An Ontological-Semantic Framework for Text Analysis

Boyan A. Onyshkevych

2 May 1997

CMU-LTI-97-148

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213

Submitted to Carnegie Mellon University in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science.

ACKNOWLEDGEMENTS

This work would not have been possible without many colleagues, faculty, and family members who have provided support, inspiration, and encouragement.

First among them is my advisor, Sergei Nirenburg, who has been a source of not only guidance and insight but also friendship during the long process of converting preliminary ideas into the final document. Victor Raskin buoyed me with encouragement and confidence over the past few years. Jaime Carbonell and Lori Levin not only served on the dissertation committee, but also provided invaluable mentoring through three incarnations of computational linguistics programs.

My colleagues at work, especially Lynn Carlson, have been a source of inspiration and support through many years of heavy work loads and difficult times.

Many thanks to other participants in the DIANA and MIKROKOSMOS projects, who have created the environment in which my work was conducted, and who are responsible for much of infrastructure that I used; in addition to the people mentioned above, thanks also go to Steve Beale, Ralf Brown, JK Davis, Kavi Mahesh, Ingrid Meyer, Evelyne Viegas, and others.

Many thanks to my parents, brother, and sister for years of encouragement and support.

Most importantly, my wife Oksana has been responsible for keeping me sane and our children happy while I labored over this dissertation, without which the work would never have come to completion.

I extend my deepest gratitude to them all.

ABSTRACT

The Knowledge-Based Machine Translation paradigm requires a comprehensive analysis of input texts into an unambiguous machine-tractable representation of the propositional and meta-propositional meaning of that text, for which we use a particular framework referred to as ontological semantics. The work presented here begins with a definition of a representation language for lexical semantic specification (and syntax/semantics interface) to support such an analysis, as well as a generalized algorithm for building the meaning representation from these lexical semantic specifications, utilizing the ontology and a syntactic parse as knowledge sources. The core of the algorithm is an algorithm for semantic constraint satisfaction and relaxation, involving finding the best path over the ontology between a candidate filler of a relation and semantic constraints on that relation. The ontology is viewed as a multi-dimensional graph, with distinct topologies in each dimension reflecting specific semantic relations between nodes (representing concepts), where weights or arc distance reflects strength of semantic relatedness in context (where the path-so-far context is maintained in a state transition table). Simulated annealing is used for acquiring these weights from training corpora. The selectional restriction satisfaction algorithm is imbedded within a framework which traverses the search space of all possible semantic interpretations, using both a data-driven operator and an expectation-driven operator.

This algorithm and framework for meaning interpretation are applied in the generic semantic dependency structure-building case (involving satisfaction and relaxation of semantic constraints), word sense disambiguation (WSD), as well as metonymy processing. WSD relies on this very rich set of constraints (generalized from traditional selectional restrictions) where any concept in the ontology can serve as a constraint; using this notion of constraints and the ontological graph search for checking constraint satisfaction (which combines traditional syntagmatic and paradigmatic approaches) provides encouraging results for WSD. The approach to metonymy processing uses both an extensive language-specific inventory of frequent metonymic relations and a mechanism for allowing any other semantic relation or chain of relations which exist in the ontology to provide the metonymic relation that is recovered from text.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS ii
ABSTRACT iii
TABLE OF CONTENTS iv
LIST OF FIGURES AND TABLES vii
1. Introduction 1
1.1 Project Background 1
1.2 Setting the Stage 2
1.3 Practical Computational Semantics 4
1.3.1 First Desideratum 5
1.3.2 Second Desideratum 6
1.3.3 Third Desideratum 7
1.3.4 Fourth Desideratum 8
1.3.5 Other Considerations of a Theory 9
1.4 KBMT and Ontological Semantics 9
1.5 Scope of this Work. 11
1.6 Outline 12
2. Overview of the Semantic Analysis Process 13
2.1 Introduction 13
2.2 The Semantic Dependency Structure Building Process. 14
2.3 Knowledge Sources 16
2.3.1 The Ontology 16
2.3.2 Lexical Semantic Specification 18
2.4 Constraint Satisfaction in Combining Semantic Elements 19
2.5 An Example 23
3. Background: Knowledge Structures 27
3.1 Ontological Knowledge 27
3.1.1 The Frame-based Formalisms 28
3.1.2 Slot Specification at the Facet level 29
3.1.3 The Ontology 30
3.1.4 The Onomasticon 35
3.2 The Lexicon 36
3.2.1 Cross-Indexing and Inheritance 39
3.2.2 Lexical Rules 40
3.3 Text Meaning Representation Language 42
3.3.1 TMR Definition 43
3.3.2 Propositional Content 46
3.3.3 Attitudes and Modalities 47
3.3.4 Stylistics 48
3.3.5 Relations 49
3.4 Syntax and Syntactic Parsing 50
3.4.1 The F-Structure 51
3.4.2 Lexical Syntactic Specification 52
3.4.3 Syntactic Processing 54
4. Representation of Lexical Semantic Knowledge 57
4.1 Issues in Lexical Semantic Representation 57

4.1.1 Elements of the Language of Lexical Semantic Representation 57
4.1.2 Ontology vs. Lexicon Trade-Off: Granularity 58
4.1.3 Ontology vs. Lexicon Trade-Off: Enumerability of Senses 60
4.1.4 Metalanguage for Lexical Semantic Representation 62
4.2 Sense Mapping 62
4.2.1 Univocal Mapping 63
4.2.2 Constrained Mapping 63
4.2.3 Non-Propositional Mapping 66
4.3 Mapping / Category Correlation 67
5. The Underlying Heuristic: Ontological Graph Search 68
5.1 The Graph Search 68
5.1.1 The Ontology as a Graph 69
5.1.2 Search Algorithms 72
5.1.3 Computational Complexity 74
5.1.4 Ontology Search as Abduction 76
5.2 Determination of Arc Weights 76
5.2.1 Static Arc Weight Assignment 77
5.2.2 Dynamic Arc Weight Assignment 78
5.2.3 Transition Table Arc Weight Assignment 79
5.3 Acquisition of the Arc Weights 80
5.3.1 Manual Arc Weight Acquisition 81
5.3.2 Automated Learning of Arc Weights by Simulated Annealing 81
5.3.3 Success of Training 83
5.4 Issues involving the Ontology Search Algorithm 84
6. Semantic Dependency Structure Building as a Search Problem 86
6.1 Search States 86
6.2 The Instantiation Operator 87
6.3 Combination Operator 91
6.3.1 Triggered by Syntax/Semantics Interface 91
6.3.2 Licensed by Underlying Syntactic Cues 93
6.4 Search Parameters 94
6.5 Search Flow and Control Structure 95
6.5.1 Flow of Control 95
6.5.2 Proliferation of Search Paths 98
7. Basic Semantic Analysis as a Search Problem 101
7.1 The Semantic Analysis Process 101
7.2 Semantic Analysis States 103
7.3 The Instantiation Process 103
7.4 The Combination Process 106
7.5 Semantic Analysis Flow and Control 111
7.5.1 Parameters of Semantic Analysis 111
7.5.2 Ambiguity Representation 111
7.5.3 The Introduction of Additional Nodes 113
7.5.4 Combination of Evidence from Multiple Sources 113
8. Application of the SDS-Building Mechanism: Word Sense Disambiguation 115
8.1 Background and Related Work 115

8.1.1 Related Paradigmatic Approaches 115
8.1.2 Related Syntagmatic Approaches 118
8.1.3 Hybrid Approaches 122
8.2 The SDS-Based Model for WSD. 123
8.3 Results 126
9. Application of the SDS-Building Mechanism: Metonymy Processing 131
9.1 Why Resolve Metonymy? 131
9.2 Related Computational Approaches to Metonymy 132
9.3 Framework for Metonymy Processing. 134
9.4 Determination of Arc Weights. 136
9.5 Metonymy Processing: An Example 137
9.6 Inventory of Metonymic Relations 139
9.6.1 Semantic Metonymic Relations 140
9.6.2 Conventional Metonymic Relations 140
9.6.3 Pragmatic Metonymic Relations 141
9.6.4 Utility of this classification 142
9.7 Metonymic Relations in the Ontology. 142
9.8 Knowledge Base for Metonymy Processing 144
9.8.1 Training the Static Weight Assignment Mechanism 145
9.8.2 Training the Transition Table 150
9.9 Results 151
10. Ontology Acquisition 161
10.1 Acquisition of Ontological Knowledge 161
10.1.1 Importing Existing Ontological Resources 161
10.1.2 Facilitating Building an Ontology 163
10.2 The Effects of Gaps in the Ontology 164
10.3 Issues in Defining Relations 164
11. Conclusion 167
11.1 Future Work. 167
11.1.1 Noun-Noun Compound Resolution 167
11.1.2 Additional Microtheories 168
11.2 Summary and Heuristics 169
11.3 Significance 171
12. Bibliography 173

LIST OF FIGURES AND TABLES

Classification of approaches to interlinguas or meaning representation 3
Overall data flow of the architecture, showing knowledge sources and flow of processing: parsed input undergoes instantiation and combination, and the resulting partial TMRs are augmented by microtheories. 15
An ontology fragment. 17
Illustration of slot constraints and filler types. 19
Two views of example of Ontology, with paths identified with bold arrows. 22
Graphical representation of lexical semantics for *chihuahua*, *eat*, and *apple*. 23
Graphical representation of instantiated lexemes (3 discrete structures) 25
Graphical representation of initial TMR (one network of structures 26
Head and Facet Use in Ontology, TMR, and Lexical Semantics 31
Filler Types 32
Matrix f-structure representation 53
Toy illustration of arcs.. 71
Two views of example of Ontology, with paths identified with bold arrows. 72
Same ontology, but represented with different arc lengths for shortest-path search 73
Path over simplified ontology from *PENNY to *INGESTIBLE 78
Fragment of Example Arc-Cost Determination Transition Table 80
Example Ontology 88
LEX-MAP reference to ontological node 88
Origination-node-less **LEX-MAP** specification. 89
Non-ontological node reference 89
Example of instance structures produced by instantiation.. 89
Example of instantiation operator with input and output TMR forms. 90
Interrelationship of TMR structure and the ontology where concepts are defined (shaded arcs are *INSTANCE-OF* links). 90
Example of combination operator with input and output TMR forms. 92
Example of combination operator, in the case of an origination-node-less structure. 92
Example TMR with unlabeled arc.. 93
Example of combination operator, in case of unlabeled arc. 94
Data flow and process interaction in the SDS-building search 96
Example search space for SDS-building, showing multiple possible paths to the goal state. 97
Example of combination operator, in case where alternate TMRs result from identifying arc label.. 98
Example of search space, showing multiple paths, both successful (reaching a state marked by double circles) and unsuccessful (which reach a FAIL state). 100
Data Flow in Semantic Analysis. 102
Search states for "*the man in the shop*", assuming no ambiguity or abandoned search paths . . 112
Word Sense Disambiguation Results on Spanish Training Data 127
Word Sense Disambiguation Results on Unseen Spanish Test Data 128
Metonymy Training Results on Data Set 1 (Static Weight Assignment) 145
Metonymy Training Results On Data Set 2 (Static Weight Assignment) 148
Metonymy Training Results On Data Sets 1 and 2 (Transition Table Weight Assignment) . . 151
Metonymy Test Results On English Test Data 151
Metonymic Inferences in 5 Spanish Texts 153

Manually Acquired Arc-Cost Determination Transition Table (for basic dependency and metonymy) 155
Transition Table Equivalence Sets for Training 158

1. Introduction

The history of Machine Translation reflects an exploration of the trade-offs between the necessity of “deep” understanding of the meaning of the input text, as reflected in the interlingual MT paradigm, and the pragmatic considerations of knowledge acquisition and processing speed, as espoused by the various flavors of the transfer and statistical MT paradigms (see, for example, Hutchins (1986)). Although we are far from the point where both considerations can be fully satisfied, it appears clear that it will not be possible to achieve practical high-quality machine translation without addressing the issues of primary concern to both approaches. The work described here addresses the formalisms, knowledge sources and processing algorithms necessary for effective representation, resolution, and processing of the meaning of text in the interlingual Knowledge-Based Machine Translation paradigm, using a particular framework for computational semantics dubbed ontological semantics, while attempting to favorably position this paradigm relative to the pragmatic concerns of the latter paradigm.

1.1 Project Background

The research described here stems from the DIANA project at CMU/CMT in 1987-1988; during the course of that project, an initial specification of the knowledge sources and formalisms was developed, and an initial toy implementation was built. This work resumed in earnest in 1993 as the core of the MIKROKOSMOS effort. That project, jointly worked by NMSU/CRL, CMU/CMT, and a US DoD research lab, funded by the US DoD, is an effort to push these lexical semantic theories, both for core semantic analysis and for development of specific *microtheories*, i.e., processing specialists which focus on a particular component of language analysis (such as aspect, time, or coreference). Also within the scope of the MIKROKOSMOS effort is system building, which includes large-scale knowledge acquisition (the ontology and lexicons for Japanese and Spanish), as well as development of a full-scale implementation of the semantic analysis algorithms and disambiguation mechanisms, as described in the remainder of this document. To support the knowledge acquisition aspect of MIKROKOSMOS, a set of tools is being developed for use by lexicographers to facilitate lexical and ontological entry, as well as for corpus analysis and induction of information from MRDs. The expected products of the MIKROKOSMOS effort include Japanese-to-English and Spanish-to-English machine translation system prototypes, utilizing components of the PANGLOSS system from ISI, CMU, and NMSU, funded by DoD and ARPA.

The components of the DIANA and MIKROKOSMOS efforts that are within the scope of the research described in this document include: definition of the syntax/semantics interface, development of the lexical semantic specification language, definition of the overall semantic analysis search space (see Figure 2A), and definition of two operators for traversing the search space. Other contributions of the author, such as definition of the lexicon format and content (jointly with others), is outside the scope of this work. The core of the effort described here, however, is the use of the graph search over the ontology as the mechanism for constraint satisfaction and relaxation, and the application of that search to building the Semantic Dependency Structure (SDS) that is the basis of the meaning representation that is built be the semantic analysis process. This constraint satisfaction process is generalizable well beyond straightforward constraint satisfaction and modest relaxation, in that it can also be used for resolution of metonymy and word sense disambiguation in general.

