it means an addition of a new feature to the set, it can be done very easily, without changing or disturbing the rest of the system. This is a very desirable property for a practical system, which is somewhat more difficult to achieve in an ontology based on multiple inheritance. As more and more relationships are encoded in the ontology, it becomes more difficult to determine all the parents of a given concept. Having a set of features, along with a description that guides the assignment of the values, considerably simplifies this tasks.

2.4.3 Comparison with VerbNet features

As a way of evaluating our feature set, it is interesting to compare it with the VerbNet feature set [Schuler, 2002], an independently developed feature set to express selectional restrictions in the VerbNet verb lexicon. It is also based on the EuroWordNet top ontology, therefore it shares many similarities with our approach. Our set is more extensive: we have 14 features with 129 possible values, when their binary features allow at most 72 values (discounting possible feature conjunctions in restrictions in both lexicons). Similarly to our lexicon, VerbNet adds an *intentional-control* feature, however, in VerbNet lexicon more objects come under intensional division - it also includes machines, and forces. This is a different approach from our system, because we treat machines doing actions as a result of a coercion which would be reflected

in the logical form of an utterance, while VerbNet would treat both humans and machines as intentional in the same context.

VerbNet does not contain our detailed features for tense and aspect. This is due primarily to the fact that they use compositional verb semantics based on Moens and Steedman [1988], thus the same features are encoded directly into their representations.

VerbNet system includes several features not present in our system. In particular, for physical objects these are shape (pointed or elongated) and rigidity (rigid or non-rigid), used to differentiate instruments and patients for verbs such as *poke* and *coil*.⁹ There are also several specific subclasses of abstract objects, namely *idea*, *sound*, *scalar*, *currency* and *organization*. Given our design guidelines, the number of the examples in VerbNet is too small to justify those features, however, we may consider including them in the future. Particularly, the *organization* feature appears on many verbs. Currently these entities are covered by our (*Intentional +*) feature, and it may be useful to make a more fine-grained division of intentional objects, between animate or organizations.

As can be seen from this discussion, our feature set and VerbNet share some similarities, which indicate that there is a core set of features useful for expressing selectional restrictions in computational lexicons. However, there are also differences. On the one hand, VerbNet contains a number of verbs we have not yet added to our system. Therefore, we have not yet found some features from VerbNet system useful, but it can clearly change in the future. On the other hand, VerbNet does not explicitly encode adverbs or nouns, so it has no direct need to express selectional restrictions for them. Our *Situation* features were primarily used on selectional restrictions for adverbs in our system, thus we need them present explicitly in our feature set, and not just implicitly in the semantic form. Similarly, our system has a more complex set of features for times, spatial properties and abstract measures and scales, reflecting our need for selectional restrictions on temporal and spatial prepositions, as well as adjectives and functional noun phrases, which are not part of the VerbNet lexicon.

2.5 Evaluation

We developed a natural language lexicon with selectional restrictions using our feature set. The lexicon organization is discussed in more detail in Chapter 3. Here we provide the statistics

⁹Currently, *+elongated* may be modeled with (*Spatial-abstraction Line*), and we have no nothing that corresponds to rigidity or for *pointed* feature.

on how features are used in our system, and which features are frequently used in selectional restrictions. These statistics reflect to some extent that our domain-independent lexicon was build together with the development of several application domains, therefore the coverage is best on words related to those areas.

Table 2.8 describes the distribution of different feature list types in selectional restrictions and grammar. The most frequent feature type in our restrictions is physical object, used in selectional restrictions of 449 words. This reflects the fact that most verbs in our domain involve an agent, or some physical object. Abstract objects are the least frequently used restriction. This may be because in our domains so far we didn't need to deal much with abstract objects like units and quantities. Selectional restrictions and coverage for abstract objects, especially involving measure phrases, are part of our ongoing research.

Feature List	Sel. Restr.	Grammar	Comment
TIME	51	14	
SITUATION	166	24	
PROPOSITION	7	1	wrong true so right incorrect false correct
PHYS-OBJ	449	4	
ABSTR-OBJ	25	0	

Table 2.8: Feature types used in grammar and selectional restrictions

Table 2.9 list the frequency of physical objects features. We provide two counts: the number of times a feature was used in selectional restrictions on lexical entries, and the number of times it was used in the grammar rules to directly restrict the application of the rule. The most frequently used values are (*Mobility Movable*), (*Intentional +*) and (*Origin Human*). This makes sense, because they are related to the most basic properties of objects: intentionality and origin are relevant to being an agent or experiencer of actions, an argument most of the actions and states have, and *Mobility* is relevant to verbs and prepositions connected with motion and physical locations. Since our system was initially developed for transportation domains, the verbs of motion are a large and well-developed class, and therefore the related features, *Mobility*, as well as (*Trajectory +*) for situations, appear frequently in the selectional restrictions.

Tables 2.12, 2.11 and 2.13 provide statistics for situation, abstract object and time features respectively. Features for abstract objects, especially, are used less frequently in our system, because until recently we had very few abstract objects in our lexicon. Adding more adjectives

Feature	Sel.restr.	Grammar	Comment
Form	76	1	
Substance	14		
Solid	0		
Liquid	0		
Gas	0		
Object	62		
Solid-object	5		wounded sick injured dead
Hole	0		
Geographical-object	12	1	
Enclosure	0		
Mobility	11	0	
Movable	106		
Self-moving	2		fly drive
Non-self-moving	0		
Fixed	5		
Origin	146	0	
Natural	142		
Living	141		
Plant	0		
Human	88		
Creature	0		
Animal	13		
Non-living	1		with
Artifact	2		with (clock speed)
Spatial-abstraction	98	3	
Strip	32	1	
Spatial-region	23	1	
Spatial-point	16		
Line	27	1	

Table 2.9: Physical object features in selectional restrictions

Feature	Sel.restr.	Grammar	Comment
Container	17	1	
-	5		wounded sick injured dead
+	12	1	
Information	55	0	
-	0		
Information-content	55		
Mental-construct	7		
Data	0		
Intentional	161	13	
-	14	13	
+	147		

Table 2.10: Features shared between physical objects and other types in selectional restrictions

and functional nouns, which are normally classified as abstract objects, is an active area of our current research.

It is important to note that while some features appear with count 0 in the tables, it does not mean that they are not used effectively in the system. An example of this is (*Aspect Indiv-level*), which never comes up in selectional restrictions. However, the opposite value, (*Aspect Stage-level*), is used frequently in our restrictions, thus we can say that both these values the binary *Aspect* feature are useful, because setting (*Intentional Indiv-level*) on a word prevents it from satisfying a frequently used restrictions for progressive constructions.

Another infrequently used set of features is *Measure-function* and *Scale*. They have rather fine-grained values, which explains why they are infrequently used. However, physical measurements such as weight and volume are an extremely important part of any language, and we felt that including those features is justified because they describe a set of very common phrases, which is likely to bring performance benefits in many domains.

Another way to evaluate the importance of features in our system is to see the impact selectional restrictions have for parsing. We conducted an experiment to see the impact of selectional restrictions on speed and accuracy of our system. To do that, we selected two corpora: a 99-sentence corpus in our medication adviser domain, and a 368 sentence corpus which consisted of all sentences in dialogue s12 in the Monroe domain. We parsed both corpora with our parser with domain-independent restrictions enabled, and compared its performance to a version of a

Feature	Sel.restr.	Grammar	Comment
Gradability	5	13	
-	0	13	
+	5		that most more less least
Measure-function	13	0	
Value	13		
Unit	0		
Term	0		
Scale	10	0	
Weight	1		weight
Volume	1		volume
Temperature	1		temperature
Rate	2		speed (clock speed)
Quantity	0		
Other-scale	0		
Money	1		budget
Length-s	4		long length height dis- tance

Table 2.11: Abstract object features in selectional restrictions

Feature	Sel.restr.	Grammar	Comment
Aspect	187	2	
Static	77		
Stage-Level	77	1	
Indiv-Level	0		
Dynamic	110	1	
Unbounded	9		
Bounded	2		within in
Time-span	14	0	
Extended	14		
Atomic	0		
Cause	29	0	
Force	29		
Phenomenal	0	1	
Agentive	29	1	
Stimulating	0	1	
Mental	0	1	
Trajectory	38	0	
-	0		
+	38		

Table 2.12: Situation features in selectional restrictions

Feature	Sel.restr.	Grammar	Comment
Time-function	25	6	
Time-of-day	13	3	^bout near in bout at around about
Clock-time	6	1	^bout near bout at around about
Day-part	7	1	^bout near in bout at around about
Day-point	6		^bout near bout at around about
Day-period	1		in
Time-of-year	6	1	on in
Day-of-week	1		on
Month-name	1		in
Year-name	1		in
Date	0	1	
Time-interval	6	1	within wait long in for during
Time-unit	6		within wait long in for during
Frequency	0	1	
Time-scale	3	13	
-	0	5	
Point	0	6	
Interval	3	2	take save leave

Table 2.13: Time features in selectional restrictions

parser which did not check selectional restrictions at all. The results are presented in Table 2.14. We report results on two sets of the data: our entire corpus (Medadvisor and Monroe-s12) and the portion of the corpus which consists of sentences with 3 or more words (Medadvisor-3 and Monroe-s12-3). The latter discounts “easy” sentences like “OK” or other acknowledgments where there is only one possible interpretation. We compare the performance at 3 levels of restrictions: domain-specialized restrictions using our features (the semi-automatic specialization procedure is discussed in Chapter 4), parsing using domain-independent restrictions only, and parsing without any selectional restrictions at all.

This table reveals that domain-specialized selectional restrictions are important for best parsing performance, and we will return to it in Chapter 4. However, even in comparison with the weaker generic restrictions, not using them results in 50% increase in the number of constituents built by the parser, with the corresponding drop in parsing speed. Moreover, there is 12% absolute (20% relative) decrease in accuracy on our test set, most of it over the longer sentences. On sentences with 3 or more words, where there is 66% decrease in efficiency on the sentences without any restrictions compared to sentences using selectional restrictions, and 96% decrease in speed compared to sentences with domain-specialized restrictions.

The differences in performance are even more pronounced in the Monroe corpus: there is 2.2 times difference in speed between the version with domain-specialized restrictions¹⁰. There is also an associated 11% absolute (13% relative) drop in accuracy. The accuracy drop is especially pronounced on 3 word sentences - there is a 22% relative decrease in accuracy when parsing without selectional restrictions compared to parsing with selectional restrictions.

Thus, using our selectional restrictions, even weak domain-independent ones, has a significant impact on the quality of parsing, and this effect is consistent across our application domains.

2.6 Conclusions and Future Work

In this chapter, we described a feature set for expressing selectional restrictions in a natural language parser and developed a formal semantics for selectional restrictions. Our method is used in a multi-domain parsing system described in the next chapter. As our statistics show, our feature system is best developed for the physical objects and situations. We developed an initial classification of features for abstract objects, but additional analysis could be useful, especially in

¹⁰There is not a substantial difference in our Monroe domain between a domain-specialized and domain-independent lexicon because we do not have a complete domain description

Corpus	Sel. restr.	# of sentences	Constituents Built	Accuracy
Medadvisor	kr	99	85	70.7
Medadvisor	generic	99	95	62.6
Medadvisor	no	99	158	52.5
Medadvisor-3	kr	79	105	64.5
Medadvisor-3	generic	79	118	53.16
Medadvisor-3	no	79	196	41.8
Monroe-s12	kr	368	125	78.5
Monroe-s12	generic	368	136	77.8
Monroe-s12	no	368	275	67.9
Monroe-s12-3	kr	269	163	72.1
Monroe-s12-3	generic	269	183	70.3
Monroe-s12-3	no	269	374	56.1

Table 2.14: The comparison of parsing results with and without selectional restrictions in the Medication Advisor domain

conjunction with a domain such as purchasing or financial news, where words denoting abstract entities are more common.

Our current implementation of feature list semantics is sufficient to express selectional restrictions for the cases we encountered, and we developed a theory for more complex cases, such as conjunctive phrases. One operation missing from our system is disjunction of feature sets. As described above, we allow for disjunctions of feature values, but there is a small set of examples which require full disjunction. For example, a word *that* can refer to a physical object which is not human, or to any abstract object. We cannot express this disjunction in our system and currently need to enter two separate sense entries for the word. Extension of the system to cover such cases is planned as future work.

We provided an evaluation of the usefulness of features in terms of their frequency in selectional restrictions. A more detailed quantitative evaluation could be useful. One way to achieve this is to evaluate the efficiency of our selectional restrictions when a given feature excluded from the feature set, and compare it with parsing on the complete set. However, because different features participate in different restrictions, we may not see much of the impact of the individual features. Another option turn off all features, and start adding them back to the set one by one until the performance of the system does not improve. This can be done with a hill-climbing method similar to feature selection approaches in statistical learning [Pietra *et al.*,

1997a]. This method would allow us to quantify contributions of individual features, and determine the smallest feature set useful in parsing. We are planning this evaluation as part of our future work.

The goal of this thesis is to provide a semantic type representation useful in practical parsing applications, and in the following chapters we will demonstrate that selectional restrictions using features can be used effectively in a realistic parsing system. We have discussed earlier the limitations of the method with respect to the granularity of the restrictions. Another obvious limitation is the applicability to figurative language, such as metonymy and metaphor. We partially address this problem with the use of type coercion, described later in this thesis. A more complete treatment would be a constraint relaxation system similar to Rosé [2000] or treating selectional restrictions as presupposition failure as suggested by McCawley [1968]. The latter approach is the most comprehensive, relying on the system reasoners to reason about sentence entailments, and covering many cases which cannot be analyzed completely with our simpler system. However, it has a disadvantage in performance, because reasoning can only be used after parsing is complete. We believe that a combination method is possible, which would attempt to parse a sentence using the current restrictions, and apply the more expensive methods if regular parsing fails. Such an approach could use the information which features didn't match as a guide for reasoning, and can be a topic of future research.

3 A domain-independent ontology and lexicon for parsing natural language

In the previous chapter we described a domain-independent semantic feature system and the operations necessary to implement selectional restrictions on feature sets. The question we are going to ask in this chapter is how to build a parser and a lexicon that can be used for semantic interpretation and disambiguation in a practical dialogue system. Multiple choices are involved in this process. Perhaps the most important is what is the semantic representation that will be generated by the parser. It has a major impact on the lexicon, which needs to provide the information necessary to build the logical form. But assuming that we have established what information needs to be encoded in the lexicon and how the grammar can use it to translate parse trees into semantic representations, there's still the question of how to organize the lexicon and the associated ontology (which we assume is the repository of the semantic types) in the best way to support efficient parsing and fast lexical acquisition. In this chapter, we will be most concerned with the latter question.

We will start with briefly reviewing the logical form representations used in existing systems, and describing the logical form we use in our domain independent lexicon. We then review existing large-scale ontologies and lexicon as sources for syntactic and semantic information, and motivate the design of our domain-independent ontology for parsing. We show how feature structures we described in the previous chapter are integrated in our ontology and lexicon, and describe formally the algorithms for inheritance and feature inference in our hierarchical lexicon. Finally, we discuss the issues in our lexicon and ontology design, comparing it to the design of other existing systems.

3.1 Motivation and Background

When we discussed our motivation for using a domain-independent wide coverage parser, one of the major points was the portability it provides in permitting deep language analysis in multiple domains. Let us re-examine this in more detail, with the emphasis on the semantic representation generated by the parser.

As the language is being interpreted, it ultimately needs to be converted into an internal knowledge representation, which the system reasoning components will use for inference. However, the representations used in reasoning and planning (either first order logic or frame-based formalisms such as KM[Clark and Porter, 1999]) often make it difficult to express the information needed for discourse processing. For example, first order logic offers only universal and existential quantification, and various extensions have been proposed to express various natural language phenomena, including generalized quantifiers (*e.g.*, *most*, *few*), operators (*e.g.*, *very*, *almost*), tense and aspect, speech acts and other discourse related information.

One way to approach the problem is to develop a knowledge representation used for both language interpretation and reasoning, as is done, for example, in episodic logic (EL) which provides a rich representation for deep natural language understanding [Hwang and Schubert, 1993]. However, more research needs to be done to ensure usefulness of these representations in natural language understanding. There has not been sufficient system-building with EL to demonstrate its usefulness for dialogue systems, and at present we are not sure whether the more subtle resources of EL (which are crucial for narrative analysis) are necessary for the task-oriented systems we are focusing on. We therefore chose a different solution - an intermediate semantic representation which can be easily generated from language, and which can later be converted in a domain-specific representation for efficient reasoning, based on an existing QLF formalism (discussed in detail in the next section) which has been actively used in existing dialogue systems. Moreover, we wanted to make a contribution to an existing research effort, the TRIPS system, and take advantage of its existing interpretation management capabilities[Allen *et al.*, 2001].

Therefore, we chose a neo-Davidsonian QLF representation with semantic roles, closer to frame representations typically used in the dialogue systems, including the TRIPS system. The representation we discuss in the next section does not amount to a complete formal semantics which can be immediately used in interpretation, though large portions of it are directly interpretable. Rather, it is an intermediate representation which provides pieces necessary to constructing a full logical form for reasoning, which is done during the discourse interpretation

process.

In designing such intermediate representation we need to consider both the expressivity of its syntax and the semantic content that needs to be provided in the lexicon to generate it. There has been a good deal of work on logical form representations which are easy to obtain from syntax. We review it and describe the syntax of our domain-independent representation, in Section 3.1.1. The representations we discuss usually are built under the assumption that that an ontology as a source of semantic types will be provided independently. At the same time, large-scale ontologies and lexicons aim to provide the necessary semantic information (reviewed in Section 3.1.2), but they are not necessarily concerned how the information they provide can be used in parsing. In our research, we found that bridging the gap between these two strands of research, and designing an ontology and lexicon to generate a domain-independent representation suitable for discourse processing presents interesting challenges.

We do not fully address the question of what is the best possible representation for discourse processing here. Instead, we describe the syntax of our domain-independent logical form (which is similar to QLF representation [Alshawi *et al.*, 1991]) in the next section, and then concentrate on the information we need to supply in the lexicon and ontology to build it successfully. The assumption that we are making is that if we are using a domain-independent syntax in our intermediate semantic representation, we will need a domain-independent lexicon and ontology to supply the necessary semantic information. Since discourse related information is not necessarily represented in the domain-specific language, then all words which carry it need to have domain-independent semantics; for words related to task content, we need to ensure that we have sufficient information to build our domain-independent form for all words in the sentences. Domain-specific representations do not always align well with linguistic structure, and can therefore be difficult to build directly from language - we will revisit this in more detail in Chapter 4.

The issues that interest us in this chapter are choosing the ontology for natural language processing and implementing the lexicon which uses it. Our contributions will be the design of the ontology which integrates a frame-based ontology organization with feature structures used in selectional restrictions for parsing and disambiguation; the development of a domain-independent ontology geared for parsing, and a lexicon and grammar using it; and the resulting discussion of guidelines for the structure of concepts and semantic roles in a parsing ontology, which can offer insights for future development of large scale resources needed for parsing and semantic interpretation.

```
(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 (SET-OF LF::FOOD*orange))
(THE v2 LF::Vehicle*truck)
```

Figure 3.1: The LF representation of the sentence *load the oranges into the truck*.

3.1.1 Logical Form Representation

Devising a domain-independent semantic representation easily computable from syntax in a large-scale system is a non-trivial problem. General purpose semantic formalisms for use with large coverage syntactic grammars include, for example, Minimal Recursion Semantics [Copestake *et al.*, 1995], GLUE semantics [Dalrymple *et al.*, 1995] and Quasi-Logical Form (QLF) [Alshawi *et al.*, 1991]. We designed our logical form to be most similar to QLF, because it has been used in other practical dialogue systems, and in some cases it can be translated into commands and database queries without the need to resolve scope ambiguities [van Noord *et al.*, 1999]. We discuss how to convert our domain independent form into a domain-specific representation in Chapter 4.

The representation produced by our parser is a flattened and unscoped logical form using reified events [Davidson, 1967]. A logical form representation of *Load the oranges into the truck* is shown in Figure 3.1. It consists of expressions with syntax

$$\langle \langle \text{SPECIFIER} \rangle \langle \text{VARIABLE} \rangle \langle \text{TYPE} \rangle \langle \text{RESTRICTION} \rangle \rangle$$

where $\langle \text{SPECIFIER} \rangle$ is a generalized quantifier either coming from a noun phrase specifier, or 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 brevity, the representation shown here is a simplified version that omits some of the reference and discourse information and focuses on the semantic content of an utterance.

The detailed discussion of our logical form representation is beyond the scope of this thesis. In short, this representation is similar to QLF, including for each term an index (a variable name), a quantifier, and the restriction. Like QLF, our logical form representation is not yet a full logical representation because the quantifiers are unscoped, and there are unresolved de-

criptions. For noun phrases, determiners such as *a* and *the* serve as quantifiers over the variable, and the exact quantifier, and scope, will be determined during reference resolution and contextual processing. “F”, “SPEECHACT” and “IMPRO” have a special status as a combination of quantifier and utterance type, which during discourse processing will be interpreted in a domain-specific way. For example, SPEECHACT is likely to be resolved to an existential quantifier, and “IMPRO” is most likely to be represented as a skolem constant after reference resolution. “F” has several possible interpretations. When used with events, it may be interpreted as an existential quantifier over an event. However, if we are transforming adjectives, adverbials or modifiers such as *very happy*, it will be treated as syntax for forming expressions and discarded in the target language. The definitions were left intentionally underconstrained, and the exact interpretation is determined during discourse processing. We discuss in more detail the syntax of QLF related to modifiers and how it is obtained with our parser in Sections 3.3 and 3.4.3. Then we discuss in Chapter 4 how it can be connected to domain-specific syntax for reasoning.

Unlike traditional QLF representations, which typically use n-place predicates, we use named arguments (which we call **semantic roles**) in our representations, as it is done in neo-Davidsonian representations and description logic. Speaking informally, an expression
 (F e123 LF::Filling*load :Agent pro1 :Theme v1 :Goal v2)
 can be seen as an abbreviation of the following quasi-logical representation
 $\langle F e \rangle$ (and (LF::Filling*load e) (agent e pro1) (theme e v1) (goal e v2)),
 where $\langle F e \rangle$ is a generalized quantifier (to be resolved to an existential during contextual processing), with not yet determined scope (which will be the entire utterance for event variables).

There are many ways to convert our LF representations into fully qualified logical representations. QLF in Core language engine was converted into a logical form which is an extension of first-order logic [Alshawi, 1992], and it was also used in an OVIS2 dialogue system [van Noord *et al.*, 1999] as a semantic representation from which dialogue updates were constructed directly. Similarly, in our system we convert the logical form representation directly into the forms used by our planner and scheduler, which depending on the implementation are either a first-order logic form, or frame representations. We can do this easily is because so far we didn’t have to deal with language which contained many scope ambiguities. In the future, if the need arises, we plan to convert our logical form representations into fully quantified expressions as necessary.

A major representational decision we needed to make in developing our logical form was to use semantic roles instead of n-place predicates. Although there are advantages to using traditional predicate logic as a representation for reasoning, using semantic roles simplifies lexicon maintenance and allows us to better express certain syntactic alternations. It also offers future

opportunities for syntactic generalization, as many linguists suggested for thematic roles in syntax ([Dowty, 1991],[Jackendoff, 1990],[Kipper *et al.*, 2000]). The more detailed discussion of how roles are used in our system will be continued in Section 3.1.4, after we discuss in more detail the available lexicons and ontologies as sources of syntactic and semantic information necessary for parsing.

3.1.2 Choosing an ontology for natural language processing

In the introduction (Section 1.5) we discussed the information that needs to be available from a parsing lexicon. This includes syntactic features, a subcategorization frame, a semantic type, and syntax-semantics linking. Given our goals to support efficient parsing and fast lexicon acquisition, let us consider the specific issues involved in using one of the existing large-scale lexicons or ontologies as a source of semantic types.

- **The level of abstraction.** The semantic predicates used during interpretation must be specific enough to allow the system to draw reasonable inferences about the world. For example, using the same predicate MOVE to denote verbs *run*, *walk* and *drive* loses important distinctions between the meanings, such as speed and the vehicle involved. At the same time, we want the semantic predicates to be such that the system has a reasonable chance of selecting the correct sense during the interpretation process. This is a failing of some electronic dictionaries, discussed in more detail below, when the number of senses becomes so large that the extensive reasoning about context is necessary to select the correct sense.
- **The compositionality of meaning representations** In a domain-independent ontology, we would like the meanings of the complex phrases to be compositional, built from the meanings of their components. For example, consider a sentence *Submit a purchase order*. In a system that only knows about submitting purchase orders, this is an atomic action. Therefore, it can potentially be represented as a single concept in the system ontology, SUBMITPURCHASEORDER(*P*), where *p* is parameter which corresponds to the purchase order to submit. This representation may be the most efficient for domain reasoning, but if there are other things that can be submitted, such as proposals or application, this leads to a proliferation of concepts: SUBMITPROPOSAL(*P*), SUBMITAPPLICATION(*A*). This is not a desirable situation for parsing, because it results in additional ambiguity in constructions like *submit it*, which then become multiply ambiguous between different possible meanings of *submit*.

- **Efficiency.** For a dialogue system, the speed of interpretation is crucial for effective operation, and we would like to use as much semantic information as possible during parsing to speed up and improve disambiguation.
- **Syntax-semantics mappings.** In order to use an ontology in a parsing system, we need to be able to link the syntactic structures to corresponding ontological representations. This mapping either has to be available from a lexicon connected to the ontology, or it needs to be specified in our lexicon. The properties of the ontology, including the level of abstraction and compositionality, and also the arguments associated with each type, should be such that syntax-semantics mapping is relatively easy. For example, if an ontology requires collecting phrases like *from Pittsford* and *to Avon* into a single PATH frame, then special handling for path adverbials has to be implemented in the grammar, adding to the complexity of the system. We will show additional examples in Section 4.2 as part of the discussion why domain ontologies are often not suitable for general NLP tasks.

In the next section, we discuss the existing ontologies and lexicons, the information they provide for parsing, and motivate the design of the TRIPS language ontology.

3.1.3 Large scale ontologies and lexicons

Over the years, many large scale ontologies and lexicons were developed to aid in natural language processing. There are a number of machine-readable dictionaries in development (see for example [Sanfilippo *et al.*, 1998] for a comprehensive overview); we are interested only in the resources which provide syntactic and semantic information necessary for a parsing lexicon as described in the previous section.

The properties of the largest computational ontologies and lexicons with respect to the information necessary for parsing and semantic disambiguation are summarized in Table 3.1. The resources listed here are either large-scale ontologies which encode general knowledge about the world or computational lexicons, which encode mainly linguistic knowledge.

CyC [Lenat, 1995] is a large scale commercial knowledge base which contains assertions about common sense knowledge. The full version of CyC has restricted access, but a more limited version, called OpenCyc, was released as open source software. It contains about 6000 concepts and about 60000 assertions about them, including selectional restrictions in the form of argument type constraints. There is an associated lexicon and lexicon development language, but no relevant statistics is provided. A detailed assessment of CyC for purposes of natural

Name	Lexicon size	Ontology size	Subcat. Frames	Subtype relationships	Selectional restrictions
WordNet	146350	111223	no	yes	no
OpenCYC	?	6,000	no	yes	yes
FrameNet	7500	450	no	no	no
COMLEX	35000	n/a	yes	no	no
VerbNet ¹	??	191	yes	yes	yes

Table 3.1: Large scale lexical databases.

language understanding was conducted by Mahesh *et al.* [Mahesh *et al.*, 1996]. Their conclusion was that CyC is capable of efficiently retrieving facts and making fast inferences, but encodes information in a way unsuitable for use in a NLP system doing large-scale interpretation. One of the biggest problems noted with CYC representation is presence of compound concepts. For example, *take* includes senses #TakingABath and #TakingAShower. We discussed earlier in Section 3.1.2 why compositionality is important in NLP ontologies, thus we also found CyC not suitable for lexicon development in our system.

KBMT [Nirenburg *et al.*, 1992a] is one of the systems developed by Nirenburg *et al.* as part of the ongoing knowledge based machine translation and ontological semantics projects. It consists of a frame-based ontology and a lexicon based on the CLFG grammar formalism. Several versions of the ontology with medium level of abstraction were implemented for use in different domains. The goal of the KBMT project is to provide the knowledge necessary for deep processing in large scale text-processing and machine translation systems. It attempts to cover all aspects of language understanding, including reference resolution and intention recognition. For that purpose, it provides a rich frame-based ontological language with complex syntax sufficient to express text meaning in large-scale applications. However, in our system we found that it is necessary to specialize our semantic model for specific domains to obtain complex representations with sufficient accuracy. Nirenburg makes a similar observation [Nirenburg *et al.*, 1992b], and proposes a unified architecture which integrates domain models with a large-scale general purpose ontology for NLP.

Our approach is similar in that we differentiate between domain-independent and domain-specific ontologies, and develop a formal mechanism to integrate domain-independent and domain-specific knowledge, but we made a different decision about ontology organization. In our medium scale domains most knowledge is localized in specialist reasoners which choose their domain-specific representations for maximum efficiency. At the same time, there is linguistic knowledge

which can be easily shared between the domains, which includes the syntactic knowledge and semantic knowledge needed for initial semantic processing. Therefore, we chose to limit the information in our ontology to only the semantic properties which encode important lexico-syntactic generalizations, and can potentially be (semi-automatically) derived from corpora and other large-scale lexicons. Our goal is then to develop a uniform representation which hides syntactic variations and can be used as a starting point in applications which require deeper semantic understanding.

Both CyC and KBMT aim to represent world knowledge in their ontologies, though with different emphasis. CyC is oriented primarily for the use of knowledge for reasoning, while KBMT intends to encode the knowledge primarily necessary for deep text interpretation. Because of the wide scope of these projects, they place less emphasis on the best encoding of lexical semantic information, which is crucial in parsing and interpretation. This information may be easier to obtain from computational lexicons which focus primarily on encoding linguistic and lexical semantic information.

WordNet[Miller, 1995] is a large scale semantic net that contains words, sense definitions and a set of relationships between senses, including subtype relationships. Senses are encoded as sunsets, groups of synonymous words, *e.g.*, {*walk, travel, go, move, locomote*}. The database has a very wide coverage, but does not provide information about subcategorization frames or selectional restrictions.

From the point of view of parsing lexicon design, WordNet offers the widest coverage, but the least convenient representation. It introduces sense distinctions which are difficult or impossible to determine without the use of context. For example, a word *take* has 42 separate senses in WordNet, including separate senses for *take into one's possession*; "*We are taking an orphan from Romania*"; "*I'll take three salmon steaks*" and *buy, select*; "*I'll take a pound of that sausage*". It is unclear how these senses can be disambiguated in a practical system without extensive use of reasoning, which makes it difficult to use in knowledge-based applications.

FrameNet [Johnson and Fillmore, 2000] is a project to encode verb and noun meanings as frames. The frame names and frame elements are selected based on examining a large number of examples from a corpus, and the examples are stored with frame definitions. FrameNet does not record syntactic patterns associated with frame elements, though Gildea and Jurafsky [2002] have done work on statistical recognition of frame elements based on large corpora. We found the frame descriptions useful as a basis for our own ontology, supplementing it with a manually constructed lexicon which provides mappings between syntactic and semantic arguments.

COMLEX [Macleod *et al.*, 1994] is a purely syntactic lexicon. It does not contain semantic information, but it contains the subcategorization frames and syntactic features not found in most other lexicons, and can be useful as a source of syntactic information. We did not use the COMLEX directly in our lexicon, however, Swier [2002] investigated the use of COMLEX together with WordNet to extend our lexicon coverage of adjectives.

VerbNet[Kipper *et al.*, 2000] is a verb-only lexicon which provides all the information necessary for parsing and generation, using Levin’s classes [Levin, 1993] as a basis of defining verb syntax and semantics. For every verb, it provides subcategorization frames, a semantic class and correspondence between the subcategorization frame and verb semantics, as well as basic selectional restrictions on verb arguments. It is the closest approach to ours in terms of the information provided for verbs, and we are working on using the information from VerbNet to extend our lexicon verb coverage. Our feature set is more extensive, and we have discussed the differences in the previous chapter. In this chapter we will also compare the differences between verb semantics and role representations in our lexicons.

3.1.4 FrameNet and TRIPS LF Ontology

As can be seen from Section 3.1.3, none of the existing lexicons and ontologies provide all the information necessary in a parsing lexicon. Therefore, for our lexicon, we designed a domain-independent ontology geared for natural language parsing (which we will call a **LF Ontology**) and a syntactic lexicon linked to the LF ontology to provide syntactic features and syntax-semantics mappings.

The basic representation in our ontology is frames, which are concepts with named arguments, as in FrameNet database. However, our LF ontology has wider coverage for nouns, adjectives and adverbs not completely covered by FrameNet. In addition, our LF ontology has deeper hierarchical structure than FrameNet, and utilizes a smaller set for role names in order to enable easy linking between syntax and semantics.

FrameNet consists of semantic frames, which collect together verbs and nouns that represent situations sharing the same set of arguments, called Frame Elements. It provides a level of granularity most suitable for our project, and the role based structure we wanted to use in our representations. However, FrameNet does not attempt to provide mappings between the frame representations and syntactic structure, and the differences described in this chapter are motivated by the need to provide those mappings in a lexicon which is easy to construct and maintain.

Although the current FrameNet ontology is mostly flat, it contains many frames with identical argument structures. For example, *Suasion1*, *Suasion2* and *Suasion3* denote a group of verbs such as *encourage*, *convince*, *induce*. They have the same set of roles, and the difference in meaning comes in whether the addressee is forming intention to act. From the point of view of argument structure and selectional restrictions these frames are identical, so we can collect them under a general parent, and use the same set of selectional restrictions.

The hierarchy in our ontology is partially based on an early implementation of FrameNet which had a notion of a domain, which was be a global parent to a set of frames, as well as the notion of more specific role names implementing more general roles in the domain frames. At that stage, the FrameNet database was not complete in many areas needed for processing in our system, for example, frames related to problem solving and planning, but we used the domains where available as parent nodes in our hierarchical structure.

Hierarchical structure provides a level of generalization in the ontology which makes it easier to include and maintain selectional restrictions. Another difference between our frame representations and FrameNet is the role set. FrameNet utilizes situation roles, that is, roles that actually evoke the situation in question. So in a *driving* situation, there will be a *driver*, whereas in the *communication* situation there will be a *communicator*. However, these roles may be seen as instances of the general *agent* role, which is the intentional being doing the action. Having a limited number of role names simplifies the inheritance in the LF ontology by giving us an option to define a general restriction (e.g. agents are intentional beings) on the top of the hierarchy tree.

For purposes of mapping between syntax and semantics, a smaller number of role names facilitates the definition of syntax-semantics mappings, because there is opportunity for generalization. For example, many motion verbs will use exactly the same set of mappings, and not having the distinctions between “driver” versus “self-mover” simplifies the definitions and makes it easier to add new verbs by example. More importantly, many theories exist which make syntactic generalizations based on semantic classes (see for example Levin [1993], Jackendoff [1990]). While we do not use our role names for syntactic generalizations yet, we intentionally designed our ontology to make it possible in the future, and are planning to include support for VerbNet classes (which are derived from Levin’s classes) as part of our future work.

In rest of this chapter, we discuss in detail how we use the LF ontology and the domain-independent lexicon to build our logical form, and how selectional restrictions are integrated in the LF ontology and used during parsing and disambiguation; and the guidelines for developing a domain-independent ontology for parsing, which include the

3.2 Domain-independent ontology

In the previous section we motivated the design of our LF ontology as a structure similar to FrameNet. In Chapter 2 we also described semantic feature lists as a basis for selectional restrictions. Our ontology consists of a hierarchy of **LF types** which integrate frame-based types with selectional restrictions based on feature lists.

The LF types are organized in a single inheritance hierarchy ordered primarily by their argument structure, and to include semantic features in the ontology we associate every LF type with a corresponding list of semantic features. For example, Figure 3.2 shows a set of sample definitions from our LF ontology. In this figure, LF::Physical-object is declared as a parent of LF::Substance, which in turn is the parent of LF::Drug. All of these are physical objects which do not have any semantic roles associated with them, so the definitions only specify the associated feature vectors. Each child inherits the features from the parent and adds its own set of features, so the final set of features associated with all entities of type LF::Drug is (*Phys-obj (Form Substance) (Mobility Non-self-moving) (Origin Artifact)*).

```
(define-type LF::Phys-object
  :parent LF::Any-sem
  :sem (Phys-obj))
(define-type LF::Substance
  :parent LF::Phys-object
  :sem (Phys-obj (Form Substance) (Mobility Non-self-moving)))
(define-type LF::Drug
  :parent LF::Substance
  :sem (Phys-obj (Origin Artifact)))
```

Figure 3.2: LF type definitions for physical objects in the LF ontology

In this example, the definition of LF::Drug adds a new feature (*Origin*) to the list of features defined on the parent (*Form, Mobility*). If the same feature is defined on both a parent and a child, the child can only specialize, but not override the parent’s value. For example, if the parent specifies (*Origin Living*), then the child can specify (*Origin Human*), but it cannot change this value to, for example, (*Origin Natural-non-living*). To handle exceptions, we have a default mechanism discussed in Section 3.4.1.

The types in our example were simple concepts without associated argument structure. In

general, for every LF type we declare a set of semantic arguments with selectional restrictions on them expressed in terms of feature sets. The semantic arguments are defined so that they represent different syntactic alternations within the same logical form. To illustrate, consider the verb *load* in the sense *to fill the container*. The LF type definition for *LF::Filling* is shown in Figure 3.3. Intuitively, it 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 the :agent role must be an intentional being.

```
(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 3.3: 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.

The definition of *LF::Filling* is consistent with a larger set of verbs: *fill*, *load*, *pack*, *stuff*. In our LF representation, they will all be represented as instances of *LF::Filling*, with associated selectional restrictions provided by arguments. It also covers two different alternations: *Load the truck with oranges* and *Load the oranges into the truck*. In both cases, *oranges* will fill the :theme slot, and *truck* will correspond to the :goal slot in the LF representation.

In the next section, we show how our generic lexicon is linked to the LF ontology to generate our logical form.

3.3 The Generic Lexicon

Every word in the lexicon is defined as a list of patterns which correspond to different word meanings combined with subcategorization frames. For every sense, we specify syntactic features (such as agreement, morphology, etc.), LF type, and the subcategorization frame and syntax-semantics mappings (which we call **syntax-semantics templates**). As an example in this section, we will use the *load* verb, which, as we discussed in the previous section, can participate in at least two alternate syntactic configurations involving LF::Filling sense: *Load the truck with oranges* and *Load the oranges into the truck*. The :Theme argument of LF::filling is realized as a direct object of the verb in the first case, and as a prepositional complement in the second. The lexicon definition which encodes this fact is shown in Figure 3.4. It contains three parts: the lexical definition in part (a), and two templates referenced in the definition in parts (b) or (c). We discuss the syntax of these parts in more detail below.