1.2 Setting the Stage

Using terminology from Lakoff (1988), the paradigm within which the current work is framed is in the *experientialist cognition* camp, as opposed to the *objectivist cognition* camp which includes all logic-based or set-theoretic approaches to semantics; although both models commit to links between human conceptual structures and a real world (i.e., *basic realism*), the experientialist approach "accounts for what meaning is to human beings, rather then trying to replace humanly meaningful thought by reference to a metaphysical account of a reality external to human experience" (p. 120). Instead of arbitrary symbols associated with things in the real world, manipulable by logical processes and as set-theoretic constructs (perhaps including defining, typical, necessary, or sufficient properties), the experientialist approach assumes the following mechanisms: internal structure to the symbols which represent basic-level concepts; some relations (image-schemas) defined on these concepts such as *containers, paths, links, part-wholes*; some set of *imaginative processes* for forming abstract cognitive models, such as *schematization, metaphor, metonymy,* and *categorization*. (p121). Lakoff argues that such a schema is not an "internal representation of external reality", but a representation of a reality that exists in the minds of humans, which is a conceptualization of the real world. Under Lakoff's definitions, the paradigm within which this work is framed could be considered experientialist:

- The symbols of the representation (i.e., the concepts in our ontology) are not defined in isolation, but are defined in a hierarchy which reflects relations between these concepts.
- The internal structure of our symbols identifies attributes, properties, and relations (i.e., each symbol, or concept in our ontology, has a defined set of differentia which distinguish it from its siblings).
- This model has image-schemas, reflected by the relations defined over the ontology, such as part-whole relations.
- The system has imaginative processes (complex events, metonymy, categorization).

Adopting a model which parallels Lakoff's experientialist view allows the meaning representation and reasoning processes to be capable of handling metonymy and metaphor, capturing the relationships between senses inherent in regular polysemy, in addition to performing an elaborate range of lexical, syntactic, semantic, and pragmatic disambiguation by use of constraint satisfaction over the ontology.

In fact, the basic premise of the KBMT approach is that handling all of these issues (and more) is necessary for achieving high-quality MT, and that the only way of ensuring that all such complexities and ambiguities have been resolved is by producing an unambiguous meaning representation. The depth of the meaning representation is a matter of debate within the interlingual community, with some allowances given to practical (system-building) considerations. We rely on a comprehensive well-defined ontology both to define the primitives of semantic meaning representation and to define the search space for constraint satisfaction and relaxation.

The motivation for performing full meaning disambiguation in translation has been debated for years. In addition to explicit arguments for the need for language-neutral meaning representations, such as those presented in Levin and Nirenburg (1994b), the MT literature is full of indirect arguments which support such a position. The experience of the transfer (and, more recently, of the "purely" statistical) MT paradigm researchers is that, although moderate quality of output isn't (intellectually) difficult to achieve, in order to improve quality, the approaches have to wrestle with problems with lexical gaps, lexical misalignment, divergences, one-to-many and many-to-one transfer rules, idioms, differences in predicate/argument structures, etc. (for example, Dorr (1994) expends considerable effort mapping out and classifying such divergences). All of these problems stem from the faulty premise that there is sufficient parallelism in lexical items, predicate/argument structure, and syntactic structure between source and target language texts. Although the KBMT paradigm and the ontological semantics framework are plagued with other difficulties, none of those issues is ever a problem — they are all intrinsically dispensed with by the language-neutral intermediate meaning representation of the ontological semantics model.

In Figure 1A, various approaches to interlinguae, semantic representation, and related knowl-

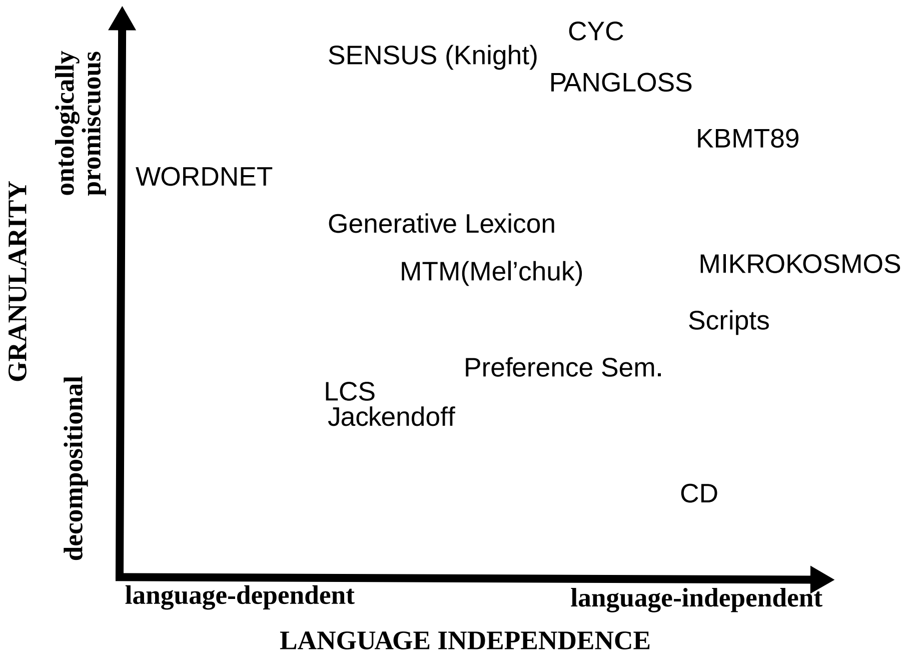


Figure 1A. Classification of approaches to interlinguas or meaning representation

edge bases (referenced in detail in subsequent chapters) are rendered against two dimensions. The first dimension, language-independence, plots the degree to which the primitives of the ontology or language are bound to or reflect a particular language. This can be difficult to establish, because terms from one language may be used to label the primitives even if they do not strictly mirror the one language. The second dimension identifies the granularity of the primitives, hence their number. At the decompositional end of the spectrum, few primitives are compositionally combined to form the meanings of all the lexemes of the language; at the other extreme, a one-to-one mapping exists between every word in the language and a primitive in the representation or ontology.

The MIKROKOSMOS approach adopts maximal language-independence in the ontology, because of the observation that using language-specific concepts will lead to misalignment of senses, lexical gaps, and all the other problems that plague the transfer-based paradigms. Our ontology includes concepts which are maximally specific while still being valid cross-linguistically, meaning that they are typically lexically (or conventionally phrasally) renderable. The specific set of concepts can be adjusted to accommodate new languages or domains (although recent experience in adding English, after extensive acquisition of Spanish, required only 0.2 of 1% of the ontology to be changed).

Recent work in using statistical models for lexical correspondences (i.e., transfer rules) would fall in the maximally promiscuous extreme of the granularity dimension, and in the minimally language-independent extreme. Since the discriminatory power of a statistical model of lexical correspondences is equivalent to a Markov chain, is seems clear that this approach, in isolation, will not be able to perform discrimination which requires the larger contexts that structures such as syntactic parse trees (not to mention semantic dependency structures) can provide. As mentioned above, the premise of the KBMT approach is that it is necessary to resolve metonymy, reference, metaphor, and a host of other phenomena to be able to produce accurate translations. In fact, the statistical transfer approach has had significant difficulty with the same sorts of problems of divergences (outlined above) as the transfer approach. Statistical MT systems, such as Brown *et al.* (1991), are increasingly incorporating symbolic techniques to address the accuracy concerns.

Where the statistical work can be useful for KBMT is in providing certain knowledge sources or processing approaches for microtheories. Even something as basic as word frequency over a corpus can be used to adjust the preferences to slightly favor a more-likely interpretation; other microtheories could overcome this factor, based on other evidence. Other statistical modeling techniques could be useful for knowledge acquisition. For example, in acquiring the ontology, techniques which provide clusters of word that appear related, based on their contextual distribution over a corpus, could be helpful in identifying concepts and their breakdown; applying this technique over corpora from multiple languages can provide the sort of comparative material that would be useful in identifying language-neutral concepts for the ontology. Additionally, statistical techniques can be exploited in generation components of KBMT, for example, in a collocation-based lexical selection microtheory.

1.3 Practical Computational Semantics

As pointed out in King (1992), semantics researchers can have various goals, and practitioners of semantics pursue a model of research that addresses their goals: there are computational linguists whose goal is AI or NLP, there are the psycholinguists, and then there are the formal semanticists (in the sense of logic-based semantics). Our goals place us squarely in the first of these three categories. That isn't to say, however, that psycholinguistic and formal semantic theory issues are of no concern to us, just that they play a significantly secondary role to the practical goal of building a working NLP application. Raskin (1990) suggests that "no significant progress in NLP semantics is possible without a comprehensive formal theory", an opinion which we share, since Raskin's notion of a formal theory is aligned with that outlined below, as opposed to the formal model-theoretic logic-based semantic theories espoused by numerous other researchers.

Thus, we need to concern ourselves with the nature and content of a practical computational semantic theory. We find that such a theory needs to address the following four desiderata:

- Adequate expressiveness of meaning representation.
- Machine-tractable representation language.
- Explicit procedure for mapping between utterances and the meaning representation.
- Recovery procedures to map and express unexpected and non-literal input.

We feel that any system or framework that meets these requirements should be considered a complete practical computational semantic theory; these are not merely engineering considerations, but formal and structural concerns that reflect the practical and computational aspects of the structure of a semantic theory, as discussed by Katz and Fodor (1963). They ask what the descriptive and explanatory goals of a semantic theory are, and we readily identify that for a practical computational one, the goals are supporting the computational application, and nothing else. This points out an obvious assumption of the above list of four desiderata: the existence of a representation of meaning. Regardless of the form of the representation, whether a set of statements in a meta-language, a procedure, or a state of an abstract model, representation of meaning is necessary for all NLP applications of interest (otherwise the applications wouldn't be concerned with computational semantics); without a representation of meaning, the NLP application wouldn't have available to it the information content of the input/output natural language.

1.3.1 First Desideratum

This brings us to the first desideratum, namely that the meaning representation needs to be sufficiently expressive to represent all the elements of meaning that the NLP application needs to respond to, whether by rendering the same content in another language, or expressing the information in a data-base format, or executing a natural language command. Jackendoff (1988) sees that any semantic theory "must be rich enough to express all the distinctions of meaning available to the intuitions of speakers of the language"; we qualify that, somewhat, for the computational semantic theory to only require the distinctions of meaning required for the practical application. Note that we aren't necessarily requiring that the practical computational semantic theory support representing the "full" or "deep" meaning of an utterance, only what is required for a particular application, which will vary widely from case to case.

This desideratum immediately points out the differences between our goals and that of the formal model-theoretic, truth-conditional semanticists, in whose number we include a wide range of semantic practitioners, including Charniak and Goldman (1988), Barwise and Perry (1983), Barwise (1989), McDermott (1978), and Johnson-Laird (1988), in addition to the other obvious candidates like Lewis, Tarski, Davidson, or Montague. We agree with the observations of King (1992), Sowa (1993), or Wilks (1992b) that the sort of logical formalisms pursued by these logical semanticists are unlikely to be developed to the point where they will capture the range of propositional and non-propositional meaning that, we believe, is necessary for higher-quality MT or for supporting complex inferences, including aspects of meaning such as stylistics, attitudes, various nuances, the range of tropes and rhetorical devices, differences of meaning due to super-sentential context or discourse setting, etc. That isn't to say that it is impossible to define a logical system that can capture all these aspects of meaning (although it certainly won't be first order); we just observe that these practitioners aren't interested in addressing the sorts of problems that practical computational linguists building systems need to worry about, such as word-sense disambiguation, ill-formed input, PP-attachment, etc., not to mention the fine-grained distinctions of meaning that speakers intend. King (1992) characterizes the goals of this groups of researchers:

> The task is to describe, in precise and rigorous terms, a relation between the sentences of a natural language and a semantic representation, where the semantic representation will in its turn provide a link between the sentence and a set of possible worlds in truth conditional terms, so the sentences are seen as truth value denoting entities, and a possible world is a model of the theory captured by the formal description of the relation between sentences and semantic representation if and only if all the sentences which evaluate to true in the theory correspond to true states of affairs of the world. (p. 287)

Practical computational applications do not, as a rule, concern themselves with truth conditions or possible worlds as an end result; it remains to be demonstrated that reliance on inferencing with truth conditional logic can add practical capabilities to a computational NLP system that "common-sense" semantics cannot (see Wilks (1992b)). As mentioned above in Section 1.2, in our application we don't worry about the state of the world, but about representing the intended meaning of speaker utterances.

Of the approaches that aren't logic-based, however, there are still problems with expressiveness. As discussed in Wilks (1992b), Onyshkevych and Nirenburg (1994), or Arnold (1996), the LCS model of Dorr (1993), Dorr *et al.* (1994), and Jackendoff (1983, 1988, 1990) and colleague is nowhere near expressive enough for full representation of meaning needed for KBMT, since they focus mostly on lexical semantic aspects of verb predication and argument structure. The early work of Katz and Fodor (1963), although perhaps headed in the direction of expressiveness, remained inadequate (Raskin (1990) criticizes that they don't distinguish, in meaning, between *man* and *bachelor*).

Certain efforts in semantics or computational semantics do, in fact, concerns themselves with adequate expressiveness, and, in fact, focus on expanding their meaning representations or formalisms to handle certain non-core aspects of meaning. Mel'chuk and Zholkovsky (1984), for example, identify (without developing a full semantic theory) a wide range of semantic relations that may exist between elements of meaning, however, in a different context and despite their interest in their Meaning-Text Model as a "logical device which associates with any given meaning *M* the set of all the texts in this language which are expressions of *M* (and which are consequently synonymous with one another)". Hovy (1988) developed a structure for representing stylistics as an aspect of meaning, an approach we follow in our framework. Other frameworks, such as Sowa (1993), Hirst (1987), Kamp (1981), Carbonell and Tomita (1987), or Dyer and Zernik (1986), do attempt to capture a range of aspects of meanings, but haven't achieved the full expressiveness that we find necessary for high-quality KBMT; in fact, we believe that we have one of the most expressive meaning representation mechanisms for computational semantics to date (but note that expressiveness does not necessarily correlate with granularity of primitives).

1.3.2 Second Desideratum

Competing with the issue of expressiveness of a meaning representation, perhaps, is the second desideratum, namely that the representational formalism be machine-tractable. The reasons for including this desideratum also lie in the practical computational nature of the theories or frameworks under discussion, in that meaning representations need to support inferencing (as part of the process of capturing the range of meaning) as part of the overall application in question. Although the meaning representation language may need to support vagueness, if the meaning representation language is ambiguous, then the inference mechanisms (either as part of the analysis process or as back-end application processes) aren't able to function deterministically without completing the disambiguation of the meaning. Although Wilks (1996) points out that some ambiguity doesn't hurt the tractability of a formal language (e.g., `NIL` in LISP), it is clear that unrestricted natural language involves too much ambiguity to be machine tractable in the manner specified above.

Although not the only way of approaching the issue of machine tractability, we choose to employ a formally defined meta-language with a concrete syntactic specification (see Section 3.3). The atoms (or primitives) of this meta-language are unambiguous and with a defined semantics (see Section 3.1). We define a set of primitives by means of a taxonomy of atoms (representing "concepts"), augmented with a network of relations, as well as features or properties on each atom. Johnson-Laird (1988) (although arguing against decomposition, an issue addressed in detail in Section 4.1), finds that "by itself, the symbol WOMAN is meaningless, but it becomes more meaningful if the specific values of a number of properties are attached to it." Our approach, as does his, has numerous types of associations, which can be used in combinations.

A group of practitioners, including Fass (1986, 1988) and the work in Boguraev and Briscoe (1989), relies on words of English as elements of the meaning representation language. As discussed at greater length in Section 4.1.1, this approach clearly doesn't satisfy the requirement of machine tractability outlined above, thus wouldn't support the range of inferencing and applications that a set of non-linguistic primitives would. However, we don't find the objections of "upper-case semantics" (see McDermott (1978)), "markerese" vs. "mentalese" (see Lewis (1972) and Wilks (1975a)) necessarily relevant to all of the computational (i.e., non-logic-based) approaches that define their own set of primitives; that is, although some of these approaches use English-language-inspired names for their concepts, we are comfortable with words as atoms, so long as the names are unambiguous (i.e., only relate to one of the English word senses of the word) and aren't interpreted by English lexical semantics (instead, are formally defined, as in the network-based model above, or are interpreted by means of the procedures that can apply to these primitives, as in Wilks (1975a)), and aren't combined by rules of English syntax (but by the syntax and semantics of a formal meta-language). For further discussion of this issue, refer to Nirenburg *et al.* (1995).

One of the points above needs further discussion here, and that is the issue of enumeration and definition of the primitives. Without some mechanism of definition (such as the network model approach, which is isomorphic, for this purpose, to the Wilks procedural interpretation approach), the primitives of a semantic representation do not have a semantics of their own, and thus the meaning representation does not convey meaning; those approaches that rely on the English denotation of their primitives alone run into the same difficulties encountered by the circular word-based semantics described above. In order to build a practical computational semantics, therefore, we find it necessary to enumerate and define all the primitives of the meta-language. A wide range of work fails to do this, resulting in incomplete semantic theories and problems in practical applications; this group includes the generative lexicon practitioners such as Pustejovsky (1991, 1995), Pustejovsky and Bouillon (1995), or Buitelaar (1997), the conceptual graphs work of Sowa (1993) and others, the lexical-conceptual structures of Dorr (1993), Jackendoff (1983, 1988, 1990) and colleagues, the naive semantics work of Dahlgren *et al*. (1989), and many others.

1.3.3 Third Desideratum

The third desideratum is an explicit discovery or construction procedure for mapping between the surface form or string and the meaning representation addressed by the former two desiderata. In some sense, it isn't the nature of such a procedure that is part of the semantic theory, merely the existence of the procedure to validate that the meaning representation is producible from the string (or v.v.) Although falling short on the first two desiderata, efforts such as the abductive reasoning models of Hobbs and Martin (1987) or Hobbs (1991), or the conceptual graph models of Sowa (1993), or Hirst (1987) make a significant effort to demonstrate the existence of such a procedure. In fact, if we used the model of Evans and Scott (1986), these procedures not only validate the representation of meaning at a sentential or propositional level, but also at a lexical level.

Another group of frameworks, including Wilks (1975a), Woods (1970), Winograd (1983), and Small and Rieger (1982), also define significant mapping procedures; their procedures effectively define (or are) the semantics of the meaning representation language, per the second desideratum, but fail to scale nearly enough to achieve sufficient expressiveness in meaning representation. Some of these approaches such as Small and Rieger (1982) or Schank (1973) require mapping procedures or lexical knowledge so complex as to defy any possible hope of scaling up to a real application.

This issue is a significant one for our work, and is addressed in Section 7 and Section 8.

In discussing the issue of mapping procedures, we need to address the question of compositionality. Because of the extent of meaning that we envision being represented, per the first desideratum, it seems unlikely that any model following strong Montagovian compositionality could achieve necessary levels of meaning representation. This follows from the lack of syntax/semantics parallelism in the cases of non-propositional meaning (and also some propositional meaning). We essentially require either no compositionality or weak compositionality, exemplified in Jackendoff (1988) by "interpretation of a noun phrase is scattered widely through the interpretation of the sentence as a whole". In fact, Wilks (1992b) finds that, in all practical cases, compositionality is either trivial or non-existent. We find that in our model, our weak degree of compositionality is often flouted, especially in the cases that the last desideratum addresses.

1.3.4 Fourth Desideratum

The fourth of our desiderata addresses the need of a mapping procedure and meaning representation for unexpected input and non-literal language. We treat this as a core desideratum and not as an engineering concern because the nature of real natural-language data that is encountered outside of the laboratory is such that one cannot avoid absolute ill-formedness in the form of typos, grammatical mistakes, incomplete constructions, etc., or, even more importantly, input in the form of non-literal language such as idioms, conventional expressions, metonymy, metaphor, irony, and a range of tropes, among other phenomena. The move from well-formed laboratory texts to real-world data is so drastic that any practical computational semantic theory needs to include a battery of procedures and representational vehicles to handle the meaning of real data. No framework that doesn't address these issues could be called a practical computational one.

Certain research efforts, in fact, focus almost exclusively on addressing issues raised by this desideratum (at the expense of the other desiderata, by the way). Fass (1986, 1988) almost exclusively focuses on metonymy, metaphor, and other such phenomena. Carbonell (1981) addresses metaphor. Fillmore *et al.* (1988) address the issue of idioms and set expressions or constructions. Grosz *et al*. (1986), Hobbs and Martin (1987), Fauconnier (1985), Kamei and Wakao (1992), Stallard (1993), and others referenced in Section 9.2, focus to various extents on the issue of metonymy.

On the other hand, a host of other frameworks that haven't addressed the issue of real corpora to any significant extent haven't yet considered the implications of all these types of unexpected input. Among them are Schank (1973), Cullingford (1981), Cullingford and Onyshkevych (1987), Pustejovsky (1991, 1995), Pustejovsky and Bouillon (1995), Buitelaar (1997), Sowa (1993), Dorr (1993), Dorr and Voss (1994), Jackendoff (1983, 1988, 1990), Dahlgren *et al*. (1989).

The approach described here has begun addressing the concerns expressed in this desideratum. For example, Section 3.4.2 addresses idioms, Section 9 addresses metonymy (a significant focus of this work), and Section 5 considers the general case of semantically unexpected input. However, to date, we haven't addressed at all the issue of absolute ill-formedness, not have we addressed any of the other tropes to any depth, other than an initial effort to address metaphor.

1.3.5 Other Considerations of a Theory

There are a few potential concerns that are conspicuously absent from the above list of four desiderata. As addressed at length in Nirenburg *et al.* (1995) and Nirenburg and Raskin (1996), the issue of reproducibility isn't relevant to practical approaches to computational semantics; in short, reproducibility addresses discovery of existing entities or phenomena, while practical computational work is concerned with construction of a meaning representation. These representations aren't evaluated on the basis of canonicality or reproducibility, but sufficiency and accuracy for the application at hand.

As alluded to in Section 1.3.1 above, we don't feel that formal logical or denotational specification of the theory or its components is necessary; however, there is certainly no reason why formal axiomatization shouldn't follow the development of a practical computational theory once it has demonstrated success.

Another concern of scientific theory is falsifiability. In the case described here, application success suggests an existence proof of the success of a theory, while application failure might suggest (but doesn't necessarily demonstrate) failure of the underlying practical computational semantic theory. More direct falsification would be demonstrated by presentation of relevant corpus data that couldn't be represented by the meaning representation language or procedurally produced by the application. This engineering-failure approach to falsifiability may not be satisfying, but reflects the practical nature of the frameworks under consideration.

1.4 KBMT and Ontological Semantics

The Knowledge-Based Machine Translation model assumes a rather "deep" analysis of the input text, with a representation of the meaning in a machine-tractable meaning representation language; the particular language used in the work described here is called Text Meaning Representation (TMR). In addition to reflecting the contribution of the (often complex) lexical semantic specification of the lexemes in the input text, the TMR attempts to capture other information which contributes to the meaning of the input text, and which might need to be rendered in the target language string in the process of Machine Translation; thus, in addition to the lexemes from the input text, semantic information encoded in the morphology, syntax, relationships between words, and discourse structure of the input text is also rendered.

In addition to basic propositional content of the input text (sometimes referred to as the *who-did-what-to-whom* component), we find it necessary to capture a range of other components of the semantics of text in order to render felicitous and accurate translations. In the approach described here, these other components of meaning include aspect, focus, modality, reference, speaker attitudes, speech acts, time, stylistics, and other pragmatic factors. Notice, however, that the inventory does not include any syntactic manifestations, such as tense. The information present in the syntax and predicate/argument structure of the input text, in addition to contributing to the meaning in some cases (such as identifying focus), contributes substantially to the disambiguation and semantic dependency structure building process, via the syntax/semantic interface. No strictly syntactic (or source text predicate/argument structure) information from the input text *per se* is represented in the meaning representation itself.

The meaning representation, i.e., the TMR, conforms to the ontological semantics approach, as first suggested (under another name) in Nirenburg and Levin (1992). In other words, the basic propositional component of the meaning is often represented by reference to an ontology. Our ontology knowledge source reflects a speaker's model of the world, which, in addition to a taxonomic organization of concepts which are the atoms of the meaning representation, also includes a variety of properties, attributes, and relations among these concepts.

The intent in building this ontology is that it be language-neutral. Although the concepts in the ontology may be labelled by one or more words from a particular language, those labels are for the convenience of the developers, but should not be interpreted using the full lexicon and semantics of the natural language from which the labels were taken. Each concept unambiguously (but possibly vaguely) represents one specific concept (or natural kind) from the world being modeled. The granularity of the ontology is pragmatically determined, although, roughly, it is intermediate between the full-decompositional approach (such as Schank (1973)) and the one-concept-per-wordsense approach (such as the SENSUS ontology assembled at ISI). The granularity of the working ontology will be as fine as necessary to discriminate between word senses in each of the languages being considered (currently Spanish, Japanese, and English). Since the concepts can be further constrained (by adding information), composed, or augmented by non-ontological structures (such as speaker attitudes, stylistics, and relations) in the lexical semantic specification, it is possible to use a smaller set of ontological concepts and still achieve a high degree of meaning discrimination.

The basic form-to-meaning correspondence in our approach is produced by a processing mechanism which consists of a cluster of *microtheories*, each of which is a specialist on a particular language phenomenon or processing issue (see Nirenburg and Levin (1992), Levin and Nirenburg (1994b)). Many of these microtheories are responsible for enhancing or refining the TMR in regard to specific language phenomena, such as aspect, definite reference, stylistics, discourse structure, metonymy, etc. While many of these microtheories are distinct processing modules which operate on private as well as shared knowledge sources, others are reflected by specific information in static or dynamic knowledge sources that are available to all microtheories (such as the microtheory of adjectival meaning described in Raskin and Nirenburg (1995)). Basic Semantic Dependency Structure (SDS) building is performed by one such microtheory, which relies on another microtheory of constraint satisfaction/relaxation. Metonymy resolution actually takes the form of knowledge which is available to the SDS and constraint-satisfaction microtheories, but without requiring an additional processing module.

In this microtheory approach, all available clues (including syntax, morphology, lexical items, etc.) from the input text (as well as expectations about human communication) can contribute to the construction of meaning. The processing of idioms, conventional language, and other constructions (using a term from Fillmore *et al*. (1988)), as well as handling of metonymy and metaphor, is done in a non-compositional manner. Even with productive language use, the analysis process is not necessarily strictly compositional, and context (syntactic, lexical, or previous TMR constructions) can affect the way in which various elements of meaning are resolved. Contradictory information could be provided by various microtheories, based on different input clues or static knowledge sources. Multiple interpretations can exist in parallel, each with a cumulative indication of likelihood of that reading. Thus the overall semantic analysis process (i.e., the construction of the meaning representation) can be viewed as an abduction process, as described in Hobbs

(1991); the construction of a TMR involves identifying the most plausible (yet defeasible) hypothesis that is compatible with the input. This conforms with the Hobbs and Martin (1987) strategy: “Language does not give us meanings. Rather, it gives us problems to be solved by reasoning about the sentence, using general knowledge. We get meaning only by solving these problems.”

The overall strategy for MT, both the analysis and generation components, in our ontological semantics framework involves the use of all available sources of knowledge, hence the appellation KBMT. The availability of a diverse range of evidence, however, raises the issue of combining that evidence to produce a single result. This issue is a more general one that just a concern for KBMT, and has been recently addressed in other NLP-related sources such as McRoy (1992), Levin and Nirenburg (1994b), Harley and Glennon (1997), Wilks and Stevenson (1997), or Jones and Onyshkevych (1997); Section 7.5.4 below addresses this issue in more detail.

One of the motivating factors for differences in depth of analysis and granularity of various MT models is the intended use of the system and the concept of operations. For example, the KANT system, described in Nyberg and Mitamura (1992) or Carbonell *et al.* (1992), is designed to work in an environment with substantial control over the grammar and vocabulary of the input text (the controlled language is even grammar-checked, resulting in no ungrammatical or non-literal input, and most metonymic and metaphorical expressions in the sublanguage are lexicalized), in a specific domain, therefor is less concerned with word sense disambiguation, and does not need the expensive mechanisms for trope resolution, stylistic analysis, etc. that might be required in a system that is addressing unrestricted text in a general domain (as is the intended application of the work described here).

1.5 Scope of this Work

There are numerous knowledge sources, formalisms, and inference mechanisms that are required to implement the mechanism described here in a working semantic analyzer for a KBMT system. The definition of some of these knowledge sources is within the scope of the work presented here, while a number of others (although described as background information) fall outside the scope of this work, which focuses on the lexical semantics, building SDS, and, most importantly, a certain approach to constraint satisfaction which involves viewing the ontology as a searchable multi-dimensional graph, and the application of this approach to a certain model of metonymy resolution and word sense disambiguation. The definition of lexical semantic specification also includes the syntax/semantics interface, which provides expectations for building the SDS.

The SDS-building mechanism, implementable under a variety of control structures, is best understood as a search through a space whose states are partial TMRs, along with other parameters. The operators for traversing this state reflect both data-driven instantiation of partial TMR structures as well as expectation-driven attempts to combine various partial structures, according to expectations either from the syntax/semantics interface or from the syntactic parse itself. In the combination operator, before two TMR fragments can be combined, the constraints on the potential relation between the fragments needs to be checked.