- ```
(a) (load (wordfeats (morph (:forms (-vb))))
 (senses
 ((LF-Parent LF::Filling) (TEMPL AGENT-THEME-GOAL-TEMPL)
 (Example "Load the oranges into the truck"))
 ((LF-parent LF::Filling) (TEMPL AGENT-GOAL-THEME-TEMPL)
 (Example "Load the truck with oranges"))
))
(b) (AGENT-THEME-GOAL-TEMPL
 (SUBJ (NP) Agent)
 (DOBJ (NP) Theme)
 (COMP (PP (ptype into)) Goal optional)
(c) (AGENT-GOAL-THEME-TEMPL
 (SUBJ (NP) Agent)
 (DOBJ (NP) Goal)
 (COMP (PP (ptype with)) Theme)
```

Figure 3.4: Defining words in the lexicon (a)Lexicon definitions for the verb *load* in the LF::Filling sense; (b) The template used to define the syntactic pattern for *load the oranges into the truck* (c) The template used to define the syntactic pattern for *load the truck with oranges*

In Figure 3.4(a), the vocabulary definition of *load* contains two entries. Each entry contains a sense, denoted by LF-PARENT (LF::Filling), and a syntax-semantics template definition denoted by TEMPL. Both LF-PARENT and TEMPL are required, because they define the two basic components of the definition: the LF sense and the subcategorization frame. In addition, we can specify an example, a preference and a special argument called LF-FORM. **Preference** is the syntactic preference that allows us to direct the search process. The default preference is 1, but we can set a preference to a lower number for rare senses or syntactic patterns, so that the more frequent ones can be tried first and the parse can be found faster for them. LF-FORM is an orthographic form of the word required to define its full type.

We said that LF-PARENT determines the sense of the word. This is true, however, using LF-PARENT alone as the semantic type of a word leads to the loss of information. Consider, for example, LF::Filling. Verbs *cram*, *fill*, *pack* and *load* are all linked to it. The LF ontology does not make any finer meaning distinctions, because there is no way to subdivide these verbs with respect to argument structure or selectional restrictions in a domain-independent fashion. However, there are shades of meaning in these words that would be lost if we translate them all into the same concept. To retain these deeper meaning distinctions, word senses are treated as leaves of the semantic hierarchy, and the complete LF type of a word is written as LF-parent\*LF-form, where the LF-parent is the type defined in the LF ontology (for example, LF::Filling for *load*), and LF-form is the canonical form associated with the word (for example, LF::Filling\*load for all morphological variations of a verb *load*).

In most cases, LF-Form is taken to be the base form of the word, but for words with identical meanings we manually define LF-FORM on all of them to be the same to cut down on the number of unnecessary distinctions during interpretation. For example *favor* and *favour* will have the same (LF-FORM **favor**). A relevant question here is what it means for the word to have identical meanings. One possibility is to assume that all entries in a WordNet synset to be identical, and assign them the same LF-Form. However, at the level of distinctions the LF ontology makes, very few words in the language are true synonyms, so there is a risk of losing information with such settings. Our current policy is to set LF form for the spelling variants of the same word, and for spoken variations such as *chopper* for *helicopter*.

The word is linked to its meaning representation via syntax-semantics templates, which are data structures which combine the subcategorization frame information with the information about syntax-semantics linking. A template consists of a set of mappings containing 3 main pieces of information for each argument: the syntactic role of the argument, similar to the roles used in LFG grammars; the subcategorized phrase type, and the role to which the argument

will be mapped in the LF representation.

For example, Figure 3.4(b) shows the syntactic template for *load the oranges into the truck*. It states that syntactically, the subcategorization frame consists of a subject (SUBJ), which is a noun phrase; a direct object (DOBJ), which is also a noun phrase, and a third argument (COMP) which is a preposition phrase with preposition *into*.<sup>2</sup> Furthermore, the filler of the subject slot corresponds to the :Agent argument in the logical form, the filler of the object slot corresponds to the :Theme argument, and the third complement corresponds to the :Goal semantic argument. The third complement is declared as optional, which means that it may be missing from the syntactic realization, as in *Now, load the truck*. A similar definition corresponding to the pattern *load the truck with oranges* is shown in Figure 3.4(c). All possible syntactic argument types are shown in Table 3.2.

|          |                                                                                                                                                                                                      |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SUBJ     | The subject of the verb                                                                                                                                                                              |
| DOBJ     | The object of the verb                                                                                                                                                                               |
| COMP     | The second object of the verb                                                                                                                                                                        |
| IOBJ     | The indirect object which undergoes dative alternation, as in <i>give him a book / give a book to him</i>                                                                                            |
| SUBCAT   | For nouns, prepositional adverbials and some adjectives, the inner argument of the phrase, e.g., <i>John in a brother of John, afraid of John, next to John</i> .                                    |
| ARGUMENT | for adjectives and adverbs, the argument to which the word applies e.g., the verb phrase in <i>go quickly</i> , noun phrase in <i>a happy man</i> , or the adjective phrase in <i>very quickly</i> . |

Table 3.2: The types of syntactic arguments in the templates

A direct object or a complement can be specified as optional in the mapping, allowing it to be absent from the syntactic realization. We can also specify a special NOROLE keyword for syntactic constituents that are not mapped to any roles explicitly. This happens, for example, in cleft constructions, e.g., *it takes two hours to complete the task*, where *it* is an expletive pronoun which does not carry a semantic load.

The syntax-semantics templates are combined with vocabulary definitions to form the entries passed to the system parser. We first present an intuitive description of the system, with the

---

<sup>2</sup>In a more realistic example, this would include a list of acceptable prepositions, for example, *into, onto, on*. We currently do not support any short-hand way to specify preposition classes for such alternations, we are planning it for our future work.

(a)  
 (LF LF::Filling\*load)  
 (SUBJ (NP (SEM (Phys-obj (Intentional +)))) (SUBJ-MAP AGENT)  
 (DOBJ (NP (SEM (Phys-obj (Mobility Movable)))) (DOBJ-MAP THEME)  
 (COMP (PP (ptype into) (SEM (Phys-obj (Container +)))) (COMP-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)  
 (COMP (PP (ptype with) (SEM (Phys-obj (Mobility Movable)))) (COMP-MAP THEME)

Figure 3.5: The definitions for verb *load* passed to the parser created from the lexicon definitions in Figures 3.4 and 3.3. (a) *load the oranges into the truck* (b) *load the truck with oranges*.

formal definitions for feature inheritance developed in Section 3.5. Figure 3.5 shows the two entries created from the definitions in Figure 3.4. In this figure, SUBJ denotes the restriction on the subject of the verb, DOBJ the restriction on the direct object, and COMP the restriction on the prepositional complement. SEM denotes the selectional restrictions propagated from the definition in Figure 3.3 based on role correspondences. These restrictions are used as follows: Given the restriction on the :Theme argument in Figure 3.3, 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 COMP-MAP denote the semantic arguments into which these syntactic arguments will be mapped, as defined in the template.

When building the logical form for LF::Filling, the parser checks the value of SUBJ-MAP. Given that it is :Agent, it adds the :Agent slot to the logical form, filling it with the variable corresponding to the subject of the sentence. Similarly, it decides on the other slots based on the mappings for direct object and COMP, and thus builds the complete logical form for an utterance.

### 3.4 Additional ontology and lexicon design issues

In the previous sections, we described the basic structure of a LF ontology and lexicon used together to generate domain-independent logical form for utterances. In this section, we explore these issues in further detail, concentrating on three main areas: feature defaults, used to make

```

(define-type LF::Manufactured-Object
 :sem (Phys-obj (:required (Origin Artifact)) (:default (Form Solid-object))))
(define-type LF::Vehicle
 :sem (Phys-obj (Mobility Self-moving)))
(define-type LF::Air-vehicle
 :sem (Phys-obj (Form Enclosure)))

```

Figure 3.6: The LF definition for LF::Manufactured-object, LF::Vehicle and LF::Air-vehicle.

lexicon writing simpler; issues in building logical form for complex sentences with adverbial modifiers; and a more detailed discussion on how selectional restrictions are implemented in the grammar.

### 3.4.1 Feature Defaults

Our LF hierarchy is monotonic with respect to feature assignments: a child can specialize the feature values specified by the parent, but not override them. This is a good property to support reasoning, because we can make conclusions about the parent and they will apply to all children. Sometimes, however, most but not all children of a given parent have a given feature set. To account for such cases, we introduced default feature assignments in our system.

For example, figure 3.6 shows a definition for LF::Manufactured-Object, some of the descendants of which are *vehicle* and *furniture*. By definition, all manufactured objects are artifacts. Most of them are also solid objects, though it is possible that manufactured things like *medicine*, if they are classified under this LF type, will have a different form. Thus (*Form Object*) is specified as a default value, which will be used by most, but possibly not all, descendants.

Using the default values allows us to greatly simplify the lexicon writing, because all features need not be repeated for every LF type. In addition to the defaults associated with the LF types, we provide a top level defaults for every feature value depending on the set type. For example, most of physical objects are not intentional. Therefore, the type declaration for the list of type *Phys-obj* specifies that all words associated with this feature set are (*Intentional -*) by default. These defaults are applied last, and provide a “base case” for all feature assignments.

The inheritance for default feature values is non-monotonic: if a child specifies a different default value than the parent, it will override the default value from the parent. For example, if in our ontology we had a type LF::furniture, it can specify *:default (Form Enclosure)*, and

all its children will use this as a default value of form, and not (*Form Solid-object*) specified on LF::Manufactured-object.

Given our default mechanisms, for a word defined in the lexicon the associated features can come from 3 distinct sources: the features specified on word or its LF parent, the default features inherited from the LF parent or from feature type, and the feature values inferred with the use of feature dependency rules described in Section 2.2.3. The algorithm for determining the complete feature set is as follows.

1. Start with a set of required features defined for the word.
2. Apply the inference rules for the features from the required set. The features inferred by those rules are also required.
3. For every feature for which the value is not yet defined, fill in the value from the default set associated with the LF parent. If the default is not defined for the parent, use the default for the feature list type.
4. Apply inference rules to the set extended with defaults. If at this stage a value inferred clashes with something already set, an error is reported, because this means that one of the default values results in an inference that conflicts with the required features.

For example, for our definitions in Figure 3.6, the features on the word *vehicle* will be determined as follows.

1. First, we look for the inference rules on the required features specified in the set and inherited from the parent, i.e., *Phys-obj (Origin Artifact) (Mobility Self-moving)*. There are no inferences specified for these features, so we proceed to the next step.
2. We examine the default feature set. The default feature *Form* is not filled in the required set, so we add (*Form Solid-object*) to the set. The resulting features are *Phys-obj (Origin Artifact) (Mobility Self-moving) (Form Solid-object)*.
3. Since not all possible feature values are filled in, we look up the default feature values for *Phys-obj* type, and fill in *(Spatial-abstraction (OR Spatial-point Spatial-region)) (Intentional -) (Information -)*. The resulting set is *(Phys-obj (Origin Artifact) (Mobility Self-moving) (Intentional -) (Information -) (Form Solid-object) (Spatial-abstraction (OR Spatial-point Spatial-region)))*

4. We attempt to apply inference rules again. There are no inferences associated with the added features, so we have the final feature set.

A similar procedure is done for *helicopter*. However, LF::Air-vehicle specifies (*Form Enclosure*), therefore, on step 3 the *Solid-object* value is not added, as it is the default value. The resulting feature set for LF::Air-Vehicle is

(*Phys-obj (Origin Artifact) (Mobility Self-moving) (Intentional -) (Form Enclosure)*  
*(Information -) (Spatial-abstraction (OR Spatial-point Spatial-region))* )

To illustrate a feature inference process, consider a word *person*. The definition specified in the lexicon has the feature set *Phys-obj (Origin Human)*. When the inference rules from Figure 2.5 are applied on step 1, (*Intentional +*) is added to the required set. This prevents addition of (*Intentional -*), as it was done on step 3 for both *vehicle* and *helicopter*, because the default is superseded by the required feature inferred with dependency rules.

The formal definition of default inheritance and inference is provided in Section 3.5.

### 3.4.2 Role implementations

Our ontology has support for thematic arguments implementing other arguments, to deal with cases where more than one thematic role can be assigned to the same syntactic argument. For example, consider the verbs of causation. In the most general implementation, a cause of an event can be either an intentional agent or a different event: *John broke the window, A storm broke the window*. It may be useful and more natural to call *John* an agent rather than just a cause. This is supported through role implementations. We can declare an agent role in the LF description, specifying that it implements a more general cause role. The agent role will then pick up the general restriction from the cause role, and possibly add it's own (intentionality) restriction. Then the lexical entries can define syntax-semantics mappings using the agent role, and during the parser process the selectional restrictions will choose between the assigning a cause or an agent role to a phrase.

This mechanism is similar to the role implementations in FrameNet, and was intended to support a similar extension. For example, in the earlier version of FrameNet, it was possible to say that a HEAR frame inherits from COMMUNICATION frame, and the Communicator frame element of COMMUNICATION corresponds to SPEAKER frame element of HEAR. We found this mechanism useful especially in defining motion verbs, where for verbs like *go*, the more general :goal argument can be described as implemented by the :to-loc role. However,

the current support is only at the lexicon level, that is, for lexicon construction the system will correctly take the restrictions from :goal and merge them in with the more specific :to-loc restrictions when making a lexical entry. Interpretation components do not make use of the information, and the full integration is planned as part of future work.

### 3.4.3 Adverbial modifiers

In the previous section we described how a parser builds the logical form of a verb phrase from a lexicon definition. The process is very straightforward for arguments which are part of a subcategorization frame of the verb, but some additional complications arise from dealing with adverbials and prepositional phrase modifiers. Consider a sentence *Send the truck from Avon to Bath*. In our grammar, we treat *from Avon* and *to Bath* as optional adverbial modifiers on *send*. The lexical entry for *send* specifies that it takes two syntactic arguments, a subject which is mapped to the :Agent slot, and an object mapped into :Theme slot. Thus, for a sentence *send a truck*, a logical form shown in Figure 3.7(a) will be generated.

(a)  
(F v123 (:\* LF::Send send) :agent y1 :theme t2)  
(IMPRO y1 LF::Person :CONTEXT-REL \*YOU\*)  
(A t2 (:\* LF::Truck truck))

(b)  
(F v123 (:\* LF::Send send) :agent y1 :theme t2 :from-loc a1 :to-loc b1)  
(IMPRO y1 LF::Person :CONTEXT-REL \*YOU\*)  
(The a1 (LF::Political-region city) :name-of AVON)  
(The b1 (LF::Political-region city) :name-of BATH)

Figure 3.7: (a) A logical form for *Send a truck* (b) a simple possible LF for *Send a truck from Avon to Bath*

Now, given that the LF definition for LF::Send has a :Goal argument, our first treatment of adverbial modifiers was to simply add “:goal b1”, where **b1** is a variable referring to the noun phrase *Bath*, resulting in the LF form in Figure 3.7(b). Unfortunately, this solution does not work for a more complex case, *Send a truck from Avon straight to Bath*, where *straight* modifies the adverbial *to Bath*. In this case, the modifier has a more complex structure, and we cannot

ignore the contribution of *straight* to the meaning of the utterance by treating *Bath* as a simple filler of a :goal slot. To address this issue, we decided to adopt a more complex representation of adverbial modifiers. The LF representations of *Send a truck from Avon to Bath* and *Send a truck from Avon straight to Bath* are shown in Figure 3.8, and we explain how they were obtained in detail below.

We are adopting an approach similar to Hobbs’s ontological promiscuity [Hobbs, 1985], where every possible modifier is defined as a predicate in our LF ontology, with a set of associated semantic arguments. For predicates corresponding to adverbials, we adopted a uniform scheme where :of refers to the entity the predicate modifies, and :val refers to the “value” of the predicate, if any.<sup>3</sup> For example, the LF predicate LF::to-loc is a concept that denotes a relation between two arguments: the action that has a trajectory denoted by the :of argument, and the destination of that trajectory, denoted by the :val argument. The LF predicate LF::direction has a single argument :of, which is either the action or the trajectory the direction of which is specified by the predicate. This representation of adverbials is particularly useful because it allows us to declare selectional restrictions on their arguments in the LF ontology.

As can be seen in Figure 3.8, we represent the adverbial modifier *to Avon* as two terms in the LF: one representing *Avon*, and another a complex term representing the *to* relation, with the corresponding semantic roles. The verb *send* then indicates that it has an external adverbial modifier by putting variable  $\tau_{11}$  on the special :MODS list in the logical form expression for the verb phrase. This assumes that the interpretation manager using this form knows that the forms in Figures 3.7(b) and 3.8(a) are essentially equivalent. We discuss how this can be accomplished in the next chapter as part of the discussion how the logical form produced by the parser can be transformed into syntax directly used by domain reasoners.

In this representation, obtaining a logical form for *Send a truck from Avon straight to Bath* is straightforward. Since *straight* is an adverb which maps to the LF type LF::Direction, we can simply add its own term to the LF, and place an additional variable on the :mods list for *to*, thus creating a uniform representation for adverbials.

Since adjectives can also have additional modifiers, as in *a very long book* we treat them similarly to adverbials, representing each adjective as a predicate on the MODS list of the NP

---

<sup>3</sup>The scheme is the convention within the LF ontology. Nothing in the grammar depends on this naming, and there are exceptions, for example, one of the senses of *with* is mapped to LF::Have-property, similar to the verb *have*, and correspondingly has :theme and :co-theme arguments. The grammar semantic interpretation rules support an arbitrary correspondence between the syntactic arguments of the adverbial and the semantic arguments of the predicate using the same scheme as employed for the subcategorized arguments of the verbs.

- (a) (F v123 (:\* LF::Send send) :agent y1 :theme t2 :MODS (t11 fl1) )  
 (IMPRO y1 LF::Person :CONTEXT-REL \*YOU\*)  
 (A t2 (:\* LF::Truck truck))  
 (F t11 (:\* LF::To-loc to) :of v123 :val a1)  
 (The a1 (LF::Political-region city) :name-of AVON)  
 (F fl1 (:\* LF::From-loc from) :of v123 :val b1)  
 (The a1 (LF::Political-region city) :name-of BATH)
- (b) (F v123 (:\* LF::Send send) :agent y1 :theme t2 :MODS (t11 fl1) )  
 (IMPRO y1 LF::Person :CONTEXT-REL \*YOU\*)  
 (A t2 (:\* LF::Truck truck))  
 (F t11 (:\* LF::To-loc to) :of v123 :val a1 :mods (s2) )  
 (The a1 (LF::Political-region city) :name-of AVON)  
 (F s2 (:\* LF::Direction straight) :of t11) (F fl1 (:\* LF::From-loc from) :of v123 :val b1)  
 (The a1 (LF::Political-region city) :name-of BATH)

Figure 3.8: Logical forms for (a) *Send a truck from Avon to Bath* (b) *Send a truck from Avon straight to Bath*.

logical form. Thus we have a uniform representation for all entities in the LF which can have attached modifiers: the modifier is represented as its own term, and linked to the entity it modifies via the `:mods` list. In the next section, we describe how this system is integrated with expressing and checking selectional restrictions during parsing.

### 3.4.4 Selectional restrictions in the TRIPS lexicon

The selectional restrictions in the lexicon govern which constituents can be combined into phrases. In our system, we identified a set of syntactic contexts where selectional restrictions can be useful for parsing. These are

- Restrictions placed by verbs on arguments and adjuncts. This is the most common case where selectional restrictions can be used. For example *\*ideas sleep* is clearly a bad sentence because of the restrictions on *sleep*. Additionally, verbs place restrictions on the prepositional phrases and adverbials which modify them, for example, we cannot say *I broke a window with a feather*
- Restrictions placed by relational nouns on their subcategorized arguments. This is especially important because the *of* argument of relational nouns can have different meanings: *the speed of the computer* is the “speed” function applied to a computer, while *the speed of 1.5MHz* is the actual value of this function. Therefore, we can say *The speed of a computer is 1.5MHz*, or *This computer has the speed of 1.5MHz*, but it is not possible to say *\*The speed of 1.5MHz is 1.5MHz*. Under the circumstances, selectional restrictions are most important to determine which argument is filled by the *of* phrase.
- Restrictions placed by adverbials on the arguments which they can modify: for example, *toward* can only apply to actions which are (*Trajectory +*)
- Restrictions placed by PP-adverbials both on their internal subcats (for example, *to* needs a location in the NP position, as in *to Avon*)
- Restrictions placed by adjectives both on the nouns they can modify (*red* can only modify physical objects), and, if they have optional PP arguments, on those arguments (*addicted to* requires either a substance or possibly an activity as an argument of *to*)

We implement selectional restrictions in all those contexts. We placed the most emphasis on implementing verb selectional restrictions, because verb senses is one of the main sources of

ambiguity in our system (see Section 3.6 for the discussion). Implementing selectional restrictions on subcategorized arguments is straightforward: as a verb phrase is being formed, the restrictions declared in the verb LF arguments can be checked against the fillers. In practical terms, when the mappings between syntactic and semantic arguments are built as described in Section 3.3, the selectional restrictions from the LF arguments are propagated into the definitions of syntactic arguments.

For example, LF::Filling declares that the :Agent role must be filled with an intentional physical object (*Phys-obj (Intentional +)*), and a syntax template maps the SUBJ argument into :Agent. During the lexical entry compilation, the restriction from :Agent of LF::Filling will be added to the SUBJ feature of *load* in the form (*sem (Phys-obj (Intentional +))*), and the grammar rule for sentences in our system is (simplified)

```
S -> NP[(sem ?sem) (agr ?a)] VP[(agr ?a) (subj (% NP (sem ?sem)))]
```

This in effect enforces the selectional restriction by requiring that the `sem` feature of the subject noun phrase unifies with the restriction on the subject of the verb phrase.

In order to influence optional adjuncts, typically coming from PP adverbials, we check the restrictions when the adjuncts are added to the verb phrase. This is implemented through the use of a special ROLES feature in the syntactic entry, which is a head feature propagated up the constituent. Consider, for example, parsing and interpretation of *move it with a stick*. There are at least two possible interpretations of this phrase: one in which *with* signifies an instrument, and another in which *with* signifies a general accompaniment relation. In most cases, it is not sufficient to know just the type of the noun phrase object to select the right meaning of *with*. When *move it with a stick* is parsed, *move it* is analyzed as a verb phrase, and *with a stick* as an adverbial. One of the LF::Move declares an optional :Instrument argument which has to be a movable object. When the parser attempts to attach *with a stick* to *move it*, it checks the list of optional roles in the ROLES feature and determines that *stick* can fill the :Instrument role. Since *a stick* is a movable object, the selectional restriction is satisfied, and the attachment is allowed. The :Accompaniment interpretation of *with* is possible, but it will be dispreferred, coming out as a 2nd or 3rd possible parse. In contrast, for *move it with a smile*, since *smile* is an abstract object and does not satisfy the instrument restrictions, the :Accompaniment meaning of *with* will be preferred, because in our lexicon it is a very general relation, which can be satisfied with most physical or abstract objects.

Nouns are handled similarly to verbs. If they have subcategorized complements, they get the restrictions propagated from the LF form via template. If they are modified by adver-

bial modifiers, the ROLES slot is used to disambiguate the meaning of the modifier whenever possible.

In our grammar, adjectives and adverbs can have two syntactic arguments: ARGUMENT and SUBCAT. ARGUMENT is always required, and it is the noun the adjective modifies, and VP or NP modified by the adverbial. SUBCAT is optional, and it refers to the “inner” arguments, necessary to complete an adjective phrase or an adverbial before it can apply to the ARGUMENT. For example, the word *to* by itself cannot apply to anything and form a coherent phrase. It requires an NP subcat to form an adverbial *to Pittsford* which can now modify verb phrases. SUBCAT is optional for adjectives in most cases: both *afraid* and *afraid of dogs* are valid adjective phrases. However, ARGUMENT is required - *afraid* must appear either as a modifier on a noun, or in a predicative position where it in effect modifies the NP subject. As we mentioned above, adverbials and adjectives place restrictions both on their ARGUMENT and their SUBCAT. This is handled straightforwardly via templates that map ARGUMENT and SUBCAT to semantic role names in the LF ontology (which are frequently, but not always :of and :val, the example of which was shown in Figure 3.8).

One special feature of our implementation is that we don’t enforce selectional restrictions on ARGUMENT of adverbials if they fit the adjunct restrictions of the verb. For example, LF::To-loc has the (*Trajectory +*) restriction on its arguments. This means that it can modify two types of verbs: those that explicitly declared that they take LF::To-loc argument (this will be the preferred modification), regardless of the semantic features set on the verb, and those that didn’t declare LF::To-loc as a semantic argument, but that are marked as (*Trajectory +*) in the lexicon. This is an implementation decision, because we found in practice that we frequently needed exceptions for just 1 or 2 verbs in a class to allow them to be modified by a given adverbial. This is especially true if specialization is involved: in our Monroe domain, for example, we want trajectory adverbials to apply to all motion actions (the default domain-independent restrictions), and to only a small set of states (*pass-by, avoid*), which is more restricted than in the domain-independent lexicon. We have an automatic scheme to specialize the lexicon to the domain, to be discussed in the next chapter, and allowing the adverbials to apply to verbs which specify a reasonable domain-specific restriction provides an easy mechanism to implement this domain-specific constraint, while in general it does not interfere with domain-independent parsing.

Adjectives are handled similarly to adverbs. They declare restrictions on their subcategorized arguments, if any, and on nouns they can modify. Currently, we don’t have preferential treatment for adjectives which could fill a slot implemented in our grammar, but we can consider it in the

future.

The place where our grammar could benefit the most from selectional restrictions is noun-noun modification, because it is in general a major source of ambiguity in semantic interpretations. We believe that there cannot be a general solution there as we achieved with verb-adjunct and adverbial-argument combinations, however, we believe N-N modification could benefit from a simple scheme which prefers modifications that can fill a slot in the ROLES list of the noun. We have not implemented this technique, but we are planning to do it as part of our future work.

### 3.5 A formal model for ontology and lexicon inheritance

In this chapter, we described a lexicon linked to an ontology with a hierarchical structure and default feature assignments. The frames in the ontology provide building blocks for a domain-independent semantic representation described above, but the selectional restrictions are implemented using feature sets assigned to words. Words are leaves in the LF hierarchy where feature sets are inherited from parent LF types.

In this section, we provide a formal model for the inheritance in the ontology, as a basis for the algorithms to assign feature set for words. We are basing our model on Carpenter’s [Carpenter, 1993] default inheritance model. Our definitions are slightly modified from Carpenter’s to reflect our goals in ontology design, and we adapted his definitions for use with our simpler feature set structures. We then extend the Carpenter’s definitions to cover the feature inference mechanism used in our system (see Sections 2.2.3 and 3.4.1). We fully incorporate feature inference in our formal system by defining a new operation, feature rule unification, and formally stating the algorithm for determining a feature set assigned to a word in this extended model.

#### 3.5.1 Default inheritance model

First, we consider the model of our default feature inheritance without the inference rules.

**Definition 3.1 ( Default Unification )** *The result of adding the default information in  $G$  to the strict information in  $F = \langle T, Q \rangle$  is given by  $F \overset{\leq}{\sqcap}_c G = \{F \sqcap G' \mid G \sqsubseteq G' \text{ and } G' = \langle T', Q' \rangle \text{ is maximally specific such that } (F \sqcap G') \neq \perp \text{ and for any feature } f, Q(f) \sqcup Q'(f) = \top\}$*

The first part of the definition means that our default unification mechanism attempts to add all the features from  $G$  structure into the  $F$  structure which won’t cause conflicts with the strict

features in  $F$ , and this part is the same as Carpenter’s definition of sceptical default unification.<sup>4</sup>

The second part of the definition means that we won’t specialize strict features with default features. In practice, if the strict features have (*Mobility Movable*), and the default is (*Mobility Self-moving*), we keep the (*Mobility Movable*) value on that definition. This is different from Carpenter’s definition, but we felt that it ultimately made more sense in our system. This way, if the developer specifies the feature value explicitly, we assume that this is more specific information than the default, which is likely in this case to be inherited from the higher levels of the ontology.

Note that the second part of the definition is slightly simplified, under the assumption of single inheritance between feature values. If the hierarchy is single inheritance, then  $Q(f) \sqcap Q'(f) \neq \perp$  means that either  $Q(f) \sqsubseteq Q'(f)$  or  $Q'(f) \sqsubseteq Q(f)$ . If this is the case, then the second part of the definition means essentially that if  $Q(f)$  is specified (that is, not equal to  $\top$ ), then  $Q'(f)$  should not be, so as to not specialize the feature  $f$ . The feature set we described in this thesis and which we use in our parsing is single inheritance by design, so this definition is sufficient for our needs. If there was multiple inheritance in our feature hierarchy, we would need a more complex check to achieve the same result.

Under this definition, we can now define a feature set associated with a lexical entry. First, let us clarify the notation. We will say that the notation

$(T (:required Q_1) (:default Q_2))$  where  $T$  is a type and  $Q_1$  and  $Q_2$  - feature assignments, will be interpreted as creating a couple typed feature sets  $(R = \langle T, Q_1 \rangle D = \langle T, Q_2 \rangle)$ , where  $R$  is the strict part of the definition, and  $D$  is the default part of the definition. For example, for feature set  $(Phys-obj (:required (Origin Artifact)) (:default (Form Solid-object)))$ ,  $T=Phys-obj$ ,  $Q_1 = (Origin Artifact)$ ,  $Q_2 = (Form Solid-object)$ , and, correspondingly,  $R = (Phys-obj (Origin Artifact))$  and  $D = (Phys-obj (Form Solid-object))$ .

**Definition 3.2 ( Default inheritance in lexical entry )** *A lexical entry in the lexicon with lf-parent  $P$  and sem  $S$  assigns the feature list*

$$(\dots(((R_S \sqcap R_1 \sqcap R_m) \overset{\leq}{\sqcap}_s D_S) \overset{\leq}{\sqcap}_s D_1) \overset{\leq}{\sqcap}_s \dots \overset{\leq}{\sqcap}_s D_{m-1}) \overset{\leq}{\sqcap}_s D_m$$

Where  $R_S$  is the required information from the word definition,  $D_S$  is the required information from the word semantics, and  $(\langle T_1, R_1 \rangle \langle T_1, D_1 \rangle), \dots, (\langle T_m, R_m \rangle \langle T_m, D_m \rangle)$  is a sequence of feature sets starting from the LF-PARENT and ending in the ontology root.

---

<sup>4</sup>To be precise, the definition is the same as Carpenter’s credulous unification, but because our feature structures do not have structure sharing in the lexicon, it is also equivalent to his definition of sceptical unification.

This definition replicates the algorithm we described in Section 3.4.1. Intuitively, it states that first the required information from the entry and all its parents is unified, producing the full set of features required in the entry (step 1 in our algorithm), and then the default features are brought in, so that the default features lowest in the tree override the default features of the parents (steps 2 and 3 in our algorithm). It is easy to see that this is the case, because once we did a (default) unification with  $D_1$ , which is the LF-PARENT, the default unification with its parent  $D_2$  will not be able to override the features added to the resulting set.

The above definition is equivalent to Carpenter’s default inheritance definition. We did not formalize the notion of node ordering in our exposition, because in a single inheritance hierarchy there is a unique path between the entry and the root. This is exactly the path that would be returned by Carpenter’s topological ordering.

This definition is implemented in our system, with one addition, namely, the requirement that children can only specialize the values set by parents. Consider the following setup: the type LF::Parent-type declares that it has a required feature value (*Mobility Self-moving*). Then its child LF::Child-type declares (*Mobility Movable*), which is more general than the parent’s value. According to our definition, the final value assigned to the lexical entry using LF::Child-type will in fact be  $(\text{Mobility Movable}) \sqcap (\text{Mobility Self-Moving}) = (\text{Mobility Self-Moving})$ . Thus, in effect, our default inheritance definition prevents the child from setting a more general feature value than the one inherited from the parent. We detect the situations when that happens and signal an error, because this indicates with high probability an error or an inconsistency in the ontology design.

### 3.5.2 Default inheritance with feature rules

We now turn to extending our definition to include the feature inference mechanisms employed in our system.

**Definition 3.3 ( Feature Rule )** *A feature rule is a tuple  $FR = \langle SF, IF \rangle$  where  $SF = \langle T_1, Q_1 \rangle$  and  $IF = \langle T_2, Q_2 \rangle$  are typed feature sets such that  $SF \sqcap IF \neq \perp$  and for any feature  $f$ ,  $Q_1(f) \sqcup Q_2(f) = \top$*

The first condition ensures that the rule is non-contradictory, that is, for example, that a rule (*Origin Plant*)  $\Rightarrow$  (*Origin Animal*) is not a valid rule because it attempts to assign a contradictory value to the same feature. The second part of the definition ensures that the rule does not attempt to specialize the feature value - that is, excludes rules like

$(Origin\ Living) \Rightarrow (Origin\ Plant)$ . While such rule is not contradictory per se, it can lead to contradictions during rule application, and therefore should be eliminated.

**Definition 3.4 ( Feature Rule Application )** *The application of a feature rule  $FR = \langle SF, IF \rangle$  to a feature structure  $F$  is a new feature structure  $FI = FR \overset{\Rightarrow}{\sqcap} F$  such that  $FI = F$  if  $F \not\sqsubseteq SF$  and  $FI = F \sqcap FI$  if  $F \sqsubseteq SF$ .*

This definition corresponds to the informal description of the feature inference process in Section 2.2.3. Assume we have a feature inference rule  $(Phys-obj (Origin\ Human)) \Rightarrow (Phys-obj (Form\ Solid-object))$  If the rule applies, there are four possible situations

1. The feature structure  $F$  to which the rule applies does not have a feature value set explicitly for the *Form* feature (which we model in our formal model by assigning  $\top$  to the feature). If this is the case, unifying  $F$  and  $FI$  will in effect add the inference  $(Form\ Solid-object)$  to the feature list.
2. The structure  $F$  has a more general value - for example,  $(Form\ Object)$ . Then, the unification will replace this more general value with the more specific value from the rule.
3. The structure  $F$  has a more specific feature value declared. There is nothing more specific than *Solid-object* in our feature hierarchy, but if we had a more specific value, say *Solid-breakable-object*, and it was specified on an entry, the unification will keep this value intact, and will in effect be just a consistency check.
4. The structure  $F$  has an incompatible feature value declared, for example,  $(Form\ Hole)$ . If this is the case, the unification will return  $\perp$ , and the system will signal an error due to inconsistency in the feature structure.

From this analysis, it is easy to see that the following two properties are true:

$$\perp \overset{\Rightarrow}{\sqcap} FR = \perp \text{ for any } FR \quad (3.1)$$

$$(F \overset{\Rightarrow}{\sqcap} FR) \overset{\Rightarrow}{\sqcap} FR = F \overset{\Rightarrow}{\sqcap} FR \text{ for any } F, FR \quad (3.2)$$

We now turn to how the rules can be applied in the system. The first thing to notice is that rule application is not commutative. Consider the following example. Assume we have two feature rules (for brevity, we will omit feature types in our examples)

*FR1:  $(Origin\ Living) \Rightarrow (Form\ Solid-object)$  and*

$FR2:(Form\ Solid-object) \Rightarrow (Spatial-abstraction\ Point)$ <sup>5</sup>. Now assume we are trying to apply these rules to a feature set  $F:(Phys-obj\ (origin\ natural))$ . If we apply  $FR1$  first, we get a feature set  $(Phys-obj\ (origin\ living)\ (form\ solid-object))$ , and then  $FR2$  will apply and add the  $(spatial-abstraction\ point)$  feature. Thus,

$$(F \overset{\Rightarrow}{\sqcap} FR1) \overset{\Rightarrow}{\sqcap} FR2 = (Phys-obj\ (Origin\ Living)\ (Form\ Solid-object)\ (Spatial-abstraction\ Point)).$$

In contrast, if we attempt to apply  $FR2$  first, it is not applicable to  $F$  directly, and then after application of  $FR1$ , the  $Spatial-abstraction$  feature will remain unchanged,

$$\text{and } (F \overset{\Rightarrow}{\sqcap} FR2) \overset{\Rightarrow}{\sqcap} FR1 = (Phys-obj\ (Origin\ Living)\ (Form\ Solid-object)).$$

To deal with the order dependency problem, we introduce a notion of rule inference:

**Definition 3.5 ( Feature Inference )** *Given a sequence of feature rules*

$FRS = \langle FR_1, \dots, FR_k \rangle$ , *we define a feature inference on a typed feature set  $F$  as*

$$F \overset{\Rightarrow}{\sqcap}_I FRS = ((F \overset{\Rightarrow}{\sqcap} FRS) \overset{\Rightarrow}{\sqcap} FRS \dots (k\ times) \overset{\Rightarrow}{\sqcap} FRS), \text{ where } F \overset{\Rightarrow}{\sqcap} FRS \text{ denotes } F \overset{\Rightarrow}{\sqcap} FR_1 \dots \overset{\Rightarrow}{\sqcap} FR_k$$

This definition introduces a notion of applying a set of rules to a given feature set, in a manner independent of the ordering of the rules, as proven by the following theorem.

**Theorem 3.1 ( Feature inference fixed point )**

$$\text{For any feature rule } FR \in FRS, (F \overset{\Rightarrow}{\sqcap}_I FRS) \overset{\Rightarrow}{\sqcap} FR = F \overset{\Rightarrow}{\sqcap}_I FRS$$

Intuitively, this theorem states is that once the feature inference is complete, no further applications of rules can possibly change the result. This means that in the original definition, we could have used a set instead of a sequence of rules, as the order of the rules is immaterial in the feature inference application.

*Proof* The proof relies on the following property: for any rules  $FR_1, FR_2$ ,

$$\text{if } F \overset{\Rightarrow}{\sqcap} FR_1 \neq F, \text{ then } ((F \overset{\Rightarrow}{\sqcap} FR_1) \overset{\Rightarrow}{\sqcap} FR_2) \overset{\Rightarrow}{\sqcap} FR_1 = (F \overset{\Rightarrow}{\sqcap} FR_1) \overset{\Rightarrow}{\sqcap} FR_2 \quad (3.3)$$

That is, once the rule has been applied, its repeated applications do not change the resulting structure. This follows directly from the analysis of 4 possible cases of rule application presented earlier in this section.  $F \overset{\Rightarrow}{\sqcap} FR_1 \neq F$  means that rule  $FR_1$  is applicable to the feature structure  $F$  and case 1, 2 or 3 are true. If case 3 is true,  $F \overset{\Rightarrow}{\sqcap} FR_1 = \perp$ . Then it is easy to see that  $\perp \overset{\Rightarrow}{\sqcap} FR = \perp$  for any rule  $FR$ , and Equation 3.3 holds.

If either case 1 or 2 are true, than  $F \overset{\Rightarrow}{\sqcap} FR_1$  will set some feature values. Then, application of  $FR_2$  will have 3 possible effects: it will not change any of the values set by  $FR_1$ , it will

---

<sup>5</sup>This particular dependency is not a rule in our system, but is used here for purposes of exposition.

specialize them, or it will result in  $\perp$  in attempting to change them into incompatible values. In any of those situations repeated application of  $FR_1$  will obviously not change the result: if the features were not changed, or were specialized,  $FR_1$  cannot replace them; if the previous result was  $\perp$ , application of  $FR_1$  will keep it  $\perp$ .

Given Equation 3.3, we will use proof by induction for our theorem.

The base case is if  $FRS = \{FR_1\}$ . Then,  $(F \overset{\rightrightarrows}{\Pi}_I FRS) \overset{\rightrightarrows}{\Pi} FR_1 = (F \overset{\rightrightarrows}{\Pi} FR_1) \overset{\rightrightarrows}{\Pi} FR_1 = F \overset{\rightrightarrows}{\Pi} FR_1$ . Now, assume  $FRS_{k-1}$  is of size  $k-1$  and  $(F \overset{\rightrightarrows}{\Pi}_I FRS_{k-1}) \overset{\rightrightarrows}{\Pi} FR_i = F \overset{\rightrightarrows}{\Pi}_I FRS_{k-1}$  for any  $FR_i \in FRS_{k-1}$ .

Now, consider  $FRS = FRS_{k-1} \cup \{FR_k\}$ , and  $(F \overset{\rightrightarrows}{\Pi}_I FRS) \overset{\rightrightarrows}{\Pi} FR_i$  where  $FR_i \in FRS$ . If  $FR_i \in FRS_{k-1}$ , then, according to the equation 3.3 and the induction assumption  $(F \overset{\rightrightarrows}{\Pi}_I FRS) \overset{\rightrightarrows}{\Pi} FR_i = F \overset{\rightrightarrows}{\Pi}_I FRS$ .

Now, consider  $FR_k$ . Let us denote  $F'_j$  the result of applying  $FRS$   $j$  times to  $F$ . Then  $F \overset{\rightrightarrows}{\Pi}_I FRS = F'_k$  and we need to show that  $F'_k \overset{\rightrightarrows}{\Pi} FR_k = F'_k$ . Now, if for some  $j \leq k-1$   $F'_j \overset{\rightrightarrows}{\Pi} FR_k \neq F'_j$ , then, by equation 3.3, it follows directly that  $(F'_j \overset{\rightrightarrows}{\Pi} FRS_{k-1} \overset{\rightrightarrows}{\Pi} FR_k) \overset{\rightrightarrows}{\Pi} (k-j \text{ times}) FRS_k \overset{\rightrightarrows}{\Pi} FR_k = (F'_j \overset{\rightrightarrows}{\Pi} FRS_{k-1} \overset{\rightrightarrows}{\Pi} FR_k) \overset{\rightrightarrows}{\Pi} (k-j \text{ times}) FRS_k$ , that is,  $F'_k \overset{\rightrightarrows}{\Pi} FR_k = F'_k$ . If, on the other hand,  $F'_j \overset{\rightrightarrows}{\Pi} FR_k = F'_j$  for any  $j \leq k-1$ , then immediately  $(F'_{k-1} \overset{\rightrightarrows}{\Pi} FR_1 \dots \overset{\rightrightarrows}{\Pi} FR_k) \overset{\rightrightarrows}{\Pi} FR_k = (F'_{k-1} \overset{\rightrightarrows}{\Pi} FR_1 \dots \overset{\rightrightarrows}{\Pi} FR_k)$ , and  $F'_k \overset{\rightrightarrows}{\Pi} FR_k = F'_k$ .

Thus, we proved the induction step and the entire theorem by induction.

The feature inference operation is fairly expensive, quadratic in the number of inference rules. However, since our system is designed to contain only a small number of feature dependency rules, this is not a serious problem. Additional optimizations can be made. First of all, a simple check ensures is that if on the  $i$ th iteration of the rule application no feature changes were made, the inference is complete, and there's no need to continue rule application. Secondly, the rules can be re-ordered so that the rules which apply to the most specific features apply last, and in combination with the above optimization it additionally speeds up the process. In practice, we observed that the average time for inference is linear in the number of rules.

Given our definition of feature inference, we can combine it with the default features as follows

**Definition 3.6 ( Default inheritance with feature rules )** *A lexical entry with lf-parent  $P$  and sem  $S$  in the lexicon with feature inference rule set  $FRS$  assigns the feature list*

$$((\dots(((R_S \sqcap R_1 \sqcap R_m) \overset{\leftarrow}{\Pi}_s D_S) \overset{\rightrightarrows}{\Pi}_I FRS) \overset{\leftarrow}{\Pi}_s D_1) \overset{\leftarrow}{\Pi}_s \dots \overset{\leftarrow}{\Pi}_s D_{m-1}) \overset{\leftarrow}{\Pi}_s D_m) \overset{\rightrightarrows}{\Pi}_I FRS$$

Where  $R_S$  is the required information from the word definition,  $D_S$  is the required information from the word semantics, and  $(\langle T_1, R_1 \rangle \langle T_1, D_1 \rangle), \dots, (\langle T_m, R_m \rangle \langle T_m, D_m \rangle)$  is a sequence of

*feature sets starting from the LF-PARENT and ending in the ontology root.*

This definition contains two feature inference applications: before and after the default features were inferred. We already discussed the reasons for that in Section 3.4.1. We apply the first inference to all required features, before the default unification is made, to ensure that the default features won't override inferences from strict features. We apply the inference for the second time after all defaults were added, to complete the set and ensure that it is consistent, which provides additional control for default feature application.

The definitions we provided in this section rely on the fact that there is no structure sharing in our semantic feature lists. It is straightforward to extend our default unification definitions to structures with path sharing. However, it is not entirely clear how feature inference will be affected by structure sharing. This is not an issue in our lexicon, because by construction our feature sets cannot share values, but it may be an issue if more complex construction is desired, and is a part of the future work on the system.

## 3.6 Discussion

The main contribution of this chapter is the design of a LF ontology which combines frames and feature structures, and which consists of a set of types which can be easily linked to a parsing lexicon. In this section, we discuss in more depth the structure of our ontology, the choice of semantic roles in our representation, and the coverage of our domain-independent grammar and lexicon.

Currently, our LF ontology contains 566 LF types, corresponding to different concepts. The lexicon statistics is summarized in Table 3.3. We have 1632 words total in our lexicon, out of which 1184 are open class words - adjectives, nouns and verbs. The table shows statistics for different parts of speech which include the number of words belonging to each part of speech; the average number of senses (that is, different LF types) per word, and the average number of syntactic variations (defined as a sense combined with a subcategorization pattern and a syntax-semantics mapping) per word.

Verbs are the most polysemous parts of speech in our system, with 1.79 senses per word on average, and 2.82 combinations of senses with subcategorization frames every time. The next most ambiguous item in our lexicon are adverbial prepositions (classified as ADV in our scheme) - they have on average 1.65 senses per word, 2.11 counting syntactic variations. This is because many prepositions are ambiguous between spatial location, path and temporal meanings, which

is another major problem for our parser. Nouns and adjectives have relatively low polysemy counts, 1.13 and 1.32 variations respectively.

What the data in our system shows is that the most ambiguity comes from different senses of verbs and associated adverbials, and in practice we observed our parser to have the most difficulty with disambiguating these items. Therefore, we concentrated our research heavily on developing representations of verbs and adverbials, with the focus on selectional restrictions as a means of disambiguation. Domain-independent restrictions do not do the best possible job with disambiguation, but in Chapter 4 we will discuss the way a domain model can be used to tighten selectional restrictions and improve disambiguation accuracy for in-domain items.

### 3.6.1 Semantic Roles in the TRIPS lexicon

As we discussed earlier, our ontology has conceptual structure similar from FrameNet, but a significantly simplified set of roles. This has several reasons:

- Parsimony in syntactic mappings. The more role names are available in the system, the more different syntax-semantics mappings need to be written. This makes system development more difficult, as we need to keep track of changes and consistency in the lexicon.
- Selectional restrictions. FrameNet roles sometimes conflate several different possible items, which makes it impossible to write a reasonable selectional restriction. If this is the case, we separated out the FrameNet role into several different role names.
- Linguistic generalizations. Many linguistic theories have made explicit generalizations about how semantic roles and syntax stay together. While we do not make any such generalizations in our system, we are planning them for future work, and therefore trying to keep the role set reasonably simple.

There are currently 37 semantic roles used in our system. We summarize the roles in Tables 3.4 and 3.5. Table 3.4 describes the roles which in our system appear only as subcategorized arguments of nouns, verbs and adverbs. Table 3.5 presents the roles which can come both from subcategorization frames and from adverbials. This table contains two columns. One is the number of times the role appeared as a subcategorized argument on verbs, nouns or adverbs. The other is the number of adverbials and adjectives which supply this role as modifiers. For example, a semantic argument :From-loc is most frequently expressed by an adverbial preposition

|                | Count | Core-word | Senses | Synt. variations | Comment                                                                      |
|----------------|-------|-----------|--------|------------------|------------------------------------------------------------------------------|
| NUMBER-UNIT    | Yes   | 6         | 1.0    | 1.0              | <i>dozen, hundred etc</i> , which can participate in <i>a hundred people</i> |
| VALUE          | Yes   | 45        | 1.0    | 1.0              | day names, month names, letters etc.                                         |
| PRO            | Yes   | 64        | 1.06   | 1.22             | pronouns                                                                     |
| V              | Yes   | 110       | 3.75   | 5.0              | Auxiliaries and be forms                                                     |
| V              |       | 553       | 1.79   | 2.82             | Verbs except for auxiliaries and be forms                                    |
| QUAN           | Yes   | 28        | 1.07   | 1.29             | Quantifiers                                                                  |
| ORDINAL        | Yes   | 31        | 1.0    | 1.0              | Ordinal numerals                                                             |
| N              |       | 637       | 1.07   | 1.13             | common nouns                                                                 |
| PREP           | Yes   | 20        | 1.0    | 1.0              | prepositions without semantic load (from subcategorization frames)           |
| CONJ           | Yes   | 25        | 1.08   | 1.08             | Conjunctions                                                                 |
| ADV            |       | 306       | 1.65   | 2.11             | Adverbs and adverbial prepositions                                           |
| FP             | Yes   | 17        | 1.0    | 1.0              | Filled pauses                                                                |
| ADV            | Yes   | 38        | 1.16   | 1.21             | Discourse adverbs                                                            |
| INFINITIVAL-TO | Yes   | 1         | 1.0    | 1.0              | Infinitival to                                                               |
| PUNC           | Yes   | 10        | 1.0    | 1.0              | Punctuation marks                                                            |
| ADJ            |       | 292       | 1.19   | 1.32             | Adjectives                                                                   |
| NEG            |       | 2         | 1.0    | 1.0              | not, n't                                                                     |
| CV             | Yes   | 6         | 1.0    | 1.0              | Special signs: possessive 's, o' in o'clock, ^                               |
| NAME           |       | 22        | 1.0    | 1.0              | Proper names (mostly state names)                                            |
| ART            | Yes   | 11        | 1.0    | 1.0              | Articles                                                                     |
| UTTWORD        | Yes   | 113       | 1.02   | 1.02             | Words that make utterances - <i>yes, no, good morning</i>                    |
| Total          |       | 2339      | 1.45   | 1.85             |                                                                              |

Table 3.3: Lexicon statistics in our system

*from*. However, there are several words which subcategorize for it as a direct object - for example, *Leave Avon* is almost identical in meaning to *Leave from Avon*, thus we chose to tag *Avon* as the same semantic argument both times. The table shows only arguments that come at least once from subcategorization frames, not taking into account those which are always supplied by adverbial modifiers.

We already discussed earlier our decision to keep the conceptual structure in the LF ontology similar to FrameNet structures, but decrease the number of roles used in the system to make lexicon maintenance easier and create opportunities for linguistic generalization. In that respect our lexicon is similar to VerbNet database [Kipper *et al.*, 2000] which also uses thematic roles to express verb semantics and place selectional restrictions on verb arguments.

The VerbNet role system currently consists of 28 roles. Some of those are identical to TRIPS roles. These are :Experiencer, :Beneficiary, :Instrument, :Cause, :Source, and :Destination (denoted as :Goal in our system), :Extent (called :Cost in our system). In some areas, VerbNet makes finer distinctions. In particular, it makes a distinction between a :Theme and :Patient, which are joined together under the :Theme classification in the TRIPS lexicon, and between :Agent and :Actor, which are also not distinguished in the TRIPS lexicon. On the other hand, TRIPS makes some distinctions not currently present in the VerbNet lexicon. In particular, we distinguish between a more abstract notion of a “destination” and a specific “to-loc” for verbs of motion. In addition, we have some roles that come from other parts of speech. For example, the :Contents role is used to denote the contents of a container in phrases like *A bottle of water*. Since these are outside of VerbNet coverage, they do not appear there.

We did not conduct a formal evaluation of the consistency of annotation between TRIPS LF ontology and VerbNet. However, given the roles defined in both systems, they appear to be mostly consistent on the main role names. The biggest difference comes from verb semantics: VerbNet uses explicitly the three-part verb representations from Moens and Steedman [1988], whereas we only use their classification in our feature system, but do not create the complete representations. We are currently working on resolving the differences and using VerbNet selectional restrictions and syntactic patterns to extend coverage of our verb lexicon.

### 3.6.2 Generic lexicon coverage

Our generic lexicon and grammar cover four application domains. The TRAINS domain, represented by the TRAINS-95 corpus, is a planning domain where the users collaborate on tasks involving complex train routing schedules. It is partially covered by the current lexicon. We

| Role Name   | Words | Comment                                                                                                            |
|-------------|-------|--------------------------------------------------------------------------------------------------------------------|
| Action      | 4     | want wanna need like                                                                                               |
| Addressee   | 14    | Same as in communication frame in FrameNet                                                                         |
| Agent       | 240   | The intentional entity doing the action                                                                            |
| Cause       | 17    | A force (non-intentional, such as a storm) causing the action                                                      |
| Co-Agent    | 1     | confirm                                                                                                            |
| Co-Theme    | 78    | A secondary entity undergoing the action together with Theme                                                       |
| Cognizer    | 39    | A mental agent for words like <i>know</i> , <i>love</i>                                                            |
| Container   | 3     | involve hold contain                                                                                               |
| Contents    | 10    | An argument of noun phrase containers, such as <i>a bottle of water</i>                                            |
| Cost        | 4     | The amount in actions <i>take (time)</i> , <i>save</i> , <i>leave</i> , <i>cost</i> - similar to Extent in VerbNet |
| Effect      | 29    | The result achieved by an action such as <i>cause</i>                                                              |
| Entity      | 14    | The existential object for verbs like <i>be</i> , <i>exist</i>                                                     |
| Experiencer | 20    | Unintentional agent in an action                                                                                   |
| Goal        | 7     | An object to which (abstract) motion is directed                                                                   |
| Of          | 541   | The argument to which predicates apply                                                                             |
| Property    | 18    | The predicate in stative verbs like <i>be</i> , <i>feel</i>                                                        |
| Result      | 3     | The result of an action, as in FrameNet                                                                            |
| Sit-Val     | 9     | The situation which defines a time point, such as <i>meeting</i> in <i>After the meeting</i>                       |
| Source      | 13    | The source of (abstract) movement                                                                                  |
| Theme       | 398   | The entity undergoing an action                                                                                    |
| Val         | 116   | The value supplied by functional noun or adverbial, <i>e.g.</i> , noon in <i>at noon</i>                           |

Table 3.4: The semantic roles in the TRIPS system which appear only in subcategorization frames.

| Role Name              | words | adverbials | Comment                                                                                          |
|------------------------|-------|------------|--------------------------------------------------------------------------------------------------|
| Along                  | 2     | 6          | The trajectory of an action, for example <i>The truck followed route 31 to Avon</i>              |
| Associated-information | 3     | 1          | Topic in FrameNet frames                                                                         |
| Beneficiary            | 1     | 1          | find                                                                                             |
| From-loc               | 2     | 2          | An implementation of :Goal argument for explicit motion verbs. An object of <i>leave, depart</i> |
| Instrument             | 13    | 1          | The instrument used to do the action                                                             |
| Purpose                | 1     | 2          | The purpose of an action - only subcategorized by <i>go</i>                                      |
| Spatial-loc            | 1     | 38         | The spatial location of an action.                                                               |
| Time-duration-rel      | 1     | 3          | The duration of an action, currently only <i>3 hours</i> in <i>we waited for 3 hours</i>         |
| To-loc                 | 9     | 9          | An implementation of :Goal argument for explicit motion verbs. An object of <i>reach</i>         |
| Via                    | 5     | 6          | The location through which a movement passes, <i>e.g., avoid Delta</i>                           |

Table 3.5: The semantic roles in the TRIPS system which can either be subcategorized arguments or adverbial modifiers.

| Dialogue     | Total Utterances | Good | Bad | Incomplete | Ungrammatical | % Good |
|--------------|------------------|------|-----|------------|---------------|--------|
| s2-accuracy  | 405              | 285  | 83  | 35         | 2             | 77.4%  |
| s2-coverage  | 405              | 335  | 33  | 29         | 1             | 91.03% |
| s4-accuracy  | 386              | 200  | 125 | 50         | 10            | 61.3%  |
| s4-coverage  | 386              | 234  | 92  | 50         | 10            | 71.8%  |
| s12-accuracy | 189              | 136  | 31  | 20         | 2             | 81.4%  |
| s12-coverage | 189              | 151  | 17  | 20         | 2             | 89.9%  |
| s16-accuracy | 383              | 271  | 83  | 21         | 8             | 76.55% |
| s16-coverage | 383              | 291  | 63  | 9          | 9             | 82.2%  |
| s17-accuracy | 389              | 267  | 96  | 23         | 3             | 73.55% |
| s17-coverage | 389              | 295  | 68  | 21         | 3             | 81.3%  |

Table 3.6: Corpus Statistics

did not conduct a formal evaluation, but the parser is capable of parsing many noun phrase and sentence types in the domain, and has been used on a cleaned-up version of the corpus as part of the evaluation of a PHORA pronoun resolution system [Byron, 2002a].

The Pacifica domain is a small domain where the users collaborate with a computer on planning an emergency evacuation of an artificial island using a variety of vehicles. In Chapter 4 we describe our evaluation which uses a corpus of 200 utterances in the Pacifica domain. We show that the error rate on parsing 50-best speech lattices is 32%, and that it improves with the addition of our domain-specialization algorithm.

The Monroe domain is a domain of human-human dialogues where the users are asked to plan and coordinate an emergency rescue task with a realistic city map. We used the Monroe corpus [Stent, 2000] as the primary testing domain for our parser. We selected 6 dialogues in the domain for further evaluation. They were marked for speech repairs [Swift *et al.*, pear] and other disfluencies which our parser is currently not capable of handling. In addition, we marked incomplete and ungrammatical utterances in the corpus. The corpus statistics are shown in Table 3.6.

We report two statistics for the Monroe domain: the raw accuracy and the coverage in our corpus files. The raw accuracy is obtained by running the current version of the parser over the corpus and calculating the number of sentences parsed correctly. The final corpus files also contain manually selected parses which do not come up as a first choice parse for that utterance,

because our parse scoring function based on sense and attachment preferences does not always make the correct choice. In this sense, the final files provide a more complete picture of our grammar coverage because they represent the utterances which are covered by the grammar, the upper bound of current grammar coverage which could be reached with a better scoring strategy.

The Medication adviser domain [Ferguson *et al.*, 2002] is the domain where users can ask questions of a computer system about the medications they are taking and possible side effects and interactions. We have a corpus of wizard of Oz dialogues. We have annotated 5 dialogues for repairs similarly to Monroe corpus, and run the coverage evaluation. The results are shown in Table 3.7. We have not yet evaluated the utterances to see if there are valid parses weighted lower than the best-first parse, so only the raw accuracy is reported, and thus the overall coverage is comparable to the raw coverage in the Monroe corpus.

| Dialogue | Total Utterances | Good | Bad | Incomplete | Ungrammatical | % Good |
|----------|------------------|------|-----|------------|---------------|--------|
| med1     | 60               | 47   | 12  | 1          | 0             | 79.66% |
| med2     | 39               | 23   | 16  | 0          | 0             | 58.97% |
| med3     | 48               | 30   | 18  | 0          | 0             | 62.5%  |
| med4     | 90               | 67   | 20  | 3          | 0             | 77.01% |
| med5     | 57               | 32   | 19  | 4          | 2             | 61.53% |

Table 3.7: Coverage statistics for our Medadvisor corpus

Note that in both cases the results represent sentence level accuracy, not just the constituent accuracy which is often used to evaluate syntactic parsers. The reason is that we are interested in a parser which produces correct logical form. Therefore, we counted parses as good only if both the syntactic analysis and the semantic form produced by the parser were correct.

To give an idea of the complexity of the sentences we are trying to parse, Table 3.8 lists some sample sentences in our test domains. We list a variety of complex sentences our grammar is capable of parsing, and two examples which are not covered. The table highlights the major difficulties we encounter in parsing speech. These are, on the one hand, long and complex sentences, which present significant challenges due to attachment ambiguities. On the other hand, there are short fragments, which make it impossible to assume that we should expect a complete sentence as a result of parsing, and thus further complicate grammar and search algorithms.

As the table shows, our parser is capable of handling quite complex sentences - there is one 11-word sentence shown which is interpreted properly as the top choice, and another 16 word sentence which is covered, though the correct interpretation comes up as a second choice in our current parser. The most common problem failures in our parser are due to ellipsis, common in natural language. We could add syntactic rules in our grammar, but we have not yet formulated a reasonable logical form to express elliptical sentences. This is planned as part of our future work.

| Sentence                                                                  | Parsed                                                                       |
|---------------------------------------------------------------------------|------------------------------------------------------------------------------|
| Oh I take glyburide ritalin and amoxicillin at breakfast at seven am      | yes                                                                          |
| When will you be eating after your surgery                                | yes                                                                          |
| and I need to find out when to take them                                  | yes                                                                          |
| the little symbol                                                         | yes                                                                          |
| and there are three people on a stretcher at the airport                  | yes                                                                          |
| to saint mary                                                             | yes                                                                          |
| i don't know where the guy with the broken leg was                        | yes                                                                          |
| and then in highland park there is a person with a broken leg in the park | as 2nd choice - first interpretation incorrectly attaches "in the park"      |
| oh fifteen a and two fifty two                                            | as 2nd choice - first interpretation of 252 is a time                        |
| Then I have the second ritalin at lunch and amoxicillin at lunch as well  | no - ellipsis in the second part of the sentence                             |
| and there's one broken leg walking person at highland park                | no - noun-noun modification too complex to produce a reasonable logical form |

Table 3.8: Sample sentences from our test corpora

One level of complexity not covered by our parser is handling of speech repairs. As we noted earlier, we cleaned up our corpora from speech repairs and ungrammatical sentences. Table 3.6.2 shows some examples of original sentences and the way the repairs were cleaned up. The repairs are indicated by square brackets. So far, we were unable to include handling of repairs in our parser, due to drastic increase of ambiguity in the parses. We are currently working on a new architecture [Stoness, 2001] which allows for incremental processing with incremental syntactic

and semantic interpretation. We hope that it will allow us to handle repairs in a more efficient way than possible within our current architecture.

| Original sentence                                                                                        | Repaired version or judgment                                              |
|----------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------|
| if you can find the [ r three ninety ] road where it says three ninety                                   | if you can find the road where it says three ninety                       |
| [ and th ] and then in highland park there is [ a broken leg or ] a person with a broken leg in the park | and then in highland park there is a person with a broken leg in the park |
| actually it's right a ab                                                                                 | INCOMPLETE                                                                |
| i'm telling you i have a                                                                                 | INCOMPLETE                                                                |
| i should suggest too i don't know to change the hospitals                                                | UNGRAMMATICAL                                                             |
| maybe he can wait this guy some more for another ambulance                                               | UNGRAMMATICAL                                                             |

Table 3.9: Examples of repairs and ungrammatical utterances in our test corpora

### Comparison with other parsers

A difficult question which arises in the process of evaluating coverage and accuracy of a parser is how to compare it with other existing systems. A number of different wide-coverage parsers have been developed in recent years, including statistical parsers built from corpus grammars [Charniak, 1997; Collins, 1999], linguistically motivated unification grammars [Alshawi, 1992; Dowding *et al.*, 1994; Copestake and Flickinger, 2000] and dependency parsers [Lin, 1998; Sleator and Temperley, 1993]. However, it is difficult to compare these parsers on coverage and accuracy, for several reasons.

The core issue is that different grammars utilized by those systems. Because different grammatical theories result in different trees, the direct comparison is difficult. Parsing accuracy is typically measured as bracket crossing accuracy or precision-recall for different dependency relations within a sentence, provided an agreement can be reached as to the relevant levels of annotation (typically supplied by the Penn Treebank annotation) and dependencies. Lin [1995] discusses in more detail the issues involved. We could do a similar evaluation on our corpora, and are considering implementing it in the future. At present, however, we felt that it was not appropriate for our goals.

As we stated in the introduction, we are interested in parsing as necessary for semantic interpretation. Measures which tell us accuracy in terms of the number of syntactic constituents interpreted correctly are not very informative about the impact of those numbers on sentence level accuracy, and even a single error in a sentence may mean an incorrect interpretation as a whole. For example, MINIPAR interprets *Is it good to take aspirin* as

[<sub>S</sub> Is [<sub>NP</sub> it ] [<sub>AP</sub> good [<sub>PP</sub> to [<sub>NP</sub> take aspirin]]]]

From the point of view of constituent evaluation, this sentence is mostly good, because only one phrase, *to take aspirin*, was parsed as a noun phrase instead of a verb phrase. From the point of view of semantic interpretation, however, this sentence has no chance to be interpreted correctly, because that error is crucial for understanding.

Existing corpus and dependency grammars were not developed for semantic interpretation from the outset, and significant effort is necessary to utilize them in semantic interpretation. We do not have a formal evaluation, but in our experience it is non-trivial to obtain reasonable semantic interpretations from existing large-scale systems, because the rules there frequently do not include distinctions related to argument structure, such as argument-adjunct distinction or predicate-argument relationships. In one case, over 6 months of work was necessary to develop a set of semantic rules to interpret MINIPAR's output on text data.<sup>6</sup> The problem becomes worse if we need to parse speech corpora, such as those in our domains. Since the existing parsers were all developed for text, they have poor coverage for questions, imperatives and fragments common in our data, which makes the comparison even more difficult. Additional evaluation is required to determine whether modifications to existing parsers will be sufficient to overcome the difficulties mentioned above to make them work for parsing and semantic interpretation in our domains. This is planned as part of our future work.

An additional difficulty in both training and evaluating corpus-based parsers is that they typically rely on having a corpus annotated with some annotation scheme. In order to annotate the corpus for our needs we in effect need the understanding of how the annotation will relate to the semantic interpretation, and we found that working that out is in many cases comparable in difficulty to writing the appropriate grammar rules. Because of that, it is easier to judge the correctness of parse trees output in our parser than attempt to annotate from scratch.

Developers of linguistically-motivated deep parsers are faced with similar issues, because these grammars build deep analyses geared toward semantic interpretation. From the 3 wide-coverage parsers for language understanding reviewed in the next section, we could only find

---

<sup>6</sup>Len Schubert, personal communication.

coverage figures for the LINGO parser. On the LINGO Redwoods corpus [Oepen *et al.*, 2002], based on 4 dialogues from the VerbMobil domain, the LINGO grammar has 91% coverage. However, it is not possible to compare this figure directly to our parser, because it is over a different corpus (appointment scheduling), and currently there is no easy way to compare the complexity of two corpora, on which the coverage depends to a large extent. For example, our corpora contain many fragments which are not complete sentences but which are complete utterances in an answer to a question - for example, *at 5 pm*. Our parser finds parses and semantic interpretations for these sentences. The results we found specified that they excluded fragments. Thus, the data sets given to the parsers are substantially different in terms of utterance types, and cannot be directly comparable.

We obtained the LINGO parser, and from our informal tests, there are sentence types, especially involving conjunction, which are covered by LINGO but not our parser. At the same time, especially in our medical domain, we found sentences which were covered by our parser, but for which LINGO parser did not produce appropriate semantic representations. An example would be *Is it good to take aspirin*. The version of LINGO we tested<sup>7</sup> did not produce a representation which represented *to take aspirin* as an argument of predicate *good*, something necessary for correct interpretation in our domain. We are considering a more detailed evaluation of our parser which attempts to analyze in more detail various grammatical phenomena covered by it in our future work.

### 3.6.3 Selectional restrictions in wide coverage parsers

As we noted earlier, selectional restrictions are used very actively in case-frame systems, because the interpretation process depends on finding the correct fillers for slots in a case frame. Selectional restrictions have been used less frequently in the large-scale syntactic parsers. In this section, we review three state-of-the-art systems using selectional restrictions and compare them to our approach.

The CORE Language Engine [Alshawi, 1992] is a large-coverage unification grammar which actively uses selectional restrictions (called sortal constraints in the system). The sortal constraints are checked during building the Quasi-Logical Form, which is executed after the syntactic parsing is completed. The system provides a domain-independent sortal constraint hierarchy which consists of 31 concepts. The hierarchy is multiple inheritance, and efficiency is achieved by coding up hierarchy entries as paths in the tree. Selectional restrictions are checked by unifica-

---

<sup>7</sup>as of May 2002

tion, and the system employs union terms, which are informally defined similar to our collection sets, to represent sortal constraints with conjunction. The CORE Language approach is closest to our approach in the sense that it places emphasis on semantic processing for reasoning, and uses syntactic parsing as a support tool for semantic processing. Our approach is more limited in the sense that we do not attempt to do any pragmatic processing, as CORE does. However, we believe that ultimately pragmatic processing has to be done using the domain-specific semantic information, and therefore we place emphasis at lexicon acquisition and design issues, and the explicit support of the ontology that underlies the domain-independent semantic processing. We then develop an algorithm to semi-automatically specialize our representations to the domain, discussed in Chapter 4.

The SRI GEMINI system [Dowding *et al.*, 1994] uses a unification formalism similar to CORE to parse spoken language in the ATIS travel domain. GEMINI combines syntactic and semantic processing for efficiency, however, it defers the sortal constraint checking till after the parse is done. This is necessary for the efficiency purposes, because, in the Gemini encoding of sortal constraints as terms, if, for example, a subject of a verb phrase has not been bound, then edges for all possible sorts of subjects need to be added to the chart. Our system does not suffer from the same problem. The reason is in the different matching algorithm. For example, verb entries are added to the chart with restrictions on all 3 of their arguments, by the time a verb phrase has been formed, the selectional restrictions eliminate most of the possible verb senses, thus eliminating possible ambiguities. The only case when the ambiguity remains is when pronouns are used as objects, thus not constraining the verb senses, but these cases comprise a relatively small subset of our current corpora, and therefore do not have a major effect on the system performance.

The VerbMobil system [Kay *et al.*, 1994] uses the HPSG LINGO [Copestake and Flickinger, 2000] grammar for parsing and semantic interpretation for speech translation in the scheduling domain. The original grammar developed for VerbMobil did not have built-in selectional restrictions. A separate process has been documented for use in checking selectional restrictions of spatial prepositions [Stede *et al.*, 1998]. Strong selectional restrictions through subtyping were used in that case. We were unable to find information about handling conjunctions and other special cases of selectional restrictions in the system. Recently, Androutsopoulos and Dale [2000] developed two ways to formally integrate selectional restrictions with a HPSG grammar: using the BACKGROUND field, which results in deferred checking of selectional constraints after a parse has been built, and using index subsorts, which allows selectional constraints to be checked during parsing. They found that the latter approach was more efficient in their practical

applications, similar to what we observed in our parser.

### 3.7 Conclusions

In this chapter, we presented a domain-independent ontology and lexicon for parsing in dialogue systems. We showed how they can be integrated with selectional restrictions based on feature lists, and described the techniques like default features that simplify lexicon acquisition and maintenance of lexicons with features. We described how the selectional restrictions are implemented in our grammar, and how a domain-independent logical form can be built with our parser. We then formalized model for default feature inheritance and feature inference to provide a basis for our algorithms and facilitate comparisons with other methods. Finally, we discussed the issues that guide lexicon design for parsing systems, namely, the choice of predicates in the LF ontology and semantic arguments, and discussed how our parser implementation differs from other existing systems.

Our lexicon is still comparatively small, 2339 words. However, this is sufficient to deal with four different application domains, and its design highlights the important tradeoffs that need to be considered in designing natural language lexicons for medium-sized NLU systems. Especially important is maintaining the general coverage - so that concepts and role names are not specialized to a single domain and can be re-used in multiple domains - while also providing a set of concepts and role names that is reasonably easy to manage and link to syntax in the parsing grammar. Part of our solution to achieving this balance is limiting the scope of the information in the LF ontology: we encode the selectional restrictions with correspond to significant linguistic generalizations, and we define LF concepts and arguments in a way that we believe there is a reasonable chance to disambiguate them during a domain-independent parsing stage. This is similar to the distinction made by the CORE [Alshawi, 1992] language engine, where sortal constraints encode only the information that can be checked efficiently during parse time.

We extend a similar approach to our ontology design, and further enhance it by the introduction of an automatic mapping between domain-independent and domain-specific ontologies, which is used to both convert our domain-independent representations into domain-specific representations suitable for reasoning, and to propagate domain-specific restrictions into the lexicon to use them efficiently. Our specialization mechanisms, and the improvements they bring to parsing speed and accuracy, are described in the next chapter.

## 4 Using features for domain specialization

In the previous chapter, we described a domain-independent ontology and lexicon which used features to express selectional restrictions to aid in parsing and disambiguation, and which generate a domain-independent logical form which can be used for discourse processing. In developing dialogue systems for multiple domains, another question which needs to be addressed is domain customization. Different back-end applications require different ontologies, and the parser needs to produce a semantic representation in the format suitable for the reasoners. In this chapter, we show how domain customization can be achieved with the help of the ontology transform rules, and how using a feature representation for selectional restrictions allows us to easily customize the lexicon to suit the needs of the domain.

We begin with the review of existing work on building customizable parsers, in Section 4.1. In Section 4.2 we discuss in detail the reasons to separate domain-independent and domain-specific knowledge representations in multi-domain dialogue systems. We then describe our customization method to convert the domain-independent LF representation into domain customized representations used for reasoning, in Section 4.3. In addition to obtaining customized knowledge representation, we use our method to specialize the lexicon and selectional restrictions to the domain. We discuss our lexicon specialization method and its evaluation in Section 4.5. We conclude with discussing the tradeoffs between portability and efficiency in our method in Section 4.7.

This chapter is largely based on the work published by Dzikovska *et al.* in [Dzikovska *et al.*, 2002; Dzikovska *et al.*, 2003].

## 4.1 Background

In the introduction, when we discussed lexicon development and semantic representation needs for practical dialogue systems, one of the reasons we cited for using syntactic parsers is portability. Once we have a dialogue system developed in one domain, we would like to move as many components as possible to other domains with minimal development costs. This includes porting the parser and semantic interpreter for the new domain. Let us review some of the arguments we made for using a domain-independent parser in more detail, in context of the previous work on building portable dialogue systems.

Currently, 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]).

We already discussed (Section 1.1) why using syntactic information is important in parsing and semantic interpretation for medium scale domains. Our alternative is to use a linguistically motivated deep parser as a basis of domain-independent parsing and interpretation in our system. 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.*, 1994]. Selectional restrictions encoded in the system lexicon considerably speed up parsing and improve disambiguation accuracy in parsing of in-domain sentences.

While parsing speed and accuracy are the most important consideration, we need to balance their improvement with the cost of lexicon development. One of the major bottlenecks there 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 needs to be changed for each new domain.

One possible way to address this re-linking problem is to work on separating domain-independent and domain-specific information in the parser. For example, McDonald [1996] maps the output of a partial parser to the semantic representation for information extraction to improve parsing speed and accuracy.

The AUTOSEM [Rosé, 2000] system makes linking a system lexicon with domain-specific knowledge representation 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.

While AUTOSEM re-uses syntactic information across domains, it does not provide a way to re-use common semantic properties of words. We have already discussed an organization of a domain-independent lexicon and our domain-independent LF ontology. In this chapter, we show how using it as an intermediate layer of abstraction allows us to easily customize our system to different domains. In our system architecture, 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 [Allen *et al.*, 2000]. Our parsing and interpretation architecture mirrors this division of labor. The LF ontology is designed primarily for parsing and discourse processing, and is correspondingly used by the parser and interpretation manager. 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 4.3.

The next section discusses this two-layer architecture in more detail, by outlining the differing needs of domain-independent and domain-specific ontologies, and their corresponding design principles.

## 4.2 Linguistic ontology versus a domain model

In the process of developing our dialogue system in multiple domains, it has become clear to us 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 4.1. 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.

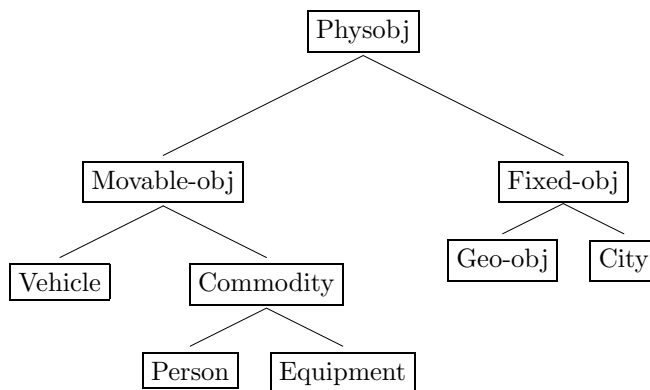


Figure 4.1: Ontology fragment for physical objects in TRIPS Pacifica domain.

The planner knows 3 main actions: MOVE, TRANSPORT and LOAD. MOVE is the action when a vehicle is moved between two 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 4.2.

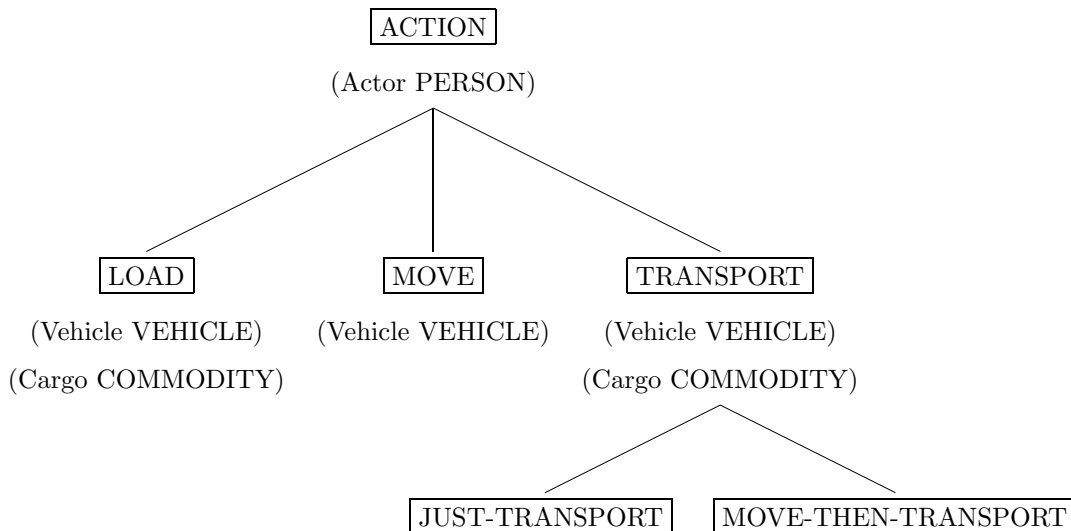


Figure 4.2: Action ontology fragment in TRIPS Pacifica domain.

Although this ontology is optimally suited for planning and 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*. This is a general definition in our domain-independent ontology, and it includes other subtypes, such as transfer or filling a container, as described earlier in Section 3.2. Similarly, 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. In our LF ontology it is achieved through defining the feature *Mobility*.

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 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 applicable 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 4.1.

We covered the design and implementation of the LF ontology and an associated lexicon in Chapter 3. To summarize, 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::Consume and LF::Removing senses of *take*, since these two senses can be distinguished with different syntactic and semantic patterns. The LF::Consume 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 movable 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 this chapter, 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.

| <b>LF Ontology</b>                                                            | <b>KR Ontology</b>                                                                      |
|-------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| 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 4.1: Design considerations for LF and KR ontologies.

### 4.3 From LF representation to domain-customized KR representation

In the previous section we discussed the differing needs of language and reasoning ontologies, and our architecture which separates the domain-independent LF ontology and the domain-specific KR ontology. We integrate domain-independent and domain-specific information by defining a set of transforms to convert the generic LF representation into a domain-specific KR representation. 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 classes).

Conversion between LF and KR representations is a two-step process. First, given a LF type in our domain independent semantic representation, we need to determine what is the corresponding KR type in the domain. Then the associated semantic arguments and modifiers coming from adverbs and adjectives need to be re-mapped. We consider these questions in turn:

- Given a LF type, how do we determine a KR type of a resulting expression?
- Given a LF type and a set of associated semantic arguments, how do we transform it into a corresponding KR expression, given that there is not always 1-to-1 correspondence between the LF and the KR argument structures.
- How do we transform adverbial modifiers, adjectives and other adjuncts?

Throughout this section, we will discuss only the question of making the correspondence between LF and KR arguments, because we found that this was the most difficult part of the task. Once the correspondence has been established, it is relatively simple to convert between the syntax of the LF representation and the domain-specific KR syntax. We will discuss this conversion briefly in Section 4.4.

We start our discussion by describing two target KR languages we will use in examples throughout this section.

### 4.3.1 Target KR languages

To illustrate how a domain-independent logical form can be transformed into different domain-specific representations, we chose two different target KR languages: the knowledge representation used by the TRIPS World Knowledge Base (TRIPS WKB Language), and the KM knowledge representation language [Clark and Porter, 1999]. The TRIPS WKB is a FOL-like language which supports predicates with multiple arguments in its syntax. However, most of the predicates in practical implementation are 2-place predicates, and each expression must describe an object or an action. Each description contains a special TYPE predicate that denotes the type of the object being described. In that respect, the representation resembles a frame structure. Figure 4.3 shows a full description of a sentence *Load the truck with oranges* in the TRIPS WKB language.

```
(AND (TYPE e1 LOAD) (ACTOR e1 YOU123)
 (CARGO e1 oranges2) (VEHICLE e1 truck3))
(The oranges2 (type oranges2 ORANGE))
(The truck3 (type truck3 TRUCK))
```

Figure 4.3: A sample description in the TRIPS WKB language

KM is a powerful frame-based language for reasoning about objects and situations. It allows us to describe entities and relationships between entities. Clark [Clark] suggests that the KM prototype mechanism can be used to convert between natural language and KM. However, his implementation is limited to keyword-spotting in a small domain. We are going to describe how to convert the LF representations produced by our parser into KM prototypes which can be used for further reasoning in a KM-based system. To implement it in practice we are going to convert the sentences into query fragments, which, when input into the KM database, will return a prototype or an object matching the description.<sup>1</sup> A set of KM descriptions that corresponds to *load the truck with oranges* is shown in Figure 4.4.

```
e12 is (a LOAD with
 (actor _YOU123) (CARGO _oranges2) (VEHICLE _truck3))
(_oranges2 is (a ORANGE))
(_truck3 is (a TRUCK))
```

Figure 4.4: A sample description in the KM language

In the rest of the examples in this chapter, we consider how a given LF representation can be transformed to obtain a KR representation consistent with the ontology described in Section 4.2. We assume that we have the descriptions of this ontology in the TRIPS WKB and in KM which are identical except for one distinction. We will assume that the definition of KR class MOVE in the KM ontology includes `:source` and `:destination` slots to encode the source and the destination of the movement, and `:mode` slot, to encode the manner of the movement, as shown in figure 4.5. In contrast, we will say that the definition of MOVE in the TRIPS WKB language has a `:path` slot, which is a complex-valued slot which contains `:source`, `:destination` and `:mode` slots within it. We are going to use this distinction to illustrate various capabilities of our transform system, and to show how we can transform between LF and KR representations

---

<sup>1</sup>An alternative would be to generate prototypes directly, which is also possible with our approach.

when argument structures on both sides are not parallel.

```
(a) (define-class MOVE :isa ACTION
 :slots ((:vehicle VEHICLE)
 (:actor TRANS-AGENT)
 (:cargo COMMODITY)
 (:path GEO-PATH)))
 (define-class GEO-PATH
 :slots ((:source GEO-LOC)
 (:destination GEO-LOC)
 (:via GEO-LOC)
 (:mode))))
```

```
(b)every MOVE has
 (actor (a PERSON))
 (cargo (a COMMODITY))
 (vehicle (a VEHICLE))
 (destination (a GEO-LOC))
 (source (a GEO-LOC))
 (via (a GEO-LOC))
 (mode)
```

Figure 4.5: The definitions of the MOVE action in a) TRIPS WKB language b) KM language

### 4.3.2 Determining the KR type of an expression

The simplest way to define the correspondence between LF and KR types is to specify a list of mappings between them where each LF type goes to a constant KR class, *e.g.* (:\* LF::Vehicle truck) corresponds to the TRUCK class in the domain ontology. The transform in Figure 4.6 encodes that information. The syntax of the transform is domain-independent, and to adapt the LF representation to the syntax of different target languages and generate the form that can actually be used in reasoning, we augment the transforms with a procedural conversion semantics, described later in Section 4.4.

The transforms can utilize the hierarchical structure of the LF ontology to remap larger

```
(define-lf-to-kr-transform truck-transform
 :type-pattern ((:* LF::Vehicle truck) → TRUCK)
```

Figure 4.6: A Simple LF-KR transforms to establish the correspondence between the LF type `LF::Vehicle*truck` and KR type `TRUCK`.

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.

The type pattern syntax described above is sufficient to remap between all the types in the LF ontology and KR types. However, in many cases listing these type remappings may lead to unnecessary duplication. Consider a case of determining the correct class for medication names, *e.g. aspirin, tylenol, claritin*. On the LF side, there are no significant lexico-semantic differences in definitions of these words, all of them have very similar distributions, so all of them have the same LF type `LF::Drug`, and are represented as `LF::Drug*aspirin`, `LF::Drug*tylenol`, `LF::Drug*claritin`. To map these to the correct KR types, we need to list the correspondences `LF::Drug*aspirin → ASPIRIN`, `LF::Drug*tylenol → TYLENOL`, `LF::Drug*claritin → CLARITIN`. To eliminate the repetition, we can use the lexical item in the remapping.

The simplest case of using a lexical form, to handle the example above, is to declare that the KR type of an item is the same as the lexical form of the word. Two sample transform utilizing the lexical form are shown in Figure 4.7. In both cases, the transforms state that the value of the `?lf-form` variable, which stands for the lexical form part of the LF type, is to be used as the resulting KR type. For example, in case of a transform in Figure 4.7(a), when we apply it to `LF::Drug*aspirin`, it binds `?lf-form` to `ASPIRIN`, and thus determines that the resulting expression will have type `ASPIRIN`.

Since the above transform uses an unconstrained variable `?lf-form` as the resulting type, we need to provide a default value in case the result does not exist in the KR ontology. We do that in the `:default` parameter in the transform definition. For instance, the transform in Figure 4.7(a) states that the default of `?lf-form` is `SUBSTANCE`. Assume that in our ontology the medication `ASPIRIN` is defined, but the medication `ZOLOFT` is not. Then, as described above, given an entity of type `LF::Drug*aspirin`, we obtain an entity of KR type `ASPIRIN`. However, we obtain an entity of type `SUBSTANCE` from `LF::Drug*zoloft`, because `SUBSTANCE` was specified as the default value for the `?lf-form` variable, and `ZOLOFT` is not defined in the KR ontology.

```

(a)
(define-lf-to-kr-transform drug-transform-medadvisor
 :type-pattern ((:* LF::Drug ?lf-form) → ?lf-form)
 :defaults ((?lf-form SUBSTANCE)))

(b)
(define-lf-to-kr-transform geo-loc-transform-monroe
 :type-pattern ((:* LF::Political-region ?lf-form) → ?lf-form)
 :defaults ((?lf-form GEO-OBJECT)))

(c)
(define-lf-to-kr-transform geo-state-transform-monroe
 :type-pattern ((:* LF::Political-region state) → GEO-STATE))

```

Figure 4.7: Transforms using a lexical form (a) A transform for medications in the medication adviser system (b) a transform for geopolitical regions in the Monroe system, and (c) an additional transform for handling *state* in the sense *GEO-STATE*

We also use the default variable value to provide a consistency check for the result values. We require that if a default value of a variable is defined, all other values assigned to that variable should be subtypes of the default. This condition is helpful in detecting problems in transforms. Assume, for example, that in our system we only have the transform in Figure 4.7(b) defined. It says that given an entity of type *LF::Political-region*, the resulting KR type should be the same as the lexical form of the word. Thus, *LF::Political-region\*city* will be converted to an object of KR type *CITY*, and *LF::Political-region\*county* to *COUNTY*. This works well in most cases, but, as it turns out, it produces a bad result with the word *state*. Figure 4.8 shows a simplified version of a type hierarchy used in our Monroe domain. When the hierarchy was designed, the KR concept *STATE* was chosen to represent static eventualities, while the KR concept *GEO-STATE* denotes a state as a geo-political region. Now, if we we pass *LF::Political-region\*state* into a transform which uses the lexical form without additional checks, it will conclude that the corresponding KR type is *STATE*. But we know *STATE* is a kind of *EVENTUALITY*, not a *GEOGRAPHICAL-OBJECT* in that hierarchy, so this correspondence should not be made. A consistency check that requires that all result types must be a subtype of a default, *GEOGRAPHICAL-OBJECT*, will uncover the problem and signal an error, requiring that the exception rule should be defined in the system to tell it what is the correct type for *LF::Political-region\*state*.

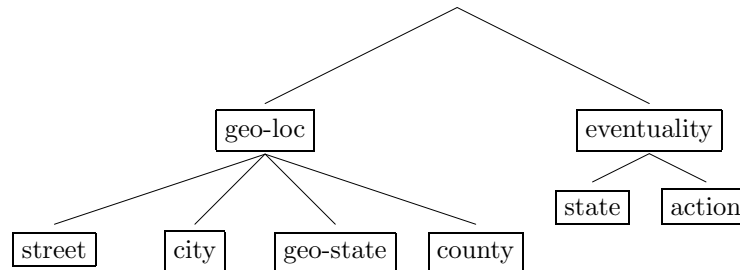


Figure 4.8: A sample hierarchical KR structure

Such exceptions in applying rules with `?lf-form` are handled through defining rules that apply to specific `lf-parent*lf-form` combinations. These are the regular rules introduced in the beginning of this chapter, for example, the `geo-state` rule in Figure 4.7(c), which says that `LF::Political-region*state` goes to `GEO-STATE`. This rule will apply in preference to the more general rule for political regions which uses the `lf-form`. This preference is the result of our transform selectional mechanism, which requires that if several possible transforms apply, transform which relies on the most specific information should be used.

The specificity is defined as follows:

- The most specific transform is the one that specifies explicitly the LF type and the lexical form of the word. If there is more than one such transform, and there are no other deciding factors (like the restrictions on arguments, described in the next section), the system will signal an error.
- The next most specific transform is the transform that specifies the LF type and uses LF form as a variable
- If neither the transform with the specific lexical form, nor the transform with the lexical form as a variable are defined, a transform which uses the lowest parent of a word is the most specific.

In our case, if for `LF::Political-region*state` `geo-state-transform-monroe` is the most specific, because both the LF parent and the lexical form specified in the transform match the type. It will be applied in preference to `geo-loc-transform-monroe`, which uses the lexical form as a variable. Thus, `LF::Political-region*state` will be correctly transformed into `GEO-STATE`. In contrast, for `LF::Political-region*city`, `geo-loc-transform-monroe` is the most specific, because there is no transform applying to both LF type `LF::Political-region` and the lexical form `city`, and `geo-loc-transform-monroe` uses LF type `LF::Political-region` and a `:lf-form` variable.