The fundamental heuristic for constraining the search space is a particular view on the constraint satisfaction process. Each type of relation in the ontology specifies a dimension of the ontology, with a certain cost for traversing arcs in each dimension. Viewing the entire ontology (with all the arcs) as a multi-dimensional graph, it is possible to view constraint satisfaction as a best-path problem between the constraint and the candidate concept (or instantiation) that is being judged against the constraint. The core of basic SDS-building is verifying that candidate arguments satisfy selectional restrictions on the role; this constraint satisfaction process is performed by this ontological graph search process.

In order to demonstrate that the SDS-building algorithm, with constraint satisfaction via the ontological graph search, can be used to address traditionally distinct phenomena in addition to basic selectional restrictions, several issues are treated below: relaxation of such constraints for unexpected semantic arguments (such as in *the baby ate a penny*), metonymy resolution, and word-sense disambiguation. In other words, basic semantic dependency structure building, metonymy resolution mechanisms, and word-sense disambiguation mechanisms all have at their core the same fundamental constraint satisfaction/relaxation requirement which can be addressed by the same algorithm. Metonymy resolution has specific triggering mechanisms; other phenomena which might be addressed (but not necessarily completely solved, of course) by the same fundamental mechanism, including metaphor processing, nominal compounding, and reference resolution, as well as some problems in generation, would have their own set of triggering conditions.

1.6 Outline

The organization of this document is as follows. Section 2 presents a shallow overview of the paradigm, the key knowledge sources, and the base semantic analysis process (but it does not cover the application of the ontological graph search process to constraint relaxation, to metonymy processing, or to word-sense disambiguation). The various knowledge sources that are used in the analysis process, namely the ontology, the lexicon, the meaning representation language, and syntactic representation, are described in much more depth in Section 3; the subsequent section focuses on the lexical semantic specification language, since it is so crucial to semantic analysis and the representation of text meaning. Section 5 presents the actual core of the approach: the ontological graph search, the central heuristic which underlies semantic constraint satisfaction and relaxation. Section 6 discusses the traversal of the search space of possible text meaning representation in a somewhat abstracted manner, whereas Section 7 discusses SDS-building, specifically, as a traversal of the search space. This generalized SDS-building process, using the ontological graph search, is applied to the general problem of Word Sense Disambiguation (WSD) in Section 8, to the processing of metonymy in Section 9, and Section 11.1 speculates on the application of the process to the resolution of N-N compounds as well. Since the subtask that is common to many task in NLP is WSD, whether in a meaning-based approach or not, Section 8 also incorporates a discussion comparing the general SDS-building and constraint-satisfaction approach to other work, on the basis of their model of WSD. The ontology is the knowledge source which provides both the primitives for lexical semantic specifications, and provides the graph over which constraint satisfaction is determined; one of the most frequent criticisms of the knowledge-based approach to Machine Translation is the difficulty in building the ontology, so Section 10 briefly suggests some ways of automating ontology acquisition, along with some acquisition methodology directions.

2. Overview of the Semantic Analysis Process

This section outlines our overall approach to semantic analysis, as discussed in the rest of the document. Section 2.3 outlines the various knowledge sources detailed in Section 3 as well as the discussion of lexical semantic specification from Section 4. The state space search is outlined in Section 2.2, and discussed in more detail in Section 6 and Section 7, with a discussion of the ontological graph traversal component in Section 2.4, and in more detail in Section 5. A brief example is presented in Section 2.5.

2.1 Introduction

This section overviews the foundations and methodology of the approach to semantic analysis espoused in this work, where semantic analysis is viewed as an embedded element of a knowledge-based machine translation (KBMT) system. The overall goal of this work has been called "deep" semantic analysis, because of our attempt to capture as much as possible of the linguistic meaning of an input text stream (what is meant by "meaning" is discussed in Section 3.3), and to represent that meaning by using a set of well-formed structures in an unambiguous machine-tractable knowledge representation language. A basic premise of the approach taken here is that in order to perform such analysis, it is necessary to have substantial knowledge about the language and about the world, hence the appellation knowledge-based.

We consider semantic analysis to combine the construction of a basic *semantic dependency structure* (SDS) and the augmentation of this structure (by specialist processes called *microtheories*) with additional constraints and other information (such as reference, resolution of deixis, etc.) gleaned from the available lexical, syntactic and other evidence in the input. In its most straightforward incarnation, the SDS-building process relies on meanings of atomic lexical units, as defined through links to the ontology and by non-propositional meaning elements; the SDS-building process is guided by the syntax-semantics interface manifested in the lexical syntactic and lexical semantic specification of lexical entries.

The bulk of the work described here falls in the category of SDS-building; a brief description of microtheory-based augmentation follows. Each microtheory treats a particular language phenomenon, whether language-specific or general; a working machine translation system in this paradigm would need to have a battery of such microtheories to handle such phenomena as definite reference and aspect. In the process of adding information to (thus further constraining) the semantic analysis produced by the SDS-building process, the microtheories also assist in traversing the search space (described below) by selecting from among candidate analyses generated by the SDS process or other microtheories, by pruning out certain readings entirely, or by adjusting the *preference* (defined below) of a particular reading or set of readings. The remaining discussion will treat the basic SDS-building process and the treatment of language phenomena covered within its general framework (e.g., metonymy, word-sense disambiguation).

The readings or semantic interpretations generated by the SDS-building process (both as intermediate and as final results) are expressed in terms of a meaning representation language. The particular language that is used is not of critical importance, as long as it meets a number of criteria regarding its expressiveness and deterministic properties; a more detailed discussion of this representation language can be found in Section 3.3. Regardless of the language, the meaning is represented as an augmented network of instantiations of concepts from the ontology. The particular language used for illustrative purposes in this discussion is called TMR (in an earlier incarnation: TAMERLAN), reflecting the language used in the experimental implementation of this approach. Since the meaning representation language is necessarily unambiguous, a particular meaning representation can serve as input to a language generation mechanism (as would be the case in Machine Translation), or could be used as input to an inference, analytic, or fusion application.

The goal of the SDS-building process is to find the most appropriate semantic interpretation of the input text. Candidate readings are ranked according to their *preference* values, a cumulative measure of evidence or likelihood that is used to order competing interpretations (the use of this term is different from its familiar meaning introduced by Wilks (1975b)). If the assignment of preferences by the search process is appropriate, then the interpretation with the highest preference value at the end of processing should indeed be the one which human translators would choose. Preference values are used by the search heuristic both for pruning paths with low preferences, and for guiding a best-first search method. The preference in the current implementation is a value in the interval *[0.0, 1.0]*, with adjustments to the preference typically made by a multiplier.

Both incrementing and decrementing adjustments are possible, reflecting an increased likelihood on that reading (for example, if the reading reflects the use of a typical collocation or idiom) or a decreased likelihood on that reading (as is the case when any constraint violation occurs). Determining how to adjust preference values in a particular case is an issue of critical importance to the success of this approach, and a variety of factors influence this decision (our current implementation actually only uses decrementing adjustments).

The process of building semantic dependency structure is considered here from the point of view of traversing a search space of all possible semantic constructions (both well-formed and incomplete) in order to find the semantic construction that best represents the meaning of the input text. Each state in the search space is referred to as a *reading*, and has an associated preference reflecting the likelihood of that reading. A particular state may be final (i.e., well-formed and complete), or incomplete (where portions of the meaning of the text have not been incorporated into the reading yet). Some further information on the search process is given in Section 2.2 below.

The two operators for expanding or traversing nodes in the search space (i.e., the processes which actually build the SDS) are *instantiation* and *combination*. Section 2.4 discusses in some detail the process by which a step in building the SDS from component pieces (i.e., the *combination* process) uses knowledge from the lexicon and the ontology to determine the likelihood of that step.

2.2 The Semantic Dependency Structure Building Process

In the abstract view, the SDS-building process is a state-space search process. Each state in the state space is characterized by a TMR structure and a syntactic parse structure. The initial states of the search all have a null TMR structure and one of the parse structures from the parse forest. The goal states in the search space are characterized by a fully-consumed parse forest (i.e., complete), and a fully-connected TMR structure with no unfilled obligatory argument positions (i.e., well-formed). The state-to-state transitions in the state space are over either instantiation arcs or over combination arcs. When applied to a state, both the combination and the instantiation process can result in multiple succeeding states (branching the search graph).

The flow diagram in Figure 2A illustrates the top-level data flow in the semantic analysis process. The input to the process is a text string, initially segmented into sentences. The sentences are

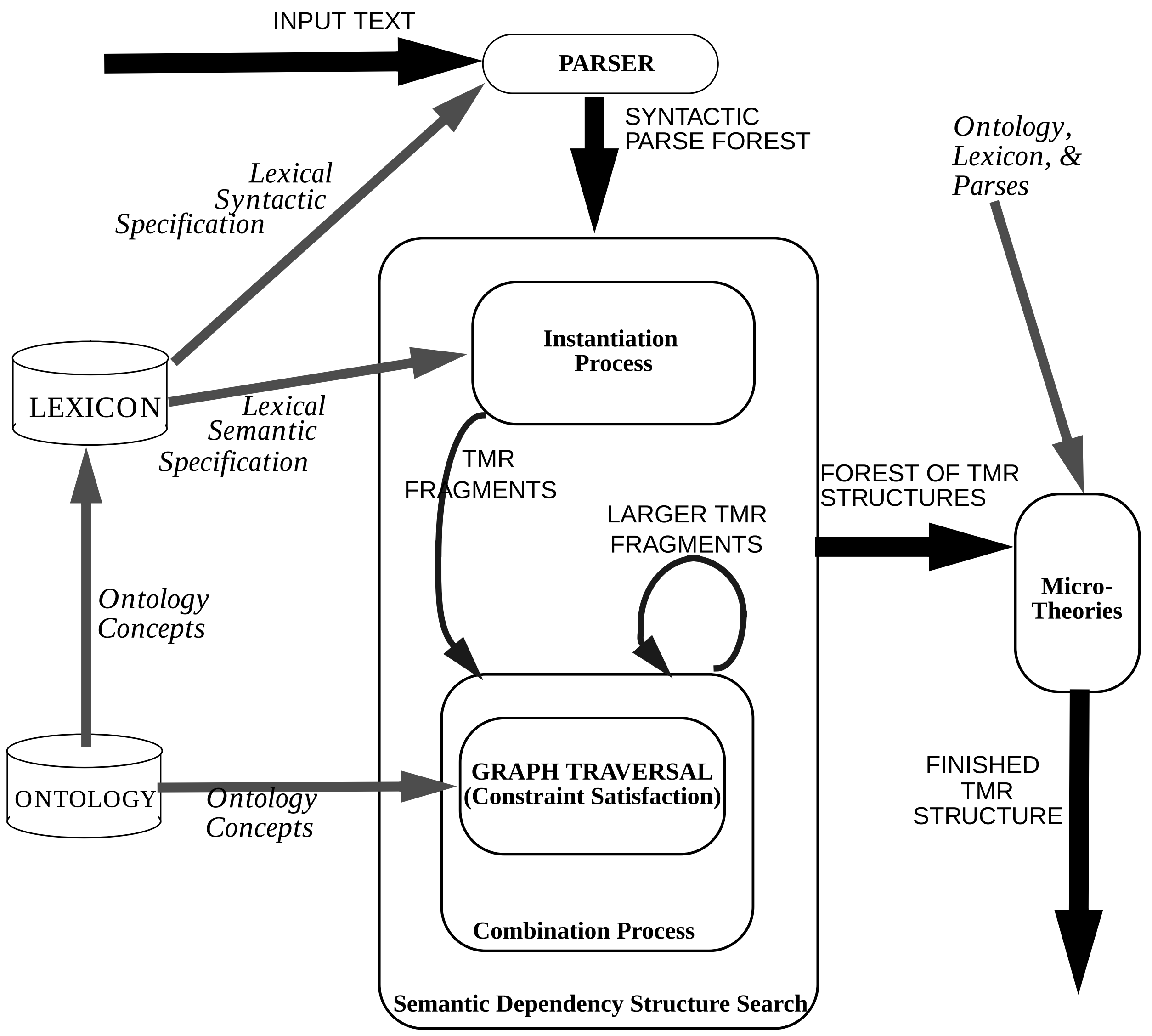


Figure 2A. Overall data flow of the architecture, showing knowledge sources and flow of processing: parsed input undergoes instantiation and combination, and the resulting partial TMRs are augmented by microtheories.

parsed by a syntactic parser to produce a forest of full or fragment parses (if a single full parse is not available). If there are multiple parse paths, using a packed forest representation allows efficient processing of that set of parses. The forest of parses is passed to the overall SDS-building control process.

Given one parse or parse fragment, the instantiation process *instantiates* each syntactically appropriate word sense from the input syntactic structure according to the lexical semantic specification of that word sense. Note that the final TMR does not include any syntactic information about the input string. Syntactic information is used as a set of clues necessary (though, certainly, not sufficient!) to guide the semantic analysis process. The syntactic parse identifies the lexemes corresponding to words, idioms, or morphemes in the input string, and eliminates those lexemes which do not meet basic syntactic constraints. The `MORPH`, `CAT`, and `SYN-STRUC` zones of the lexicon entry for each lexeme are utilized during the course of the syntactic parsing process; the

SYN-STRUC zone provides the most information about the local syntactic context in which the lexeme appears.

If there are multiple senses of a lexical item in the source text which are not eliminated by categorial or subcategorization constraints in the parse, then each sense is instantiated (thus producing branching arcs from that state in the search). Each instantiation of a lexical item is, essentially, a TMR fragment; combining such fragments produces a (proto-)TMR for the source text.

The local syntactic information in each lexical entry contains reference variables in each argument or head position; the lexical semantic definition references those variables in the appropriate head or argument positions in the lexical semantic definition. Together, these elements form the syntax-semantics interface (without the use of other explicit mapping rules).

The *combination* component (described in more detail in Section 2.4 below), utilizes this syntax-semantics interface to indicate in what capacity to combine two TMR fragments; in many cases, the syntax-semantic interface indicates that the semantic head of one lexical unit is to fill a specific role of the other. In addition, the lexical semantic definition specifies constraints that the filler of the role must meet, or constraints that the head must satisfy in order for a unit to participate in its representation. In some cases (for example, many English prepositions) the lexical semantics of a lexical unit instantiate no head, but just indicate what role one local argument would play in the semantics of another.

The SDS-building control process iterates through each instantiated lexical-semantic unit, walking through the lexical-semantic information, and combining it with other instantiated lexical-semantic units or TMR fragments, according to the information in the syntax-semantic interface. Thus the instantiation process always results in one or more subsequent states in the search space, whereas the combination process can result in the termination of that path (in either a final, successful state, or in a non-final failure state if the combination fails). The combination process could also result in one or more subsequent states, reflecting the one or more ways in which the combination process could succeed (typically with different preferences).

2.3 Knowledge Sources

Before proceeding to discuss the heart of the semantic analysis process, the combination process, we need to briefly overview the knowledge sources that are used in the semantic analysis process. The *lexical semantic specification* found in each entry in the lexicon is the repository of low-level semantic information. The *syntax-semantics interface* (see Section 3.4.2 and Section 4.2) links into that specification, guiding the search process by suggesting what element is a candidate for combination with what other element, and in what relation (i.e., what slot). The syntax-semantics interface as a knowledge source is sketched out above, and is not further detailed here. Most lexical semantic specifications call for the instantiation of one or more concepts from the *ontology*, which is the grounding of the entire semantic representation. There are other knowledge sources that are used in the semantic analysis process, but they are not reviewed here in this overview section.

2.3.1 The Ontology

The concepts in the ontological world model include *objects* (such as airplanes, ideas, or giraffes), *events* (such as buying or eating) and *properties* (such as has-as-part or temperature). The ontology is organized as a tangled taxonomy (an *IS-A* hierarchy) for reasons of storage and access efficiency. Thus, the concept HAMMER may be a child (i.e., a specialization) of the concept of HAND_TOOL, while concepts of BALL_PEEN_HAMMER and CLAW_HAMMER could be children of HAMMER (CLAW_HAMMER *IS-A* HAMMER *IS-A* HAND_TOOL). Ontological entities could also be understood as the perception of Platonic ideals or natural kinds, as represented in a speaker's model of the world. In other words, the HAMMER concept does not refer to a particular hammer, but to a speaker's generic notion of a hammer. Ontological concepts can be *instantiated*, that is, a representation of a specific instance of the concept is produced to signify a particular mention of this concept in a text or discourse (but does not necessarily correlate to an existing real-world entity). Thus, **CONTRACT-132** may refer to the contract counterfactually referred to in the seventh sentence of the text that a semantic analyzer is processing at the moment.

In addition to the organization into a taxonomy via *IS-A* links, the ontology also contains numerous other links between concepts. These additional properties are used as background knowledge for building and disambiguating semantic dependency structures in TMRs. Figure 2B

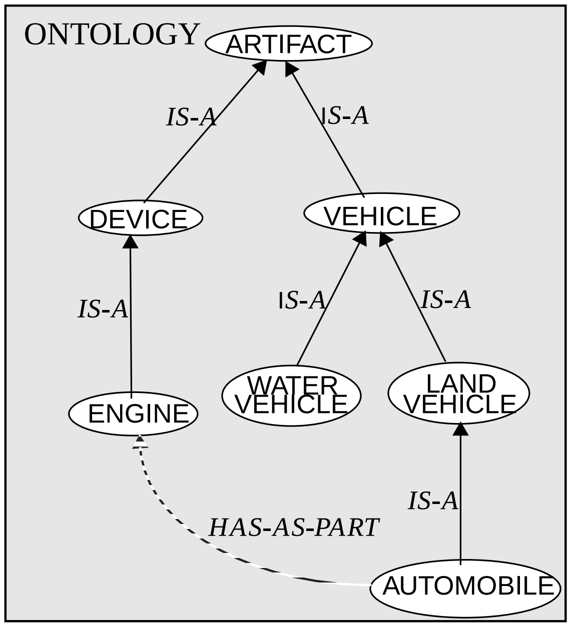


Figure 2B. An ontology fragment

illustrates a fragment of a hypothetical ontology, with mostly taxonomic (*IS-A*) links shown. An ontology that will actually be used in an application will include such properties as, for instance, *IS-PART-OF*, *IS-AN-OCCUPANT-OF*, *MANUFACTURED-BY* as well as semantic dependency relations that have been traditionally referred to as *case roles* in Case Grammar and its many practical applications. In our system, we represent ontological concepts as *frames* (see Section 3.1.1), while properties are represented as *slots* in FRAMEKIT or FRAMEPAC frame languages, described in Nyberg (1988) and Brown (1996), respectively. Graphically, concepts are represented as nodes, and properties as labelled links between nodes. For example, the EAT concept may have case role slots such as *AGENT* and *THEME* (reflecting the eater and what is being eaten), as well as slots that are more general, such as *LOCATION* (probably *inherited* from an ancestor of EAT in the ontology and not directly acquired for the concept EAT). The properties themselves represent concepts from the ontology, and appear in the hierarchy and inherit slots and properties from their ancestors.

All the above properties are, in fact, relations between ontological concepts. Another kind of property in our system is called *attribute* and signifies a link between a concept and a specially defined set of values (numerical, literal or scalar). Properties like *ENGINE_TYPE* or *TEMPERATURE* are attributes.

Properties are defined in frames for particular concepts and, in accordance with the semantics of the representation language, apply to all concepts below them in the hierarchy. Constraints are placed in the definition of a property on domain and its range; these constraints are also concepts from the ontology. When the property appears as a particular slot in the frame for a concept, additional semantic constraints may be locally defined in this frame. These will be more specific than the constraints already specified in the definition of the property. For example, there might be a LOCATION relation defined in the ontology. The domain of this relation might be specified as any EVENT or any PHYSICAL_OBJECT (in other words, events and physical objects may have locations). The range of the relation might be PLACE (that is, only places can be the locations of events or physical objects). The concept of an AIRPLANE_LANDING_EVENT would have a *LOCATION* slot (being, presumably, a descendent of EVENT, this concept is within the domain of the relation). However, it may be useful to further constrain the range of the relation (i.e., the allowed value of the slot) in this particular concept to be, say, LANDING_STRIP, a descendent of PLACE. This further constraint may be overridden in some text occurrences (as in texts about forced or crash landings), and the algorithm discussed in Section 2.4 incorporates a constraint relaxation technique to take care of such situations. In FRAMEKIT and FRAMEPAC, the constraints on the allowed fillers of various slots are maintained in the `SEM` facet of the slot, whereas the fillers themselves are in the `VALUE` facet.

2.3.2 Lexical Semantic Specification

Each lexical entry (i.e., each word sense) includes a lexical semantic specification. The base case of this specification is an indication that the word refers to a concept from the ontology, and in the process of semantic analysis, the word would result in an instantiation of that concept. In many cases that concept has further constraints on the allowable fillers for various slots (in the same manner as the local constraints in the ontology, as discussed above) or specific values filled in for literal (non-relational) slots. Some lexical semantic specifications include multiple concepts to be instantiated in a particular structure (i.e., one instantiation will be specified to be the head, and another as a filler of a particular slot). Other lexical semantic specifications might not invoke the instantiation of a concept, but just provide filler information for another concept (the adjective *blue*, for example) or relate two other concepts to be instantiated by other words (the preposition *in* might just specify that something is to be in the *LOCATION* slot of something else).

Interwoven with these semantic specifications is the syntax-semantics interface component. Particular slots in the specification may have a reference variable as the filler; the variable is bound to a headed syntactic structure during processing, and the instantiated concepts that result from the semantic processing of that syntactic structure are inserted into the indicated slot's value. For example, in the specification for *eat*, the concept `EAT` may be called for, and the `AGENT` slot of that concept may dereference the syntactic subject head; thus the resulting construction after the SDS-building process would result in an instantiated EAT concept, with its *AGENT* slot filled by an instantiation which refers to the eater, for example, **DOG23**.

2.4 Constraint Satisfaction in Combining Semantic Elements

The basic premise of the SDS-building approach is that *constraint satisfaction* controls the combination of any two elements of meaning representation (initially instantiated lexical semantic definitions, which are incrementally combined to form the meaning of the entire utterance). Given the meanings of individual words (possibly augmented instantiations of ontological concepts), the combination operator attempts to combine such structures into the meaning of a phrase. For example, the syntactic specification for a sense of *eat* may subcategorize for an object, and the syntax-semantics interface (i.e., the `$vars`) indicates that the meaning associated with the object serves as the *THEME* of the meaning of *eat*. Thus the combination operator builds a semantic structure by attempting to insert the meaning representation produced by instantiating the syntactic object into the *THEME* role in the meaning representation of *eat*.

In the example illustrated in Figure 2C, the constraining concept for the filler of the *AGENT*

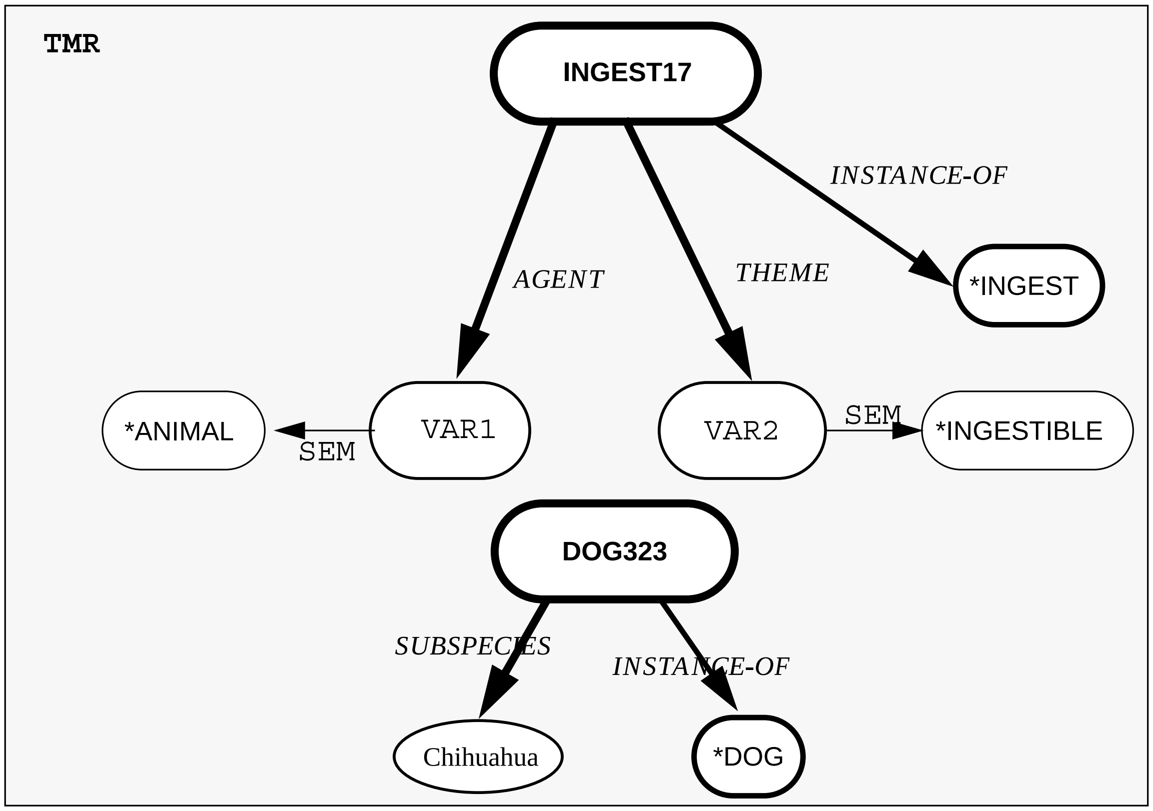


Figure 2C. Illustration of slot constraints and filler types

slot is ANIMAL, and a candidate filler might be an instantiation of DOG (for example, the **DOG323** instantiation in the figure). Similarly, the *THEME* slot is constrained to be filled by a concept instantiation which is an INGESTIBLE. Details of this figure aren't important at this point, and will be revisited in the longer example in Section 2.5 below.

The combination operator can be applied recursively to phrases; it is expected that, after a series of applications, the (proto-)meaning of a complete utterance will be produced (in practice, this process in actually implemented in a bottom-up manner). The combination operator is not typically applied at the supersentential levels of semantic analysis. This SDS processing is, in practice, augmented by additional microtheories which take the proto-TMR and add specific information which is gleaned from non-lexical sources.

Technically, the combination process creates relations between two concepts. These relations are made manifest in the formalism by allowing a slot in one concept to point to another concept as its `VALUE`. A number of constraints guides this linking — a) the constraints on the range (and domain) of the relation in the ontology, b) possible further constraints on the relation's range appearing in the head concept's entry in the ontology, and c) the lexical semantics specifying idiosyncratic constraints on the head concept. For example, the case role *AGENT* has a constrained range (animals or other animate entities may be agents); the concept INGEST constrains the agent to be ANIMAL; the German word *freßen* maps to the concept INGEST and further constrains the *AGENT* to be non-human.

The SDS-building process is also expected to determine which slot will contain a link to the meaning of a dependent element (i.e., the specific relation that holds between the two instantiated meanings) as well as which element is to be the head and which is to be the filler. Three eventualities can be distinguished in this process:

• The syntax-semantics interface explicitly identifies the slot (e.g., the meaning representation of the syntactic subject of *eat* is directed by the content of the lexicon entry for the verb to be inserted into the *AGENT* slot of the head concept representation).

• An explicit syntactic indicator of the filler's role is available, indicating which element is to be the head, which is to be the filler, and what relation holds between them (e.g., the preposition *in* may indicate that the meaning produced by the object of the preposition fills the *LOCATION* slot of the meaning produced by the syntactic head to which the prepositional phrase attaches.)

• When no syntactic clue is available as to the nature of the relation between the two elements (in fact, no indication may be available as to which is the head), the SDS-building process undertakes a search over all candidate slots. The head concept of a meaning representation will have a number of allowable ontological properties. The SDS building process includes attempting the slot-filling constraint-satisfaction process over each of these. (This case occurs, for example, in noun-noun compounding in English.)

As mentioned above, the constraint on a candidate slot filler can be specified in one of three locations. Constraints on slot fillers are defined in terms of ontological concepts. Thus, since the candidate filler is a constrained ontological concept, and the constraint is marked by an ontological concept, too, the constraint satisfaction process can be a matter of verifying that the filler is subsumed by the concept marking the constraint. In other cases, the constraint satisfaction process involves exploring other (non-taxonomic) paths between the candidate filler and the constraint over the ontology. These other paths may define a metaphorical or metonymic relationship between the candidate filler and the constraining concept.

Given this view of semantic composition as a constraint satisfaction problem, and given also that the candidate filler and the constraints are both ontological concepts, thus, representable as nodes in a connected graph, this process can be interpreted as the problem of finding a low-cost path through a graph, well-known in the graph theory literature. Although any of a variety of shortest-path graph search algorithms could be used, we use an A*-style modification of the Dijkstra algorithm with heap-based priority queues. This algorithm gives us the desired expected-case complexity, with worst-case complexity of only $O(|E| log_2 |N|)$, where $|E|$ is the number of edges and $|N|$ is the number of nodes, and $|E| << |N|^2$.