```
(define-lf-to-kr-transform load-transform-trips
 :type-pattern (LF::Filling → LOAD)
 :argument-patterns (((:agent ?a) → (actor ?a))
 (:theme ?t) → (cargo ?t))
 ((:goal ?g) → (container ?g))))
```

Figure 4.9: Transform between the domain-independent form for LF::Filling and a domain-specific LOAD action

As an additional example, in the next section, we will give examples where transforms are defined on LF type LF::Motion. If we take a word LF::Removing\*load, in our system there is no transform defined which uses the LF type LF::Removing. Therefore, the most specific applicable transform is the one which uses its parent, LF::Motion.

### 4.3.3 Transforming semantic arguments

Once we have determined the KR type that corresponds to a given LF type, we need to specify how the semantic arguments and modifiers associated with that LF type are going to be matched with the KR representation. First, we discuss the simplest case, when we transform only the internal semantic arguments associated with a given LF type. Next section will deal with more complex transformations where the adjuncts in the LF representation are involved.

One of the most frequent cases in writing transforms is when a set of semantic roles goes to a set of KR slots, *e.g.*, :theme becomes *cargo*, :instrument becomes *vehicle* and so on. An example of such transform for LOAD actions in the Pacifica domain is shown in Figure 4.9. It maps any LF representation of type LF::Filling into an instance 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).

The example above is the simplest possible in our system, and the variable names in role mappings may appear superfluous. This is not the case, as we will show in the next section with more complex transforms.

As a rule, all argument mappings specified in transforms are optional. For example, the :goal argument may or may not be filled explicitly in the sentence. For example, both *Load the*

*oranges into the truck* and *Load the oranges* are valid utterances, but the latter has an implicit destination value which is derived from context. The transform we showed earlier will still apply in both cases to the semantic arguments present in the form, and leave it to the discourse interpretation to fill the missing pieces. 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 4.2). 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 4.10(a), which specifies that the `:Theme` semantic argument that fills the *cargo* slot is required for the transform to apply.

```
(a) (define-lf-to-kr-transform transport-transform
 :preconditions ((:obligatory theme))
 :type-pattern (LF::Motion → TRANSPORT)
 :argument-patterns (((:agent ?a) → (actor ?a))
 ((:theme ?t) → (cargo ?t))
 ((:instrument ?vh) → (vehicle ?vh))))

(b) (define-lf-to-kr-transform move-transform
 :type-pattern (LF::Motion → MOVE)
 :argument-patterns (((:agent ?a) → (actor ?a))
 ((:theme ?t) → (vehicle ?t))))
```

Figure 4.10: The transforms for (a) *TRANSPORT* and (b) *MOVE* actions in TRIPS Pacifica domain. The precondition limits the *TRANSPORT* transform application only to the cases when a cargo slot can be filled.

Figure 4.10 lists two transforms which apply to the same LF type `LF::Motion`, in order to differentiate the cases when motion verbs will be interpreted either as *MOVE* or *TRANSPORT* actions in the Pacifica domain. The transform system relies on the preconditions and KR type restrictions to choose the correct transform. Consider three sentences: *Go to Avon*, *Send a truck to Avon* and *Send the oranges to Avon*. In the first case, `:theme` argument is not filled explicitly, therefore, `transport-transform` is not applicable, because it requires that `:theme` be present as a precondition; `move-transform` is applicable because *theme* is optional there. In the second

case, the `:theme` is present. However, *truck* is not a suitable filler for the *cargo* slot of `MOVE`.<sup>2</sup> Thus again `transport-transform` is not applicable, but `move-transform` is, because *truck* can fill the *vehicle* slot. In contrast, `move-transform` is not applicable to our last case, because the `:theme` filler *oranges* cannot fill the *vehicle* slot; `transport-transform` will be used because `:theme` is present and consistent with the restrictions on the *cargo* slot.

Another common case for transforms is when a group of semantic arguments from the LF representation needs to be combined into a complex-valued slot (in the KM language) or in an external modifier (in TRIPS-WKB language). For example, in the Pacifica domain, the sentence *leave Abyss for Delta* will represent *Abyss* as a `:From-loc` argument, and *Delta* as a `:To-loc` semantic argument. As we discussed earlier, in our TRIPS WKB representation the `MOVE` class has a single complex-valued slot, *Path*. It has to be filled with a frame of type `GEO-PATH`, which itself is a class with multiple slots. To implement such transform, we need a special operator to instantiate new classes. An example is shown in Figure 4.11. The special variable `*1` of instantiating a slot value. During transform application, we check whether *path* slot is already filled. If it is, then `*1` will be bound to the frame which fills the slot. If not, we will create an instance of `GEO-PATH` object, because the definition of `MOVE` (shown in Figure 4.5(a)) specifies it as the KR type associated with the *path* slot of `MOVE` in TRIPS-WKB. This allows us to write multiple rules as necessary - the same transform in Figure 4.11 could be written as three separate rules transforming `LF::From-loc`, `LF::To-loc` and `LF::Via` slots, and the semantics of `*1` is such that `SOURCE`, `DESTINATION` and `MID-POINT` predicates in effect collect all path arguments into a single description.

The transform in our example is declared as `abstract`. This means that it is not applicable unless some more general transform applied to the relevant LF term (`LF::Motion`) to transform the rest of the arguments. This is important, because the rest of the arguments can be coming from different places. Depending on the motion verb, *vehicle* may be coming either from an `:instrument` slot, as in *The truck should take the people to Avon*, or from the `:theme` slot, as in *Send the truck to Avon*, and we have different transforms to accomplish this re-mappings. Yet the same transformation is applied to *to Avon* independent of how the *vehicle* slot was filled. To avoid unnecessary duplication (which makes system maintenance considerably more difficult), we have a separate transform for paths, but declare it as `abstract`, so that it applies only in combination with some other motion transform. We will return to discussing tradeoffs involved in declaring a transform `abstract` again, when we discuss a very general transform for

---

<sup>2</sup>This is determined by the transform of *truck* which assigns KR type `TRUCK` to it, and the restriction that the *cargo* slot of `MOVE` must be filled with `COMMODITY`.

```

(define-lf-to-kr-transform path-transform-trips
 :abstract t
 :type-pattern (LF::Motion → MOVE)
 :argument-patterns ((NIL → (path *1))
 ((:from-loc ?fl) → *1 (source ?fl))
 ((:to-loc ?tl) → *1 (destination ?tl))
 ((:via ?va) → *1 (mid-point ?va))))

```

Figure 4.11: A transform to collect all path adverbials into a single path frame in the Trips Planner language. \*1 denotes an operation of creating a new variable for the path frame.

transforming *red* into a color slot on a physical object.

Note that the transform shown in Figure 4.11 is somewhat of a simplification. It applies directly to LFs from verbs such as *leave*, for example, *The train left Avon* for which `:from-loc` and `:to-loc` are subcategorized arguments and not adverbial adjuncts. In most cases, however, `:From-loc` and `:To-loc` are not simply slots in the logical form, they are coming from adverbials and therefore have a more complex representation. We have two options in dealing with those cases. First is finding a way to establish a regular relationship between an expression which describes a slot and the corresponding adverbial. We will discuss how this can be done later, in Section 4.4.2. Another option is to match directly on the logical form produced by the adverbials and other external LF modifiers. This will be the focus of our next section.

#### 4.3.4 Dealing with external LF modifiers

The assumption in the previous section was that we are dealing with transform arguments coming only from the internal LF arguments, represented as semantic argument values in our LF representation. This does not cover a large number of cases when the arguments to the LF are coming from external modifiers, namely, adverbials and adjectives, and for which the parser generates the the complex logical form we discussed in Section 3.4.3. To summarize, the parser creates a LF representation in which every adverbial modifier is a separate LF term modifying the main phrase. This is necessary because adjectives and adverbials can themselves be modified by other adverbs. In this section, we will discuss how different types of external modifiers are handled during the transform process.

Our examples will deal with logical forms for three expressions, *Go to Avon*, *a red truck* and *Go straight to Avon*. In the first case, we have a “binary” modifier, that is a word *to* which has two arguments: the city to which the movement is directed, and the movement action. In the second case, the modifier *red* is unary: it modifies a noun *truck*, but in itself it does not create an entity which can be used to fill a slot in the KR representation. In the third case, which is the most complex, the adverb *straight* modifies the adverbial *to Avon*, and we are interested in how the complex LF representation in this case can be transformed in a variety of domain-specific interpretations possible for this utterance.

### Binary Modifiers

Consider our example *Go to Avon*. Its LF representation is shown in Figure 4.12, and the phrase *to Avon* is represented by a separate LF::To-loc term listed on the :MODS list for LF::Move\*go. If the KR Ontology has a similar structure, then all we need to do is to write a regular transform which will, for example, map the LF::To-loc predicate to the DESTINATION-LOC predicate on the KR side. However, in most cases of domain-specific languages, and in particular in the Trips WKB and KM languages we are using, the structure is different. In KM, *destination* is a slot in the MOVE frame. In Trips WKB, *destination* is a slot in the complex GEO-PATH predicate filling the *path* argument of MOVE. Thus, we need a way to convert the complex argument structures from the LF representation into a different set of (possibly) complex argument structures in the target KR representation.

```
(F E1 LF::Move*go :MODS (V1234))
(F V1234 (:* LF::to-loc to) :of E1 :Val a456)
(F a456 (*: LF::Political-region city) :name Avon)
```

Figure 4.12: The LF representation for *Go to Avon*

Figure 4.13 shows transforms which are applied to adverbial modifiers in both languages. For KM, the transform states that if a LF::Motion term is transformed into a MOVE frame, then the filler of the :Val slot (the variable ?t1) in the LF::To-loc modifier will fill the *destination* slot on the MOVE frame. So when that argument pattern is matched against the logical form in Figure 4.12, ?t1 is bound to *Avon*, which will be the filler for the *destination* slot in the resulting MOVE frame.

The transform for TRIPS-WKB language, shown in Figure 4.13(b) is an extended version of the transform in Figure 4.11 used for matching on the adverbial modifiers. It uses the same

syntax to instantiate the *Path* slot in the frame it modifies, but uses a more complex expression on the left-hand side of the argument to match against the adverbial.

(a)

```
(define-lf-to-kr-transform to-loc-transform-KM
 :abstract t
 :type-pattern (LF::Motion → MOVE) :lf-var ?vv
 :argument-patterns (((F ?tl LF::To-loc :Of ?vv :Val ?g) → (destination ?g))))
```

(b)

```
(define-lf-to-kr-transform to-loc-transform-trips
 :abstract t
 :type-pattern (LF::Motion → MOVE) :lf-var ?vv
 :argument-patterns ((NIL → (path *1))
 ((LF::To-loc ?tl :OF ?vv :Val ?tl) → *1 (destination ?tl)))
```

Figure 4.13: Transforming the LF::To-loc modifier (a) The transform for *go* and LF::to-loc adverbial together in the KM language (b) the same transform in the TRIPS-WKB language

While the expressions in the transforms match closely with the LF syntax, there are several special features which need to be implemented in order to make them match against the logical forms. First, all the transforms here apply in the context of a wider transformation which transforms a LF::Motion event into an instance of MOVE. This is important, because, for example, the filler of the :Val argument in LF::To-loc goes to a *destination* slot in a MOVE frame, and we need to make sure that such frame and slot indeed exist. Moreover, our LF expressions are just lists of terms, so for a complex enough sentence there may be LF::To-loc terms in our logical form that modify something else in the sentence (and therefore may need to be transformed differently). Therefore, the matcher needs to know how to verify that a given adverbial modifies the relevant LF term. This is achieved by stating that the value of the :of argument in the LF::To-loc term should be bound to the variable variable of the main “context” term, declared as `:lf-var ?vv`.

Secondly, these expressions are not exact matches in the sense that they must allow for additional possible arguments in the LF representations. This is necessary to correctly handle additional modifiers possible in language. For example, in the sentence *Go straight to Avon*, since *straight* modifies *to*, the corresponding LF::To-loc term will have a :mods list among its arguments. It does not change the fact that *Avon* needs to fill the *destination* slot, and therefore

it is not relevant to our current transform. Thus, the matcher needs to ignore any additional arguments which may be present in the LF term. This will become even more important later, when we talk about transforming the adverbs which themselves modify other adverbials.

Putting it all together, let's consider an example of how LF::Go to Avon is transformed to a corresponding KM representation. We will do the TRIPS WKB example later, when we talk about the details of the transform application algorithm. Figure 4.14 is a complete example of a logical form for *go to Avon*, the transforms which apply to it, and the resulting KM representation. To transform the form in Figure 4.14(a), we first look for the transform to transform the LF::Move term, and apply `move-transform` shown in Figure 4.14(b). It transforms the :agent argument into the *actor* slot in the KR representation. The remaining untransformed argument is LF::To-loc\*to. The matcher looks up the transforms which are related to LF::To-loc\*to, finds `path-transform-km`, shown in Figure 4.14(c). This transform essentially states that it would be applicable if there is a LF::Motion term, which was transformed into an instance of MOVE frame (this is specified by `:type-pattern`), and the LF::To-loc argument modifies this frame (specified by declaring `:type-var ?vv` and the `:Of ?vv` argument in the argument pattern). Both of these are true in our case, thus the transform is applied and used to find the filler of the *destination* slot in the MOVE frame. The resulting representation is shown in Figure 4.14(d).

Notice that we set up our transforms so that they can be easily extended to account for additional modifiers. Consider the case of *Go straight to Avon*. Its logical form representation is very similar to *Go to Avon*, but *to* will have a term in its :mods list for an additional modifier, *straight*. The entire set of transforms from our example will apply, and transform everything except the term for *straight*. So we need to write only one more transform for that term to handle it. We discuss how to do it later in this section, after we deal with a simpler case of unary modifiers.

### Unary modifiers

In addition to dealing with binary modifiers, the system needs to know how to transform unary modifiers coming from most adjectives and adverbs, such as *red* or *quickly*. As we noted earlier in this section, in contrast to PP adverbials these don't have an internal argument such as *Avon* which could fill a slot in the resulting representation. Instead, when they are transformed, there will usually be a constant value to fill in the slot in the result. For example Figure 4.15(a) contains the LF for *the red truck*. In our Pacifica KR representation class TRUCK has a `Color`

- (a) (F E1 LF::Move\*go :agent pro1 :MODS (V1234)  
 (IMPRO pro1 LF::Person :context-rel \*YOU\*)  
 (F V1234 (:\* LF::to-loc to) :of E1 :Val a456)  
 (F a456 (\*: LF::Political-region city) :name Avon)
- (b) (define-lf-to-kr-transform move-transform-trips  
 :type-pattern (LF::Motion → LOAD)  
 :argument-patterns (((:agent ?a) → (actor ?a))  
 ((:theme ?t) → (vehicle ?t))))
- (c) (define-lf-to-kr-transform to-loc-transform-KM  
 :abstract t  
 :type-pattern (LF::Motion → MOVE) :lf-var ?vv  
 :argument-patterns (((F ?tl LF::To-loc :Of ?vv :Val ?g) → (destination ?g))))
- (d) (E1 is (a MOVE with (actor pro1) (destination a456)))

Figure 4.14: Transforming the utterance *go to Avon* (a) The LF representation; (b) The main transform for LF::Move\*go; (c) The applicable transform for LF::To-loc in context of LF::Motion; (d) The resulting KM representation

slot that needs to be filled with a constant value RED. This is handled with the transform shown in Figure 4.15(b), which says that the lexical form of the LF::Color modifier should be used as a slot filler for the *color* slot associated with classes of type PHYS-OBJECT.

Note that this transform utilizes the hierarchical properties of the LF ontology, saying that it applies to any LF object modified by the LF::Color modifier, and that it will be transformed in an instance of a PHYS-OBJECT frame with a *color* slot. In practice, this transform is used in conjunction with the main transform that applies to trucks and determines the actual KR type of the resulting expression, TRUCK in this instance. In addition, the transform in this example has been declared as **:abstract t**. This means that the transform won't be applied unless there is a more specific transform applicable to the main type.

It is an application-specific decision whether to declare transforms as abstract. The problem here arises because the transform in our example basically says that anything modified by color should be an instance of PHYS-OBJECT, but we cannot specify the type more explicitly in this transform. Now, assume we have a phrase *a red rocket* and we don't know how to transform *a rocket*. If this transform were not declared as abstract, and we applied it, the KR representation would be (V2 is (a PHYS-OBJ with (color RED))). This is significant loss of information,

```

(a)
(A V2 LF::Truck :MODS (V7890))
(F V7890 (:* LF::Color red) :of V2)

(b)
(define-lf-to-kr-transform color-transform-KM
 :abstract t
 :type-pattern (LF::Any-sem → PHYS-OBJECT) :type-var ?vv
 :argument-patterns (((F ?cl (:* LF::Color ?cform) :Of ?vv) → (color ?cform))))

(c)
(V2 is (a TRUCK with (color RED)))

```

Figure 4.15: Transforming a LF with a unary modifier. (a) The LF representation for *a red truck* (b) the transform for colors; (c) the resulting KM representation

and it depends on the capabilities of the rest of the system whether it is acceptable. It is possible that an interpretation manager knows how to handle such underdetermined objects and can therefore ask a clarification question to determine the more exact type. In our current implementation, however, this is not the case. Our reference resolution does not have a mechanism to detect that the type is seriously underconstrained, and, when presented with a form like that, will return some referent consistent with KR type PHYS-OBJECT, for example, a truck. This is clearly unhelpful, and therefore we decided that until a better way to handle these constructions can be found, the transform should be declared as abstract, and not used unless a more specific type determination can be made.

### Complex adverbial modifiers

We now go back to the question of transforming complex adverbial modifiers. One property that differentiates adverbials and other external modifiers from the internal semantic arguments in our LF representation is that adverbials can themselves be modified with additional modifiers, for example, *straight to Avon*, where *straight* modifies *to*, as shown in Figure 4.16.

The exact treatment of these cases depends on the capabilities of the target KR language. The mechanisms discussed above cover a variety of possibilities. Table 4.2 summarizes the different ways the same sentence, *Go straight to Avon*, can be represented in different systems. These choices are the summary of different possibilities we encountered in our work on the TRIPS system, shown as applicable to the same form for illustration purposes. In the rest

```

(F v123 (:* LF::Move go) :agent y1 :MODS (t1))
(IMPRO y1 LF::Person :CONTEXT-REL *YOU*)
(F t1 (:* LF::To-loc to) :of v123 :val a1 :mods (s2))
(The a1 (LF::Political-region city) :name-of AVON)
(F s2 (:* LF::Direction straight) :of t1)

```

Figure 4.16: A logical form for *Go straight to Avon*.

of this section we are going to show how they can be used to obtain different representations depending on the choices made for KR ontology design.

| Representational choice                                             | KR representation for <i>Go straight to Avon</i>                                                        |
|---------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|
| (a) Represent modifiers as slots in the complex frame for adverbial | (TYPE e1 MOVE) (path e1 p123)<br>(type p123 GEO-PATH) (mode p123 DIRECT)<br>(destination p123 Avon123)) |
| (b) Represent (manner) modifiers as slots in the main frame         | E1 is (a MOVE with (mode DIRECT)<br>(destination Avon123))                                              |
| (c) Ignore the modifiers completely                                 | E1 is (a MOVE with (destination Avon123))                                                               |

Table 4.2: Choices in representing modifiers to adverbial adjuncts. (a) In TRIPS WKB, *straight* fills a slot in the *path* frame; (b) In KM, *straight* fills a slot in the main MOVE frame; (c) In some cases, modifiers may be ignored, we use KM syntax as an example

When the adverbials in question are interpreted as complex frames, the easiest treatment for additional modifiers is to see them as additional slots in the corresponding frames, as shown in Table 4.2(a). The example assumes the TRIPS WKB representation, which re-maps the adverbial modifiers on paths to slots in GEO-PATH frames. In this case, *straight to Avon* will be interpreted as a complex GEO-PATH type with *destination Avon* and *direction straight*, as shown in Figure 4.17.

This transform has two levels of indirection: it applies to the LF::Direction modifier, which modifies the LF::To-loc modifier, which itself modifies the LF::Motion frame. Note that it is used only to transform *straight*, and in transforming *straight to Avon*, it has nothing to say about the mappings of *to Avon* itself. They are accomplished with the path transforms discussed earlier in this chapter. This is an important feature, because there may be different treatment

```

(define-lf-to-kr-transform direction-transform-trips
 :abstract t
 :type-pattern (LF::Motion → MOVE) :type-var ?vv
 :argument-patterns
 ((NIL → (path *1))
 ((F ?tl LF::To-loc :Of ?vv) (F ?dr (:* LF::Direction straight) :Of ?tl)
 → *1 (mode DIRECT)))

```

Figure 4.17: Transform to move the external modifiers into the slot of a GEO-PATH frame attached to MOVE

for different modifiers: we may transform *straight* into a slot filler, but we may want to ignore (or transform in some other way) *up* in *Go up to Avon*.<sup>3</sup> Therefore, the transforms for *straight* and *up* should be independent of the transform for *to Avon*, in order to avoid unnecessary duplication and simplify system maintenance.

This is the place when partial matching on the LF form representation we mentioned earlier plays an important role. In the real LF representation for *go straight to Avon*, the LF::To-loc adverbial will also have a :Val argument filled with *Avon*. We do not include it here, because it is not relevant to the purpose of this transform. Instead, we have a matcher which knows about the way LF form is composed from a set of arguments, and matches by checking that all arguments mentioned in the transform (only :Of in this case) are filled, and ignores any other arguments present in the LF representation. This helps us to easily combine this transform with a more general transform we discussed earlier, which applies to all paths regardless of whether they have additional modifiers.

Another option, from Table 4.2(b), is to interpret external modifiers as slot values in the main KR frame. For example, in our KM Pacifica ontology, *go straight to Avon* can then be interpreted as MOVE with *destination Avon* and *mode DIRECT* to indicate the “movement mode”, as shown in Figure 4.18.

Finally, it is possible that the system cannot deal with finer distinctions and wants the modifiers are discarded. For example, the system router may not care about the difference in *to Avon* and *straight to Avon*, and choose to ignore the modifier, as shown in Figure 4.19.

---

<sup>3</sup>There is the question here of whether *down* modifies *to* or really *go*. We found that this was not an unreasonable representation, to uniformly handle other cases such as *send an ambulance up along Mt.Hope*. Therefore we assume that this is a correct representation for purposes of this example.

```
(define-lf-to-kr-transform direction-transform-KM
 :abstract t
 :type-pattern (LF::Motion → MOVE)
 :argument-patterns
 (((F ?tl LF::To-loc :Of ?vv) (F ?dr (* LF::Direction straight) :Of ?tl))
 → (mode DIRECT))))
```

Figure 4.18: Transform to move the external modifiers into the MOVE frame

```
(define-lf-to-kr-transform direction-transform-ignore
 :abstract t
 :type-pattern (LF::Motion → MOVE) :type-var ?vv
 :argument-patterns
 (((F ?tl LF::To-loc :of ?vv) (F ?dr (* LF::Direction straight) :of ?tl) → NIL)))
```

Figure 4.19: Transform to ignore the direction modifiers

As this section shows, many complex cases of LF to KR transformation are supported straightforwardly in our system, using a small set of standard tools: argument mappings, instantiating nested expressions with `*` variables, and using lexical forms of the word. We simplified our examples of handling *straight to Avon* a little by saying that they apply only to `LF::Direction*straight` modifier, which is transformed into `:mode direct` slot. For a more general treatment, we can use the lexical form of a modifier exactly as it would be used in regular transforms.

Thus far, our transforms cover all major cases we have seen in our systems when argument structures of the LF and KR ontologies diverge. However, we have only actively used the TRIPS WKB language in the system. We are currently working to provide a KM implementation, based on the procedural transform semantics discussed in the next section.

## 4.4 Implementing the transforms

So far, we have discussed only how to establish the correspondence between LF and KR types and their argument structures using LF-KR transforms. This part of the transformation between LF and KR representation is the most difficult part of the task, and we express it in a way inde-

pendent of the syntax of the actual KR language. Once the correspondence between the types and arguments has been established, we need to actually build the resulting KR representation using the syntax of the target KR language.

In the implementation used for evaluation later in this chapter, we use a simpler form of the transform syntax with a built-in processing algorithm that knows how to transform between the LF syntax and either a simple frame-based language or a predicate logic [Dzikovska *et al.*, 2002]. We have also proposed using more general production rules to transform directly into the syntax of the target language [Dzikovska *et al.*, 2003]. The former implementation was too dependent on the specifics of the TRIPS knowledge representation syntax. Using a production rule system was our attempt to provide a better solution, but on additional analysis we discovered that production rules worked better for KM, which permits multiple assertions about the objects, but not so well with the TRIPS WKB representation. TRIPS WKB requires that all relevant information about an object be collected in a single conjunction, and therefore we needed a single production rule to transform all possible arguments, a setup not convenient for transform writing.

For a more general solution, we present here a formalized model which defines the basic operations performed during the transformation, and thus provides a procedural semantics for the transform syntax we describe in this chapter.

#### 4.4.1 Transform application algorithm

Figure 4.20 shows the BNF for transform syntax. These main things happen during the transformation:

- An object of the type specified by  $\langle \text{KR-type} \rangle$  in the `type-pattern` is created on the KR side
- We transform each argument from the LF representation in turn. To do this, we match the arguments in the LF representation against the patterns from `:argument-pattern`.
- If a pattern  $\langle \text{Transform-argument-pattern} \rangle$  matches any of the arguments on the LF side, a corresponding argument is added to the KR representation being built
- We view the KR representation being created as a tree rooted in the main node of type  $\langle \text{KR-type} \rangle$ . If during the transform application we see a  $\langle \text{Create-argument-pattern} \rangle$ , which is a pattern with NIL on the left-hand side, we create a subnode, which can be

either a built-in complex slot in the KM representation, or a subformula in the TRIPS WKB representation. We assign to this subnode a unique identifier  $\langle \text{s-var} \rangle$ . Then, if any  $\langle \text{Transform-argument-pattern} \rangle$  specifies  $\langle \text{s-var} \rangle$ , we add the resulting KR argument not to the root node, but to the node specified by  $\langle \text{s-var} \rangle$ .

The BNF syntax we show here is fairly straightforward. The most complex expression is the one which describes the complex F arguments we use to match adverbial modifiers. The syntax reflects the fact that there can be an arbitrary number of F expressions which form the link between the main LF term and the modifier, and only one  $\langle \text{Match-F-Argument} \rangle$  with two instances of  $\langle \text{LF-slot-argument} \rangle$ . One is to provide the link to the LF term the adjunct modifies, and another is to identify the semantic argument which will be mapped into a slot.

$$\begin{aligned} \langle \text{Transform} \rangle &::= (\text{define-lf-kr-transform } \langle \text{transform-name} \rangle \\ &\quad [ \text{:abstract t} ] \\ &\quad \text{:type-pattern } (\langle \text{LF-type} \rangle \rightarrow \langle \text{KR-type} \rangle) \\ &\quad [ \text{:argument-patterns } (\langle \text{Argument-Pattern} \rangle+) ]) \\ \\ \langle \text{Argument-Pattern} \rangle &::= \langle \text{Create-argument-pattern} \rangle | \langle \text{Transform-argument-pattern} \rangle \\ \langle \text{Create-Argument-pattern} \rangle &::= \text{NIL} \rightarrow [ \langle \text{s-var} \rangle ] (\langle \text{KR-Argument-name} \rangle \langle \text{new-s-var} \rangle) \\ \langle \text{Transform-argument-pattern} \rangle &::= \langle \text{LF-Argument} \rangle \rightarrow \langle \text{Kr-argument} \rangle \\ \\ \langle \text{LF-Argument} \rangle &::= \langle \text{LF-Slot-argument} \rangle | \langle \text{F-argument} \rangle \\ \langle \text{LF-slot-argument} \rangle &::= (\langle \text{Sem-argument-name} \rangle \langle \text{Filler-var} \rangle) \\ \langle \text{F-Argument} \rangle &::= \langle \text{Link-F-Argument} \rangle^* \langle \text{Match-F-Argument} \rangle \\ \langle \text{Link-F-Argument} \rangle &::= (\text{F } \langle \text{Mod-var} \rangle \langle \text{Sem-argument-name} \rangle \langle \text{LF-slot-argument} \rangle) \\ \langle \text{Match-F-Argument} \rangle &::= (\text{F } \langle \text{Mod-var} \rangle \langle \text{Sem-argument-name} \rangle \\ &\quad \langle \text{LF-slot-argument} \rangle [ \langle \text{LF-slot-argument} \rangle ]) \\ \\ \langle \text{KR-Argument} \rangle &::= [ \langle \text{s-var} \rangle ] (\langle \text{KR-argument-name} \rangle \langle \text{Filler-var} \rangle) \textit{mid} \text{NIL} \\ \langle \text{s-var} \rangle &::= * \langle \text{node-id} \rangle \end{aligned}$$

Figure 4.20: The BNF for lf-kr transforms

The formal set of operations which have to be defined on the KR representation to implement the transform application algorithm is summarized in Table 4.3. A general algorithm for applying

the transforms is shown in Figure 4.21, and we will illustrate its implementation here with TRIPS-WKB and KM examples.

|                                        |                                                                                                                                                                                                                            |
|----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Create-object(type, lf-var)            | Create an object of a given type, and establish its correspondence with the variable on the LF side. Return the identifier for the new node.                                                                               |
| Establish-object (node, argument-name) | If an argument with a given name is defined in the node, return the node identifier for this argument. Otherwise, create a new node, and fill the given argument with a newly instantiated object of the appropriate type. |
| Add-argument(node,arg-name,arg-value)  | Add an argument with a given name and value at the level specified by the node.                                                                                                                                            |

Table 4.3: The abstract operations in the transform application algorithm

First, let us consider the TRIPS-WKB example. Assume we are given a logical form for *Send a truck to Avon*, shown here in Figure 4.22(a). We look for the most specific transform which applies to LF::Move\*send and find `move-transform-trips-or-km`, shown in Figure 4.22(b). Now we start applying the transform. The call to *Create-object* starts the process, by creating a new KR object of type MOVE: (AND (Type E1 move)). Now we proceed to applying argument transforms. `:agent ?a` matches the LF, with ?a assigned to `*YOU*`. There is no explicit variable in the transform, thus we are adding an argument to the top-level node, and we get (AND (Type E1 move) (Actor E1 \*you\*)) Similarly, we match on `:theme` and get (AND (Type E1 move) (Actor E1 \*you\*) (Vehicle E1 truck123)).

The application of `move-transform-trips` is completed, but there is one more argument which has not been transformed - LF::To-loc\*to. Therefore, we continue looking for more general transforms which apply in this case, and we find `path-transform-trips` shown earlier in Figure 4.11. It is an abstract transform, so we don't apply the type pattern, it is only used to check that the transform is applicable. We start applying argument transform patterns from `move-transform-trips`. The first one in the list starts with NIL, which means that *establish-object(E1,path)* will be called. This will add a new node (AND (type newvar1 Path)), establish the correspondence between `*1` and the newly created object, and add the path slot to the current KR representation:

```
(AND (Type E1 move) (Actor E1 *you*) (Vehicle E1 truck123) (path E1 newvar1).
```

Finally, we apply the transform for `:to-loc` argument. It matches (not a direct match, but we will return to it below), and ?t1 is assigned value A456. The argument transform specifies `*1`,

```

Given: A transform T and a logical form lf
if matches(lf , '?spec ?ev ?LF-type') then
 if not(abstract(T)) then
 Root-kr-object := Create-object(LF-type)
 endif
 foreach Argument-pattern in Argument-patterns(T)
 if (LF-Argument-Pattern is NIL) then
 /* Create-argument-pattern - make a new subformula */
 vars[new-s-var] = Establish-object(s-var,KR-argument);
 elseif matches(lf , '?sem-argument-name ?filler-var') then
 /* Transform-argument-pattern which matches the LF
 - add the appropriate argument to the KR side */
 Add-argument(s-var, kr-argument-name, filler-var);
 endif
 endfor /* Done looping over transform arguments */
endif

```

Figure 4.21: The algorithm for transforming LF into KR representation

```

(a)
(F E1 (:* LF::Move send) :Agent *YOU* :Theme truck123 :MODS (V1234))
(F V1234 (:* LF::to-loc to) :of E1 :Val A456)
(F A456 (:* LF::Political-region city) :name Avon)

(b)
(define-lf-to-kr-transform move-transform
 :type-pattern (LF::Motion → MOVE)
 :argument-patterns (((:agent ?a) → (actor ?a))
 ((:theme ?t) → (vehicle ?t)))

```

Figure 4.22: (a) A logical form for *Send a truck to Avon* (b) The applicable transform

so we add `(destination A456)` to the PATH conjunction identified by `*1`. None of the other argument transforms match, and there are no other arguments in the LF form, so the final TRIPS WKB representation is shown in Figure 4.23.

```
(AND (Type E1 MOVE) (Actor E1 *you*) (Vehicle E1 truck123) (path E1 newvar1)
(AND (Type newvar1 PATH) (Destination newvar1 A456))
```

Figure 4.23: The KR form for *Send a truck to Avon* in TRIPS WKB language after all possible transforms were applied

Let us consider how the same logical form will be converted into KM language. First, we apply `move-transform-trips`. We are applying the type pattern, and a call to `Create-object(MOVE)` results in the assertion `E1 is (a Move)`. We then start applying the transforms. The calls to `add-argument` make additional assertions about the slots of `E1`, so we end up with `E1 is (a Move with (Actor *you*) (Vehicle truck123))`

Unlike in our TRIPS-WKB representation, in our KM representation source and destination are not collected into a special PATH object, they are just defined as slots on the MOVE action, we add the `Destination` slot directly to the top-level node, and the obtain representation shown in Figure 4.24.

```
E1 is (a MOVE with (Actor *you*) (Vehicle truck123) (Destination A456))
```

Figure 4.24: The KR form for *Send a truck to Avon* in the KM language after all possible transforms were applied

As long as we can define the three operations specified in this section, we can re-map the LF ontology into an arbitrary KR. These operations are in fact quite general. The optional variables in the argument transforms identify “nodes” in the tree of formulas being built with transforms, and allow us to build formulas of arbitrary depth. There is also little syntax dependency. KM is a purely frame based language. TRIPS-WKB is FOPC based, though in the examples all predicates had just one argument, and we used it to simplify processing. In complete FOL based language, each predicate would be a node, and “argument names” will be the positions of the arguments in the predicate argument list (argument name “1” for the first argument, “2” for the second argument, *etc* ), so we can use this system to transform to first order logic just as well as to the frame-based languages.

We mentioned earlier that we chose this two-level architecture with separate type and syntax transformation over a generic production rule architecture. This is because the split between

type mapping and syntax transformation makes it easier for us to utilize the hierarchical structure of the LF ontology in the mappings. For example, we would like to combine the more specific information in the mapping between LF::Motion and TRANSPORT which applies only when :theme is present (shown in Figure 4.10) with the more general mapping between :to-loc and *destination* slots in our path transform, which applies to all motion verbs. This information is easier to combine when the contributions of different LF arguments to the final KR representation are separated out, rather than when they are mixed together with syntactic transformations in the final form. Then these transforms can be combined on the more abstract level, and the final form obtained with the syntax transform described in this section.

One possibility we have not considered in our transform algorithm is that the additional modifiers such as *straight* can be represented as operators or functional terms in the KR representation. This is an important question for transforming into FOL-like representations which allow operators. We plan to develop a scheme to include functional terms in our mappings as part of our future work.

#### 4.4.2 Matching on adverbial modifiers

In the previous exposition, we used the transform `path-transform-trips`(Figure 4.11). It specifies that :to-loc maps to a *destination* slot, but in our examples we used it to match on the complex argument (F v123 (:\* LF::To-loc to)). The issue here arises because we have two possible representations for semantic arguments coming from adverbials or subcategorized arguments (discussed in detail in Section 3.4.3). If :to-loc is realized as a subcategorized argument, as in *leave Delta for Abyss*, we can treat it as an internal slot with a value which is a variable ?s referring to the source slot filler *Delta*. If this is the case, `path-transform-trips` applies directly. Alternatively, in sentences like *go from Delta to Abyss*, LF::From-loc is treated as a complex-valued modifier coming from an adverbial, and the variable ?g is located in the :Val slot of the complex expression. This is exemplified by the transform `path-transform-km` in Figure 4.13. The problem is then how to reconcile these two representations to use one transform if possible, so that developers can be insulated from such specific details of the LF syntax.

We can conceptualize the knowledge about the transform as separated into two main sub-parts: given a semantic argument, either a internal semantic slot or an adverbial modifier, we need to know what is the corresponding slot on the KR side; and if there are modifiers, we need to determine what to do with them. The former question can be answered with the help of slot equivalence macros. These macros express the basic knowledge about the semantics of the

LF hierarchy: for example, if we have LF::To-loc coming as an adverbial, the value of LF::Val argument will be exactly the same as when LF::To-loc is a simple slot with a single variable value. This follows directly from the semantics of LF::To-loc, for which LF::Val encodes the destination. We declare that fact in a macro shown in Figure 4.25.