The cost of graph traversal is a function of arc traversal costs, whose relative values are empirically determined. Seeking a low-cost path becomes, then, the control strategy for the process. Conceptually, this process determines an abstract distance between two ontological concepts. The more ontologically related two concepts are, the "cheaper" the path between them. The relation between the two concepts can be vertically taxonomic, or any of a variety of other relations that reflect conceptual relatedness between two concepts (such as a composer and his work, *sword* and *scabbard*, a part and the whole, *to taxi* and *airplane*, *landing strip* and *airplane*).

Arcs in the graph are directional; the cost of traversing the inverse of a link (and all links have inverse links associated with them) is typically different from the cost of traversing the link itself. Graph traversal is typically computed as originating at the candidate filler, and ending at the constraining concept. In the example in Figure 2C above, the ontology must be traversed to find the best path from DOG to ANIMAL (in this case a trivial hierarchical traversal) in order to verify that the candidate **DOG323** may indeed fill the *AGENT* slot of the event. In the trivial case of verifying that the candidate filler is in a subtree headed by the constraint, the graph is treated as a tree (i.e., non-taxonomic links are ignored); the cost of an *IS-A* arc is set to be very low, and the cost of a *SUBLASSES* arc (the inverse of the *IS-A* arc), as well as all other links, is set higher. Thus the constraint satisfaction test is treated trivially. Section 8 below describes how this framework is used to disambiguate among multiple senses of words in an input utterance.

In many cases, however, the simple *IS-A* test will fail, because the base constraints are established for literal meaning, whereas the input contains a meaning shift or unexpected input. Thus, in metonymic or metaphoric text the *IS-A* constraints fail. Then the graph traversal is expanded to include other, appropriately weighted, arcs (relations) in the ontology. The sorts of relations that are used in treating the cases of metonymy and (some) metaphor are identified by those additional relations in the ontology (in fact, they are included in the ontology often with the express purpose of helping to treat metaphors and/or metonymies). For example, in *The White House said yesterday...* the *AGENT* for *say* is a metonym; since the constraint for *AGENT* on the appropriate word sense of *say* is HUMAN, *White House* does not satisfy the trivial hierarchical constraint. Thus the shortest path that the graph search finds includes an *OCCUPANT* arc (inherited by all concepts below RESIDENCE in the ontology); traversing this arc identifies the likely existence of a occupied-for-occupant (or institution-for-member) metonymy. The traversal of this arc has a greater cost than the traversal of vertical hierarchical arcs, thus it wouldn't be preferred unless there were no uni-directional vertical path available. Section 9 below focuses on metonymy resolution using the overall framework described above.

Additionally, a variety of words are subject to regular polysemy (see, e.g., Pustejovsky (1991), Apresjan (1974)), namely systematic and productive formation of secondary senses of words from base forms of certain classes, such as *duck* or *rabbit* as a food (vs. a creature), *door* as an opening (vs. the device that covers the opening), and so on. Instead of attempting to cover each sense of such words (as reflected by lexical semantic definitions incorporating concepts from the ontology), these regular polysemies are reflected (generatively) by appropriate arcs in the ontology.

For any two concepts in the ontology, there will be many possible paths between them, however, typically with different weights. The processing paradigm espoused here postulates that the best path between the two concepts will identify the correct relationship between them (if the weighting mechanism is appropriate and the relative weights are appropriately assigned). The example in Figure 2D illustrates how two paths over the ontology can have different weights. In the

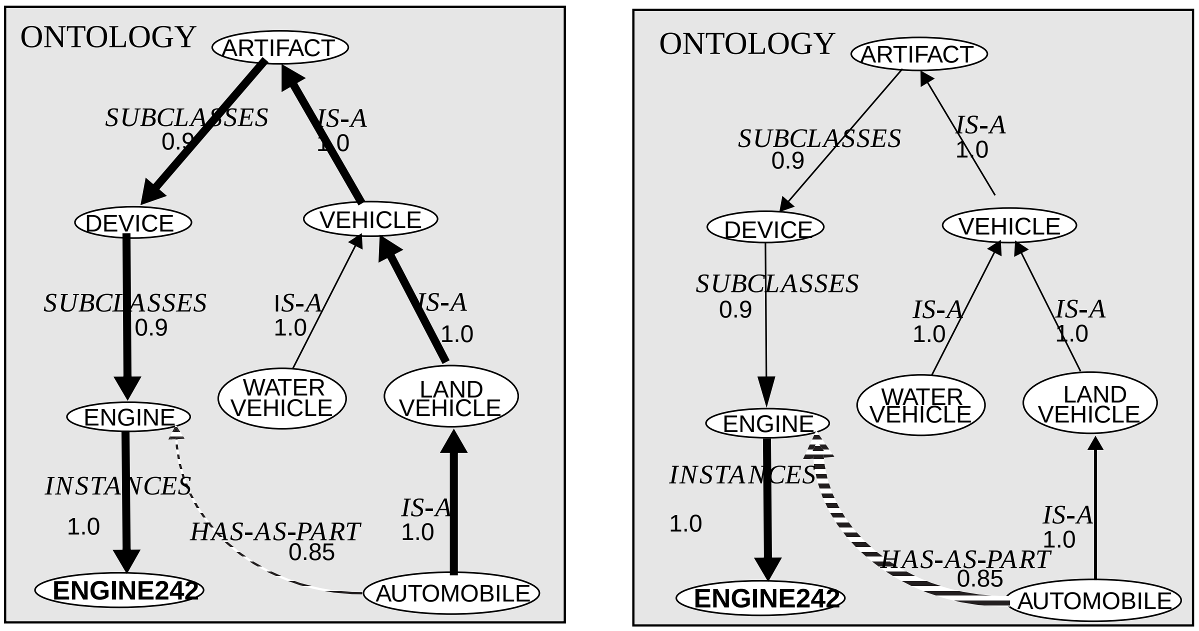


Figure 2D. Two views of example of Ontology, with paths identified with bold arrows

sentence *Fred drove his dual-cam V8 down Main street*, the phrase referring to the engine is used metonymically for the vehicle; in the semantic representation for *drive*, the constraint on what could be driven would specify the VEHICLE concept from the ontology. The two paths illustrated in this figure show how different weights on individual arcs lead to differing path weights (namely, *1.0 * 1.0 * 1.0 * 0.9 * 0.9 * 1.0 = 0.81* for the left view, and *0.85 * 1.0 = 0.85* for the right view). If the arc weights are set appropriately, the shortest path from the filler to the constraint will reflect the metonymy, by traversing the arc capturing the part-for-whole relation embodied in the metonymic expression. It is clear from this example that the success of this approach is dependent on the richness of the ontology (not just in terms of concepts, but in terms of links as well) and on appropriate determination of weights.

In cases where the syntax-semantics interface provides no clue as to the appropriate slot (the third case above) as is the case in noun-noun compounding, the search originates at one of the two noun's resulting concept, and attempts to traverse from there to the other noun's head concept. Either locally or at an ancestor of the originating concept, all the relations that the concept may participate in are available to the search. In other words, an attempt is made to link the two concepts over any relation that they, or their ancestors, can participate in.

The weights that are in the transition table are critical to the success of the heuristic. The cost assessed for traversing a metonymic (or other) arc may be dependent on the previous arcs traversed in a candidate path, because some arc types should not be repeatedly traversed, while other arcs should not be traversed if certain other arcs have already been seen. We use a state transition table to assess the appropriate cost for traversing an arc (based on the current path state) and to assign the next state for each candidate path being considered. Our weight assignment transition table has 40+ states, and has individual treatment for 40 types of arcs; the other arcs (of the 350 total arc types) are treated by a default arc cost mechanism.

The arc weights are learned by an automatic training method. After building a training set of inputs (candidate fillers and constraints) and desired outputs (the “correct” paths over the ontology, i.e., the preferred relation), a *simulated annealing* numerical optimization method identifies the set of arc costs that results in the optimal set of solutions for the training data. A similar approach is used to optimize the arc costs so that the cheapest cost reflects the preferred word sense from a set of candidates.

2.5 An Example

Before launching into a more detailed discussion (in subsequent chapters) of the processing paradigm, the lexicon, or its support knowledge sources and reference formalisms (i.e., the ontology, the syntactic f-structure, and the TMR), a simple illustration is in order. As our research concentrates on semantics, we do not emphasize the syntactic information in the examples. Suffice it to say that we assume a syntactic parse as a tree structure with heads projecting constituents (this will be significant in the syntax-semantics interface).

The lexemes from the example sentence *The chihuahua ate the apple* are presented in abbreviated form below (this example ignores tense, aspect, determiners, etc.) Figure 2E presents a

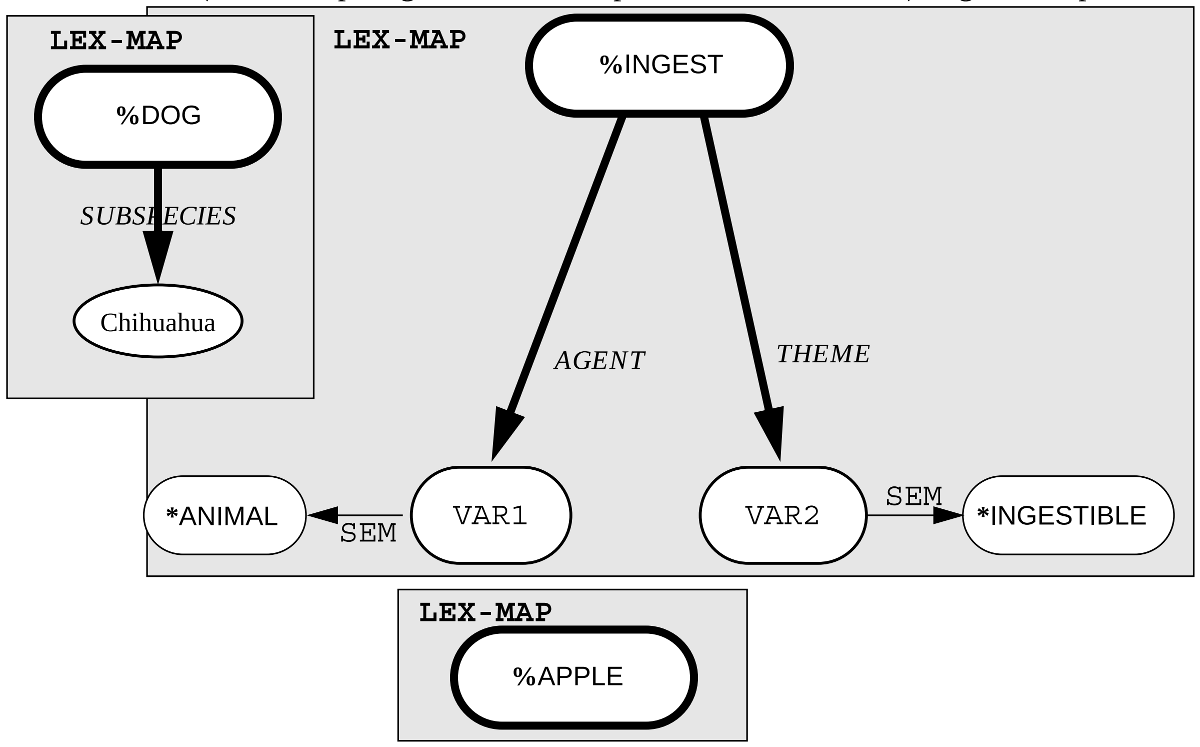


Figure 2E. Graphical representation of lexical semantics for *chihuahua*, *eat*, and *apple*.

graphical view of the lexical-semantic representation for the nouns and the verb.[1] The simplest is

1. The heavy vertical links represent slots (case roles) on a concept, while lighter horizontal links represent constraints on the expected/possible fillers of those slots. Note that there are three distinct structures, one for each of the three words.

for *apple*: the semantics zone of this lexicon entry (for simplicity, we ignore polysemy for the time being and refer to the basic "fruit" sense of *apple*) simply indicates that there is a concept in the ontology equivalent to the meaning of this lexeme. The **%** marks an ontological concept which is to be instantiated as part of the SDS-building process.

The representation of the other noun is somewhat different. It so happens that the ontology used to support our sample dictionary does not have a concept for chihuahuas (the question of the grain-size trade-off in designing ontologies and lexicons is far from settled; for further discussion see Section 4.1). Thus, our lexicon entry for *chihuahua* contains in its **SEM-STRUC** zone a request to instantiate a DOG concept, but with further (lexicon-stipulated) specification that the dog is of the subspecies called Chihuahua.

The representation for *eat* has different complexity, as it is an argument-taking lexical unit whose semantic description must include information about building a semantic dependency structure comprising the meaning of the unit itself and the meanings of its arguments. This structure-building operation, with a concomitant disambiguation process, is supported by listing semantic constraints on the meanings of arguments of the argument-taking lexical unit. In this case, the INGEST concept has (at least) two slots, named *AGENT* and *THEME*. The semantic constraints on those slots are represented in "facets" of those slots, specifically SEM facets, represented by the lighter arrows. The semantic constraints are themselves concepts from the ontology; any concept (or instantiation of a concept) which falls below the constraint in the ontology tree satisfies the constraint.