```
(define-slot-macro to-loc-macro :type-var ?vv (:to-loc ?tl) → (F ?tr LF::To-loc :Of ?vv :Val ?tl))
```

Figure 4.25: The slot equivalence macro for LF::To-loc.

Given this equivalence macro, the developer now has two choices. If the modifiers are always to be ignored, or unlikely to exist (*e.g.*, for LF::Instrument slot, which can come from the PP adjunct *with*, but which is not going to have any extra modifiers), the simplest syntax, similar to Figure 4.11 can be used, and the more complex rule will be inserted automatically based on the macro. If the modifiers need to be processed, a more complex syntax corresponding to the external modifier case can be used, and the simpler rule will be derived automatically based on the slot macro.

Matching on adverbial modifiers in our logical form is a complex problem, with multiple possible solutions. Our current implementation does not yet use the declarative representations described here. We have a matcher that matches directly on the representations that would be produced by these macros. Here we presented an improved design which offers the advantage of declaratively encoding much of the knowledge about the LF form structure, and the relationships between different pieces. The slot macros are only dependent on the domain-independent semantics of an adverb. For example, we know that :Val for LF::To-loc is the place to which the trajectory is directed, and :of is the action whose trajectory it is. Thus, it follows directly that it is the value of :Val which is the filler of the semantic argument :to-loc on an action. This knowledge do not change between domains, so the slot macros are properly part of the LF Ontology and can be attached to it.

## 4.5 Lexicon Specialization

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.<sup>4</sup>

We illustrate the lexicon specialization process with the verb *load*. Given the definitions of *load* and LF::Filling in Figure 3.3, and the transform definitions in Figure 4.9, 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;
- Query 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;
- Apply similar transforms to the rest of the arguments.

This process creates a new definition of *load*, shown in Figure 4.26, that has two entries corresponding to the same two senses described in Section 3.3, but with stricter selectional restrictions on the arguments as the result of domain specialization. Once the specialization is completed, 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, the general mechanism in the system ontology described in Chapter 3. During the specialization process we

---

<sup>4</sup>We keep unspecialized entries with a lower preference, so parses for out of domain utterances can be found if no domain-specific interpretation exists.

(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 4.26: 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*

(a) (Origin Human) => (Form Solid-object)

(b) (Kr-type Person) => (Origin Human)

Figure 4.27: 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.

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 4.27(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 4.27(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 4.26.

The feature inference is also used in many cases to prevent application of clearly unsuitable transforms. For example, consider the case when a transform writer made a mistake and instead of correctly re-mapping the :agent argument to *actor* slot, specified the mapping between :agent and :goal. This can be easily detected through feature inference. The goal is of type GEO-LOC, and when defining the LF to KR mapping, we define a feature inference rule (KR-type Geo-loc) => (Intentional -). The mapping between :agent and :goal will trigger this inference, and result in the conflict with the domain-independent restriction of :Agent, which specifies (*Intentional -*). Thus, a problem rule can be caught early on in the transform writing process.

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 (*Mobility Self-moving*) and therefore does not satisfy the restriction.

### 4.5.1 Domain type coercion

As part of lexicon specialization, our system implements domain type coercion, which is an additional mechanism which helps parser deal with certain semantic type coercions 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 (6).

(6) Which prescriptions do I need to take?

Intuitively, this is understood to mean (7).

(7) 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 (6) 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 LF::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 4.28) would be sent to the medication knowledge base, eliciting an empty result, since *prescriptions* are not recognized in the knowledge base as consumable objects.<sup>5</sup>

A correct interpretation needs a semantic representation for (6) that resembles the semantic representation for (7). For this, additional information must be interpolated - in particular, the relation that holds between prescriptions and medications. We accomplish this in our framework

---

<sup>5</sup>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.

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

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 4.29a.

- (a) declare-coercion-operator prescribed-medication  
 :arguments prescription  
 :return medication
- (b) (prescription  
 :semfeatures (Phys-obj (Information +)  
 (Kr-type PRESCRIPTION))  
 :lf LF::Information-object\*prescription  
 :coercion ((operator prescribed-medication)  
 (semfeatures (Phys-obj (Kr-type Medicinal-substance)  
 (Form Substance))))))

Figure 4.29: (a) Operator to coerce prescriptions to medications. (b) Lexical entry for *prescription* with coercion feature generated from operator in (a)

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 4.29b). The coercion feature specifies the coercion operator that will be applied during the interpretation process, and a set semantic features derived from the operator returns type

medicinal-substance, allowing the coerced NP *prescriptions* to be accepted in parsing contexts that require substances. For every noun phrase with a nonempty coercion feature, the grammar produces an additional NP representation for that item with a feature that indicates that a coercion has occurred, and the original semantic features replaced by the features specified by the coercion. During parsing, we apply the specified coercion rule, and generate a special logical form which uses the specified coercion operator. It results in the (simplified) representation for *prescriptions* in Figure 4.28b. This coerced representation is then plugged into the query in Figure 4.28 in place of (PRESCRIPTION ?x). The new query is sent to the medication database and results in the correct answer: a set of medications.

Adding the information about possible coercion rules into the lexical entries gives us an edge in efficiency, because only the coercions relevant to our domain are attempted, and the parser can apply the correct rule immediately. At the same time, the declarative coercion rules can also be used in reference resolution. For example, consider the sequence of utterances: *I have a new prescription. When do I need to take it?* The obvious referent for *it* is *prescription*, but there is a type mismatch, because *take* in this context prefers an argument that is a medicinal substance. The PHORA system [Byron, 2002b] can use the declarative coercion rules in the resolution process. When a type mismatch is encountered, it searches the library of coercion rules for an operator with suitable argument types to perform the coercion and attempts to resolve the pronoun.

In this section we discussed handling coercion as a domain-specific phenomenon. There are also cases in different domains which can be seen as instances of domain-independent coercion. For example, saying *go to Elmwood Avenue* can be taken to really mean *go to some contextually specified point on Elmwood Avenue*. We have started work on a more general treatment of coercion in our domain-independent lexicon, and plan to fully integrate it with our grammar as part of our future work.

## 4.6 Evaluation

Improving parsing speed and accuracy is one of the main goals of our approach. In this section, we discuss the impact our lexicon specialization method has on parsing in two domains. Portability, our other goal, is more difficult to evaluate numerically. We present some statistics about lexicon and KR ontology sizes in our domains in this section, and discuss the portability improvements from our approach in the next section.

Lexicon specialization considerably speeds up the parsing process. We already showed some initial results in Section 2.5. On our Medadvisor and Monroe-s12 corpora, parsing with specialized selectional restrictions results in 10% fewer constituents created compared to parsing with generic selectional restrictions, which leads to corresponding improvement in parsing speed. In this section we report the results on evaluating our method on a considerably more complex set of inputs, namely, parsing speech lattices.

Getting perfect speech recognition from a speaker-independent system is a difficult task. Depending on our domain, the sentence accuracy of our speech models could be as low as 50%, and it takes time to build better models in spoken dialogue domains, where there the data for language modeling is scarce and fairly expensive to collect and transcribe. One way to get around speech recognition errors 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* is shown in Figure 4.30. 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.

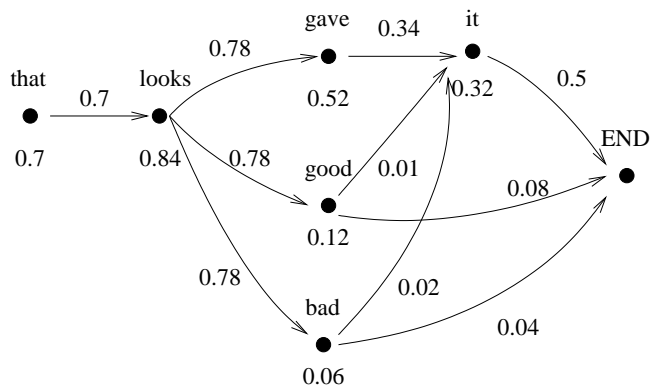


Figure 4.30: A sample lattice for *that looks good*, misrecognized as *that looks gave it*. Numbers at nodes denote confidence scores for lexical items, numbers on arcs denote the confidence scores on transitions.

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. We show results for two Pacifica sets: the original evaluation was conducted 1.5 years ago, and Pacifica-03 evaluation, conducted at the time of

|                 | Generic | Pacifica | Medical | Generic-03 | Pacifica-03 |
|-----------------|---------|----------|---------|------------|-------------|
| # of senses     | 1947    | 2028     | 1954    | 2547       | 2656        |
| # of KR classes | -       | 228      | 182     | -          | 210         |
| # of mappings   | -       | 113      | 95      | -          | 98          |

Table 4.4: Lexicon statistics.

|                       | Pacifica | Medical  | Pacifica-03 |
|-----------------------|----------|----------|-------------|
| # of sentences        | 200      | 34       | 188         |
| Time with KR (sec)    | 4.35     | 2.5      | 3.61        |
| Time with no KR (sec) | 9.7      | 4.3      | 9.42        |
| Errors with KR        | 47 (24%) | 8 (24%)  | 50 (27%)    |
| Errors with no KR     | 65 (32%) | 16 (47%) | 75 (40%)    |

Table 4.5: 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.

writing. Table 4.4 describes the lexicon and ontologies used in these domains. The results presented in Table 4.5 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.<sup>6</sup>

The improvement comes from two sources. The tighter selectional restrictions limit the search space and help to correctly disambiguate according to our domain knowledge. The increased preference values for specialized entries ensure that they 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 help the parser find an interpretation for in-domain utterances faster, though there are compromises between speed and accuracy involved in setting up boosting parameters. We will discuss these later in Section 4.7.2.

To show how specialization helps with picking better word sequences from lattices, Table 4.6 lists several example sentences from our evaluation, listing the words that were said, and

---

<sup>6</sup>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.

the sentence that was picked out of the lattice with an without specialization. The first two examples where the non-specialized version makes errors are interpretations plausible in some context, especially if we want to interpret fragments. For example, a city can be out of electricity, and then the domain-independent interpretation will make sense. We know, however, that in our Pacifica domain only bridges can be out of order. This restriction allows the domain-specialized version pick up the right interpretation from the lattice. Similarly, in the second example we know nothing about manner of waiting in our domain, but we can answer questions about the duration of actions, so the specialized parser prefers the correct interpretation. In the third case, the domain-independent version makes a mistake because there is no easy way to restrict the subject of the verb *use* in a domain-independent fashion - there are other plausible contexts where physical objects are subjects, for example, *this map uses many colors*. The tight domain constraints in Pacifica allow for a better domain-specific restriction which helps the parser make the correct choice.

Especially interesting is the last example in this table. There is one very long utterance, which also had a lattice with a very large number of possible paths. The non-specialized version run over the limit of chart size in our parser and gave up before an interpretation could be found. The specialized version actually found a plausible interpretation. It uses *drive* instead of *get*, so it was counted as wrong in our evaluation. However, in this case the interpretation selected by our specialized version is equally good for the task. We noted multiple cases like that in our evaluation, where the words picked up by the domain-specific version were not exactly right, but still good enough to produce the correct response. We have not found a reliable way to evaluate such cases, but we are planning it as part of our future work.

| Real words                                                           | Without specialization          | With specialization                                                    |
|----------------------------------------------------------------------|---------------------------------|------------------------------------------------------------------------|
| Delta Bridge is out                                                  | Delta is out                    | Delta Bridge is out                                                    |
| 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 INTERPRETATION FOUND         | Meanwhile use the other truck to drive the people from Exodus to Delta |

Table 4.6: Example sentences picked out of the same lattice with and without specialization

One case where we found specialization to hurt the parser's performance was the use of numbers in our Monroe domain. Interpreting numbers is a difficult problem, and it is especially

difficult in Monroe, where people commonly use route numbers to identify routes, as in *Go to 15*. We use a domain-specific mapping between numbers and routes to bias our parser toward this interpretation. In most contexts, this works very well, because syntactic context supplies the rest - for example, *to 15* is a locational phrase that likely involves a route, whereas *15 people* clearly refers to a number. But there were several cases when, in response to a question, the speaker answered *six*, meaning *6 people*. Our domain-specialized interpretation bias toward numbers as route names was very strong, and the top interpretation produced by the parser corresponds to *route 6*. In contrast, the non-specialized parser correctly interprets it as *a set of 6 objects*, leaving it up to the reference to determine the object type. However, there are very few examples like that in our corpora, and they are highly ambiguous in any case - *six* as an utterance could equally well mean *route 6* in a different context. We are currently working on an algorithm in which the parser, before producing the final interpretation, asks the reference whether a given interpretation is plausible in the current context. We expect it will help us to achieve better disambiguation on such contextually dependent utterances.

In addition to evaluating the efficiency and accuracy gains from domain specialization, we have some data to show that 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 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 4.4), this represents an improvement over hand-crafting custom lexicons for every domain.

Comparing the results of two evaluations over time also shows that the system maintains its portability. In the time which passed between the two evaluations, we added a new domain to the lexicon, correspondingly expanding it by 600 word senses. We do not have a measure of grammar change, however, many additions were made to improve the parser coverage. We also made some changes that excluded a small number of utterances from our covered corpus. This is due to the fact that we used to treat *Okay now load the truck* as a single utterance, but as a result of the additional analysis we conducted on our new domain, we decided that our grammar should treat it as a sequence of two utterances, the acknowledgment *Okay* and the command. This and a couple of similar changes resulted in 12 utterances that we needed to remove from our corpus.

In addition to the sentences no longer covered, the increased ambiguity in the grammar caused our parser to perform slightly worse than it was a year ago. However, it is still clear that lexicon specialization helps. In the original Pacifica evaluation, there was 2.23 time increase

in parsing speed and 25% relative increase in accuracy between the generic and the specialized versions of the parser. In the Pacifica-03 domain, there is 2.64 speed and 33.3% accuracy improvement.<sup>7</sup> We needed to make a few changes in the mappings to accommodate the changes that happened in the LF and KR ontologies, but this affected only 15 mappings, most of which had to be deleted because we used to represent tense and modality information with special KR types, and have since changed the representation.

In the future we expect to improve the accuracy of our system so that it will be the same as of the older Pacifica parser. The three utterances picked out correctly from the lattices by the old Pacifica grammar, which were not handled properly by the new grammar, are all short (3 words) utterances. In developing our grammar, we added more rules which now allow more fragments such as adjective phrases or numbers to be interpreted as valid utterances, and on this evaluation we found that sometimes our parser preferred a fragment to a complete sentence. We are currently working to improve our grammar rule preferences and scoring function to take this into account and prefer full sentences whenever possible.

## 4.7 Discussion

Our architecture is aimed at developing a portable parser and semantic interpreter while not sacrificing the speed and accuracy that comes from domain customization. In the previous section we presented the numerical result which show that we can achieve a 2-fold increase in parsing speed in two of our domains at a cost of writing a relatively small number of mappings. In this section, we discuss in more detail the portability of parser, and the limitations of our current approach.

### 4.7.1 Portability

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 subcategorization frames such as those used in COMLEX. Developers can instead use descriptive labels for semantic

---

<sup>7</sup>We did not measure this in the original evaluation, but in Pacifica-03 we also measured the speed improvement in terms of the number of constituents built in the domain. The non-specialized version builds 752.9 constituents per utterance compared with 441.4 in the specialized version, an 1.7 times improvement. The rest of the difference in speed comes from the increased garbage collection because of the larger number of constituents.

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 4.3.2). 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.

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.*, pear]. 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 4.4 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. This is a minor increase, which does not have significant effects on the speed and accuracy in our system.

The above numbers were obtained under an additional restriction: if an entry was specialized, the non-specialized version was removed from the lexicon. This made sense in our application. Both domains had tight constraints, and there was no point in looking for out of domain interpretations if a given word had a domain-specific entry. In a more general case, we keep the non-specialized entries in the lexicon at a lower preference, to allow for out-of-domain interpretations of in-domain words. Under those conditions, the increase in the lexicon size is considerably larger (22% in our Pacifica-03 domain). When running with these settings, the parse accuracy remains the same, but we pay the price in decreased performance. We are currently working

on improving our search methods so that the entries which violate domain constraints are not added to the chart until all other possibilities have been explored. We hope this will help us to improve the speed of the system despite the increased number of lexical entries in those cases.

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 VerbNet 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, 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 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.

As an informal example of how the portability of our system is improved with the use of mappings consider our Medadvisor domain. The bulk of the development in the domain, including writing the mappings, was conducted two years before the date of writing this thesis. Since then, we added a large number of words and word senses to our lexicon, and added a new domain, a purchasing assistant. In the Medication adviser domain, the preferred meaning of *take* is LF::Consume, as in *take aspirin*, and this is reflected in the mappings. The preferred meaning of *take* in our purchasing assistant domain is LF::Select, as in *I will take this computer*. *Take aspirin* is actually ambiguous between the two, and on the set of medication adviser dialogues the domain-independent version frequently selects the wrong meaning. In the domain-specialized version, the LF-KR mappings and lexicon specialization helped the parser to always select the correct sense of *take* for the domain. Moreover, there were several sentences which were not covered by our parser two years ago, but which were handled correctly in this evaluation because new grammar rules were developed in the meantime. This is clearly the example of how the portability of the system is improved by the mappings - in this case we observed the improvement in coverage through the development of domain-independent grammar which applies to both domains, while there was no loss of accuracy, even though the word sense ambiguity was worsened due to the addition of a new sense to the same word.

One more advantage of using a domain-independent semantic 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.

### 4.7.2 Balancing parsing speed and accuracy

Our evaluations on parse lattices demonstrated that parsing speed and accuracy improve with adding our domain-specialization to a domain-independent system. In chapter 2, as part of the general discussion on the usefulness of selectional restrictions, we also showed statistics on parsing Monroe and Medadvisor corpora with customized and with generic selectional restrictions. The evaluation shows that there is a 10% decrease in the number of constituents built during parsing, and 8% absolute (11% relative) increase in sentence disambiguation accuracy if we are parsing sentences from human corpora with customized restrictions, compared to parsing with generic restrictions. This is a substantial improvement, and because of the corresponding decrease in memory usage, it results in even bigger improvement in parsing speed.

However, one issue that remains not fully resolved in our system is boosting the probabilities for specialized entries. We mentioned earlier that we increase the preference of specialized entries to make the parser use them first and thus arrive at a parse faster. The boosting factor is currently selected manually. What we found is that the best boosting factor we currently have is 1.03. This means that many of our entries have preference over 1, and the search in our parser is no longer guaranteed to be best first. We found that this highly focuses the search and provides the best possible efficiency.

To some extent, we can counteract the effects of not having the best-first search by telling the parser to parse until a given number of hypotheses is found (we used 5 in our medication advisor disambiguation tests). This increases the accuracy of the search, because there is a good chance that even if the best-first order was broken on the original search, if we search for a larger number of possible interpretations, the highest one will be found. There is a performance penalty for it in the increased time to parse, but at least for single sentences we found that our average parse time is currently less than a second on our sentences, so we could afford to have a little more search time to gain accuracy.

Efficiency is crucial when parsing lattices, and we found that this setting was the best possible choice for us. However, it results in some errors because now the parser frequently prefers words to silences: *I send a truck to Calypso* is weighted higher than *Send a truck to Calypso* if *I* is boosted to 1.03. If we do not boost the preferences of specialized entries, relying only on specialized domain restrictions, the domain-specialized model still performs significantly better in terms of selecting the right words from the lattices, because tighter restrictions cut down the number of possible combinations. However, we do not get any improvement in performance without boosting the probability of specialized entries, and under some settings we even experienced

decrease in performance.

Obviously, a better system is needed for properly weighting the entries. Recently, a set of tools has been developed to visualize search in our grammar. We are currently working on improving our search strategies, so that we do not need to give up the accuracy and can still get performance gains from changing the preferences of specialized entries.

### 4.7.3 Comparison with other approaches

We already compared our approach to the AUTOSEM system [Rosé, 2000], which maps directly between the syntactic information and the semantic structures in the lexicon. The distinguishing feature of our approach is that it maps between a domain-independent logical form and a domain-specific representation, and it offers a number of advantages in a dialogue system which needs to do discourse processing.

Another approach which uses a domain-independent representation which is converted into a domain-specific form is Abductive Equivalent Translation (AET) [Rayner, 1993]. Rayner used it to convert natural language utterances into domain-specific database queries. He developed a system in which the first order logical forms obtained from utterances were converted into database queries in first order logic with the aid of a theorem prover and logical axioms which linked words with domain-specific predicates. The advantage of his approach is that it is guaranteed to produce logically equivalent results, something we have not proven for our system. However, our representation has built-in controls to ensure that the transforms are sound. These include consistency checks with default feature values when variables are used in transforms, and consistency checks with feature rules during lexicon specialization.

In terms of portability our approach offers more flexibility, because it allows linking our representations to both predicate logic and frame-based KR representations, including representations with nested frame arguments, something not easily supported by translation between the logical forms. As we stated above, our domain-independent form includes information for discourse processing, something that is not necessary for database queries, and therefore not supported by AET. However, it is of major importance for a dialogue system, so using a representation which includes some discourse information is an important benefit.

Finally, using LF-KR mappings for lexicon specialization, with proven speed and accuracy benefits, is a unique feature of our method. Our transforms were designed from the start with an abstract syntax which makes it easy to extract the information needed for lexicon specialization from the KR representation based on the transform. Mosny [1996] adopted Rayner's system

was later adapted to extract selectional restrictions to use in a parser for database interfaces. However, this required further restricting the form of permissible correspondences between domain independent and domain-specific representations. In particular, his system relies on the assumptions that every predicate on the language side gets translated into a single predicate on the database side, and that there were no predicates applying to formulas. This is a significant restriction. The first assumption is violated in our examples which translate a set of semantic arguments into a complex-valued slot. The second assumption does not apply to our examples where the adverb *straight* modifies a phrase *to Avon*, which in itself is a formula. Our approach handles many of these complex cases which occur frequently in our domains.

An open issue with our approach is the soundness of the conversions that we do. AET offers guarantees that the domain-specific logical form will be equivalent to the domain-independent form based on a set of axioms which establish correspondence between the lexical predicates produced by the parser and the database predicates. In our approach, there is no proven equivalence. In part, this is due to the wider range of possibilities we handle - for example, we found it important that some modifiers should be ignored during the conversion between LF and KR representations in our domains, and, of course, dropping modifiers does not create an equivalent KR representation. However, a more detailed analysis of the invariants which must be supported during conversion could be helpful, and is planned as part of our future work.

## 4.8 Conclusions

In this chapter, we described a two-layer architecture for building multi-domain parsers. We separate domain-independent and domain-specific information by maintaining two ontologies: a domain independent LF ontology, and a domain- and application-dependent KR ontology. The ontologies are linked with the help of LF-KR transforms. The architecture allows us to utilize in our system a portable linguistically motivated parser capable of interpreting complex natural language sentences. The disadvantage of using domain-independent parser is primarily speed and low disambiguation accuracy, and our specialization technique allows us to semi-automatically specialize the lexicon to alleviate those problems, at a cost of writing a relatively small number of transforms between the ontologies. We already demonstrated in Chapter 3 that our domain-independent parser and grammar have good coverage of several different domains. In this chapter, we also showed that using our customization process improves the speed and accuracy of our parser in two domains, and so our approach is viable for building parsers for multi-domain medium-scale NLP systems.

There are several questions that need to be explored further in our system. One is the exact influence of boosting the domain-specialized entries on our search process. We are currently building search visualization tools to aid in this process. Another potential issue is grammar specialization. This involves both employing domain-specific grammar rules, or using statistical learning methods to adjust weights on existing rules, as in [Rayner and Carter, 1996] and [Copestake and Flickinger, 2000]. To date, we did not have parsed corpora of large enough size to successfully employ learning algorithm. We are currently working on collecting such corpus in the Monroe domain [Swift *et al.*, pear]. The corpus is being built using our parser specialized to the domain, to improve parsing speed and accuracy and decrease the need for hand-selecting the relevant parser. In that, our specialization method contributed to building our corpus, and we plan to implement statistical learning methods on this corpus to further improve parsing speed and accuracy as part of our future work.

## 5 Statistical Methods for Learning Features

In the previous chapters we described how to express selectional restrictions using feature lists in a practical dialogue parser. This discussion was built on two basic assumptions: that we have a lexicon with (hand-assigned) features and restrictions, and that all our selectional restrictions are strict. Both of these assumptions are useful when building a practical system. However, for large-scale projects we need to consider automating the task of acquiring selectional restrictions, or, more generally, selectional preferences and determining their strength.

In this chapter, we are going to explore a statistical model of selectional preferences based on using features. Ideally, we would like to have a mechanism to learn both the features associated with words and the selectional restrictions from an appropriate training corpus for the domain. This chapter does not fully achieve this goal, because we were unable to show that our method performs better than existing class-based models. However, we offer a general statistical model for selectional preferences based on feature lists, describe the experiments we conducted for learning feature-based selectional preferences from corpora, and discuss the improvements which may produce better results in future work. We also discuss how our model can be integrated with the parsing and customization model described in previous chapters, and how it relates to the questions we originally asked about selectional restrictions.

### 5.1 Background

Let us first consider a model for learning selectional restrictions from corpora. A selectional restriction between words  $w_1$  and  $w_2$  can be seen as a binary function  $r(w_1, ap, w_2)$ , true if  $w_2$  is acceptable as an argument  $ap$  of  $w_1$ , and false otherwise. In general, however, it is better to think of selectional restrictions as preferences: we highly prefer certain combinations of words,

but when no parse is possible, for example, when figurative language is involved, the restriction can be relaxed to allow for other uses of language. The idea of selectional preferences dates back to Wilks [1975], whose original implementation involved manually defined preference strengths in the lexicon. In this chapter, we discuss a probability model as a theoretically well-founded approach to expressing selectional preferences, and develop a probabilistic algorithm for learning them from corpora.

When asking the question whether  $w_2$  is a good argument to  $w_1$ , it is natural to cast it as a question about the probability of seeing  $w_2$  as an argument  $ap$  of  $w_1$ . Thus, if we know the probability distribution  $P(w_2 \mid ap, w_1)$ , we can answer the question of how likely a given combination is. A probability of 1 would mean that  $w_2$  is the only word allowed as an argument  $ap$  of  $w_1$  (unlikely in natural language, but possible in a small number of idioms). A probability of 0 would mean that the combination is impossible, and probabilities between 0 and 1 reflect different degrees of association between the two words. Provided that the probability function is continuous between 0 and 1, it can express gradable preferences between weak (described by lower numbers), and strong (described by higher numbers), and we could trade off different preference strengths against each other when we need to choose the best parse among the list of possible options.

In this section, we review a way to automate this task by learning selectional preferences from corpora. The basic task is, given a set of corpus observations, derive a probability distribution  $P(w_1, w_2 \mid ap)$ , to judge the fitness of the pair  $\langle w_1, w_2 \rangle$  when  $w_2$  is in a given argument position  $ap$  governed by  $w_1$ . Note that this formulation is slightly different from the question proposed above, which used the conditional probability of  $w_2$  given  $w_1$ . This is because our conditional distribution can be calculated from the non-conditional distribution using the Bayes rule, and the latter is easier to model in many cases.

The training data for learning our probability distribution is a training corpus  $\langle w_1, ap, w_2 \rangle$  collected from some natural language source, for example, extracted from a text corpus. Given this training corpus, the simplest possible approach would be to calculate the probability based on counts collected from corpora as

$$P(w_1, ap, w_2) = \frac{\text{count}(\langle w_1, ap, w_2 \rangle)}{\text{tripleCount}}$$

where *tripleCount* is the training corpus size. This gives the probability estimate which maximizes the likelihood of our training corpus. Unfortunately, it does not produce satisfactory results due to data sparsity problems. Many triples  $\langle w_1, ap, w_2 \rangle$  would have never appeared

in the training corpus, and therefore would be assigned probability 0. Some better estimation technique is necessary to account for these cases.

An obvious solution is suggested by the traditional approach to selectional restrictions, where restrictions are expressed as relationships between word classes and not individual words. In estimating the probability  $P(w_1, w_2 \mid ap)$  we can use a more general probability  $P(c_1, c_2 \mid ap)$  to smooth the distribution, where word  $w_1$  belongs to class  $c_1$ , and  $w_2$  to class  $c_2$ . As an example of this technique Grishman and Sterling[1992] used a text corpus with words annotated for word senses from a domain-specific hand-constructed semantic hierarchy. They collected triples  $\langle verb, ap, noun \rangle$  and conducted the evaluation in the way that effectively defined the probability distribution for verb-object pairs as:  $p(n, object, v) = \frac{count(v, object, n)}{tripleCount}$  where  $count(v, object, n)$  is the number of times the noun or noun class  $n$  occurred as an object of the verb or the verb class  $v$ , and  $tripleCount$  is the total number of triples collected.<sup>1</sup> They evaluated several different distributions, always generalizing the noun class, but with or without generalizing the verb, and showed that the model performed better in parsing than selectional restrictions manually constructed with their hierarchy.

The major issue with this approach is the availability of training data. The corpus in the experiments needed to be manually annotated with a domain ontology. This is a labor-intensive approach, and we need to look for better sources of semantic information for statistical modeling. We discuss the options in the next section, together with the approaches to evaluate the models obtained from corpora.

Our discussion starts with the general discussion of class-based models for learning selectional restrictions. The underlying principle in estimating probabilities is that we can back off from probabilities of words to probabilities of (semantic) word classes. The precise statistical formulation is discussed in Section 5.1.1. The intuitive explanation can be seen in the following example. Assume we want to estimate a probability of  $\langle remove, snowstorm \rangle$ . We may have never seen this pair in the training data, but this is not a conclusive judgment, because there will be many valid verb-object combinations never seen in training data (e.g.,  $\langle remove, digger \rangle$  is good, but it never appeared in our Monroe corpus). Yet we know that *remove* usually applies to physical entities, and *snowstorm* is an event. If we could learn this generalization from corpora, this could be helpful in getting a good probability model, and we can see this approach as a probabilistic counterpart of selectional restrictions in traditional lexicons, stated in terms of ontological categories.

---

<sup>1</sup>We re-cast the work presented in the original paper in terms of probability models so that it is easier to compare with other approaches.

The ultimate goal of this chapter is to propose a parallel probabilistic model, and a method to learn selectional restrictions expressed in terms of feature sets from corpora. There are two main reasons for this. We already showed in the previous chapters that a feature representation offers advantages as a semantic representation for selectional restrictions, such as flexibility, extensibility, ease of customization. We would like to keep the connection with our symbolic feature-list models, and extend them in probabilistic terms, so that they could be integrated into our grammar as preferences instead of strict restrictions. In addition, we hope that feature-based models may provide benefits in learning.

Semantic classes allow us to back off to class probabilities from word probabilities, and we will discuss the details in Section 5.1.1. But, especially with a large number of classes (corresponding, for example, to nodes in a large-scale ontology), we may still have difficulties with data sparsity and learning enough about a single class. Feature sets, which consist of smaller components, may be a way to pool the information from different word classes. For example, consider our Monroe corpus. The word *crew* appears 112 times as an object of *send* in the corpus, and the next most frequent object of *send* is *ambulance* - 39 times. If we are learning with traditional classes, combining information from these entries would be difficult, because their common parent is *physical-object*, a very general class to back off to. But they share one feature in common from our feature set — (*Mobility Movable*). Moreover, all objects of *send* for which we know the feature assignments had (*Mobility Movable*) in their feature lists. If we could build a model to utilize this information, we may then apply it to other words. For example, we may first conclude that *send* prefers to take movable objects given the evidence from its objects with known features. Then, we may deduce that *digger*, which appeared 7 times as an object of *send*, also has that feature, and thus can be used as object for *move*, which also requires (*Mobility Movable*). Thus, building a statistical model based on feature list offers us an opportunity to pool properties together from different words and utilize them in learning.

In this chapter we develop a log-linear probabilistic model for selectional restrictions using feature sets. We use the traditional approach to class-based models, reviewed in the next section, as the basis for the extended model using feature sets. We propose our model in Section 5.2, and discuss training and the experimental evaluation in the rest of the chapter. Our experimental results are mixed, and we were unable to show conclusively that learning with features presents the advantages we hoped for. However, the model is still useful as a generalization of the feature-based selectional restrictions we use in our parsing lexicon, and we describe some additions which may improve the results in the future work.

Before we can discuss in detail a probability model based on feature sets, we need to outline

the issues which typically arise in class-based models, and which will need to be addressed in our model. The two major questions include the source of semantic classes and evaluation methods. We review the existing class-based models with the emphasis on how semantic classes can be obtained for a probabilistic model in Section 5.1.1. The two basic choices are classes from an established symbolic ontology, for example, WordNet, or automatically derived classes using a clustering method. The choice between these is an important issue which extends to the feature-based models, and we discuss the options as the motivation for choices we made later in our learning experiments.

We review the evaluation methods for probabilistic selectional restrictions in Section 5.1.2. The methods applicable to standard class-based models transfer directly to our extended model, and we review and motivate the choice of Pseudo Word Sense Disambiguation and perplexity reduction as the evaluation methods for our experiments.

After we define our probability model and discuss the choices involved in data modeling, we describe an Iterative Maximization algorithm to learn the model parameters in Section 5.3, and, finally, discuss our experimental results in Section 5.5. One more question which needs to be considered is the integration of our statistical model with the parsing model discussed in the previous chapters. We do not develop a fully integrated model here, however, we discuss the possibilities of integrating our model with parsing in Section 5.6.1.

### 5.1.1 Probability models of selectional restrictions

We mentioned in the previous section that semantic classes can be used to smooth the probability distribution for  $P(w_1, w_2 \mid ap)$ . Before we go into details, let us make one simplification in notation: we assume that we hold the argument position  $ap$  fixed, and therefore we drop it from our notation. Therefore, we  $P(w_1, w_2)$  and assuming that  $w_1$  and  $w_2$  are in a known relationship, for example, verbs and direct objects in our learning experiments discussed later. The task of learning these probabilities can be re-cast as learning the probability distribution  $P(w_1, w_2, c_1, c_2)$ , because if we know it, the probability  $P(w_1, w_2 \mid ap)$  can be expressed as  $P(w_1, w_2) = \sum_{c_1, c_2} P(w_1, w_2, c_1, c_2)$ .

The distribution  $P(w_1, w_2, c_1, c_2)$  is sparse, but we can decompose it into components which are easier to learn by making independence assumptions. For example, we can estimate it as

$$P(w_1, w_2, c_1, c_2) = P(w_1 \mid c_1)P(c_2, c_1)P(w_2 \mid c_2) \quad (5.1)$$

that is, assuming that in a given argument position  $w_1$  and  $c_2$  are independent given  $c_1$ , and  $w_2$  and  $(c_1, w_1)$  are independent given  $c_2$ .

This model can be interpreted as a generative model, where the word pairs are in effect generated from the underlying (hidden) semantic classes, and probability distributions  $P(w_1 | c_1)$  and  $P(w_2 | c_2)$  describe the generation probabilities. We can further simplify the model by assuming that there is a single class generating  $w_1$  and  $w_2$ ,  $c_1 = c_2 = c$ . Then, Equation 5.1 is simplified to

$$P(w_1, w_2, c) = P(c)P(w_1 | c)P(w_2 | c) \quad (5.2)$$

The latter model is the assumption used most frequently in the current research on learning selectional restrictions [Rooth *et al.*, 1999; Hofmann and Puzicha, 1998]. Much of the research focuses on learning probabilities for word pairs in a given position, for example, verb-noun object pairs. Then, the probability is written as

$$P(v, n, c) = P(c)P(v | c)P(n | c) \quad (5.3)$$

where  $v$  is the verb and  $n$  is the noun in the object position. In the context, a single class is a class of  $\langle v, n \rangle$  pairs — for example, we may have a class  $c_1$  which corresponds to  $\langle v, n \rangle$  pairs where something physical is moved, and it has influence both on choosing the verb (with *move*, *take*, *remove* etc generated from  $c_1$  with high probability) and noun (with *truck*, *person*, *luggage* highly likely with  $c_1$ ).

One of the crucial choices for a class-based model is determining the appropriate set of semantic classes for this model, and obtaining the training data which can be used with those classes. We gave an example of using a hand-written ontology in the previous section, a labor-intensive approach not practical in large domains. A frequently used possibility is to have a large-scale lexicon as a source of classes. For example, Resnik [1993] formulated a class-based model for selectional restrictions and used WordNet as a semantic dictionary to learn selectional restrictions for verb-object combination, with WordNet synsets as semantic classes of nouns in the model. This model was extended by Ribas [Ribas, 1994; Ribas, 1995] to all types of selectional restrictions.

The problem with this approach, however, is the suitability of classes from the external dictionary to the selectional restrictions task. We already evaluated in Chapter 3 the existing dictionaries as sources of symbolic selectional restrictions in our system, and found them not particularly suited for the task. It remains an open question whether they are the best possible source of semantic information also for statistical learning. Considering WordNet in particular, Ribas [1994] outlines several problems with selecting the appropriate class in the restrictions based on WordNet, due to overgeneralization and acquiring restrictions to incorrect senses.

Approaches to overcome the problems with WordNet senses as semantic classes were proposed in [Ribas, 1995; Abney and Light, 1999], but rather than trying to overcome the difficulties, a better choice may be to generate the appropriate classes automatically. In particular, Li and Abe [Li and Abe, 1996] used word clustering based on Minimum Description Length as a basis for deriving probability distribution in a prepositional phrase attachment task. Using automatically derived clusters as the appropriate word classes they obtained 93% accuracy, compared with 88% accuracy on the same task when the probability model uses WordNet classes. This shows that automatically derived clusters used as semantic classes in the back-off model in some cases are better suited for selectional restriction tasks than hand-annotated restrictions. Similarly, Lee [1997] defines a similarity measure for nouns based on their appearance as verb objects, and then used the clusters of similar words to smooth the probability  $p(v | n)$ , and improve the model performance on pseudo word sense disambiguation and perplexity reduction tests (see next section for the discussion of the relevant evaluation measures).

In our work, we do not have the classes generated completely automatically. We use a feature set model as a guidance on the class structure, largely because we would like to derive features similar to our lexicon features. However, we carried on experiments (discussed in Section 5.5) which specify the number of features, but which are trained on completely unsupervised data, in effect deriving clusters of verb-noun pairs automatically, subject to (few) constraints on the structure of the feature set.

In the models mentioned above, the clusters are defined separately from the word model which uses them. Rather than using special clustering methods, we would like instead to learn the clustering structure best suited for our model. This can be done by treating semantic classes as unobserved data, and assuming that observed word pairs are generated from the unseen classes. [Rooth *et al.*, 1999] uses the probability model from Equation 5.3, and verb-object co-occurrence data with an Expectation Maximization algorithm to induce the clustering which maximizes the probability of the training corpus. It assumes that probabilities  $p(c)$ ,  $p(v | c)$  and  $p(n | c)$  are parameters in generating observations from hidden data, and uses an Expectation Maximization algorithm to derive a set of probabilities which maximize the probability of a training corpus of word pairs. It thus simultaneously learns the soft clustering of the data as probabilities of verbs and nouns being associated with specific clusters, and the full probability model with the addition of  $p(c)$ .

The general model for clustering from co-occurrence data has been developed by Hofmann and Puzicha [1998]. They discuss a variety of formulations for EM-based clustering models, and introduce the use of deterministic annealing to overcome local maxima problems with the EM

algorithm, as we discuss in Section 5.3.2. The clustering model in Equation 5.3 is the symmetric clustering model by Hofmann, which we will use as a comparison in our evaluation.

In our research we take the same idea of deriving soft clusters together with the probability distribution, but, instead of using a clustering model relying on single classes, we model the data with a log-linear probability model based on feature sets associated with words, in parallel to the approach taken within our lexicon. We describe our model and training algorithm a little later, after discussing evaluation methods for statistical models of selectional restrictions.

### 5.1.2 Evaluating models of selectional restrictions

Once a probabilistic model has been learned from a corpus, we need to evaluate its quality. Ideally, we want to have a measure of its usefulness in parsing. For example, Seagull and Schubert [1999] developed a parser that combines a linguistically motivated grammar with a probability distribution for head word sense patterns estimated from a corpus, and evaluated the resulting model on precision and recall on Penn Treebank bracketing, and performed significantly better compared to a baseline PCFG. However, this approach requires a full PCFG model, which is non-trivial to obtain in a new domain. Since we intend to estimate only the word pair probabilities in our model (and we discuss in Section 5.6.1 how it can be included in a full PCFG model), we need evaluation measures which approximate this test.

The two tasks related to parsing which utilize selectional restrictions are attachment decisions and word sense disambiguation. If we build a statistical model with probabilities  $p(\textit{preposition} \mid n)$  and  $p(\textit{preposition} \mid v)$  of a given prepositional phrase to be associated with a noun  $n$  or a verb  $v$ , we can use the probabilities to decide on the correct attachment of the prepositional phrases [Stetina and Nagao, 1997; Ratnaparkhi, 1998]. This will be an open possibility with the model we propose in this chapter, however, we wanted a task to evaluate a model which describes only the relationships between verb-object pairs we had available in our experiments.

Pseudo word sense disambiguation (PWSD) is a test which takes two similar verb-object pairs and determines which one is a better match. This can be done in two ways: we can compare the probabilities of two pairs  $p(v_1 \mid n)$  and  $p(v_2 \mid n)$  and decides which verb is more appropriate with a given noun, as done by Lee [Lee, 1997]; alternatively, we can compare probabilities  $p(n_1 \mid v)$  and  $p(n_2 \mid v)$  to decide which noun is a more likely object of the verb [Rooth *et al.*, 1999]. Either of these variations is good if a full probability model  $P(v, n)$  is available, and we chose

the latter version in our own evaluation, because these are the probabilities which would be used in a lexicalized, generative PCFG parsing model, such as Collins[1997] model 1.

In addition to PWSD tests, we use the perplexity of the model on the test data as a measure of the model quality. Assume we have a corpus of verb-object pairs  $Y = \{y = \langle v, n \rangle\}$ , with  $y$  appearing with frequency  $\tilde{p}(y)$  in the corpus. For a given probability model  $P(y)$  the cross-entropy per-word is defined as

$$H(P) = - \sum_y \tilde{p}(y) \log P(y)$$

and the perplexity of a language model  $P$  is defined as

$$B(P) = 2^{-H(P)} = 2^{\sum_y \tilde{p}(y) \log P(y)}$$

Perplexity is a measure of model quality traditionally used in language modeling, which predicts the level of uncertainty associated with a given model. The higher the perplexity, the more uncertain is the set of words which can appear with a given verb. Thus, reducing the perplexity provides a measure of goodness of our selectional restrictions and their predictive power. However, perplexity is less directly related to the parsing task, and in our experimental results we will see that perplexity reduction is not always consistent with the improvements in PWSD results.

## 5.2 Probability Model

Now that we discussed the general principles for modeling and evaluating selectional restrictions based on semantic classes, let us extend the class-based model to represent our data by relying on semantic feature lists rather than just classes of words. In our symbolic lexicon, selectional restrictions are modeled as a restriction between a verb and a head of the noun phrase it governs, and our statistical model mirrors this assumption. Given two word tokens  $w_1$  and  $w_2$ , we assume that they are associated with feature sets  $\bar{f}_1 = \langle f_{11}, \dots, f_{1m} \rangle$  and  $\bar{f}_2 = \langle f_{21}, \dots, f_{2n} \rangle$ . Then our task is to derive a probability function  $P(w_1, w_2, \bar{f}_1, \bar{f}_2)$  which represents the probability that a word  $w_1$  is assigned a feature set  $\bar{f}_1$  and takes as an argument a word  $w_2$  with a feature set  $\bar{f}_2$ . The words may be any pair of words which employ selectional restrictions: a verb and an object, a preposition and a noun it governs, an adjective and a noun, etc. In our practical experiments we will narrow down the range of possible relationships to verb-object pairs, but the model covers a large class of possible relationships.

Since our data is defined as dependent on a set of features assigned to words, it is natural to model it with a log-linear probability distribution, which is a distribution based on a set of properties associated with words [Berger *et al.*, 1996]. The distribution has the form

$$P(w_1, w_2, \bar{f}_1, \bar{f}_2) = C^{-1} e^{\sum_i \lambda_i g_i(w_1, w_2, \bar{f}_1, \bar{f}_2)}$$

where  $C$  is a normalization constant, and  $g_i(w_1, w_2, \bar{f}_1, \bar{f}_2)$  are property functions<sup>2</sup> which depend on the words. The choice of property functions determines the properties of the underlying data which can be expressed by the model. In this section, we discuss in detail how to model the data with the appropriate property functions, and describe the set of property functions we used in our experiments.

### Property Functions

Table 5.1 gives some example property functions and their use in a probability model. Broadly speaking, we can have three types of properties: properties which reflect features assigned to words, properties which reflect selectional restrictions, and properties which reflect structure of the feature set and correlations between features.

An example of a property which reflects features assigned to words would be a property  $g(w_1, w_2, \bar{f}_1, \bar{f}_2) = 1$  if  $w_1 = v$  and  $f_{1m} = vf_i$ ; and 0 otherwise. This property function is a representation of a feature set assigned to the verb: it is true for a verb  $v$  if the feature  $f_{1m}$  is set to the value  $vf_i$ . If the weight of this property is positive, then this feature is highly likely to be associated with the verb; if it is negative, then this association is highly unlikely.<sup>3</sup>

For brevity, we will write this property as  $g(v, vf_i)$ , noting the values which must be set for the property to be true.<sup>4</sup> Similarly to the verb assignments, we can have properties  $g(n, nf_i)$  which correspond to having a noun to which a feature value  $nf_i$  is assigned. Learning high weights for those properties corresponds to learning features associated with the noun.

---

<sup>2</sup>The standard terminology calls  $g_i$  “features”. However, in this thesis we used the term “features” to denote a semantic feature list associated with the word. Therefore, we are using the term “property functions” to denote the “features” in the log-linear model which use our semantic features as described in this chapter.

<sup>3</sup>More precisely, increasing the weight of this property increases both the association of  $v$  with  $vf_i$ , and the overall probability of an unannotated pair with verb  $v$ . We ignore this in the rest of our discussion, since we believe it does not have a significant impact on our evaluation results.

<sup>4</sup>We assume in this notation that verbs have names distinct from nouns, and all feature values are distinct, so the form of the original property can be recovered from this abbreviated notation.

| Property           | Example                                                                | Usage                                                                                                          |
|--------------------|------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| $g(v, vf_i)$       | $w_1 = send$ and $vf(aspect) = dynamic$                                | To model features associated with words                                                                        |
| $g(v, nf_i)$       | $w_1 = send$ and $nf(mobility) = movable$                              | To model selectional restrictions between the verb and features of its noun object                             |
| $g(vf_i, nf_j)$    | $vf(trajjectory) = +$ and<br>$nf(mobility) = movable$                  | To model selectional restrictions as dependencies between verb features and noun features                      |
| $g(v, vf_i, nf_j)$ | $w_1 = take$ and $vf(trajjectory) = +$ and<br>$nf(mobility) = movable$ | To model selectional restrictions as dependencies between verb senses determined by features and noun features |
| $g(nf_i, nf_j)$    | $nf(origin) = human$ and $nf(form) = object$                           | To model dependencies between features associated with the same word.                                          |

Table 5.1: Sample property functions for modeling feature sets. Properties can express features assigned to words, selectional restrictions, and dependencies between feature values.

There are several ways to encode selectional restrictions with properties. The simplest way is to use properties of the form  $g(w_1, w_2, \bar{f}_1, \bar{f}_2) = 1$  if  $w_1 = v$  and  $f_{2m} = nf_i$ , and 0 otherwise (abbreviated as  $g(v, nf_i)$ ). A high weight for this property indicates that a verb is placing a selectional restriction on its objects, namely, that a noun which appears with it should have the feature value  $nf_i$  assigned. A negative weight would indicate that a noun with a specified feature cannot occur with a given verb. The properties can use more features, for example,  $g(v, nf_i, nf_j)$ , where the selectional restriction requires a noun with features  $nf_i$  and  $nf_j$  simultaneously.

We can also encode selectional restrictions based on features only, for example  $g(w_1, w_2, \bar{f}_1, \bar{f}_2) = 1$  if  $f_{1m} = vf_i$  and  $f_{2m} = nf_i$ , and 0 otherwise (abbreviated as  $g(vf_i, nf_i)$ ). In this case, the high property weight encodes the high likelihood that verbs with feature  $vf_i$  assigned take the nouns with feature  $nf_i$ . Obviously, a more general version of this property is a selectional restriction where a verb with an assigned set of values requires a noun with a different set of values.

Finally, we can have properties of the form  $g(w_1, w_2, \bar{f}_1, \bar{f}_2) = 1$  if  $w_1 = v$  and  $f_{1m} = vf_i$  and  $f_{2m} = nf_i$ , and 0 otherwise. These can encode selectional restrictions dependent on the verb polysemy. A high weight on this property indicates that a sense of verb  $v$  associated with the verb feature  $vf_i$  prefers the noun senses associated with the feature  $nf_i$ . Again, properties dependent on sets of features can be included in the model to express conjunctive selectional restrictions. These properties are different from the properties based on verb features only in that they can give a better representation to different verb senses. In our symbolic lexicon, in particular, many verb features are too general to define reasonable selectional restrictions between their values and verb objects. However, verb feature assignments often differentiate between different verb senses, so a combination of a verb with its assigned features is a good representation of a given verb sense for selectional restrictions.

For expressing the structure of a feature set, we can use properties of the form  $g(w_1, w_2, \bar{f}_1, \bar{f}_2) = 1$  if  $f_{1m} = vf_i$  and  $f_{1n} = vf_j$ , and 0 otherwise. This property describes the dependence between two feature values assigned to a verb sense: if its weight is high, the values are highly likely to co-occur, while a low weight indicates that co-occurrence is unlikely. Adding properties of this form to the set describes in a probabilistic manner the feature dependencies hard-coded in our symbolic lexicon, described in Section 2.2.3.

We assumed above that the properties which are the most relevant and therefore the most useful are those which treat verb and its features as “context” which places restriction on the noun. The converse is also possible. For example, we could have a property  $g(n, vf_i)$  which would indicate that a specified noun tends to co-occur with verb senses assigned the feature  $vf_i$ .

The choice of properties depends more on the complexity of the model: adding more properties is likely to result in a more precise model, but may make learning more difficult because the complexity of the property space increases.

Using properties based on features may give our model an advantage compared with a simple class-based model. Assume we only have to learn a model with two verbs: *see* and *move*. We can see many things: *city, building, truck, person, digger*. But we can move only a subset of those: *truck, person, digger*. In fact, the set of physical things which can be moved is a subset of a larger set of physical entities which can be seen. Now, assuming that *truck* and *person* are frequent words in our corpus, Hofmann’s class-based model should be able to easily derive two clusters for nouns: one for (*city, building*), which can appear as objects of both *see* and *move*, and another cluster for (*truck, person*) which can only appear as objects of *move*. Or, alternatively, there will be two clusters, with *city* and *building* generated with high probability from the first cluster, and *truck* and *person* generated with equal probability from both clusters. But now suppose that *digger* is not a very frequent word, and that it only appeared a couple of times, and only with *move*. We would still like to learn the generalization that all movable objects can also be seen, and therefore we should assign a high probability to  $\langle \textit{see}, \textit{digger} \rangle$  even if we have never seen it before. But, as it turns out, it is difficult for a class model. On several of our test runs with this example, we got the models which assigned a high probability for  $\langle \textit{move}, \textit{digger} \rangle$ , but not for  $\langle \textit{see}, \textit{digger} \rangle$ , missing this important generalization.

In a feature-based model, this generalization can be captured in three ways.

First, we could have two features,  $f_1$  and  $f_2$ , corresponding to the “seeable” and “movable” noun sets, and then add to our property set the property  $g(-f_1, f_2)$  which denotes a dependence between those two features. We hope that in this corpus we can learn a low weight for this property, indicating that the probability is low that a word which has feature  $f_2$  will not have a feature  $f_1$  assigned to it.<sup>5</sup> Then the model first learns that *digger* has the feature  $f_2$ , because it appears as a direct object of *move*. Then the learning algorithm can guess that *digger* should also have the feature  $f_1$ , then learn a higher weight for  $g(\textit{digger}, f_1)$  and a correspondingly higher probability for  $\langle \textit{see}, \textit{digger} \rangle$ .

Second, we may believe we know something about the way the data is structured. Then, we may put a pre-defined constraint on the feature combinations allowed in our model. In this

---

<sup>5</sup>We are using  $-f_1$  to denote the value “-” of the binary feature  $f_1$ . The argument here can be generalized to multiple-valued features by saying that we will learn negative weights for all values of  $f_1$  except for that correlated with  $f_2$ , or by introducing a property  $g(-f_1 = fv, f_2)$ , true for all values of  $f_1$  except for  $fv$ .

example, if we can guess that  $f_2$  can only be true where  $f_1$  is true, then we can use this as a strong restriction on the underlying feature structures and learn the appropriate model.

Third, we can use patterns in feature co-occurrence to learn the appropriate weights. In particular, from our corpus the model is likely to learn not only learn that the weight of  $g(\textit{move}, f_2)$  is high (corresponding to the fact that movable objects are in class  $f_2$ , but also that  $g(\textit{move}, -f_1)$  is low (corresponding to the fact that we rarely see non-physical objects such as *happiness*). Assuming that we had enough instances of *move* in the corpus to make this conclusion, from observing  $\langle \textit{move}, \textit{digger} \rangle$  the algorithm can now conclude that *digger* also has the feature  $f_1$ , or it would not appear in the context of *move*.

The first of those alternatives is more flexible because it may help us to discover explicitly generalizations about our data sets which were not previously known. However, it is more computationally expensive, because adding more properties increases the number of parameters to be learned and the general complexity of the model. Therefore, in our experimental setup we chose the combination of second and third options, using the structure of the TRIPS feature set as our guess about the feature co-occurrence restrictions, and learning selectional restrictions through interaction of features which simultaneously occur (or don't occur) on the observed pairs, as we discuss in the next subsection.

### The probability model for experimental evaluation

So far, we have discussed a design for a general model in which each word has an associated feature list. In our experiments, we make a simplifying assumption that given a verb-object pair only the feature list associated with the noun is important. We then use properties of the form  $g(v, nf_i)$  and  $g(n, nf_i)$ . As we discussed above, the former corresponds to a selectional restriction placed by the verb on the noun features, and the latter to the feature list associated with the noun. In addition, we placed the restrictions on the allowed feature combinations based on feature list type, as discussed below.

For a specific example, consider our 3-feature set discussed in detail in Section 5.5.2:

- *Main-type* with values *phys-obj*, *abstr-obj*, *situation*, *time*, *proposition*, *-*;
- *Mobility* with values *-*, *Fixed*, *Movable*
- *Aspect* with values *-*, *Static*, *Dynamic*.

If we see a data item annotated as  $y = \langle \textit{send}, \textit{truck} \textit{ (Main-type Phys-obj) (Mobility Movable) (Aspect -)} \rangle$ , then there will be 6 properties with non-zero values for it, listed in Figure 5.2.

| Noun property      | Weight         | Selectional restriction | Weight         |
|--------------------|----------------|-------------------------|----------------|
| g(truck, Phys-obj) | $\lambda_{11}$ | g(send, Phys-obj)       | $\lambda_{31}$ |
| g(truck, Movable)  | $\lambda_{18}$ | g(send, Movable)        | $\lambda_{38}$ |
| g(truck, Aspect-)  | $\lambda_{20}$ | g(send, Aspect-)        | $\lambda_{40}$ |

Table 5.2: Properties with non-zero values for the word pair  $\langle \textit{send}, \textit{truck} \rangle$  annotated with features  $\textit{(Main-type Phys-obj)}$ ,  $\textit{(Mobility Movable)}$ ,  $\textit{(Aspect -)}$

The corresponding probability will then be taking into account the sum of corresponding weights listed in the table:

$$P(\textit{send}, \textit{truck} \textit{ (Main-type Phys-obj) (Mobility Movable) (Aspect -)}) = C^{-1} e^{\lambda_{11} + \lambda_{18} + \lambda_{20} + \lambda_{31} + \lambda_{38} + \lambda_{40}} \quad (5.4)$$

If both the weights of properties corresponding to noun features and the weights of properties corresponding to selectional restrictions are high, then the probability will be high. If some of the weights were low or negative, then the probability would be correspondingly lower.

If we have an unannotated word pair, for example,  $y = \langle \textit{send truck} \rangle$ , then its probability will be the sum of probabilities over all possible feature assignments, which involve all possible combinations of feature values subject to structural restrictions. The restrictions in our set are that *Mobility* is “-” with anything except  $\textit{(Main-type Phys-obj)}$ , and *Aspect* is “-” with anything except  $\textit{(Main-type Situation)}$ . This simulates the typed feature lists we used in previous chapters. In our case, there are 8 possible feature-value assignments consistent with each word pair, and the probability of an unannotated pair is

$$\begin{aligned} P(\textit{send truck}) &= P(\textit{send truck} \textit{ (Main-type Phys-obj) (Mobility Movable) (Aspect -)}) \\ &+ P(\textit{send truck} \textit{ (Main-type Phys-obj) (Mobility Fixed) (Aspect -)}) \\ &+ P(\textit{send truck} \textit{ (Main-type Situation) (Mobility -) (Aspect Dynamic)}) \\ &+ P(\textit{send truck} \textit{ (Main-type Situation) (Mobility -) (Aspect Static)}) \\ &+ P(\textit{send truck} \textit{ (Main-type Proposition) (Mobility -) (Aspect -)}) \\ &+ P(\textit{send truck} \textit{ (Main-type Abstr-obj) (Mobility -) (Aspect -)}) \\ &+ P(\textit{send truck} \textit{ (Main-type Time) (Mobility -) (Aspect -)}) \\ &+ P(\textit{send truck} \textit{ (Main-type Other) (Mobility -) (Aspect -)}) \end{aligned}$$

Each of the probabilities in this list expands similarly to the example in Equation 5.4.

In our model, different properties work together to provide selectional restrictions. Consider two word pairs:  $y_1 = \langle \textit{send}, \textit{truck} \rangle$  and  $y_2 = \langle \textit{send}, \textit{city} \rangle$ . In the appropriate model, the following will be true

- Weight for  $g(\textit{send}, \textit{movable})$  will be high, and weight for  $g(\textit{send}, \textit{fixed})$  will be low;
- Weight for  $g(\textit{truck}, \textit{movable})$  will be high, and weight for  $g(\textit{truck}, \textit{fixed})$  will be low;
- Weight for  $g(\textit{city}, \textit{movable})$  will be low, and weight for  $g(\textit{city}, \textit{fixed})$  will be high;
- The weights for other features associated with *truck* and *city* will be comparable (because all the other features of those words are presumably the same in our model)

Then, it is easy to see that the model will assign a high probability to  $\langle \textit{send}, \textit{truck} \rangle$  and a low probability to  $\langle \textit{send}, \textit{city} \rangle$ , in effect creating a selectional restriction. Note, however, that this may be dependent on the word frequencies. If *city* is much more frequent in the corpus than *truck*, and used in a variety of contexts, all weights for properties using *city* are likely to be higher, and the contrast will not be so evident after summation. Moreover, it is possible that if we compare  $\langle \textit{send}, \textit{city} \rangle$  with  $\langle \textit{send}, \textit{digger} \rangle$ , the model will decide that *city* is better because *digger* is a very infrequent word. The numbers produced by the model are appropriate in the sense that we are learning probabilities of the pairs, which also depend on the word frequency, but it means that we have to be careful how we use the probabilities in evaluation. It may be the case that the model has not learned good selectional restrictions, but appears to be making correct judgments because the “good” object is a more frequent noun. Because of this, in our evaluation when we generated PWSD sets with two contrasting noun objects, one “good” and one “bad”, we limited the set of possible bad objects only to the nouns with frequency close to the frequency of a “good” noun, to make sure that the results we have are due to the appropriately learned selectional preferences and not just frequency effects (see Section 5.5.1 for specifics of how our PWSD sets were created).

The advantage of our property set design is that it aligns well with our symbolic lexicon, where selectional restrictions are expressed as dependencies between individual verbs and the features of their arguments. Thus, with this setup we would hope to learn property weights similar to our selectional restrictions. The disadvantage is that we are not learning explicit generalizations about feature dependencies as we discussed in the previous subsection, and also that we are not learning the features associated with verbs, and we have to depend on the

correctness of our guess about the feature co-occurrence restrictions encoded in the structure of our feature set. The results we obtained on training our model were mixed. We discuss the related issues in Section 5.5, after we explain in more detail our training algorithm.

### 5.3 Training

Once we have selected a property set for modeling the data, we can start learning weights associated with features from a training corpus. We start by collecting a corpus of observations  $Y$ , which consists of tokens  $y_1, \dots, y_n$  appearing  $c_1, \dots, c_n$  times respectively. The tokens in our model are of words and associated features as discussed later in this section. We assume that this corpus is derived from the (larger) complete sample space  $Z$ , with individual tokens  $z \in Z$ . The observations  $Y$  are drawn from this space with the unknown probability distribution  $P(z)$ . The goal of training is to estimate the true probability distribution  $P(z)$  based on the available training corpus. The basic idea of the learning algorithm is to find the weights which maximize the likelihood of the training corpus. More formally, we need to find a set of weights  $\bar{\lambda}$  which maximizes the corpus log-likelihood<sup>6</sup>

$$\log P(Y) = \log \prod_y P(y)^{c_y} = \sum_y c_y \log P(y)$$

Here  $c_y$  is the number of times a token  $y$  appeared in the corpus. We can multiply the function above on  $1/N$ , where  $N$  is the total number of tokens in the corpus, and re-cast our problem in terms of empirical distributions. We therefore define the cross-entropy per word  $L_Y$  as

$$L_Y = 1/N \sum_y c_y \log P(y) = \sum_y \tilde{p}(y) \log P(y) = E_{\tilde{p}} \log P(y) \quad (5.5)$$

where  $\tilde{p}$  is the empirical distribution of the observed data,  $\tilde{p}(y) = \frac{c_y}{N}$ , and  $E_{\tilde{p}}$  is the expectation calculated over the empirical distribution.

The form of observations  $Y$  depends on the availability of training data. In the completely supervised case, our corpus will consist of verb-object pairs and all features associated with each verb and object. If this is the case  $L_Y$  is a convex function, and an iterative scaling algorithm (discussed in Section 5.3.1) can be used to find its global maximum. Unfortunately, completely annotated data is difficult to find, and in general the corpus of observations will not be annotated

---

<sup>6</sup>Since log is a monotonically increasing function, maximizing log-likelihood is equivalent to maximizing the corpus likelihood.

with all features. This is the incomplete data case, which is of most interest to us. We will return to it after we discuss learning property weights for fully annotated data.

### 5.3.1 Training on completely annotated data

If we have a corpus of observations of the form  $y = \langle w_1, w_2, \bar{f}_1, \bar{f}_2 \rangle$  where verbs and nouns are fully annotated with feature values, we can use the maximum entropy principle, which states that we should seek the distribution which maximizes the entropy of the corpus [Jaynes, 1982]. It can be proven that for log-linear models the distribution which maximizes the model cross-entropy is the same as the one which maximizes the corpus likelihood subject to feature expectation constraints [Pietra *et al.*, 1997a]. If this is the case, we can use either the Generalized Iterative Scaling algorithm [Darroch and Ratcliff, 1972] or Improved Iterative Scaling algorithm [Pietra *et al.*, 1997b] to find the model parameters.

Both of those algorithms are iterative algorithms guaranteed to converge to the global maximum of the corpus likelihood, which is a convex function for log-linear models. At each step, we change the parameters so that  $\bar{\lambda}^{(k+1)} = \bar{\lambda}^{(k)} + \bar{\delta}$ , where  $\bar{\delta} = \{\delta_1, \dots, \delta_n\}$  is determined by the training algorithm. For IIS algorithm,  $\bar{\delta}$  is the root of the equation

$$\sum_z P_{\bar{\lambda}^k}(z) g_i(z) e^{\delta_i g_{\#}(z)} = \sum_y \tilde{p}(y) g_i(y) \quad (5.6)$$

where  $g_{\#}(z) = \sum_i g_i(z)$ .

The improved iterative scaling algorithm is guaranteed to increase the value of  $L_Y(\bar{\lambda})$  at every step, until it converges to its maximum. This is a property we will find useful when we discuss the partially annotated data case, where the iterative scaling can be adapted to find local maxima in the corpus likelihood function.

One difficulty which arises from training the incomplete models is due to data sparsity. Imagine that in our corpus we only observed instances of  $\langle \textit{kill}, \textit{person} (\textit{Origin Human}) \rangle$ . However, the *Origin* feature in our has multiple values, including *Living-non-human*, which encompasses other animals, and *Artifact* for man-made objects. It is easy to check that the corpus likelihood is maximized if the weights of features *Living-non-human* and *Artifact* are set to  $-\infty$ . However, this is not a good thing to learn - for example, it will cause the probability of  $\langle \textit{kill}, \textit{animal} (\textit{Origin Living-non-human}) \rangle$  to be equal to 0, which is obviously not correct.

To correct for the data sparsity problem, a smoothing method is necessary. The goal is to learn a set of weights which assigns small but non-zero probabilities for previously unseen feature

combinations. There are different possible approaches for smoothing (see [Chen and Rosenfeld, 2000] for a review). For example, we may smooth the observed counts, pretending that each feature was observed some small number of times. Another option, which we will use, is to utilize a Gaussian prior.

The idea of the method is to say that the vector of weights is drawn from the diagonal Gaussian distribution, so that  $P(\bar{\lambda}) = \prod_i \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{\lambda_i^2}{2\sigma_i^2}}$ . This means that we believe that some parameter sets (and, correspondingly, some languages) are more likely than others, and the languages with weights close to  $-\infty$  are highly unlikely. Therefore, if we can take this prior distribution into account, we can discourage weights for rare combinations from going to  $-\infty$ .

Thus, we maximize the joint probability of the corpus and the parameters

$$L_y^*(\bar{\lambda}) = \log P(Y) = \log\left[\prod_y P(y)^{c_y}\right] \times \prod_i \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{\lambda_i^2}{2\sigma_i^2}} = \sum_y c_y \log P(y) - \sum_i \frac{\lambda_i^2}{2\sigma_i^2} + \sum_i \log \frac{1}{\sqrt{2\pi\sigma_i^2}}$$

The last term in this equation is a constant with respect to  $\bar{\lambda}$  and can therefore be dropped during maximization. Chen and Rosenfeld [1999] describe how the IIS algorithm can be adapted for use with a Gaussian prior. The resulting equations for  $\bar{\delta}$  are

$$\sum_z P_{\bar{\lambda}^k}(z) g_i(z) e^{\delta_i g_i(z)} = \sum_y \tilde{p}(y) g_i(y) - \frac{\lambda_i}{\sigma_i^2} \quad (5.7)$$

Smoothing with priors has been shown to help with language modeling for sparse data cases. The additional complexity of the method, however, is choosing an appropriate values of  $\sigma_i^2$ . It depends on the size of the training set, as well as on the number of properties in the model. In our experiments, we assume that there is a unique value  $\sigma^2 = \sigma_i^2$  for all  $i$ , and report our results with different values of  $\sigma^2$ . We find the optimal value by searching through the model space until the probability of a held-out development dataset can no longer be improved.

### 5.3.2 Training on incomplete data

As discussed in the previous section, if we had a corpus of word pairs with associated feature sets, we could use the IIS algorithm to learn the weights of our properties. Unfortunately, our hand-defined lexicon does not have complete coverage in many domains; moreover, because of word sense ambiguity it is not possible to determine the correct feature list automatically in many cases, and the manual annotation is difficult and time-consuming. Instead, we can relatively easily obtain from corpora a partial version of this data set, for example, all verb-object pairs without the associated feature assignments, or a dataset in which only a subset of the data items

is annotated with corresponding feature lists. We can then view words with feature assignments as hidden data  $z = \{w_1, w_2, \bar{f}_1, \bar{f}_2\}$ , as described below.

We model the unannotated pairs collected from corpora as generated by a hidden dataset obtained from a log-linear probability distribution described above. More formally, we have a set of hidden data  $Z$ , which consists of individual instances  $z \in Z$ , which correspond to the pair together with a complete set of features associated with it. However, we cannot fully observe the hidden data. Instead we sample a corpus  $Y = \{y_1, \dots, y_N\}$  — observed data, with  $y_i$  appearing  $c_i$  times in the corpus. For every observed data item  $y$ , there is a set of feature assignments  $X(y)$  consistent with a given  $y$ , and a corresponding set  $\{z = (x, y) \mid x \in X(y)\}$  of hidden data consistent with the observation  $y$ . For example, if we have only unannotated dataset of verb-object pairs as our observed data, then for every observed data item  $y = \langle n, v \rangle$ ,  $X(y)$  is a set of all possible feature assignments in our feature set (see Section 5.2, Equation 5.5 for an example). If we have a partially annotated dataset, for example, if we observed a data item  $y = \langle \text{send}, \text{truck}, (\text{mobility movable}) \rangle$ , that is, a verb-object pair annotated with  $(\text{mobility movable})$  feature, then our set  $X(y)$  is a set of all possible feature assignments including  $(\text{mobility movable})$ .

The data is distributed according to a log-linear probability distribution with parameters  $\bar{\lambda} = (\lambda_1, \dots, \lambda_n)$

$$P_{\bar{\lambda}}(z) = C^{-1} e^{\sum_i \lambda_i g_i(z)}$$

where  $C$  is a normalization constant

$$C = \sum_z e^{\sum_i \lambda_i g_i(z)}$$

and  $g_i(z)$  are property functions that describe the data set.

Since we are dealing with incomplete data, the corpus log-probability formula (5.5) is now expressed as

$$L_Y(\bar{\lambda}) = \sum_y \tilde{p}(y) \log P(y) = \sum_y \tilde{p}(y) \log \left[ \sum_{x \in X(y)} P_{\bar{\lambda}}(y, x) \right] = E_{\tilde{p}} \log \left[ \sum_{x \in X(y)} P_{\bar{\lambda}}(y, x) \right]$$

Similarly, we can generalize the IIS algorithm for learning with incomplete data. Intuitively, we can generalize the IIS equations (5.7) by taking into account the fact that we are now dealing with incomplete data, and  $y$  on the right-hand side are observed data consistent with multiple instances of hidden data. The resulting equation for iteration  $k$  of the algorithm will be

$$\sum_z P_{\bar{\lambda}^k}(z) g_i(z) e^{\delta_i g_i(z)} = \sum_y \tilde{p}(y) \sum_{x \in X(y)} P_{\bar{\lambda}^k}(x \mid y) g_i(y, x) - \frac{\lambda_i}{\sigma_i^2}$$

We will call this method IM-Gaussian, and we present the complete proof that our method works, namely that it increases the value of function  $L^*$  on every step until it converges to a critical point, in Section 5.4. In general, the method uses different parameters  $\sigma_i^2$  for every property function  $g_i(y, x)$ . As mentioned earlier, in our practical experiments, for simplicity we will use a single value  $\sigma_i^2 = \sigma^2$  for all  $i$ .

A major difference between learning on complete and incomplete data is that for incomplete data the corpus likelihood is no longer a convex function. Therefore, the IM can no longer guarantee finding the global maximum in corpus likelihood. Instead, it is likely to converge to a local maximum. This is a common problem observed for EM class-based probabilistic models, and there are several possible ways to deal with it.

The general idea of the methods we implemented is a gradual relaxation of constraints: we first place relatively strong constraints on the parameters, hoping that there will be fewer local maxima. Then, as we learn a more constrained solution, the constraints can be relaxed and we can hope that the more constrained solution will provide us with a starting point in a less constrained space closer to a global maximum, thus providing a better overall solution.

One possibility for constraint relaxation is to add properties gradually, instead of training on the whole set at once. Thus, we start with a set with a small number of properties, in order to learn reasonable weights for them before adding more. A principled way of doing this can be implemented with feature selection in the IM algorithm [Riezler, 2000]. In our experiments, we employ a simpler version. First, we train the model with just 1 feature *Main-type*. We initialize the weights for that feature randomly, and run the training until convergence. Then, we add more features, bringing the total to 3 or 11 depending on the experiment. We use the weights for properties using *Main-type* learned with a simpler model, and initialize the weights for all other properties randomly. This way, we may hope that weights for *Main-type* are already reasonable when we start training a more complex model and we can avoid (some) local maxima by using them as an intelligent guess.

We can also use  $\sigma^2$  to influence training. We can start with a really low  $\sigma^2$ , which penalizes weights which are far from 0. When  $\sigma^2$  is increased, the parameters get more freedom. Again, we hope that we can first learn a strongly constrained distribution in the space with fewer local maxima, and then as we relax the restrictions, we have a good guess about the starting point in the parameter space, since we already know that these are reasonable weights, albeit in a more constrained model. Our pilot experiments showed that this technique prevented our algorithm from getting stuck in local maxima in some cases, and we utilize it in all our learning experiments.

In addition to the methods we implemented, another widely used approach for avoiding local maxima is called annealing. The general idea of the annealing methods is to add a temperature parameter to the optimization process, and gradually change it in a way to bring us closer to the optimal solution. Simulated annealing [Kirkpartick *et al.*, 1983] is a probabilistic algorithm which allows for random variation of parameters at every step of the optimization. The changes which increase the optimized function (viewed as a Gibbs free energy function by analogy with statistical physics) are always accepted. The changes that decrease it are accepted with the probability controlled by a temperature parameter. When the temperature is large, the probability of accepting a “bad” change is high. As the system cools down, the probability of accepting “bad” changes decreases. The process is guaranteed to find a global maximum provided that the temperature is cooled down slowly enough, but in general it is highly inefficient.

The deterministic annealing algorithm [Rose, 1998] replaces the random search through the parameter space with a deterministic method. The corpus log-likelihood  $L_Y$  is understood as a cost function in the energy space, and adding a constraint on entropy with respect to some prior distribution over the hidden states produces a free energy function dependent on the temperature  $T$ . The energy function at high  $T$  has fewer local maxima, and gradually cooling it often results in finding a better maximum for  $L_Y$ . Hofmann [Hofmann, 2001] demonstrated that it helped its class-based models to avoid local maxima and improve the results.

The deterministic annealing could be extended to the IM algorithm (which is our training method discussed in the next section, and which can also be seen as a case of generalized EM [Riezler, 2000]), though we haven’t yet implemented the extension. Our algorithm with increasing  $\sigma^2$  was inspired by deterministic annealing, with  $1/\sigma^2$  as a gradually decreasing temperature parameter in the equations. However, it is not directly related to the Gibbs free energy functions, which are the theoretical justification for deterministic annealing. A better investigation of its usefulness is planned as part of our future work.

## 5.4 The Iterative Maximization Algorithm

In the previous section, we stated the problem of learning selectional restrictions in terms of learning a probability distribution from an observed corpus of partially annotated data generated from a hidden dataset with a log-linear probability distribution. Riezler [2000] developed an Iterative Maximization (IM) algorithm for learning log-linear models from incomplete data. We already discussed the IM algorithm and its extension with a Gaussian prior, the IM-Gaussian

algorithm. We also presented an informal description of these algorithms which should be sufficient for understanding the experimental results in Section 5.5.

In this section, we develop a rigorous proof that the IM-Gaussian procedure converges to a critical point of the probability function  $L^*$ . We do this by providing a summary of Riezler's proofs that the IM algorithm converges to a critical point of corpus likelihood, and then use these results as a basis for proving convergence for IM-Gaussian. Our proof is based on auxiliary functions, the technique originally used in [Pietra *et al.*, 1997b] to prove convergence of the IIS algorithm, and later by Riezler to prove convergence of IM. Note that Eisner [2001](section 8.2) points out that the IM algorithm, with or without a prior, can be regarded as a Generalized EM algorithm [Dempster *et al.*, 1977], which may be regarded as the alternative proof of convergence.

### 5.4.1 Auxiliary functions

The idea of the IM algorithm is based on a general algorithm for maximizing functions presented by Della Pietra [Pietra *et al.*, 1997a] as a basis for an Improved Iterative Scaling algorithm. The central idea is the notion of an auxiliary function.

Assume we have a function  $L(\bar{\lambda})$  that we need to maximize with respect to a vector of parameters  $\bar{\lambda} = (\lambda_1, \dots, \lambda_n)$ , but which is difficult to maximize directly. We can devise a maximization procedure if we can find an auxiliary function  $A(\bar{\gamma}, \bar{\lambda})$  which depends on  $\bar{\lambda}$  and another set of parameters  $\bar{\gamma} = (\gamma_1, \dots, \gamma_n)$  and satisfies the definition below.

**Definition 5.1 ( Auxiliary functions ).** *A real-valued function  $A(\bar{\gamma}, \bar{\lambda})$  is auxiliary for maximizing function  $L(\bar{\lambda})$  if*

(1) *For all  $\bar{\lambda}, \bar{\gamma}$*

$$L(\bar{\gamma} + \bar{\lambda}) \geq L(\bar{\lambda}) + A(\bar{\gamma}, \bar{\lambda}), \text{ where } \bar{\gamma} + \bar{\lambda} \text{ is a vector sum } (\lambda_1 + \gamma_1, \dots, \lambda_n + \gamma_n)$$

(2) (a)  *$A(\bar{\gamma}, \bar{\lambda})$  is continuous in  $\bar{\lambda}$  and  $\bar{\gamma}$  with  $A(\bar{0}, \bar{\lambda}) = 0$ , where  $C^1$  is a class of functions which are continuous and have a continuous first derivative, and  $\bar{0}$  is a vector consisting of all 0 values*

(b)

$$\frac{d}{dt} \Big|_{t=0} A(t\bar{\gamma}, \bar{\lambda}) = \frac{d}{dt} \Big|_{t=0} L((t\bar{\gamma}) + \bar{\lambda})$$

where  $t \in \Re$  and  $t\bar{\gamma} = (t\gamma_1, \dots, t\gamma_n)$ .

(3)  $A(\bar{\gamma}, \bar{\lambda})$  has a unique maximum  $\hat{\gamma}$ .<sup>7</sup>

Intuitively, property (1) of this definition is the basis of the maximization procedure: if we can find a value of  $\bar{\gamma}$  such that for a given  $\bar{\lambda}$   $A(\bar{\gamma}, \bar{\lambda}) > 0$ , we can use it to increase  $L$ , because it will be the case that  $L(\bar{\gamma} + \bar{\lambda}) > L(\bar{\lambda})$ . Specifically, Della Pietra [Pietra *et al.*, 1997b] proves that for a fixed  $\bar{\lambda}$ ,  $\hat{\gamma} = \operatorname{argmax}_{\bar{\gamma}} A(\bar{\gamma}, \bar{\lambda})$  ensures that  $A(\bar{\gamma}, \bar{\lambda}) > 0$  if condition (2) is satisfied. Then, if  $A(\bar{\gamma}, \bar{\lambda})$  has a unique maximum (condition 3), we can use  $A$  to maximize  $L$  iteratively as follows:

$$\bar{\lambda}^{(k+1)} = \bar{\lambda}^{(k)} + \bar{\gamma}^{(k)}, \text{ with } \bar{\gamma}^{(k)} = \operatorname{argmax}_{\bar{\gamma}} A(\bar{\gamma}, \bar{\lambda}^{(k)}) \quad (5.8)$$

Riezler proves the following:

**Theorem 5.1 ( Riezler 4.11 — Convergence )** *Let  $\bar{\lambda}^{(k)}$  be a sequence of values defined by Equation 5.8 and  $\{L(\bar{\lambda}^{(k)})\}$  be a sequence of likelihood values bounded from above. Then it converges monotonically to some critical point of  $L$ .*

The application of this maximization algorithm to learn the probability model which maximizes the likelihood of a training corpus yields the IM algorithm.

## 5.4.2 IM

In the previous section we summarized a learning process which can be used to obtain a local maximum of any function  $L$ . To apply it to our case, we cast our learning problem as learning parameters of a log-linear distribution which maximize the log-likelihood of our observed dataset  $Y$ .