During semantic analysis, all the lexical semantic specifications are instantiated, as illustrated in Figure 2F. A uniquely numbered instance of each relevant concept is created. Each instantiated concept has in its frame representation the slot *INSTANCE-OF* whose filler indicates from what concept this instance was produced. The instantiations are combined in well-defined ways to produce the initial TMR for the text, as illustrated in Figure 2G. In the system-internal representation, each instantiation remains an independent structure, with pointers (in the form of the structure name) as the filler of the relevant slot:

```
(DOG323
      (INSTANCE-OF *DOG)
      (SUBSPECIES "CHIHUAHUA")))

(APPLE23
      (INSTANCE-OF *APPLE))

(INGEST17
      (INSTANCE-OF *INGEST)
      (AGENT        (VALUE DOG323)
                    (SEM *ANIMAL))
      (THEME        (VALUE APPLE23)
                    (SEM *INGESTIBLE)))
```

If more properties of a concept were known, for example if we knew that its color was white, there would be another slot in the **DOG323** structure called *COLOR* with a value of WHITE. The graph notation represents pointers as direct links to the node (instance structure).[1]

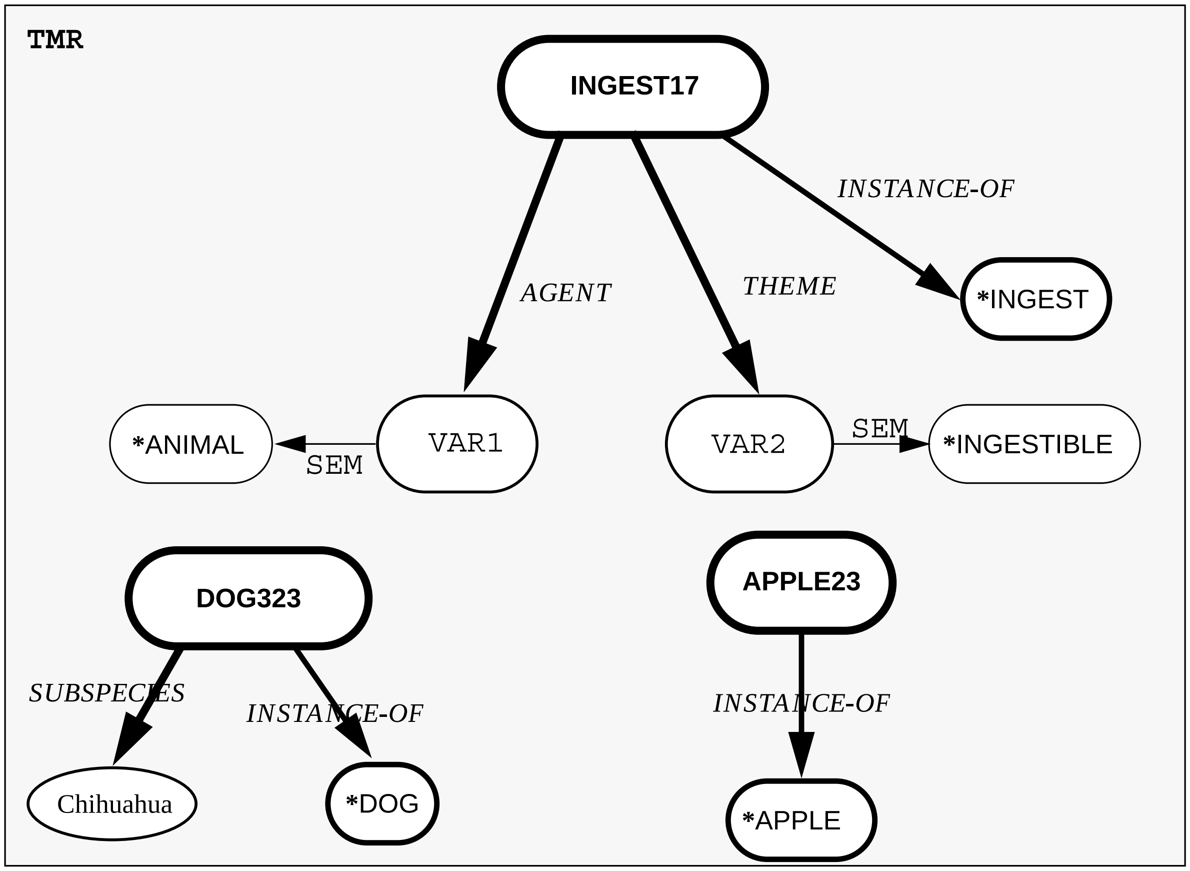


Figure 2F. Graphical representation of instantiated lexemes (3 discrete structures)

A structure may participate in multiple other structures, which would be illustrated in the graph by having multiple arrows pointing to a node. For example, if *chihuahua* were modified by the adjective *horrible*, a structure (of type ATTITUDE) would be added to the graph which would point to **DOG323** in the same fashion as the pointer from the **INGEST17** concept instance. The details of the TMR notation or the illustrative graph are not salient for our current purpose which is to illustrate how semantic patterns found in lexicon entries are instantiated combined in order to produce the initial TMR.

1. This leads to a bit of confusion between facets of slots vs. slots of concepts (e.g., the SEM facet in the graph). Also note that some liberties were taken with the VALUE facet in the text structure; in the graph, the VALUE facet is represented by the label on the nodes.

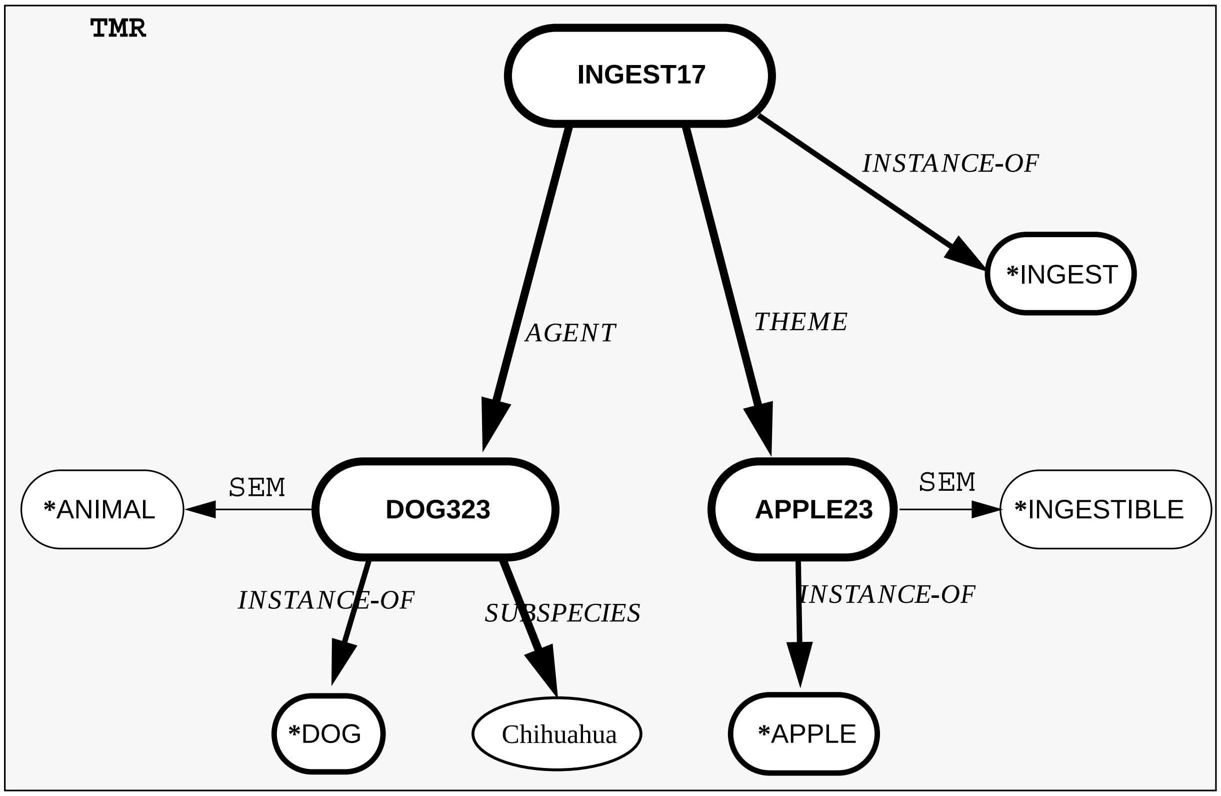


Figure 2G. Graphical representation of initial TMR (one network of structures

3. Background: Knowledge Structures[1]

This section identifies the primary knowledge sources, representation formalisms, and preliminary processing necessary for the knowledge-based semantic analysis approach treated in the following sections.

The term *knowledge source* is used to refer to a static data source, representing some regularized machine-tractable body of knowledge or data, coupled with routines for accessing and manipulating this knowledge (see Nirenburg *et al*. (1992) for further discussion of knowledge sources). The separation of static knowledge from the algorithm of a system is a well-known software engineering principle needing no further motivation here. The primary knowledge sources used in the approach to semantic analysis presented in this paper are the ontology (outlined in Section 3.1) and the lexicon (discussed in Section 3.2); the interaction between these two knowledge sources takes place in the lexical semantic specification within the lexicon, and is discussed in detail in Section 4. The representation of each lexical unit's meaning is a building block of the sentential semantic dependency structure (SDS), that is, the core of the representation of the meaning of the sentence.

As mentioned above, the goal of semantic analysis is to capture the meaning of input text in an unambiguous machine-tractable representation; Section 3.3 introduces the formalism used here for this representation language, called TMR, in which that unambiguous machine tractable representation of meaning is rendered.

Syntactic parsing, whether performed prior to or concurrently with semantic analysis, needs to be surveyed (in Section 3.4) before proceeding to the exposition of the semantic analysis approach, for the reason that syntactic parsing functions as a dynamic knowledge source, producing knowledge (i.e., parse trees) used by heuristics in the semantic analysis below.

3.1 Ontological Knowledge

The formalism for lexical semantic definitions of lexemes (as described in Section 4), and, by construction, the meaning representation of a text (as rendered in the TMR meaning representation language defined in Section 3.3) both need to have a semantics for the respective languages. In order for a semantic specification to have explanatory power, the atoms or primitives of the meaning representation language must be interpreted in terms of an independently motivated model of the world as perceived by a speaker. Our approach to semantics shares this tenet with logical semantic theories (e.g., Kamp's DRT, in Kamp (1981)). However, we differ from the logical semanticists in that we believe that for any realistic experiments to be performed with an NLP system using the algorithms and formalisms suggested by a semantic theory, this world model must be actually built, not just defined algebraically. The issue of grounding symbols which form the primitives of such a language has been widely debated in AI, linguistics, philosophy of language, and cognitive science (e.g., McDermott (1978)). Although we do not address this problem directly here, it is relevant to mention that a different paradigm of KBMT attempts to ground the representation of the semantics of a language in the language itself, by using numbered word senses as primitives in meaning representation, thus equating the description language and the language being described, as in Farwell *et al.* (1993). In some cases, this is augmented with a

1. Much of this section has appeared in Onyshkevych and Nirenburg (1991), Onyshkevych and Nirenburg (1994), Meyer *et al.* (1990), or Nirenburg *et al.* (1990)

small number of special predicates (e.g., Jackendoff (1983, 1990)). The resulting semantic descriptions are language-dependent, which necessitates extra work in building multilingual application. Nirenburg and Levin (1992) and Levin and Nirenburg (1994b) call this approach to semantics *syntax-driven*, while the semantics advocated in this paradigm is called *ontology-driven.*

The term *ontology* is used here to denote a body of knowledge about the world. Our ontologies (see Carlson and Nirenburg (1990), Skuce and Monarch (1990), or Monarch (1989) for an earlier exposition) are structured as directed graphs, or, more specifically, tangled trees. The knowledge in the world model is separated into two (interconnected) knowledge bases. The first knowledge base, referred to as the *ontology* proper, contains knowledge about *concepts*, or natural kinds or Platonic ideals. The second knowledge base, which we call the *onomasticon*, is a collection of specific instantiations of ontological concepts "remembered" by the system. Thus, the concept "U.S. President" will be found in the ontology, while the knowledge that the system may have about Harry Truman will be found in the onomasticon.

Section 10 and Section 3.1.3 below discuss some issues regarding the mechanism for acquisition of the ontology, but it should be noted here that in encoding ontological knowledge, we are acquiring a practical resource, not defining a merely theoretical construct. Nor are we attempting to merely discover a natural construct or entity (see Mahesh and Nirenburg (1995)). In fact, the feasibility of building the ontological resource is validated by an existence proof, in that the ontology that has been acquired for the Mikrokosmos project exists and satisfies pragmatic needs of the overall effort, as stated; currently the ontology consists of almost 5000 concepts. The top level distinguishes between OBJECT, EVENT, and PROPERTY, with a maximum depth of 15 concepts.

Both the ontological knowledge described in the second and third parts of this section, and the TMR representation language (described in Section 3.3) use a frame-based representation language, which is described first.

3.1.1 The Frame-based Formalisms

FRAMEKIT and FRAMEPAC are implementations of a generic frame-based knowledge representation language (in LISP and C++, respectively). Knowledge bases in these frame formalisms take the form of a collection of *frames*. A frame is a named set of *slots*. A slot is a named set of *facets*. A facet is a named set of *fillers*[1]. A filler can be any symbol or expression (such as a string or list). This structure defines the basic set of constraints and features of FRAMEKIT and FRAMEPAC. Although these formalisms specify some extensions to this basic expressive power, such as inheritance, they are still quite general and semantically underspecified. The actual interpretation and typing of the basic entities is generally relegated to the particular application, for example, the ontology or the TMR representation language. The constraint languages of the ontology and of TMR are built on top of the frame representation. These languages' approach of semantic and functional underspecification is different from many other knowledge representation languages and environments, in which the basic representation language is made much more expressive at the expense of its relative awkwardness, difficulty in learning, and some format-related constraints on application development. Details of FRAMEKIT and FRAMEPAC may be obtained from Nyberg (1988) and Brown (1996), respectively.

1. There is actually another layer between the facet and the filler (the *view*), which is ignored here as an unnecessary complication for present purposes.

A typical example of a frame with specified slots, facets, and fillers might be:

```
(MACINTOSH
        (IS-A                  (VALUE  PERSONAL-COMPUTER))
        (PRODUCED-BY           (VALUE  APPLE))
        (SUBCLASSES            (SEM  PLUS CLASSIC POWERBOOK165
                               POWER_PC6001 II  ...))
        (HAS-AS-PART           (SEM  DISK-DRIVE SYSTEM-UNIT
                               MONITOR CPU MEMORY HARD-DRIVE  ...)))
```

In the above example, MACINTOSH is a frame name, *IS-A* is a slot name, VALUE and SEM are examples of facet names, and APPLE and MEMORY are fillers (although they, themselves, may be names of other slots), for example. The semantics of the structure needs to be defined by the application. Once a frame is defined within FRAMEKIT or FRAMEPAC, it may be *instantiated*. The semantics of an instantiation will vary with the model intended by the knowledge base, but, in general, an instantiation is meant to refer to a specific instance of the concept represented by the frame. For example, an instantiation of the MACINTOSH frame from the above example might be used to model or refer to a specific Mac in the real world. So, for example, the instance **MACINTOSH23** might be produced by the instantiation process (instantiations are usually identified by frame name followed by a unique number). An instantiation may be given specific information, i.e., specific fillers for given slots. In the case of TMR representations, each instantiation refers to an individual entity in the discourse, which might or might not refer to a real-world entity.