We earlier defined the corpus likelihood  $L_Y$  for a given probability distribution  $P$ . If we hold the corpus fixed, but vary the parameter set  $\bar{\lambda}$  associated with the distribution, we can see the corpus likelihood as the function  $L(\bar{\lambda})$ , and cast our problem as finding a set of parameters  $\bar{\lambda}$  which maximizes the corpus probability as a function of  $\bar{\lambda}$ . To denote dependence on  $\bar{\lambda}$  we will write  $P_{\bar{\lambda}}(y)$  — the probability distribution  $P$  given the set of parameters  $\bar{\lambda}$ , and  $E_{\bar{\lambda}}(g_i)$  — the expectation computed using the function  $P_{\bar{\lambda}}(y)$ .

$$L_Y(\bar{\lambda}) = \sum_y \tilde{p}(y) \sum_{x \in X(y)} \log P_{\bar{\lambda}}(x | y) = E_{\tilde{p}} \sum_{x \in X(y)} \log P_{\bar{\lambda}}(x | y)$$

---

<sup>7</sup>This is different from the Della Pietra definition, we changed it slightly to accommodate our proofs.

Given our target function  $L_Y$ , we can define an auxiliary function

$$A(\bar{\gamma}, \bar{\lambda}) = E_{\bar{p}}\{1 + E_{\bar{\lambda}}[\sum_i \gamma_i g_i(y, x) \mid y] - E_{\bar{\lambda}}[\sum_i \frac{g_i(z)}{g_{\#}(z)} e^{\gamma_i g_{\#}(z)}]\}$$

where  $g_{\#}(z) = \sum_j g_j(z)$  — the sum of all property values for a given hidden data item  $z$ . This function is the generalization of a known auxiliary function used in a generalized iterative scaling algorithm for complete data case [Pietra *et al.*, 1997a].

Note that we are somewhat abusing the notation here by introducing the  $x$  and  $y$  variables which are not overtly bound. What they mean is that the expectation  $E_{\bar{p}}$  is computed over a set of observed data  $y \in Y$ . Then, the expectation  $E_{\bar{\lambda}}$  is computed over a set of hidden data items  $x \in X(y)$  consistent with the observed item  $y$ . Thus, the formula expands to

$$A(\bar{\gamma}, \bar{\lambda}) = \sum_y \tilde{p}(y) \{1 + \sum_{x \in X(y)} P_{\bar{\lambda}}(x \mid y) [\sum_i \gamma_i g_i(y, x)] - \sum_z P_{\bar{\lambda}}(z) [\sum_i \frac{g_i(z)}{g_{\#}(z)} e^{\gamma_i g_{\#}(z)}]\}$$

We will keep the terse notation here to provide for easier parallels with Riezler's proofs, but we will provide the expanded versions of the formulas in places important for actual implementations of the algorithm.

Riezler proves that this function satisfies all the conditions for an auxiliary function, and we re-state his results as list of lemmas.

**Lemma 5.1 ( Riezler Proposition 4.4 — Auxiliary function condition (3) )** *For each  $\bar{\lambda} \in \mathfrak{R}^n$ ,  $\bar{\gamma} \in \mathfrak{R}^n$ :  $A(\bar{\gamma}, \bar{\lambda})$  takes its maximum as a function of  $\bar{\gamma}$  at the unique point  $\hat{\bar{\gamma}}$  satisfying for each  $\hat{\gamma}_i$*

$$E_{\bar{p}}\{E_{\bar{\lambda}}[g_i(y, x) \mid y]\} = E_{\bar{\lambda}}[g_i(z) e^{\hat{\gamma}_i g_{\#}(z)}]$$

*i.e.*

$$\sum_y \tilde{p}(y) \sum_{x \in X(y)} P_{\bar{\lambda}}(x \mid y) g_i(y, x) = \sum_z P_{\bar{\lambda}}(z) g_i(z) e^{\hat{\gamma}_i g_{\#}(z)}$$

The proof is based on two results we will use later:

$$\frac{d}{d\gamma_i} A(\bar{\gamma}, \bar{\lambda}) = E_{\bar{p}}\{E_{\bar{\lambda}}[g_i(y, x) \mid y]\} - E_{\bar{\lambda}}[g_i(z) e^{\gamma_i g_{\#}(z)}] \quad (5.9)$$

$$\frac{d^2}{d\gamma_i^2} A(\bar{\gamma}, \bar{\lambda}) = E_{\bar{p}}\{E_{\bar{\lambda}}[g_i(y, x) g_{\#}(z) e^{\gamma_i g_{\#}(z)}]\} < 0 \quad (5.10)$$

**Lemma 5.2 ( Riezler Lemma 4.5 — Auxiliary function condition 1 )**

$$A(\bar{\gamma}, \bar{\lambda}) \leq L(\bar{\gamma} + \bar{\lambda}) - L(\bar{\lambda})$$

**Lemma 5.3 ( Riezler Lemma 4.5 — Auxiliary function condition 2 )**  $A(0, \bar{\lambda}) = 0$

**Lemma 5.4 ( Riezler Lemma 4.5 — Auxiliary function condition 2 )**

$$\frac{d}{dt} \Big|_{t=0} A(t\bar{\gamma}, \bar{\lambda}) = \frac{d}{dt} \Big|_{t=0} L((t\bar{\gamma}) + \bar{\lambda})$$

This satisfies all the conditions for A to be an auxiliary function to L, and therefore allows us to maximize  $A(\bar{\gamma}, \bar{\lambda})$  as a way of maximizing L. In the next section, we extend this algorithm with a Gaussian prior on property weights to provide smoothing in case of data sparsity problems.

### 5.4.3 Extending the IM algorithm with a Gaussian prior

Consider the application of the IM algorithm to a partially annotated corpus where all occurrences of a word  $w$  have been annotated with the feature value  $f_i$ . Since the empirical probability of assigning any other feature  $f_j$  to word  $w$  is 0, the IM algorithm from the previous section will learn this 0 value by driving the weight of properties which use  $f_j$  to  $-\infty$ . This may be not the best possible result if the data is sparse: we would like to learn a low, but non-infinite weight for these properties to improve learning on sparse data. An identical problem arises in a complete data case for learning Maximum Entropy models from n-gram data. Chen and Rosenfeld [Chen and Rosenfeld, 1999] used a Gaussian prior on property weights to provide smoothing in log-linear models. We now show how a Gaussian prior can be integrated into the Iterative Maximization algorithm.

Instead of maximizing the probability of the training data alone, we are going to maximize the joint probability of the training data and weights, assuming that weights are distributed according to a normal distribution with standard deviation  $\sigma^2$ . Then the corpus probability is

$$\log P^*(Y) = \log \left[ \sum_y c_y \sum_{x \in X(y)} \log P(y, x) \right] \left[ \prod_i \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{\lambda_i^2}{2\sigma_i^2}\right) \right]$$

The first term in this product is corpus probability as we defined it previously, and the second term is the probability of the parameter set  $\bar{\lambda}$  assuming that the parameters are distributed with a Gaussian distribution. The corresponding corpus likelihood function is

$$L^*(\bar{\lambda}) = L(\bar{\lambda}) + \sum_i \log \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{\lambda_i^2}{2\sigma_i^2}\right)$$

This can be re-written as

$$L^*(\bar{\lambda}) = L(\bar{\lambda}) - \sum_i \frac{\lambda_i^2}{2\sigma_i^2} + \text{const}(\bar{\lambda})$$

where  $\text{const}(\bar{\lambda})$  is a constant term which does not depend on  $\bar{\lambda}$  and can be dropped during maximization. We will therefore use the definition

$$L^*(\bar{\lambda}) = L(\bar{\lambda}) - \sum_i \frac{\lambda_i^2}{2\sigma_i^2}$$

in the rest of the section.

The auxiliary function we use in maximizing  $L^*(\bar{\lambda})$  can then be defined as

$$A^*(\bar{\gamma}, \bar{\lambda}) = A(\bar{\gamma}, \bar{\lambda}) - \sum_i \frac{(\lambda_i + \gamma_i)^2 - \lambda_i^2}{2\sigma_i^2}$$

This formulation follows pretty straightforwardly from taking the first derivative of  $L^*(\bar{\lambda})$ , an operation necessary to maximize it. We will show it in detail below when proving that  $A^*(\bar{\gamma}, \bar{\lambda})$  is auxiliary to  $L^*(\bar{\lambda})$ . To prove this, we will use Lemmas 5.1 - 5.4 from the previous section.

**Lemma 5.5** (  $A^*(\bar{\gamma}, \bar{\lambda})$  — **auxiliary condition 3** ) *For each  $\bar{\lambda} \in \mathfrak{R}^n$ ,  $\bar{\gamma} \in \mathfrak{R}^n$ :  $A^*(\bar{\gamma}, \bar{\lambda})$  takes its maximum as a function of  $\bar{\gamma}$  at the unique point  $\hat{\gamma}$  satisfying for each  $\hat{\gamma}_i$*

$$E_{\bar{p}}\{E_{\bar{\lambda}}[g_i(y, x) \mid y]\} = E_{\bar{\lambda}}[g_i(z)e^{\hat{\gamma}_i g_{\#}(z)}] + \frac{\lambda_i + \hat{\gamma}_i}{\sigma_i^2} \quad (5.11)$$

*i. e.*

$$\sum_y \tilde{p}(y) \sum_{x \in X(y)} P_{\bar{\lambda}}(x \mid y) g_i(y, x) = \sum_z P_{\bar{\lambda}}(z) g_i(z) e^{\hat{\gamma}_i g_{\#}(z)} + \frac{\lambda_i + \hat{\gamma}_i}{\sigma_i^2}$$

*Proof.*

$$\frac{d}{d\gamma_i} A^*(\bar{\gamma}, \bar{\lambda}) = \frac{d}{d\gamma_i} A(\bar{\gamma}, \bar{\lambda}) - \frac{d}{d\gamma_i} \frac{2\lambda_i\gamma_i + \gamma_i^2}{2\sigma_i^2} = \frac{d}{d\gamma_i} A(\bar{\gamma}, \bar{\lambda}) - \frac{\lambda_i + \gamma_i}{\sigma_i^2}$$

Thus,  $A^*(\bar{\gamma}, \bar{\lambda})$  reaches a critical point for  $\hat{\gamma}_i$  for which

$$E_{\bar{p}}\{E_{\bar{\lambda}}[g_i(y, x) \mid y]\} - E_{\bar{\lambda}}[g_i e^{\hat{\gamma}_i g_{\#}(z)}] - \frac{\lambda_i + \hat{\gamma}_i}{\sigma_i^2} = 0$$

We proved that  $\hat{\gamma}$  is a critical point of  $A^*(\bar{\gamma}, \bar{\lambda})$ . To prove that this is a maximum, it is sufficient to show that its second derivative is strictly less than 0.

$$\frac{d^2}{d\gamma_i^2} A^*(\bar{\gamma}, \bar{\lambda}) = \frac{d^2}{d\gamma_i^2} A(\bar{\gamma}, \bar{\lambda}) - \frac{d}{d\gamma_i} \frac{\lambda_i + \gamma_i}{\sigma_i^2} = \frac{d^2}{d\gamma_i^2} A(\bar{\gamma}, \bar{\lambda}) - \frac{1}{\sigma_i^2} < 0$$

because  $\frac{d^2}{d\gamma_i^2} A^*(\bar{\gamma}, \bar{\lambda}) < 0$  by equation (5.10).

**Lemma 5.6** (  $A^*(\bar{\gamma}, \bar{\lambda})$  — **auxiliary condition 1** )

$$A^*(\bar{\gamma}, \bar{\lambda}) \leq L^*(\bar{\gamma} + \bar{\lambda}) - L^*(\bar{\lambda})$$

*Proof.*

$$A^*(\bar{\gamma}, \bar{\lambda}) = A(\bar{\gamma}, \bar{\lambda}) - \sum_i \frac{(\lambda_i + \gamma_i)^2 - \lambda_i^2}{2\sigma_i^2} \leq L(\bar{\gamma} + \bar{\lambda}) - L(\bar{\lambda}) - \sum_i \frac{(\lambda_i + \gamma_i)^2 - \lambda_i^2}{2\sigma_i^2}$$

by Lemma 5.2

$$= L(\bar{\gamma} + \bar{\lambda}) - \sum_i \frac{(\lambda_i + \gamma_i)^2}{2\sigma_i^2} - (L(\bar{\lambda}) - \sum_i \frac{\lambda_i^2}{2\sigma_i^2}) = L^*(\bar{\gamma} + \bar{\lambda}) - L^*(\bar{\lambda})$$

**Lemma 5.7** (  $A^*(\bar{\gamma}, \bar{\lambda})$  — **auxiliary condition 2** )  $A^*(\bar{\gamma}, \bar{\lambda})$  is continuous in  $\lambda$  and  $C^1$  in  $\gamma$  with  $A^*(0, \bar{\lambda}) = 0$  and

$$\frac{d}{dt} \Big|_{t=0} A^*(t\bar{\gamma}, \bar{\lambda}) = \frac{d}{dt} \Big|_{t=0} L^*((t\bar{\gamma}) + \bar{\lambda})$$

*Proof.*  $A^*(\bar{\gamma}, \bar{\lambda})$  is in  $C^1$  as a sum of functions all of which are in  $C^1$ .

By Lemma 5.3

$$A^*(0, \bar{\lambda}) = A(0, \bar{\lambda}) - \sum_i \frac{(\lambda_i + 0)^2 - \lambda_i^2}{2\sigma_i^2} = 0 - 0 = 0$$

$$\frac{d}{dt} \Big|_{t=0} A^*(t\bar{\gamma}, \bar{\lambda}) = \frac{d}{dt} \Big|_{t=0} A(t\bar{\gamma}, \bar{\lambda}) - \frac{d}{dt} \Big|_{t=0} \sum_i \frac{(\lambda_i + t\gamma_i)^2 - \lambda_i^2}{2\sigma_i^2}$$

By Lemma 5.4

$$= \frac{d}{dt} \Big|_{t=0} L((t\bar{\gamma}) + \bar{\lambda}) - \sum_i \frac{2\lambda_i\gamma_i + 2t\gamma_i^2}{2\sigma_i^2} \Big|_{t=0}$$

Thus we have that

$$\frac{d}{dt} \Big|_{t=0} A^*(t\bar{\gamma}, \bar{\lambda}) = \frac{d}{dt} \Big|_{t=0} L((t\bar{\gamma}) + \bar{\lambda}) - \sum_i \frac{\lambda_i\gamma_i}{\sigma_i^2} \quad (5.12)$$

Similarly,

$$\begin{aligned} \frac{d}{dt} \Big|_{t=0} L^*((t\bar{\gamma}) + \bar{\lambda}) &= \frac{d}{dt} \Big|_{t=0} L((t\bar{\gamma}) + \bar{\lambda}) - \frac{d}{dt} \Big|_{t=0} \sum_i \frac{(t\gamma_i + \lambda_i)^2}{2\sigma_i^2} \\ &= \frac{d}{dt} \Big|_{t=0} L((t\bar{\gamma}) + \bar{\lambda}) - \sum_i \frac{2t\gamma_i^2 + 2\lambda_i\gamma_i}{2\sigma_i^2} \Big|_{t=0} \end{aligned}$$

Thus,

$$\frac{d}{dt} \Big|_{t=0} L^*((t\bar{\gamma}) + \bar{\lambda}) = \frac{d}{dt} \Big|_{t=0} L((t\bar{\gamma}) + \bar{\lambda}) - \sum_i \frac{\lambda_i\gamma_i}{\sigma_i^2} \quad (5.13)$$

QED.

Thus we proved through lemmas 5.5-5.7 that  $A^*(\bar{\gamma}, \bar{\lambda})$  is an auxiliary function and we can use it in an algorithm to maximize  $L^*$ .

Based on equation (5.11) which defines a critical point for A, we derive the formulas for our IM-Gaussian algorithm

$$\lambda_i^{(k+1)} = \lambda_i^{(k)} + \delta_i$$

$$\sum_y \tilde{p}(y) \sum_{x \in X(y)} P_{\bar{\lambda}^k}(x | y) g_i(y, x) = \sum_z P_{\bar{\lambda}^k}(z) g_i(z) e^{\delta_i g_{\#}(z)} + \frac{\lambda_i^{(k)} + \delta_i}{\sigma_i^2}$$

If  $g_{\#}(z) = K = \text{const}$  this simplifies to

$$\sum_y \tilde{p}(y) \sum_{x \in X(y)} P_{\bar{\lambda}^k}(x | y) g_i(y, x) - e^{\delta_i g_{\#}(z)} \sum_z P_{\bar{\lambda}^k}(z) g_i(z) - \frac{\lambda_i^{(k)}}{\sigma_i^2} - \frac{\delta_i}{\sigma_i^2} = 0$$

Or

$$C_1 e^{C_2 \delta_i} + C_3 \delta_i = C_4$$

where

$$C_1 = - \sum_z P_{\bar{\lambda}^k}(z) g_i(z)$$

$$C_2 = K, C_3 = -\frac{1}{\sigma_i^2}$$

$$C_4 = \frac{\lambda_i^{(k)}}{\sigma_i^2} - \sum_y \tilde{p}(y) \sum_{x \in X(y)} P_{\bar{\lambda}^k}(x | y) g_i(y, x)$$

In the next section, we document our experiments on applying this algorithm to learning verb selectional restrictions from corpora.

## 5.5 Learning Experiments

In the previous sections we described the model for learning selectional restrictions from corpora using feature sets as an underlying representation. In this section, we report on our experiments using this model attempting to learn selectional restrictions on verb objects from corpus data. We use two corpora, one derived from the TRIPS lexicon, and another from the Wall Street Journal data, evaluating our results on the Pseudo Word Sense disambiguation test and on perplexity reduction. We start with the description of the corpus data we collected (Section 5.5.1), describe the experimental setup (Section 5.5.2), and present the results of our learning experiments in Section 5.5.3. We conclude with the general discussion of the results we obtained, future directions to improve them, and the way our model can be integrated with a general parser described in previous chapters.

### 5.5.1 Corpus Data

For the experimental evaluation of our method we collected corpora which consisted of verb-object pairs and used the IM-Gaussian algorithm to obtain the probability models of those data. We evaluated the decrease in the perplexity of a data set, and the accuracy of our model on a pseudo word sense disambiguation task set up similar to Rooth [Rooth *et al.*, 1999]. For PWSD evaluation we created datasets consisting of matching verb-object pairs, with each good  $\langle verb, noun_1 \rangle$  pair matched with a corresponding contrastive  $\langle verb, noun_2 \rangle$  pair which was not a good verb-object combination. We then evaluated the accuracy with which the model assigns higher probability to a good pair than to the bad pair.

We collected two training corpora, with characteristics summarized in Table 5.3. The TRIPS-lex dataset is an artificial data set based on the TRIPS lexicon. It is useful for experimentation, because the data could be automatically annotated with features as described below, and we can expect good results from our learning algorithm on these data because we know the feature model used to generate it. Testing on this dataset allows us to confirm that our algorithm works well, and makes it easier to evaluate the impact of seeding in training data. The WSJ dataset is a set of verb-object pairs collected from a Wall Street Journal corpus. It is more interesting in the sense that it is a realistic set obtained from natural data, and testing on it is a better estimate of how well the learning happens in practice.

To collect the TRIPS-lex corpus, we assembled a complete dataset of acceptable verb-object pairs from the version of TRIPS lexicon specialized to the Medication Adviser domain using our specialization method. We used the specialized lexicon to provide tighter restrictions and thus a smaller set of acceptable pairs. This makes the data set more interesting, because the restrictions are stronger, whereas with a less restrictive dataset there are more acceptable pairs and less interesting structure to learn. The data was generated as follows. For every verb, we generated as many instances as there were different lexico-syntactic patterns in the lexicon. For example, for *load*, we have two possible patterns, corresponding to *load the truck with oranges* and *load the oranges into the truck* (see Figure 3.4 in Section 3.3), so we would generate two instances of the verb *load*. Then, for every instance we generated all possible pairs including all nouns in the lexicon consistent with the selectional restriction on its direct object. For example, the first instance generated pairs  $\langle load, truck \rangle$ ,  $\langle load, ambulance \rangle$  *etc*, while the second generated pairs  $\langle load, oranges \rangle$ ,  $\langle load, people \rangle$  *etc*.

Most of the generated pairs in the TRIPS-lex set were unique, though some repetitions were possible. For example, *make* can be used in constructions *make the people happy* and *make the*

*people go to Avon*. Both of these will generate identical pairs for training data,  $\langle \text{make, people} \rangle$ . Therefore, all verb-object pairs in the training corpus have very close frequencies. This is not typical of natural language distributions, which are usually Zipfian, with a small number of very frequent items, and a large number of infrequent pairs. This may have an impact on our evaluation results, as discussed in Section 5.6.

From the complete list of verb-object pairs we collected from the TRIPS lexicon, we set aside 10% of the total number of pairs in our corpus for testing, and 10% for development purposes. We removed the pairs which occurred in the test dataset from the training data, to ensure that we are testing on completely unseen data. Then, we matched the data in our test set with an equal number of bad pairs to create a PWSD test dataset as described earlier. A pair was considered bad if it was a combination not included in our set of acceptable pairs. Since the dataset we collected was complete, any pair not included in it violates selectional restrictions.

WSJ2 is a dataset based on Wall Street Journal corpus. We extracted a list of verb-object pairs from the BLLIP corpus, which is the WSJ corpus parsed with the Charniak’s statistical parser [Charniak, 1997], using the TGrep pattern matching tool. Out of 18400 extracted pairs, we retained 10% for testing and 10% for development. The probability of selecting a pair for testing was proportional to its count in the corpus, and some pairs were selected multiple times, hence the number of unique testing pairs was smaller than the total number of pairs. We then removed from the training data all pairs selected for the test corpus to ensure that our testing data consisted entirely out of the previously unseen pairs. We created the PWSD dataset similarly to the TRIPS-lex dataset, by adding a pair with a matching verb but a different noun to each “good” pair, with the additional constraint that the “bad” noun should have a frequency similar to that of a “good” noun object.<sup>8</sup>

An additional complication with the PWSD test data based on the Wall Street Journal corpus is how to ensure that the contrastive pairs are truly “good” and “bad” combinations. It is clear that something that appeared in the training data is a good combination, but it is less clear how to generate a bad combination. For our test set, we simply considered any pair which did not appear in the training data as bad. However, our corpus size is quite small and therefore on manual examination it turned out that many of the pairs automatically generated as “bad” were in reality perfectly acceptable. In an attempt to ensure better reliability of the testing results, we obtained a human-annotated PWSD dataset. We collected the data on a web site, asking our subjects to rate the felicity of every verb-object pair on a 5-point scale. We

---

<sup>8</sup>We defined similarity as at most 30% difference in the frequency of the “bad” object from the frequency of the “good” object.

obtained ratings for 1860 pairs, each rated independently by 2 annotators. These pairs were matched so that each annotator had to rate both a good and a bad pair with the same verb. We then retained the pairs which both annotators rated either “4” or “5” for a good pair, or “1” or “2” for a bad pair. This resulted in our WSJ2-ann corpus, which we used for additional testing of our models.

| Corpus    | N   | V   | Pairs (Unique) | Train (Unique) | Test (Unique) |
|-----------|-----|-----|----------------|----------------|---------------|
| TRIPS-lex | 487 | 235 | 95592 (59500)  | 76474 (51177)  | 9559 (8871)   |
| WSJ2      | 633 | 512 | 18400 (10048)  | 11029 (7763)   | 1789 (1518)   |
| WSJ2-ann  | 143 | 72  | 204(188)       | n/a            | 204(188)      |

Table 5.3: The data sets for our learning experiments

### 5.5.2 Experimental setup

We originally intended to train our model with the feature set based on the TRIPS feature set, but it is quite large, allowing over  $10^6$  valid feature-value combinations. Summing over all possible feature combinations (as in the example in Equation 5.5) for unannotated data items, and over all possible feature-value combinations in the complete data space, was too inefficient to make training feasible. Therefore, we simplified it by keeping only the top-level values for each feature.<sup>9</sup> The resulting subset is shown in Figure 5.1.

In our symbolic lexicon, we used typed feature lists. To make the representation more uniform for learning, we converted the typed lists into untyped lists by converting a feature list type into a special feature *Main-type* which corresponds to the type of the feature list involved: *Phys-obj*, *Abstr-obj*, *Situation*, *Proposition* or *Time*. To capture the fact that only certain features can be associated with a given type, we placed restrictions on feature value co-occurrences enforced during learning. It requires that all features in the feature set inconsistent with the *Main-type* value are assigned the value “-”. For example, if *main-type* is *Phys-obj*, the only acceptable value of feature *Aspect* is “-”.

Based on this feature set, we built a property set as discussed in Section 5.2, to simultaneously learn the features associated with nouns and selectional restrictions verbs place on noun objects. We then trained three different models. The simplest model is the model using just the *Main-type* feature. The idea behind it is that the feature list type gives the first approximation of selectional

---

<sup>9</sup>Another option would be to use a sampling technique over the complete feature space. It remains part of our future work.

- Common features
  - Information +/-
  - Container +/-
- Phys-obj
  - Form
    - , *Substance, Object*
  - Origin
    - , *Natural-non-human,*
    - human, non-living, artifact*
  - Mobility
    - , *Fixed, Movable*
  - Intentional +/-
- Situation
  - Aspect
    - , *Static, Dynamic*
  - Time
    - , *Atomic, Extended*
  - Cause
    - , *Force, Stimulating, Mental*
  - Trajectory +/-
- Abstr-obj
- Proposition
- Time

Figure 5.1: A simplified feature set used in our learning experiments

restrictions: we cannot move anything but physical objects, for example. We will refer to this model as 1-feature model in our evaluation. The second model we trained has 2 more features in addition to *main-type*: *Aspect* for situations and *Mobility* for physical objects, with the intuitions that they make distinctions likely to be important in many selectional restrictions. We will refer to this model as the 3-feature model in our results. Finally, we trained a 12-feature model with the entire feature set shown in Figure 5.1.

In all our models, we use a Gaussian prior to deal with data sparsity, and we use the 1-feature model to initialize 3 and 12 feature models, as described in detail in Section 5.3, to avoid getting stuck in local minima.

### 5.5.3 Experimental results

To show how learning happens in our model, consider this example from our simplest model, trained on just the type of the feature list associated with the noun, in our TRIPS domain. Table 5.4 shows feature weights learned for verb *remove* and nouns *snowstorm* and *orange* in our one-feature model on the TRIPS dataset which had about 10% of the annotated data included. As can be seen from this table, the model learns a high weight for  $g(\text{remove}, \text{Phys-obj})$ . Looking

at the noun properties, *orange* has the highest weight for *Phys-obj*, while *Snowstorm* has a low weight for *Phys-obj* and a high weight for *Situation*. Thus, this probability distribution will assign a considerably higher probability to  $\langle \text{remove}, \text{orange} \rangle$  than to  $\langle \text{remove}, \text{snowstorm} \rangle$ , which corresponds to the selectional restriction in our lexicon.

| word      | Abstr-obj | Time   | Proposition | Phys-obj | Situation | Other  |
|-----------|-----------|--------|-------------|----------|-----------|--------|
| remove    | -0.146    | -0.143 | -0.217      | 1.165    | -0.221    | -0.121 |
| snowstorm | -0.352    | -0.342 | -0.329      | -0.625   | 1.036     | -0.328 |
| orange    | 0.001     | -0.007 | -0.003      | 0.474    | -0.031    | -0.003 |

Table 5.4: The comparison of feature weights for *remove*, *snowstorm* and *orange*

In this case, we could make the explicit associations between words and features because we were learning with a partially annotated model. *Snowstorm* was annotated with (*Main-type Situation*) in the training dataset, thus a high association weight was learned. Instances of *Orange* were not annotated in the training data, but *Orange* occurred as a direct object of verbs like *move*, which in turn tended to occur with nouns like *helicopter* and *shipment*, tagged as *Phys-obj* in the training set. Thus, the model also learned a generalization to associate *Orange* with the *Phys-obj* feature. This also shows that the model is indeed capable of generalization, using the annotated data as a guide for learning appropriate weights also for items which were not annotated in the training data. In the next section, we will show a more formal evaluation that confirms that the model performs well on unseeded nouns.

### TRIPS-lex evaluation results

Since the TRIPS-lex dataset was artificially generated from the TRIPS lexicon, it was easy to evaluate how seeding impacts our training results, because we could automatically generate a variety of models with the desired amount of seeding. We selected 4 different versions of the training set, with statistics described in Table 5.5. To generate a partially seeded corpus, we randomly selected a subset of noun types, and then deleted features from all verb-noun pairs not containing those nouns. We generated two partially annotated corpora: one in which we kept features on nouns only, corresponding to roughly 8% of total training data set, and another in which we kept features on half of the nouns (including the 40 from the previous training corpus), corresponding to 50% of annotated data set. During learning, the seeds were used as a hard constraint on the features associated with a given pair in the dataset at every step in the learning algorithm.

| Corpus      | Nouns seeded | Pairs seeded | % Total |
|-------------|--------------|--------------|---------|
| Unseeded    | 0            | 0            | 0       |
| Seeded-40   | 40           | 5945         | 7.8%    |
| Seeded-244  | 244          | 37905        | 49.5%   |
| Seeded-full | 485          | 76474        | 100%    |

Table 5.5: Training sets with different amounts of seeding in the TRIPS-lex corpus. The columns show the number of nouns for which seeds were retained, the corresponding number of training data pairs, and the total percentage of data annotated.

Table 5.6 summarizes the learning results from our experiments with the TRIPS-lex data set. In the table, the accuracy is shown as the number of cases where the probability of a good pair was strictly greater than the probability of a bad pair. Perplexity is calculated over the held-out test corpus, which was also used to create the PWSD test set. We report results at three levels of  $\sigma^2$  — 256, which is a very low value in our experiments; 2048, which is an average value, and the best  $\sigma^2$  value we could find which maximizes the probability of the development set. The latter varies depending on the number of features and seeding in the model. We also report results with  $\sigma^2 = -\infty$ , which corresponds to the IM algorithm without the prior.

Different levels of seeding are separated by double lines in the table. The data show that with little seeding adding more features is helpful, but the prior does not help much: models without a prior (that is, with infinite  $\sigma^2$ ) perform just as well as models with large prior values, and smaller values of  $\sigma^2$  make the results worse. The dependency on seeding is summarized in Figures 5.2 and 5.3. The graphs show two trends: the performance for all models improves as the  $\sigma^2$  parameter is relaxed, and as more features are added. We discuss these results, and possible reasons for this behavior, in Section 5.6.

One potential problem with this evaluation measure is that there may be a number of PWSD contrastive pairs with probabilities so close to be almost indistinguishable, and the judgment the model makes on this pair cannot therefore be considered reliable if we need to compare two probabilities, for example, to rule out a parse or to select the best parse among multiple alternatives. Figure 5.4 is based on our Seeded-40 model. The figure shows the percentage of the cases where the difference in good and bad probabilities is higher than a specified percentage  $x$  shown on the horizontal axis. If  $x = 0$ , it means that the difference between probability values within a PWSD pair is greater than 0%, so all PWSD pairs are covered. When  $x = 200$ , then the difference between the probabilities in a PWSD pair should be 200%, and the  $y$  axis shows that, for example, for the model with  $\sigma^2 = 2048$  and 12 features around 70% of the PWSD pairs

| Training data      | Features    | $\sigma^2$    | Test size   | Accuracy    | Perplexity   |
|--------------------|-------------|---------------|-------------|-------------|--------------|
| Unseeded           | 1           | 256           | 3507        | 78.09       | 111100       |
| Unseeded           | 1           | 2048          | 3507        | 79.08       | 106867       |
| Unseeded           | 1+12        | 256           | 3507        | 78.1        | 110641       |
| Unseeded           | 1+12        | 2048          | 3507        | 89.4        | 84996        |
| <b>Unseeded</b>    | <b>1+12</b> | <b>524288</b> | <b>3507</b> | <b>91.3</b> | <b>62096</b> |
| Unseeded           | 1+12        | inf           | 3507        | 91.4        | 63419        |
| Seeded-40          | 1           | 256           | 3507        | 78          | 112866       |
| Seeded-40          | 1           | 2048          | 3507        | 80.5        | 103942       |
| Seeded-40          | 1           | 524288        | 3507        | 90.8        | 66349        |
| <b>Seeded-40</b>   | <b>1</b>    | <b>inf</b>    | <b>3507</b> | <b>90.9</b> | <b>63563</b> |
| Seeded-40          | 1+12        | 256           | 3507        | 78.4        | 109038       |
| Seeded-40          | 1+12        | 2048          | 3507        | 89.1        | 85159        |
| <b>Seeded-40</b>   | <b>1+12</b> | <b>524288</b> | <b>3507</b> | <b>91.5</b> | <b>63752</b> |
| <b>Seeded-40</b>   | <b>1+12</b> | <b>inf</b>    | <b>3507</b> | <b>91</b>   | <b>62883</b> |
| seeded-244         | 1           | 2048          | 3507        | 87.4        | 94845        |
| seeded-244         | 1           | 524288        | 3507        | 90.9        | 67616        |
| <b>seeded-244</b>  | <b>1</b>    | <b>inf</b>    | <b>3507</b> | <b>90.9</b> | <b>65848</b> |
| seeded-244         | 1+12        | 2048          | 3507        | 89.8        | 81361        |
| <b>seeded-244</b>  | <b>1+12</b> | <b>524288</b> | <b>3507</b> | <b>90.9</b> | <b>63064</b> |
| <b>seeded-244</b>  | <b>1+12</b> | <b>inf</b>    | <b>3507</b> | <b>91.2</b> | <b>62944</b> |
| Seeded-full        | 1           | 2048          | 3507        | 90.4        | 88311        |
| Seeded-full        | 1           | 524288        | 3507        | 90.6        | 67991        |
| Seeded-full        | 1           | inf           | 3507        | 90.8        | 67646        |
| Seeded-full        | 1+12        | 2048          | 3507        | 91.2        | 74519        |
| <b>Seeded-full</b> | <b>1+12</b> | <b>524288</b> | <b>3507</b> | <b>90.9</b> | <b>61726</b> |
| Seeded-full        | 1+12        | inf           | 3507        | 91.3        | $-\infty$    |
| Hofmann            | Unseeded    | 6 clusters    | 3507        | 73.4        | 117592       |

Table 5.6: PWS test accuracy on the TRIPS-lex set. “1” in features column denotes models using a single *Main-type* feature, and “1+12” denotes a 12 feature models initialized from the 1-feature model. The differences in PWS accuracy of more than 1.6 percentage points are statistically significant at the 0.05 level. The best results are highlighted in bold. Hofmann row describes the performance of the class-based Hoffman model.

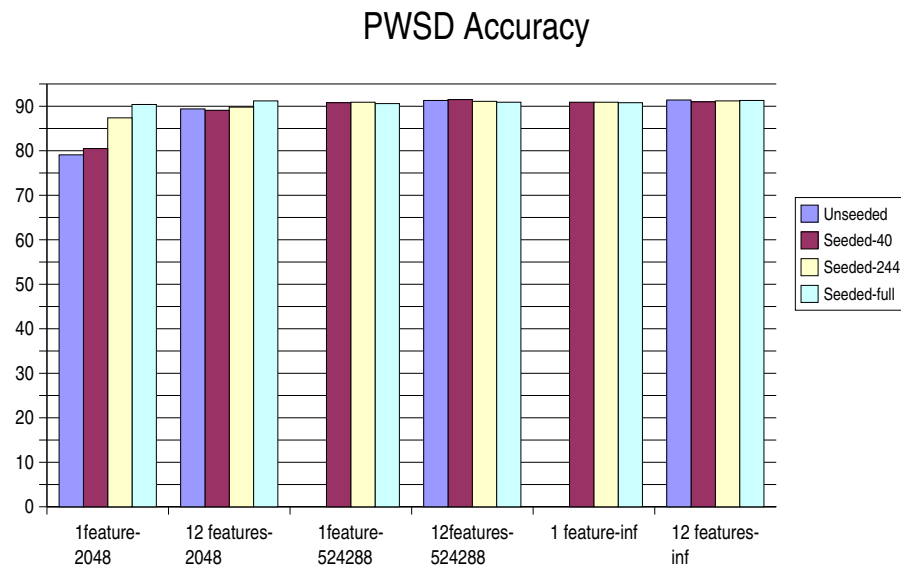


Figure 5.2: Pseudo word sense disambiguation results on TRIPS-lex dataset depending on seeding

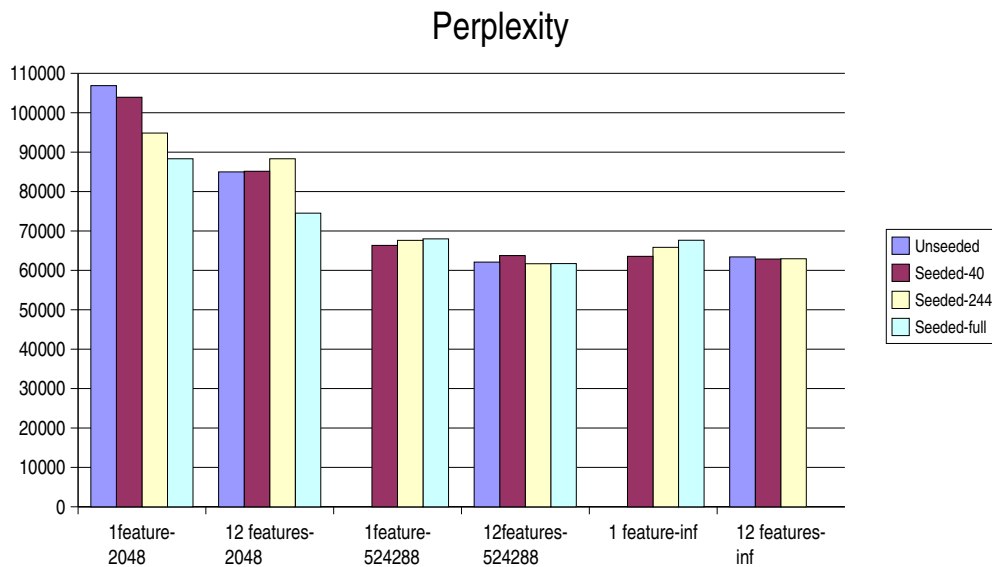


Figure 5.3: Perplexity evaluation results on TRIPS-lex dataset depending on seeding

have a difference between a “good” and a “bad” probability of 200% or more. Obviously, and as the difference increases, the number of pairs which differ that much decreases.

The graph gives an idea of how trustworthy the numbers in Table 5.6 are. For example, for our models with  $\sigma^2 = 256$  with 1 feature no PWSD pairs have more than 25% difference in probabilities. This means that if we try to make good/bad judgments using this model, they will not be very reliable, because the difference will frequently be in the last digits of probabilities, which may not be very reliable due to round-up errors. In contrast, for  $\sigma^2 = 2048$  with 12 features, around 95% of the PWSD pairs have at least 25% difference in probability, and therefore judgments made by that model are likely to be reliable.

Note that for infinite  $\sigma^2$ , a large percentage of bad pairs has 0 or near-0 probability (this arises from over-fitting without a prior, as discussed in Section 5.3), therefore almost 80% of PWSD pairs are shown as having the difference of 400% or more. For small  $\sigma^2$ , the derived distribution has relatively small differences between different data pairs, because weights are forced closer to 0 by the prior.

Given that model coverage is important, another way to compare the models is to show how the accuracy is related to coverage, which is a way of seeing tradeoffs between the precision and recall of a given method. Figure 5.5 graphs the accuracy of the algorithm in contrast with the percentage of pairs it can distinguish. As it illustrates, all models do better if only the pairs with higher difference are taken into account, and therefore there is a tradeoff between deciding whether the two pairs are distinguishable by a given model, and the accuracy of the decision, or to balancing precision and recall of a given model.

For space reasons, we only provide accuracy and coverage graphs for our seeded-40 model in this chapter. Other models behave similarly on this evaluation.

We compared our model to a pairwise data clustering model of Hofmann [Hofmann, 2001], using the PennAspect toolkit[Schein *et al.*, 2002]. We trained the model with 6 clusters, which should produce performance comparable to our performance with 1 feature model, which had a single feature with 6 values. The resulting accuracy was on the TRIPS-lex dataset was 73.4%, comparable to our 1-feature models with priors, and lower than any of our 12 feature models. This comparison shows that adding more parameters to the model is helpful. However, it is not complete, and we discuss it in more detail in Section 5.6.

The experiments we report here included data sets with different amounts of seeding. To make sure that seeding helps, we compared the results of learning on seeded versus unseeded data in our training set. For that purpose, we took our seeded-244 set, and selected two subsets

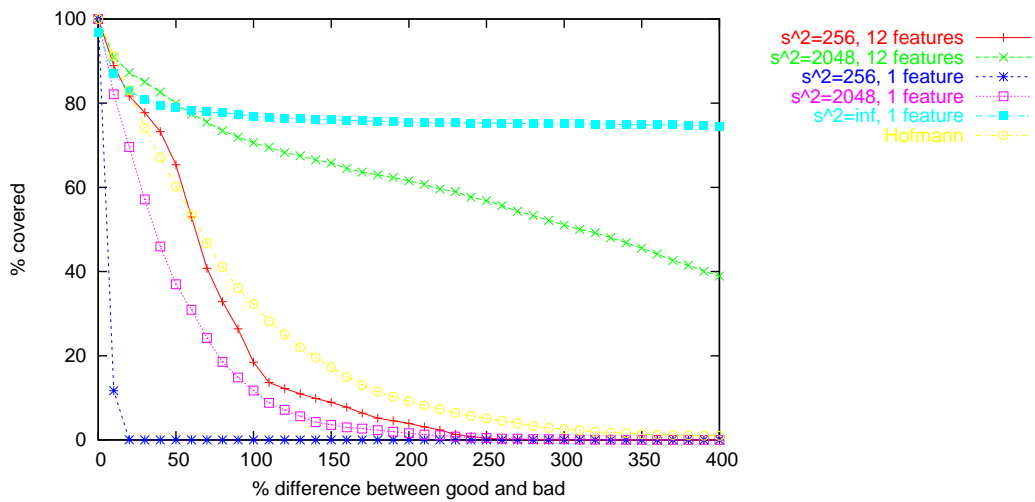


Figure 5.4: Coverage graph for TRIPS-lex dataset seeded-40 model. The  $x$  axis plots the percentage difference between the good and bad probabilities in test pair. As it increases, the number of pairs with that much difference in probability decreases.

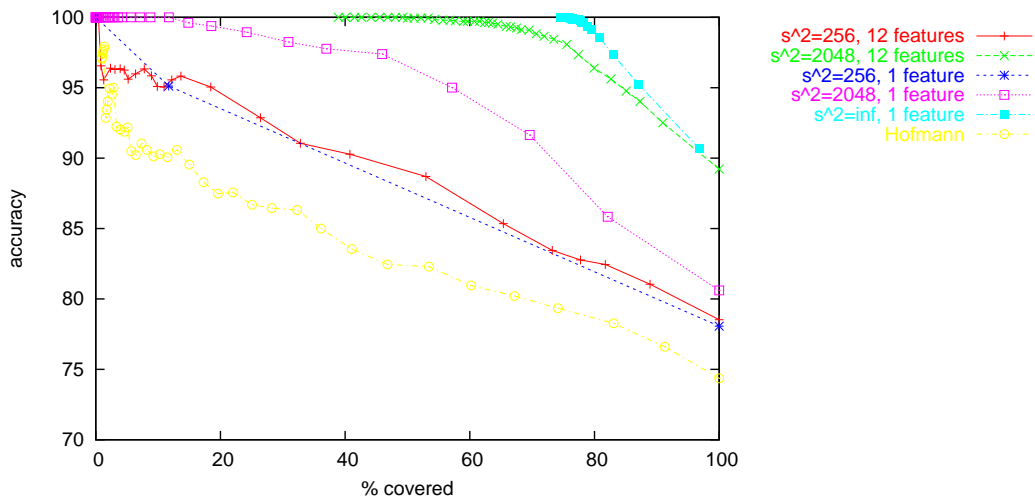


Figure 5.5: Coverage vs. accuracy graph for TRIPS-lex dataset seeded-40 model. The  $x$  axis plots the % of pairs covered (recall), the  $y$  axis — the accuracy at that level (precision)

from our test data: one in which both the noun in the “good” PWSD pair, and the noun in the “bad” PWSD pair had seeds in the training data, and another in which both nouns didn’t have seeds in the training data. There were 843 PWSD pairs for which both nouns were seeded, and 842 PWSD pairs for which both nouns were unseeded. We then evaluated the PWSD accuracy, and the perplexity of the model on those sets. The results are presented in Table 5.7.

| Features | $\sigma^2$ | PWSD accuracy |        | Perplexity |        |
|----------|------------|---------------|--------|------------|--------|
|          |            | unseeded      | seeded | unseeded   | seeded |
| 1        | 2048       | 85.3          | 90.7   | 107206     | 82003  |
| 1        | 524288     | 90.3          | 91.2   | 69172      | 65036  |
| 1        | inf        | 91.3          | 92.4   | 65563      | 64955  |
| 12       | 2048       | 90.0          | 92.2   | 74044      | 71617  |
| 12       | 524288     | 89.9          | 92.5   | 61948      | 60054  |
| 12       | inf        | 90.5          | 91.4   | 64030      | 61252  |

Table 5.7: Evaluation on seeded and unseeded data with the seeded-244 model. The “Unseeded” columns contain PWSD accuracy and perplexity calculated over the test pairs for which the nouns did not have seeds in the training data; the “Seeded” column is over test pairs for which the noun objects had seeds in the training data. Differences of PWSD accuracy over 2.3 percentage points are statistically significant at the 0.05 level.

This evaluation makes it clear that seeding helps with learning on unseeded data. On the test pairs for which there were no seeds in the training set, our model performs significantly better than chance, thus, it clearly learned some good weights for all nouns, not just those for which there were seeds. Note that the pairs from the test data never appeared in the training set, and all our properties depend only on the features associated with words, thus the model is learning some non-trivial feature based properties for all words in the training data.

The data also underscore that increasing the value of  $\sigma^2$  significantly improves the results, both on seeded and on unseeded subsets. The comparison is not as clear for the large  $\sigma^2$  in comparison with the model with no prior (infinite  $\sigma^2$ ). The perplexity is slightly better without prior for 1 feature models, and slightly worse for 12 feature models. Also, for 12-feature model without prior, both seeded and unseeded data perform equally well, but there is a statistically significant difference in PWSD accuracy if a prior is added. This is consistent with the overall evaluation results we showed earlier, and adds additional evidence that adding a prior was not helpful on this data set. In the next section there will be some examples where adding a prior was helpful in evaluation, and we will return to the question of the usefulness of smoothing with

priors in Section 5.6.

### WSJ evaluation results

Table 5.8 summarizes the learning results from our experiments with the WSJ data set. The training dataset consisted of 11029 tokens (7763 unique verb-object pairs), and we manually annotated 5998 tokens (3984 unique pairs), corresponding to 54% of the corpus. Since manual annotation is very time-consuming, we only annotated the pairs with the values for *Main-type* feature.

Using our training set, we trained a baseline model with a single *Main-type* feature, a model with 3 features (*Main-type*, *Mobility* and *Aspect*), and a full model with 12 features. Table 5.8 shows the evaluation results for those models, and Table 5.9 shows the same testing results on a human-annotated data corpus described in Section 5.5.2. Figures 5.6 and 5.7 show the coverage vs. accuracy plots on our WSJ2 set (see previous subsection on the discussion of how to interpret the graphs).

Similarly to the TRIPS-lex results, we report the results for a small  $\sigma^2 = 300$ , the average  $\sigma^2 = 2400$  and for the best value of  $\sigma^2$  we could find for a given training set. To find an optimal value of  $\sigma^2$  we started with a fairly small  $\sigma^2 = 150$ , trained the initial model, gradually increased the value of  $\sigma^2$ , using the results of training with each consecutive  $\sigma^2$  as the initialization for the run with the next value. We stopped increasing  $\sigma^2$  when training would decrease the perplexity of the development data compared to the results obtained at the previous level. With 12 features, the best value of  $\sigma^2$  was 9600. With 3 features, the best value was  $\sigma^2 = 19200$ . We report the results both for 19200 and 9600 for comparison purposes. To check that this  $\sigma^2$  was indeed close to the best possible value, we trained our models with  $\sigma^2 = 1e8$  and random initialization, and the results showed increased perplexity and worse PWSD results compared to our optimal models.

We compared our results with the results obtained from training Hofmann’s model with 6 clusters. A more detailed discussion is in the next chapter, but as the table demonstrate here, the performance of our best models was similar to the Hofmann’s model, which was simpler (only 6 clusters), and considerably faster to train. The differences in PWSD accuracy are not statistically significant, though the perplexity of our best model (12 features at  $\sigma^2 = 9600$ ) has perplexity 13% lower than the Hofmann’s model. The tables and graphs also show that for larger number of features (3 or 12), adding a prior has been helpful for learning, and we discuss that in the next section.

| Model          | Features          | Iterations | $\sigma^2$   | Test size   | Accuracy    | Perplexity    |
|----------------|-------------------|------------|--------------|-------------|-------------|---------------|
| basic          | 1                 | 100        | 2400         | 1789        | 57,9        | 142309        |
| basic          | 1                 | 500        | inf          | 1789        | 62.1        | 2971743041    |
| basic          | 1+3               | 100        | 300          | 1789        | 53,6        | 155510        |
| basic          | 1+3               | 400        | 9600         | 1789        | 64.1        | 126722        |
| <b>basic</b>   | <b>1+3</b>        | <b>400</b> | <b>19200</b> | <b>1789</b> | <b>65.5</b> | <b>133355</b> |
| basic          | 1+3               | 100        | inf          | 1789        | 63.8        | +Infinity     |
| basic          | 1+12              | 100        | 300          | 1789        | 55,3        | 137007        |
| <b>basic</b>   | <b>1+12</b>       | <b>100</b> | <b>9600</b>  | <b>1789</b> | <b>66.2</b> | <b>126430</b> |
| basic          | 1+12              | 100        | inf          | 1789        | 60,9        | 152486        |
| <b>Hofmann</b> | <b>6 clusters</b> | <b>n/a</b> | <b>n/a</b>   | <b>1789</b> | <b>67.1</b> | <b>144821</b> |

Table 5.8: PWSD test accuracy on the WSJ2 set. The table shows that adding a prior improves both PWSD test accuracy and model perplexity. The differences in PWSD accuracy of more than 2.6 percentage points are significant at 0.05 level. The best 3 and 12 feature models and the comparable class model are highlighted in bold. The difference in PWSD results for those models is not statistically significant at the 0.05 level

| Model          | Features          | Iterations | $\sigma^2$  | Test size  | Accuracy    | Perplexity |
|----------------|-------------------|------------|-------------|------------|-------------|------------|
| basic          | 1                 | 18         | 300         | 102        | 57.8        | 232665     |
| basic          | 1                 | 100        | 2400        | 102        | 65.7        | 149817     |
| basic          | 1                 | 500        | inf         | 102        | 68.6        | 349686059  |
| basic          | 1+3               | 100        | 300         | 102        | 58.8        | 162588     |
| basic          | 1+3               | 100        | 2400        | 102        | 68.6        | 138372     |
| basic          | 1+3               | 400        | 9600        | 102        | 73.5        | 131268     |
| basic          | 1+3               | 100        | inf         | 102        | 72.5        | 4819051774 |
| basic          | 1+12              | 100        | 300         | 102        | 60.7        | 146018     |
| basic          | 1+12              | 100        | 2400        | 102        | 66.7        | 133550     |
| <b>basic</b>   | <b>1+12</b>       | <b>100</b> | <b>9600</b> | <b>102</b> | <b>73.5</b> | 124382     |
| basic          | 1+12              | 100        | inf         | 102        | 67.6        | 157335     |
| <b>Hofmann</b> | <b>6 clusters</b> | <b>n/a</b> | <b>n/a</b>  | <b>102</b> | <b>81.4</b> | 129097     |

Table 5.9: PWSD test accuracy on the WSJ2-ann set. The differences of more than 10 percentage points are statistically significant.

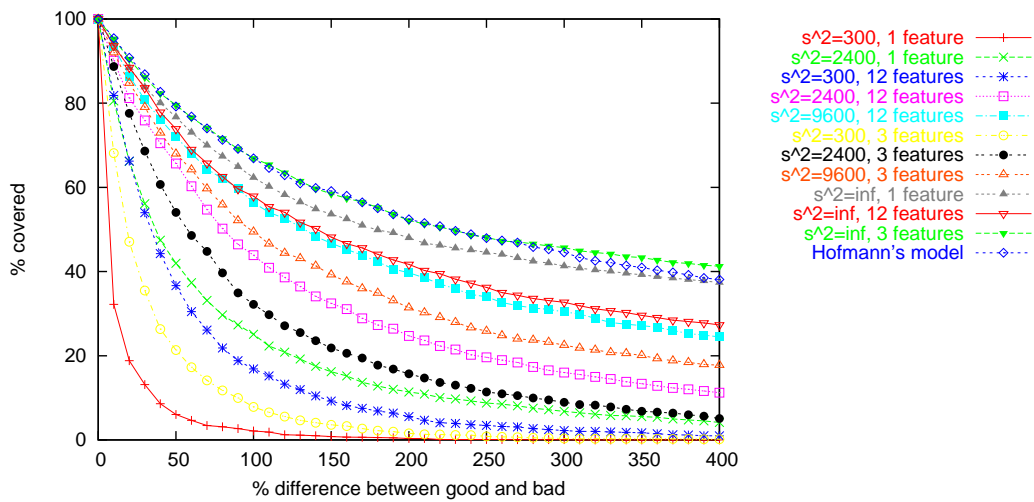


Figure 5.6: Coverage vs. accuracy graph for WSJ2 data set. The  $x$  axis plots the percentage difference between the good and bad probabilities in test pair. As it increases, the number of pairs with that much difference in probability decreases.

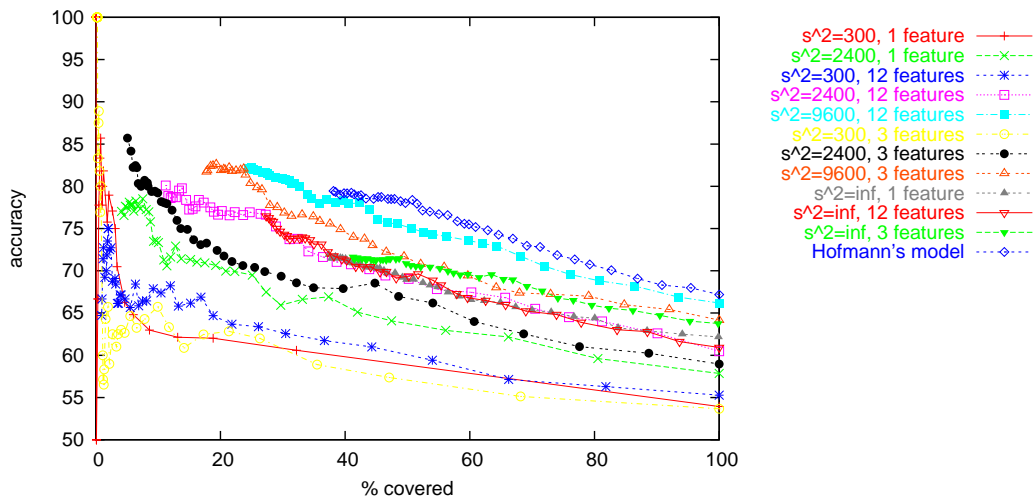


Figure 5.7: Coverage vs. accuracy graph for TRIPS-lex data set. The  $x$  axis plots the % of pairs covered (recall), the  $y$  axis — the accuracy at that level (precision)

On the human data set, the differences between most models are not statistically significant. The 3 feature model without prior performs significantly better than the rest, and Hofmann’s model is significantly better than all other models. However, it is interesting to note the big difference in performance between automatically generated and human annotated datasets. When we collected the human annotated data, on the instances where the annotators agreed, they tagged “good” 60% of the pairs marked “bad” in our PWSD corpus. The considerably improved performance of all models on the human annotated dataset is evidence that our automatically generated test data may not be the best possible way to evaluate the results, or that a more complex criterion for distinguishing good and bad pairs on the PWSD test is necessary.

## 5.6 Discussion and Future Work

The results our model shows on the TRIPS-lex dataset were promising, because it is clearly a non-trivial data set, where our algorithm performed better than the Hofmann’s algorithm. We believe that it was because Hofmann’s algorithm got stuck in a local maximum, and our model had the advantage of being tailored to the domain, because it uses a simplified version of the feature set used to generate the TRIPS-lex data set. We have done some experiments that showed that our TRIPS-based feature set produced considerably better results than using a set of binary features which provided approximately the same number of feature combinations consistent with each seen data piece. However, more research needs to be done to confirm that this comparison has been fair to Hofmann’s model, since we only run it with 6 clusters. We attempted another run with 572 clusters, which exactly the same number as the number of permissible feature-value combinations in our 12-feature model. However, training has been very slow, and after 2 weeks running we were forced to stop the training without obtaining a result<sup>10</sup>. Optimizing the number of Hofmann clusters by cross-validation would produce the strongest results, which may be a fairer comparison to our model on the TRIPS-lex dataset.

Given the performance of our model on WSJ corpus data data, where a Hofmann’s clustering algorithm outperformed our best models (and was an order of magnitude faster to train), it appears that our models may be best suited for the domains where we have a good model of the feature set involved. One possibility of why our models were handicapped on the Wall Street Journal corpora is that our data contained a lot of abstract objects, and in our feature set if *Main-type* was set to *Abstr-obj*, only 2 binary features (*Information* and *Container*) could

---

<sup>10</sup>The training was done on a 1.8Hz 2-processor machine with 1G RAM. Part of the slowness might be due to the fact that PennAspect toolkit is written in Java

have values other than “-”, compared to 6 two-valued or 3-valued features for physical objects and situations. This was initially motivated by the structure of feature set in our domains, because we originally planned to obtain distributions interpretable in terms of our symbolic features through having some fully annotated data items. In the future, adding several more features which can vary for abstract objects might bring better results by allowing the learned model to distinguish more subclasses of abstract objects, even though features have no *a priori* interpretation, and would not be connected to our symbolic lexicon.

Another possibility is that our property set was not appropriate for modeling the WSJ data. It may be too impoverished to represent the distribution properly. We use a very simple property set, while in Section 5.2 we discussed a much wider variety of properties which can be used to better represent the data. The advantage of log-linear models is that additional properties can be factored in easily. The disadvantage of adding extra properties is the increase in the number of parameters, and thus the complexity of the learning space. We discussed ways to mitigate these effects, which include feature selection and deterministic annealing, in Section 5.3. Using a more appropriate model can help us improve results, as evidenced by our results on the TRIPS data set, where the model closely matched the original data. We are considering these as part of our future work.

It is also possible that our training sets were too small for the number of properties we are using in the model. Many clustering algorithms are training on the data sets 10 times the size of ours, and certainly we saw better performance on the larger TRIPS-lex set. An analysis of the errors made by the model is necessary to determine if this is the case.

Finally, the results of the IM algorithm are strongly dependent on initialization, because there are many local maxima. Hofmann’s clustering algorithm uses deterministic annealing to deal with the local maxima problem, and it apparently produces better results. Our algorithm, too, could benefit from a similar technique. A different learning method, for example, learning using Latent Maximum Entropy principle [Wang *et al.*, 2002], may also be helpful.

The results also demonstrate that the model performance on our PWSD task is not necessarily correlated with the decrease in perplexity of the test data. Some of our better-performing models have the highest perplexity. In particular, the model trained on 3 features without a prior has infinite perplexity because it actually assigns 0 probability to one of the data pairs in the test set, but at the same time it has the third best performance among our models. In terms of perplexity reduction, our 12-feature seeded model is considerably better than Hofmann’s model, even though those results are not consistent with the PWSD evaluation results. This is a common problem in speech and natural language processing, which is why perplexity

is not considered the best possible measure of performance. Yet the PWSD test is also only an approximation of the real task of using probabilities in the corpus, and we may need a more complete evaluation within an integrated parsing model to determine which of these models performs best in parsing tasks.

The results on using the prior to smooth the data have also been mixed. Adding the prior was not helpful in our TRIPS-lex data sets. However, this may be influenced by the particular properties of the training set. First, the TRIPS-lex dataset is nearly complete, so there is no real problem with data sparsity and no smoothing is required. Second, as we noted earlier, the data in the TRIPS-lex corpus are distributed nearly uniformly, while in real corpora there will be a large number of infrequent items for which smoothing is helpful. In contrast, adding a prior to our WSJ data sets has been helpful for models with large number of features: for example, at  $\sigma^2 = 9600$ , the 12 feature model performs significantly better than the 12 feature model with  $\sigma^2 = \infty$ , on PWSD tests and especially on perplexity reduction. At the same time, it is important to note that there are no significant improvement in performance for our 3 feature model when a prior is added. This suggests that, possibly, the 12 feature model gets stuck in a local maximum because there are many more maxima with more features, and using annealing or some other method for avoiding local maxima might improve performance of the model.

It is interesting to note that the performance of the fully annotated models on our TRIPS-lex dataset was not much different from the performance of partially annotated models. The local maxima problem does not exist for fully annotated models. Therefore, the results obtained there should be the global maximum achievable with the model. The results are clearly better compared to the unseeded models, but there is no big difference in performance for completely and partially seeded models. If this cannot be explained by local maxima or overtraining, the other possibility, which we already discussed, is that the model structure is too limited to describe the dataset properly. In particular, we may be experiencing problems with sense ambiguity, because our properties depend on verb tokens, not verb senses. Adding properties that reflect selectional restrictions for verb senses (as described in Section 5.2) might also improve the model performance.

The unrealistic frequencies in the TRIPS-lex corpus may also be the reason why seeding didn't appear to make much difference in training. It is possible that a simple guess about structure is sufficient to describe the model when the word frequencies are uniform. In contrast, we tried some unseeded models on our WSJ corpus. We do not present the results there, because due to several differences in the models the comparison is not direct. However, it is worthwhile to note that these models performed at chance levels, and it appears that seeding

with a even a single feature contributed to a very significant improvement to produce the WSJ results presented here. A more formal evaluation needs to be conducted to see the impact of seeding on the WSJ corpus, and is planned as future work.

### 5.6.1 Integration with symbolic parsing

While we discussed a probabilistic model for selectional restrictions in this chapter, a question which remains open is how to integrate such model fully in a parsing model we discussed in the previous chapters. There are several options possible for integration. Ideally, we could have a lexicalized PCFG trained from the corpus for the whole parser, as was done by Seagull and Schubert [1999]. If this is the case, the selectional restrictions we described can be integrated as part of the general probabilistic model as follows. Assume that during parsing we need to evaluate the probability of a VP node *send a truck*. It may have several possible expansions, and we want to calculate a probability of it being expanded into a verb *send* followed by a direct object headed by the word *truck*. This can be seen as product of two probabilities: that *send* has a direct object, and that *truck* fills the direct object role. The second factor in this product is in fact  $P(\textit{truck}|\textit{send}, \textit{direct-object})$ , which we can obtain by our method. The first factor, the probability of *send* being a transitive verb with a direct object, can be learned separately, smoothed over many possible direct objects of *send*.

The advantage of integrating our method into a PCFG grammar is that it is theoretically well-grounded and can provide a uniform probabilistic parsing model. There are two possible disadvantages. First is the availability of training data, and second is the integration with domain-specialization methods from Chapter 4.

Training a full probabilistic grammar requires a training corpus annotated with appropriate parses. This is difficult to obtain from scratch (as discussed in detail in Section 3.6.2), because we need parses suitable for semantic interpretation, not available from the existing parsers. We are considering another approach, training a probabilistic grammar with our rule set and a corpus of parsed and hand-checked utterances. Domain customization can be used to jump-start the parsing process in the domain, and to build parsed corpora with the help of customized parser, as we have done in the Monroe and Medication Adviser domains [Swift *et al.*, pear]. Once a sufficient amount of parsed data has been collected, a probabilistic model can be trained on this information. Then since verb-object, verb-subject and other relationships can often be extracted from unparsed or partially parsed corpora even if full parses are unavailable (see for example [Lee, 1997]), we can enhance the (unlexicalized) model by combining it with a lexicalized probabilistic

model for those relationships.

One more advantage to this model is that it allows for easier integration of domain customization with probabilistic models. In effect, we proposed two separate ideas in this thesis about selectional restrictions: using domain customization to optimize them to the domain, or using a probabilistic model. They can co-exist in different ways. Domain customization is clearly necessary to produce logical forms for reasoning, but the lexicon specialization can be treated differently in that process. We discussed above using the specialized parser as a tool to create a corpus for use in training probabilistic models.

We may also decide not to build a full probabilistic grammar, but have our selectional restriction model function simultaneously with the probabilistic model in the CAFÉ architecture [Stoness, 2001]. In this architecture, instead of treating selectional restrictions as strict rules, we change to the advisor architecture, where multiple advisors weigh in on the goodness of a given constituent. Currently these include selectional restrictions and reference feedback. The probability from a probabilistic grammar can be added as another advisor contributing to the overall score assigned to the constituent.

In the integration process we need to consider the same questions which we asked about strict selectional restrictions, namely, “what does it mean to satisfy a selectional restriction and what do we do for special cases, such as conjunctions and pronouns”. The answer depends to some extent on the way the integration is accomplished. If the selectional restrictions are fully replaced by probabilities, the question about satisfying the restrictions is re-cast in terms of “what does a probability mean as a selectional preference”. We already discussed possibilities: the probability can be part of the lexicalized model, thus impacting the score of the parse; or the probability can be used as part of the advisor architecture. The latter is an open research issue. The simplest way to use probabilities is to have a threshold on  $P(n|v)$  (similar to [Grishman and Sterling, 1992]), and consider the restriction satisfied if the probability is above threshold. This is a rather ad-hoc approach, though, which does not fully utilize the advantages of having a number as opposed to a binary yes/no judgment. A better approach would be to define a scoring function which uses the probability as part of a total score of the constituent, and which can be hopefully derived from corpus data. Or we could even imagine feedback from different advisers as property functions in the general log-linear model for selectional restrictions.

Handling pronouns can be very similar in our symbolic and statistical model during parsing. As long as we are using the model as we developed it here, dependent only on features assigned to words, pronouns are just regular items with features assigned by our symbolic algorithms and then used in the probability calculation. Conjunctions are trickier because now we are not

handling a word pair, but a verb and a constituent with two heads. We have not addressed this explicitly in our model. Traditionally, just one head was selected for probability calculation [Collins, 1999]. We may also consider averaging the probabilities from both heads, or taking the lower one as the real probability. The latter can be seen as parallel to the collective set situation, where every element of the set has to satisfy the restriction to be valid, so the lowest probability would be representative of this approach. Exploring the integration issues discussed in this section and determining the best ways to integrate our probabilistic and non-probabilistic models are part of our future work.

## 5.7 Conclusions

In this chapter, we described a statistical model for learning selectional restrictions from corpus data with a feature-based representation. We evaluated our model on two data sets: a TRIPS-lex dataset generated explicitly from a feature-based representation, and a WSJ dataset extracted from WSJ corpus data. We have demonstrated on the TRIPS-lex dataset that our model is capable of learning a probability distribution which performs well both on perplexity evaluation and on PWSD data sets. We showed that adding a Gaussian distribution improves the results on sparse data sets. The results on the WSJ data were mixed: our models performed better than state-of-the-art clustering model at reducing the perplexity of the data, but they didn't perform equally well on our PWSD evaluation. We discussed the possible reasons for this, and outlined some things that can be done to improve the results in the future.

## 6 Conclusions and Future Work

The main goal of this thesis was to develop components to support parsing and semantic interpretation in multi-domain medium scale natural language understanding systems. This is a class of applications where a domain is large enough to require a complex knowledge representation for reasoning, but yet small enough to provide a reasonably sized domain and task models which can be used as sources of domain-specific semantic information. The example we used throughout this thesis is the TRIPS dialogue agent [Allen *et al.*, 2001], but the same problems are important for a larger class of applications, including tutoring systems, spoken machine translation systems, and other applications which require reasoning and intention recognition to interact with users.

Much of the existing work in the area concentrated on either single domain systems, or systems for multiple small domains [Dowding *et al.*, 1994; Seneff, 1992; Chang *et al.*, 2002]. Our work is concerned with the additional issues which arise in developing a multiple domain system in a medium scale application. We formulated a specific set of requirements for our applications. These are the following

- handling complex natural language, which requires analysis with a deep linguistically motivated parser;
- fast parsing, to enable real-time responses;
- using as much of domain-independent information as possible to improve portability and speed up system development;
- producing a semantic representation suitable the domain-specific reasoning;
- accurate semantic disambiguation.

There are many aspects to developing an interpretation system that satisfies these requirements. Our main focus was on the representation of semantic information, which is crucial to achieving all of the above goals. We started with a bottom up chart parser and a domain-independent unification grammar as a basis for our development, in order to provide deep natural language parsing. There are known problems with using deep parsers in practical systems, the most important being speed and disambiguation accuracy. We chose to implement a selectional restriction mechanism to control the parser search space and improve disambiguation results, and concentrated on developing a computational lexicon and ontology to provide the semantic information necessary during parsing.

For that purpose, we took a semantic feature representation, a well known representation in computational linguistics, and adapted it to suit our needs. We developed a feature set for use in selectional restrictions in our system. The evaluation showed that parsing with selectional restrictions expressed in our lexicon brings two-fold increase in speed and a significant increase in parsing accuracy on real language corpora collected in human-human experiments and from users interacting with a system. Our experience with developing our feature set provided insights on the advantages and disadvantages of using features as a lexical semantic representation, which also have an impact on development of any systems using semantic information, whether it is a feature set or a more traditional frame-based ontology. These include a balance between the expressiveness of the system, and the difficulty of maintaining it, especially as relevant for selectional restrictions; and the guidelines for including new feature in the set based on its disambiguating power.

In addition, we provided formal semantics for feature operations by adapting theory of typed feature structures from Carpenter[Carpenter, 1992] and Copestake[Copestake, 1992]. The formal analysis showed that there are at least two different operations which can be used to check selectional restrictions, and that the choice of the operation has a direct impact on handling of complex constructs which appear in our domains, in particular, conjunction. These results transfer directly on any unification-based system using selectional restrictions, regardless of the specific representation used for restrictions. We outlined the issues that arise in using either subtype operation or unification to check selectional restrictions, proposed solutions to some of them, and provided a formal analysis which can be used to make implementation decisions in other systems.

We then described a domain-independent ontology which combines our feature list representation with a domain-independent linguistic frame representation similar to the FrameNet representation [Johnson and Fillmore, 2000]. The FrameNet database does not contain map-

pings between syntactic structures and frame elements, and, as we discussed, is difficult to use directly as a lexical semantic representation. Therefore, we developed our own ontology in which verb and noun senses are compatible with FrameNet divisions, but which has hierarchical structure and a set of semantic roles easy to use in a parsing lexicon. Based on our LF ontology, we developed a domain-independent lexicon and grammar which use selectional restrictions to improve speed and accuracy. We demonstrated that our selectional restrictions significantly improve parsing speed and accuracy in two different application domains.

The lexicon we developed has been used in four different domains, and we conducted a formal evaluation on human corpora collected in two of the domains. Our parser achieves 80% sentence accuracy on many of the dialogues, which contain complex syntactic constructions, and provides a good basis for current development of conversational assistants in these domains. This is a high figure for human-human dialogues, which are much more complex than usual human-computer dialogues, and contain many sources of additional complexity such as valid utterance fragments which are not complete sentences.

The development of our domain-independent ontology and lexicon address the issues of portability, handling complex language, and re-using syntactic and semantic information between the domains. To address speed and accuracy problems, and to produce semantic representations customized to the domain, we developed a two-level architecture which separates domain-independent and domain-specific information. Domain-specific representations and semantic information are localized in back-end reasoners, and chosen for maximal speed and portability. We showed how these needs conflict with the needs of a domain-independent language interpretation system. To resolve the conflict, we developed an architecture in which domain-independent and domain-dependent ontologies are maintained separately, and linked with LF-KR transforms. The transforms achieve three main goals: they allow us to produce representation in the target KR language used by the reasoners; they allow us to produce representations using domain-specific ontologies needed for reasoning; and they link domain-independent and domain-specific information in our lexicon. The latter allows us to significantly speed up parsing and improve accuracy on speech lattices in two of our domains, at a relatively small cost of defining a set of transforms of size an order of magnitude smaller than the size of our domain-independent lexicon.

Finally, we experimented with learning selectional restrictions from corpora. We developed a statistical model based on the Iterative Maximization algorithm [Riezler, 2000] and augmented it with a Gaussian prior similar to [Chen and Rosenfeld, 1999] to handle data sparsity problems. We proved that our combined IM-Gaussian algorithm converges to a critical point of the corpus

likelihood, and used the algorithm to train several statistical models on data derived from the TRAINS lexicon and from Wall Street Journal corpora. Our evaluation showed that adding a prior improves the results compared with the original IM algorithm. However, the performance of our method on real data is worse than that of a state-of-the-art clustering algorithm[Hofmann, 2001]. This may be due to getting stuck in local maxima, or to insufficient number of properties in the model. We proposed a set of improvements to the algorithm to boost its performance, planned as part of our future work.

Using a feature representation has been important to success in several aspects of this work. Typed feature lists have better computational properties than unrestricted multiple inheritance, and their extensibility makes it very convenient for use in a general domain-independent lexicon that evolves over time. Moreover, the ability to add new features and specialize feature values is key to our specialization algorithms that provide easy and transparent integration of domain-independent and domain-specific information in our corpora. In combination with our general feature inference mechanism, our feature representations also allow us to transparently combine domain-independent and domain-specific entries in the lexicon, and have the domain constraints influence parsing even if we know specializations for only some of the words in the sentence.

If we can parse sentences which contain out of domain words, this opens interesting research directions in robust parsing. If we can produce a representation for an utterance which has domain specializations for all but one word, it is a good candidate for a clarifying question. Existing approaches to robust parsing use either fragment combination [Rosé, 2000; van Noord *et al.*, 1999] or word skipping and syntactic repair rules [Core and Schubert, 1999; Lavie, 1996] to recover the meaning of the input in presence of errors and extragrammatical utterances. While our approach cannot currently deal with repairs and clearly ungrammatical utterances, it offers a way to handle extragrammaticality caused by misrecognized words. We also hope that it can be combined with syntactic repair methods, which typically suffer from performance problems due to increased ambiguity. The ability to bring domain constraints to into a domain-independent parser could offer a useful way to control the ambiguity and improve the efficiency of the existing combination methods, and we hope to research this approach in the future.

## 6.1 Future work

There are other questions which need to be explored in the future work. With respect to feature set, we provided an evaluation on real data sufficient to show that selectional restrictions play an important role in achieving reasonable parsing speed and accuracy, and we provided statistics

about feature usage which indicate the importance of our features by showing how frequently they are used in different selectional restrictions. We also provided a set of guidelines for adding new features and values to our set. One such guideline was that the feature should have significant disambiguating power. The number of words using the feature in their restrictions provides a measure of that, but it would also be interesting to devise a more precise measure of it. One such possibility is measuring the changes in disambiguation accuracy over a large corpus if the feature is removed from the set. We did not have a corpus of sufficient size available to make such measurements, but we are currently working on construction of parsed corpora which can help answer such questions in the future.

Another possible extension for the use of our feature set is its usability in other reasoning tasks. Though it is designed primarily for parsing, many features in our set encode basic properties of words which could be used in reasoning, and it was one of the original goals for this research. Especially interesting is the connection with reference. The PHORA system [Byron, 2002a] uses our domain-specific selectional restrictions to improve the quality of reference resolution. In this thesis we demonstrated that having basic selectional restrictions with our set benefits parsing even without domain specialization. Would it also benefit pronoun resolution? We believe that this is may be a useful research direction, especially with the view of doing pronoun resolution in larger scale systems, where domain information may not always be available.

A question we left open in our lexicon and ontology design is lexicon acquisition. We motivated many of our design decision by the need of fast and convenient lexicon development. The fact that we successfully maintain the parser in four different domains in parallel provides proof that we achieved our goals at least to some extent. However, it remains to be seen how scalable this approach would be if we want to increase the size of our lexicon several times. This cannot be done completely manually, and we have several projects under development on automatically acquiring lexicon entries from other sources, particularly WordNet and VerbNet lexicons. Changes and additions to our architecture to better support automated acquisition are an exciting possibility for future research.

A similar issue arises with our specialization approach. We have demonstrated that it was successful in our domains in increasing parsing speed and improving disambiguation accuracy. However, it relies on several specific assumptions, which include the fact that the size of the domain is quite small and that once domain concepts are known, it should be relatively easy to figure out what relevant domain-independent concepts map into domain-specific concepts. This task may become more difficult should our domain-independent ontology size grow significantly.

There are ways in which it can be made easier - for example, tools which go over corpora collected for the domain, find out words for which specializations are not known, and propose the LF categories which may need to be included in specialization.

More importantly, the question which needs to be further investigated in our domain specialization method is the role of boosting in our search. Currently, we boost up the probability of our domain-specialized entries to improve parsing speed and disambiguation. The boost-up factor was selected manually after some experimentation to be the value which provided the best performance across domain. We have not fully investigated either why this value provided the best results, or what would be the general criteria for selecting an appropriate boosting factor. This is the area where using statistical learning methods can be very profitable. The LINGO grammar used statistical learning techniques to learn the best rule probabilities from corpus [Copestake and Flickinger, 2000] and to improve parsing accuracy in the VerbMobil domain. Similarly, we hope that our grammar could benefit from learning both domain-specific weights and domain-specific boosting factors to achieve the best possible performance.

Finally, there are open questions related to our statistical learning algorithm. We have started this research in the hope that our features will be easier to learn than the full word senses from large corpora, and that feature structures will be a good framework for learning selectional restrictions. Neither hope has been fully realized. We have an artificial data set on which our feature-based model performs better than the established methods, but it does not fully transfer to our data sets derived from a Wall Street Journal corpus, where the results have been mixed. We have suggested possible reasons for that, and ways of improving our results, but working further to determine the exact causes and see if statistical preferences for our feature set can be learned from corpora is a research question which will have to be explored in the future.

## Bibliography

- [Abney and Light, 1999] Steven Abney and Marc Light, “Hiding a semantic hierarchy in a Markov model,” In *Proceedings of the ACL’99 Workshop on Unsupervised Learning in Natural Language Processing*, pages 1–8, 1999.
- [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.
- [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.
- [Alonge *et al.*, 2000] A. Alonge, F. Bertagna, N. Calzolari, A. Roventini, and A. Zampolli, “Encoding Information on Adjectives in a Lexical-Semantic Net for Computational Applications,” In *Proceedings of NAACL’2000*, Seattle, 2000.
- [Alshawi, 1992] Hiyan Alshawi, editor, *The Core Language Engine*, ACL-MIT Press Series in Natural Language Processing. MIT Press, Cambridge, England, 1992.
- [Alshawi *et al.*, 1991] 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.
- [Androutsopoulos and Dale, 2000] Ion Androutsopoulos and Robert Dale, “Selectional Restrictions in HPSG,” In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, pages 15–20, Saarbrücken, Germany, August 23 2000.

- [Berger *et al.*, 1996] Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra, “A Maximum-Entropy Approach to Language Modeling,” *Computational Linguistics*, 22(1):39–71, 1996.
- [Byron, 2002a] Donna K. Byron, *Resolving Pronominal Reference to Abstract Entities*, PhD thesis, University of Rochester, 2002.
- [Byron, 2002b] Donna K. Byron, *Resolving Pronominal Reference to Abstract Entities*, PhD thesis, University of Rochester, 2002.
- [Carpenter, 1992] Bob Carpenter, *The Logic of Typed Feature Structures*, Cambridge University Press, 1992.
- [Carpenter, 1993] Bob Carpenter, “Skeptical and Credulous Default Unification with Applications to Templates and Inheritance,” In Ted Briscoe, Valeria de Paiva, and Ann Copestake, editors, *Inheritance, defaults, and the lexicon*, pages 13–37. Cambridge University Press, Cambridge, England, 1993.
- [Chang *et al.*, 2002] N. Chang, J. Feldman, R. Porzel, and K. Sanders, “Scaling Cognitive Linguistics: Formalisms for Language Understanding,” In *Proceedings of the First International Workshop on Scalable Natural Language Understanding*, 2002, <http://smartkom.dfki.de/Vortraege/icsi-ai2.pdf>.
- [Charniak, 1997] Eugene Charniak, “Statistical Parsing with a Context-Free Grammar and Word Statistics,” In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI-97/IAAI-97)*, pages 598–603, Menlo Park, July 27–31 1997. AAAI Press.
- [Chen and Rosenfeld, 1999] S. Chen and R. Rosenfeld, “A Gaussian prior for smoothing maximum entropy models,” Technical report, Carnegie Mellon University, 1999.
- [Chen and Rosenfeld, 2000] Stanley F. Chen and Ronald Rosenfeld, “A Survey of Smoothing Techniques for ME Models,” *IEEE Trans. on Speech and Audio Processing*, 8(1):37–50, January 2000.
- [Clark and Porter, 1999] P. Clark and B. Porter, *KM (1.4): Users Manual*, <http://www.cs.utexas.edu/users/mfkb/km>, 1999.
- [Clark] Peter Clark, “From Natural Language to KM Representations,” Working Note 24.

- [Collins, 1997] Michael Collins, “Three Generative, Lexicalized Models for Statistical Parsing,” In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23, Somerset, New Jersey, 1997. Association for Computational Linguistics, Association for Computational Linguistics.
- [Collins, 1999] Michael Collins, *Head-Driven Statistical Models for Natural Language Parsing*, PhD thesis, University of Pennsylvania, 1999.
- [Copestake, 1992] Ann Copestake, *The representation of lexical semantic information*, PhD thesis, University of Sussex, 1992.
- [Copestake and Flickinger, 2000] 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.
- [Copestake *et al.*, 1995] Ann Copestake, Dan Flickinger, Rob Malouf, Susanne Riehemann, and Ivan Sag, “Translation using Minimal Recursion Semantics,” In *Proceedings of the 6th. International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-95)*, Leuven, Belgium, July 1995.
- [Copestake *et al.*, 1993] Ann Copestake, Antonio Sanfilippo, Ted Briscoe, and Valeria de Paiva, “The ACQUILEX LKB: an introduction,” In Ted Briscoe, Valeria de Paiva, and Ann Copestake, editors, *Inheritance, defaults, and the lexicon*, pages 148–163. Cambridge University Press, Cambridge, 1993.
- [Core and Schubert, 1999] Mark G. Core and Lenhart K. Schubert, “A Model of Speech Repairs and Other Disruptions,” In Susan E. Brennan, Alain Giboin, and David Traum, editors, *Working Papers of the AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems*, pages 48–53, Menlo Park, California, 1999. American Association for Artificial Intelligence, American Association for Artificial Intelligence.
- [Dalrymple *et al.*, 1995] Mary Dalrymple, John Lamping, Fernando Pereira, and Vijay Saraswat, “Linear Logic for Meaning Assembly,” In *Proceedings of CLNLP*, Edinburgh, April 17 1995, Comment: 19 pages, uses lingmacros.sty, fullname.sty, tree-dvips.sty, latexsym.sty, requires the new version of Latex.

- [Darroch and Ratcliff, 1972] J. N. Darroch and D. Ratcliff, “Generalized Iterative Scaling for Log-Linear Models,” *Annals of Mathematical Statistics*, 43:1470–1480, 1972.
- [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.
- [Dempster *et al.*, 1977] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum Likelihood from Incomplete Data via the EM Algorithm,” *J. Royal Statist. Soc. Ser. B*, 39(1):1–38, 1977, With discussion.
- [Dorr and Olsen, 1997] Bonnie Dorr and Broman Olsen, “Deriving Verbal and Compositional Lexical Aspect for NLP Applications,” In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 151–158, Somerset, New Jersey, 1997. Association for Computational Linguistics, Association for Computational Linguistics.
- [Dowding *et al.*, 1994] 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.
- [Dowty, 1991] David Dowty, “Thematic Proto-Roles and Argument Selection,” *Language*, 67(3):547–619, 1991.
- [Dzikovska *et al.*, 2002] Myroslava Dzikovska, James F. Allen, and Mary D. Swift, “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*, May 2002.
- [Dzikovska and Byron, 2000] Myroslava O. Dzikovska and Donna K. Byron, “When is a union really an intersection? Problems interpreting reference to locations in a dialogue system,” In *Proceedings of the Fourth Workshop on the Semantics and Pragmatics of Dialogue (GOTALOG’2000)*, Goteborg, June 2000.
- [Dzikovska *et al.*, 2003] Myroslava O. Dzikovska, Mary D. Swift, and James F. Allen, “Integrating linguistic and domain knowledge for spoken dialogue systems in multiple domains,”

- In *Proceedings of IJCAI-03 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Acapulco, August 2003.
- [Dzikovska *et al.*, pear] 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.
- [Eisner, 2001] Jason Eisner, *Smoothing a Probabilistic Lexicon via Syntactic Transformations*, PhD thesis, University of Pennsylvania, July 2001.
- [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.
- [Gazdar *et al.*, 1985] Gerald Gazdar, E. Klein, Geoff Pullum, and Ivan Sag, *Generalized Phrase Structure Grammar*, Blackwell, Oxford, 1985.
- [Gildea and Jurafsky, 2002] Daniel Gildea and Daniel Jurafsky, “Automatic Labeling of Semantic Roles,” *Computational Linguistics*, 28(3):245–288, 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 On-line 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>.
- [Grishman and Sterling, 1992] Ralph Grishman and John Sterling, “Acquisition of selectional patterns,” In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING '92)*, pages 658–664, Nantes, France, July 1992.
- [Gurevych *et al.*, 2003] Iryna Gurevych, R. Malaka, , Robert Porzel, and H. Zorn, “Semantic Coherence Scoring Using an Ontology,” In *Proceedings of the Joint Human Language Technology and Northern Chapter of the Association for Computational Linguistics Conference (HLT-NAACL)*, Edmonton, Canada, May 2003.
- [Heeman and Allen, 1995] Peter A. Heeman and James Allen, “The TRAINS 93 dialogues,” TRAINS Technical Note 94-2, Department of Computer Science, University of Rochester, 1995 1995.

- [Herskovits, 1986] A. Herskovits, *Language and spatial cognition. An Interdisciplinary Study of the Prepositions in English*, Cambridge University Press, Cambridge, 1986.
- [Hobbs, 1985] Jerry R. Hobbs, “Ontological promiscuity,” In *Proc. of the 23<sup>rd</sup> Annual Meeting of the Association for Computational Linguistics. Chicago, IL, 8–12 Jul 1985*, pages 61–69, 1985.
- [Hofmann, 2001] Thomas Hofmann, “Unsupervised Learning by Probabilistic Latent Semantic Analysis,” *Machine Learning*, 42(1/2):177–196, 2001.
- [Hofmann and Puzicha, 1998] Thomas Hofmann and Jan Puzicha, “Statistical Models for Co-occurrence Data,” Technical Report AIM-1625, Massachusetts Institute of Technology, December 1998.
- [Hwang and Schubert, 1993] Chung Hee Hwang and Lenhart Schubert, “Episodic Logic: a Situational Logic for Natural Language Processing,” In Stanley Peters, David Israel, Peter Aczel, and Yasuhiro Katagiri, editors, *Situation Theory and its Applications*. Center for the Study of Language and Information, Stanford, California, 1993.
- [Jackendoff, 1990] R. Jackendoff, *Semantic Structures*, The MIT Press, 1990.
- [Jaynes, 1982] E. T. Jaynes, “On the rationale of maximum-entropy methods,” *Proc. IEEE*, 70(9):939–952, 1982.
- [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.
- [Katz and Fodor, 1964] Jerrold J. Katz and J. A. Fodor, “The Structure of a Semantic Theory,” In J. A. Fodor and Jerrold J. Katz, editors, *The Structure of Language: Readings in the Philosophy of Language*, pages 479–518. Prentice-Hall, Englewood Cliffs, New Jersey, 1964.
- [Kay *et al.*, 1994] Martin Kay, Jean Mark Gawron, and Peter Norvig, *Verbmobil: A Translation System for Face-To-Face Dialog*, CSLI Press, Stanford, California, 1994.
- [Kenstowicz, 1993] Michael Kenstowicz, *Phonology in Generative Grammar*, Blackwell Textbooks in Linguistics. Blackwell, Oxford, 1993.
- [Kiefer *et al.*, 1999] B. Kiefer, H. Krieger, J. Carroll, and R. Malouf, “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.

- [Kipper *et al.*, 2000] Karin Kipper, Hoa Trang Dang, and Martha Palmer, “Class-Based Construction of a Verb Lexicon,” In *Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00)*, pages 691–696, Menlo Park, CA, July 30– 3 2000. AAAI Press.
- [Kirkpartick *et al.*, 1983] S. Kirkpartick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” *Science*, 13 May 1983, 220(4598):671–680, 1983.
- [Lascarides and Copestake, 1998] Alex Lascarides and Ann Copestake, “Pragmatics and Word Meaning,” *Journal of Linguistics*, 34(2):387–414, 1998.
- [Lavie, 1996] A. Lavie, *GLR\*: a robust grammar-focused parser for spontaneously spoken language*, PhD thesis, CMU, 1996.
- [Lee, 1997] Lilian Lee, *Similarity-based approaches to natural language processing*, PhD thesis, Harvard University, 1997.
- [Lenat, 1995] Douglas B. Lenat, “CYC: A Large-Scale Investment in Knowledge Infrastructure,” *Communications of the ACM*, 38(11):33–38, November 1995.
- [Levin, 1993] Beth C. Levin, *English Verb Classes and Alternations: a Preliminary Investigation*, University of Chicago Press, Chicago, IL, 1993.
- [Li and Abe, 1996] Hang Li and Naoki Abe, “Clustering Words with the MDL principle,” In *Proc. of COLING*, pages 4–9, 1996.
- [Lin, 1995] Dekang Lin, “A Dependency-based Method for Evaluating Broad-Coverage Parsers,” In *Proceedings of IJCAI-95*, pages 1420–1427, 1995.
- [Lin, 1998] Dekang Lin, “Dependency-based Evaluation of MINIPAR,” In *Workshop on the Evaluation of Parsing Systems*, Granada, Spain, May 1998.
- [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.
- [Mahesh *et al.*, 1996] Kavi Mahesh, Sergei Nirenburg, Jim Cowie, and David Farwell, “An Assessment of Cyc for Natural Language Processing,” Technical Report MCCA-96-302, CRL, NMSU, New Mexico, 1996.

- [McCawley, 1968] James McCawley, “The role of semantics in a grammar,” In Emmon Bach and Robert Harms, editors, *Universals in Linguistic Theory*, pages 124–169. Holt, Rinehart and Winston, 1968.
- [McDonald, 1996] David D. McDonald, “The interplay of syntactic and semantic node labels in partial parsing,” In H. Bunt and M. Tomita, editors, *Recent Advances in Parsing Technology*, pages 295–323. Kluwer Academic Publishers, 1996.
- [Miller, 1995] G. Miller, “WordNet: A Lexical Database For English,” *Communications of the ACM*, 38(5), 1995.
- [Moens and Steedman, 1988] M. Moens and M. Steedman, “Temporal ontology and Temporal reference,” *Computational Linguistics*, 14(2), 1988.
- [Mosny, 1996] Milan Mosny, “Semantic Information Preprocessing For Natural Language Interfaces to Databases,” Master’s thesis, Simon Fraser University, November 1996.
- [Nirenburg *et al.*, 1992a] Sergei Nirenburg, Jaime Carbonnell, Masaru Tomita, and Kenneth Goodman, *Machine Translation: A Knowledge-based Approach*, Morgan Kaufmann Publishers, Los Altos, CA, 1992.
- [Nirenburg *et al.*, 1992b] Sergei Nirenburg, Jaime Carbonnell, Masaru Tomita, and Kenneth Goodman, *Machine Translation: A Knowledge-based Approach*, pages 34–35, Morgan Kaufmann Publishers, Los Altos, CA, 1992.
- [Oepen *et al.*, 2002] Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christopher D. Manning, “LinGO Redwoods - A Rich and Dynamic Treebank for HPSG,” In *Proceedings of The First Workshop on Treebanks and Linguistic Theories (TLT2002)*, Sozopol, Bulgaria, 2002.
- [Pietra *et al.*, 1997a] Stephen Della Pietra, Vincent J. Della Pietra, and John D. Lafferty, “Inducing Features of Random Fields,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.
- [Pietra *et al.*, 1997b] Stephen Della Pietra, Vincent J. Della Pietra, and John D. Lafferty, “Inducing Features of Random Fields,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.
- [Pollard and Sag, 1994] Carl Pollard and Ivan Sag, *Head-Driven Phrase Structure Grammar*, University of Chicago Press, Chicago, 1994, Draft distributed at the Third European Summer School in Language, Logic and Information, Saarbrücken, 1991.

- [Pustejovsky, 1995] James Pustejovsky, *The Generative Lexicon*, The MIT Press, Cambridge, Massachusetts, 1995.
- [Ratnaparkhi, 1998] Adwait Ratnaparkhi, *Maximum Entropy Models for Natural Language Ambiguity Resolution*, PhD thesis, University of Pennsylvania, 1998.
- [Rayner, 1993] Manny Rayner, *Abductive Equivalential Translation and its application to Natural Language Database Interfacing*, PhD thesis, Royal Institute of Technology, Stockholm, September 1993, Comment: 162 pages, Latex source, PhD thesis (U Stockholm, 1993). Uses style-file `ustockholm_thesis.sty`.
- [Rayner and Carter, 1996] Manny Rayner and David Carter, “Fast Parsing Using Pruning and Grammar Specialization,” In Arivind Joshi and Martha Palmer, editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 223–230, San Francisco, 1996. Association for Computational Linguistics, Morgan Kaufmann Publishers.
- [Resnik, 1993] P. Resnik, “Semantic classes and syntactic ambiguity,” In *Proc. of ARPA Workshop on Human Language Technology*, Plainsboro, NJ, 1993.
- [Ribas, 1994] Francesc Ribas, “An Experiment on Learning Appropriate Selectional Restrictions from a Parsed Corpus,” In *Proceedings of COLING-94*, pages 769–774, September 07 1994, Comment: 11 pages.
- [Ribas, 1995] Francesc Ribas, “On Learning More Appropriate Selectional Restrictions,” In *Proceedings EACL-95*, Ireland, 1995.
- [Riezler, 2000] Stefan Riezler, *Probabilistic Constraint Logic Programming. Formal Foundations of Quantitative and Statistical Inference in Constraint-Based Natural Language Processing*, PhD thesis, University of Tuebingen, 2000, Comment: PhD Thesis, 144 pages, University of Tuebingen, 1998.
- [Rooth *et al.*, 1999] Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil, “Inducing a Semantically Annotated Lexicon via EM-Based Clustering,” In *Proceedings of the 37th Annual Meeting of the ACL*, May 19 1999, Comment: 8 pages, uses `colacl.sty`.
- [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.

- [Rose, 1998] Kenneth Rose, “Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems,” *Proceedings of the IEEE*, 80:2210–2239, November 1998.
- [Rusiecki, 1985] Jan Rusiecki, *Adjectives and Comparison in English*, volume 31 of *Longman Linguistics Library*, Longman, 1985.
- [Sanfilippo *et al.*, 1998] A. Sanfilippo, N. Calzolari, S. Ananiadou, R. Gaizauskas, P. Saint-Dizier, and P. Vossen (eds), “EAGLES, Preliminary Recommendations on Semantic Encoding. Interim Report,” June 1998.
- [Schein *et al.*, 2002] A. Schein, A. Popescul, and L. Ungar, “PennAspect: A two-way aspect model implementation,” Technical report, Department of Computer and Information Science, The University of Pennsylvania., 2002.
- [Schubert *et al.*, 1987] L. K. Schubert, M. A. Papalaskaris, and J. Taugher, “Accelerating Deductive Inference: Special Methods for Taxonomies, Colours and Times,” In N. Cercone and G. McGalla, editors, *The Knowledge Frontier: Essays in the Representation of Knowledge*, pages 188–220. Springer-Verlag, New York (NY), USA, 1987.
- [Schubert, 1979] Lenhart K. Schubert, “Problems with parts,” In *Proceedings of the 6th IJCAI-79*, pages 778–784, Tokyo, August 1979.
- [Schuler, 2002] Karin Kipper Schuler, “VerbNet: a broad-coverage, comprehensive verb lexicon,” a dissertation proposal, Department of Computer and Information Science, University of Pennsylvania, 2002.
- [Seagull and Schubert, 1999] Amon Seagull and Lenhart Schubert, “Guiding a Well-Founded Parser with Corpus Statistics,” In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*. University of Maryland, June 1999.
- [Seneff, 1992] Stephanie Seneff, “TINA: A Natural Language System for Spoken Language Applications,” *Computational Linguistics*, 18(1):61–86, March 1992.
- [Sleator and Temperley, 1993] Daniel Sleator and Davy Temperley, “Parsing English with a Link Grammar,” In *Third International Workshop on Parsing Technologies*, 1993.
- [Sowa, 1999] John F. Sowa, *Knowledge Representation : Logical, Philosophical, and Computational Foundations*, Brooks/Cole Pub Co., 1999.

- [Stede *et al.*, 1998] M. Stede, S. Haas, and U. Ussner, “Tracking and understanding temporal descriptions in dialogue,” Technical Report 232, Technische Universitat Berlin, 1998.
- [Stent, 2000] Amanda J. Stent, “The Monroe Corpus,” Technical Report 728/TN 99-2, The University of Rochester, Computer Science Department, 2000.
- [Stetina and Nagao, 1997] Jiri Stetina and Makoto Nagao, “Corpus-based PP attachment ambiguity resolution with a semantic dictionary,” In *Proceedings of the 5th Workshop on Very Large Corpora*, pages 66–80, Beijing, August 1997.
- [Stoness, 2001] Scott C. Stoness, “Continuous understanding: A First Look at CAFE,” Area Paper for URCS, May 2001.
- [Swier, 2002] Robert Swier, “Automatically Acquiring Adjective Knowledge for Dialogue Systems,” Master’s thesis, University of Rochester, Rochester, NY, August 2002.
- [Swift *et al.*, pear] Mary D. Swift, Joel Tetreault, and Myroslava O. Dzikovska, “Semi-automatic syntactic and semantic corpus annotation with a deep parser,” In *Proceedings of LREC-2004*, to appear.
- [Tetreault, 2001] Joel Tetreault, “A Corpus-Based Evaluation of Centering and Pronoun Resolution,” *Computational Linguistics*, 27(4):507–520, 2001.
- [Thomas and Pulman, 1999] J. Thomas and S. G. Pulman, “Bidirectional Interpretation of Tense and Aspect,” In H. Bunt *et al.* (eds), editor, *Proceedings of the Third International Workshop on Computational Semantics*, pages 247–263, Tilburg, 1999.
- [Utsuro and Matsumoto, 1997] Takehito Utsuro and Yuji Matsumoto, “Learning Probabilistic Subcategorization Preference by Identifying Case Dependencies and Optimal Noun Class Generalization Level,” In *Proceedings of 5th ANLP Conference*, 1997.
- [van Noord *et al.*, 1999] Gertjan van Noord, Gosse Bouma, Rob Koeling, and Mark-Jan Nederhof, “Robust Grammatical Analysis for Spoken Dialogue Systems,” *Natural Language Engineering*, 5(1):45–93, 1999.
- [Vossen, 1997] Piek Vossen, “EuroWordNet: a multilingual database for information retrieval,” In *Proceedings of the Delos workshop on Cross-language Information Retrieval*, March 1997.
- [Vossen *et al.*, 1997] Piek Vossen, Laura Bloksma, Horacio Rodriguez, Salvador Climent, Nicoletta Calzolari, Adriana Roventini, Francesca Bertagna, Antonietta Alonge, and Wim Peters,

“The EuroWordNet Base Concepts and Top Ontology,” Ec-deliverable, CNR - Istituto di Linguistica Computazionale (Pisa), 1997.

[Wang *et al.*, 2002] S. Wang, R. Rosenfeld, Y. Zhao, and D. Schuurmans, “The Latent Maximum Entropy Principle,” In *IEEE International Symposium on Information Theory, ISIT-2002*, 2002.

[Wilks, 1975] Yorick Wilks, “Preference Semantics,” In E. L. Keenan, editor, *Formal Semantics of Natural Language*, pages 329–348. Cambridge University Press, Cambridge, MA, 1975.