Other slot values inherit from PERSONAL-COMPUTER (e.g., *MAXIMUM-NUMBER-OF-USERS*), or from COMPUTER (which might be the parent of PERSONAL-COMPUTER), INDEPENDENT-DEVICE, DEVICE, ARTIFACT, etc. Just as slots may be inherited through *IS-A* links to parents, slots are inherited by an instantiated frame from its frame "parent". So if **MACINTOSH23** were asked to provide the *PRODUCED-BY* slot in the VALUE facet, the response would be APPLE. This inherited information may be overridden; a specific filler for any slot and facet may be stored in an instantiation (which is itself a frame), and that filler would be returned in a query for that information; typically, information is inherited only if no local specific information is available, for a given slot and facet, but this parameter is configurable.

3.1.2 Slot Specification at the Facet level

The full set of facets used in the lexicon representation consists of:

- VALUE - a specific value (e.g., number of sides for a triangle = 3, sex of a man = male). This is the facet where actual information is represented; typically, the other facets are constraints on what may be a legal (or likely) filler of the VALUE facet. Typically, in the ontology, this facet is not specified. This facet is used for recording a) constrained mappings within lexical semantic specification, or b) semantic dependency structure links. In our lexical semantic specification language, fillers of this facet are often symbols consisting of "^" appended to a variable name, e.g., (%visit (*AGENT* (VALUE ^$var1))...) The caret is an operator (akin to an intension operator) which dereferences the variable (retrieves the lexeme to which the variable gets bound during the syntactic parsing process within the f-structure) and then retrieves the concepts which are instantiated by that lexeme's lexical semantic specification. So any place where a ^$var# appears is an indication to the semantic dependency-building algorithm of how to attempt to build the sentential semantic dependency structure (see Section 7). In simple terms, `^$var1` means "the meaning of the syntactic unit referenced by `$var1`."

- `DEFAULT` - typical, expected value (e.g., color of diapers = white). If a `VALUE` is needed by some inference process operating on a TMR representation, and the `VALUE` is unspecified, the `DEFAULT` is used; this usage is consistent with standard Artificial Intelligence and logic default mechanisms.
- `SEM` - akin to a traditional selectional restriction (e.g., the agent of a cognitive event has to be a HUMAN). This is essentially a constraint on what the `VALUE` may be. Instead of using some small set of binary features, we allow any concept (or boolean combination of concepts) from the ontology to be a semantic constraint; any `VALUE` then needs to be a descendent of one of the concepts listed in `SEM`. All slots have `SEM` facets in the ontology, but often these need to be modified (typically, constrained further) for a specific lexeme. This semantic restriction is not absolute; it may be relaxed or violated in specific ways, as in cases of metonymy or metaphor.
- `RELAXABLE-TO` - maximum relaxability, if any, of `SEM` restrictions; used in cases of selectional restriction violation processing (treatment of unexpected input, including metonymy and metaphor).
- `SALIENCE` - a scalar value in the range *[0.0, 1.0]* designating the significance of a specific attribute slot or role (partly reflecting the notion of "defining properties" vs. "incidental properties").

Table 3A identifies the availability and use of the five facets in entries in the ontology, TMR frames that are instantiations of ontological concepts, and **`LEX-MAP`** specifications of additional constraints to be added to instantiated concepts; specifics of these knowledge sources can be found in Section 3.1.3, Section 3.3, and Section 4, respectively. Note that the full inventory of facets is utilized in the lexical semantic specification, but only reduced sets are available in the other two knowledge sources. Table 3B defines the filler types that are used in the former table. Notice that the `VALUE` facet has some restricted occurrences in each of the four data types, namely that `VALUE` can only be filled for a concept in the ontology for a slot which is one of the hierarchical/ instantiation indicators: *IS-A*, *INSTANCE-OF*, *SUBCLASSES*.

3.1.3 The Ontology

In formal semantics, one of the most widely accepted methodologies is that of model-theoretic semantics in which syntactically correct utterances in a language are given semantic interpretation in terms of truth values with respect to a certain model (in Montague semantics, a "possible world") of reality. Such models are in practice never constructed in detail but rather delineated through a typically underspecifying set of constraints. We are committed to actually building such a model for any large-scale experimentation. Lexical semantic definitions of units in the lexicon (and, by construction, sentential semantics as represented in TMR) are linked to this detailed world model by the lexical semantic specification language defined in Section 4. Elements of this world model are densely interconnected through a large set of well-defined ontological links which enable the world modeler to build descriptions of complex objects and processes in a compositional fashion, without excessive proliferation of concepts.

Table 3A. Head and Facet Use in Ontology, TMR, and Lexical Semantics

			ONTOLOGY	ONOMASTICON	LEXICAL SEMANTICS	TMR
HEAD		TYPES	•concept	•instance	•inst. request	•instance
		EXAMPLES	*contract	**$person23**	**%contract**	**%person23**
FACET	VALUE	TYPES	•(concept, only in *IS-A* or *SUB-CLASSES* slots)	•instance •number: any •symbol •gen. instance •(concept, only in *INSTANCE-OF* slot)	•inst. request •gen. inst. req. •name •number: any •symbol •variable •(concept only in *IS-A* slot)	•instance •number: any •symbol •gen. instance •(concept, only in *INSTANCE-OF* slot
		EXAMPLES	*object	**%war23** 4 &blue **%%bottle2** *human	**%contract** **%%engine** $John 7 &blue $VAR1 *object	**%war23** 4 &blue **%%bottle2** *human
	SEM	TYPES	•concept •symbol-set •number range	—	•concept •symbol-set •number range	—
		EXAMPLES	*object {&red, &blue} (<> 1 8)	—	*object {&red, &blue} (<> 1 8)	—
	DEFAULT	TYPES	•symbol •concept •number: any	•symbol •concept •number: any	•symbol •concept •number: any	•symbol •concept •number: any
		EXAMPLES	&blue *engine 4	&blue *engine 4	&blue *engine 4	&blue *engine 4
	RELAX-TO	TYPES	•concept	—	•concept	—
		EXAMPLES	*object	—	*object	—
	SALIENCE	TYPES	•number: [0,1]	•number: [0,1]	•number: [0,1]	•number: [0,1]
		EXAMPLES	.5	.5	.5	.5

Although the ontology bears superficial similarity to semantic networks, such as those de-

Table 3B. Filler Types

Filler Type	Description of Type
concept	any concept (entry) from the ontology
instance	any instantiation of a concept from the ontology, to include entries from the onomasticon
generic instance	an instantiation which doesn't refer to a particular entity or element in the world or discourse model, but to the generic
instantiation request	a request to instantiate the named concept (only used in the lexical semantics)
generic instance request	an indication that the SDS-building process is not to produce a specific instantiation (which refers to a specific entity in the world or discourse model), but a *generic instance* of a concept.
number: any	any number in any range (integer or float)
number: [0, 1]	any float between 0.0 and 1.0, inclusive
number range	a specification of the allowable numerical values, represented as a range (using unary operators >, >=, <, or <=), the binary operator <>, or a set of allowable numbers, such as {2,3,4}.
name	a string; only to be used for names of named entities
symbol	a literal value taken from an exactly specified set of allowable literal fillers
symbol set	a finite enumerated set of symbols, any of which can be used as the filler
variable	a variable from the syntax/semantic interface, in the form $VAR#, where # is replaced by a natural number
—	an indication that this facet is not properly used in the specified data structure

scribed in Quillian (1968), Woods (1975), or Fahlman (1982), there are some differences that make the comparison not quite exact. Most semantic networks used for NLP contain a mix of types of information, including word tokens, word senses, syntactic category information, concepts, propositions, intensions, etc., while our ontology is restricted to concepts as nodes, with a limited type of proposition represented by relational slots (or by the links to instantiations). We differentiate ontological knowledge from knowledge about instances or tokens, which is represented in our onomasticon or TMR (described below). Woods (1975) insists that semantic networks represent intensional objects and propositions without regard to their truth conditions. We represent no truth conditional information at all in the ontology; that sort of information, in addition to attitudes and proposition-level constructs reside in our TMR representations, not in the ontology *per se*.

The granularity (therefore the detail and the extent) of the ontology is an open issue, with the extremes being concepts reflecting the meaning of each and every word in the lexicon, on the one hand, and just enough primitive concepts to allow construction of any desired meaning, on the other. This issue of granularity is discussed further in the context of the linkage of lexical meaning to the ontology (Section 4.1).

In the ontology, frames are used to represent *concepts*, which are the basic building blocks of the ontology. Examples of concepts might be *house*, *automobile*, *voluntary-olfactory-event*, and *specific-gravity*. Most concepts have *properties*, which are represented as frame slots on the concept frame; the properties are also well-defined concepts within the ontology. Some concept properties actually do not relate to the domain model, just capture administrative and explanatory material for the human users (see examples below). Fillers of slots (i.e., values of properties of the concept) are constrained to be names of atomic elements of the ontology, expressions referring to elements of the ontology with additional constraints, properties, or modifications, collections of atomic or modified concepts, or special-purpose symbols and strings. The top-level distinction in the ontology is between EVENT (which captures actions and processes), ENTITY, and PROPERTY (which includes ATTRIBUTE and RELATION, thus captures all states, all relational information between events or things, as well as features). Note that the ontology does not make a distinction between individuals and sets, leaving that to a meta-ontological mechanism that is orthogonal to the ontology, unlike the approach taken in Dahlgren *et al.* (1989), who make a top-level distinction between individuals and collections, which results in two mirror sub-hierarchies.

The example concepts below serve to illustrate some aspects of the constraint language used.

```
ONTOLOGY:
      (AUTOMOBILE
                     (IS-A          (VALUE LAND-VEHICLE))
                     (SUBCLASSES    (VALUE RACING-CAR PASSENGER-CAR))
                     (HAS-AS-PART   (SEM ENGINE TRANSMISSION...)))
```

The above concept is for a generic automobile. The *IS-A* slot indicates that the parent concept is LAND-VEHICLE, which means that an AUTOMOBILE inherits all properties (i.e., slots) of its parent. LAND-VEHICLE, in turn, would have *IS-A* VEHICLE, indicating that it inherits all of VEHICLE's properties; the inheritance is recursive, so AUTOMOBILE also inherits VEHICLE properties. The *SUBCLASSES* slot lists those concepts from the ontology that AUTOMOBILE is the parent of. The *HAS-AS-PART* slot list some basic components of an automobile; note that the list is in the SEM facet of the slot where semantic constraints on possible fillers of the VALUE facet are specified. Table 3A above itemizes the availability of facets in the ontology; note that the VALUE facet is typically not used in the ontology at all, except for the slots that place the concept in the ontology: the *IS-A* slot (identifying the parent) and the *SUBCLASSES* slot (identifying the children).

This concept may be *instantiated*, meaning that a particular instance of an AUTOMOBILE has been mentioned (implicitly or explicitly) in the discourse being modeled, and that instance is being associated with that instantiation. Instantiations are identified, for example, as **AUTOMOBILE23**, that is, typically the name of the concept from which the instantiation is being made, followed by a unique identifier to distinguish that particular instantiation from all others. A list of all instantiations is maintained in the instantiating concept. As an example of instantiation, the AUTOMOBILE concept and a particular instantiation for Lynn's Saab might appear as:

```
ONTOLOGY:
    (AUTOMOBILE
            (IS-A              (VALUE  LAND-VEHICLE))
            (SUBCLASSES        (VALUE  RACING-CAR  PASSENGER-CAR))
            (INSTANCES         (VALUE  AUTOMOBILE23))
            ...)
INSTANCE:
    (AUTOMOBILE23
            (INSTANCE-OF       (VALUE  AUTOMOBILE))
            (MANUFACTURER      (VALUE  SAAB))
            (COLOR             (VALUE  &MAROON))
            ...)
```

What this indicates is that the particular automobile being modeled by this instantiation is maroon and is manufactured by Saab (more properly, the VALUE in *MANUFACTURER* might be **CORPORATION 44** or **SAAB**, that is, a pointer to the instantiation of a CORPORATION concept for Saab-Scania, either derived from the onomasticon (see below) or built from current information). These two slots are not defined locally in the AUTOMOBILE concept, but inherited. *MANUFACTURER* might be inherited from ARTIFACT (and defined as *MANUFACTURER* (SEM HUMAN ORGANIZATION), indicating that whatever fills that slot needs to have as an ancestor HUMAN or ORGANIZATION, as is the case for the instantiation for Saab). The *COLOR* property (slot) might be inherited from PHYSICAL-OBJECT, which might have a list of literals indicating possible colors in the SEM facet of the *COLOR* slot; alternately, a more sophisticated model of color may be used, which might include concepts for various colors, or color-wheel/hue indicators, etc.

In addition to the SEM and VALUE facets, the ontology uses the DEFAULT facet, where defeasible knowledge about the expected fillers of a slot may be represented. For example, for the concept *AUTOMOBILE, the slot HAS-AS-PARTS could have a DEFAULT facet filled in the ontology to represent general knowledge about expected auto components, such as *ENGINE, *WHEELS, *TRANSMISSION, and so on. This list of default fillers could be augmented or overridden in a lexical semantic specification that calls for an instantiation of this concept. Thus an instantiation of this concept may have specific VALUEs in the *HAS-AS-PARTS* slot such as **ENGINE423**. The SALIENCE facet can also be specified on a slot in an ontology entry, in order to highlight a slot as being particularly significant.

Another concept might be for the notion of smelling (in the voluntary verbal sense):

```
ONTOLOGY:
    (VOLUNTARY-OLFACTORY-EVENT
            (IS-A         (VALUE  VOLUNTARY-PERCEPTUAL-EVENT))
            (AGENT        (SEM  MAMMAL BIRD REPTILE AMPHIBIAN))
            (INSTRUMENT   (SEM  OLFACTORY-ORGAN)))
```

Additional slots such as *EXPERIENCER* and *BENEFICIARY* would be inherited from among the ancestors of this concept, e.g., VOLUNTARY-PERCEPTUAL-EVENT or EVENT. Note that concepts also exist for all properties (i.e., slot names) and non-literal fillers (therefore, the ontology for this world model include concepts for AGENT, INSTRUMENT, VOLUNTARY-PERCEPTUAL-EVENT, MAMMAL, BIRD, OLFACTORY-ORGAN, EXPERIENCER, EVENT,...)

As stated above, since ontological concepts are (or at least are supposed to be) language-independent, their names should be seen only as labels (which happen to be expressed in English for our mnemonic purposes, although random symbols would serve just as well) and are not intended to be interpreted in terms of the semantics of English. In naming our ontological concepts, we use the following conventions to reduce naming ambiguity:

- Whenever possible, we use scientific, rather than lay terms.
- We try to be consistent in the names of ontological concepts going down a subtree: EVENT has subclasses MENTAL-EVENT, PHYSICAL-EVENT, and SOCIAL-EVENT.
- Whenever possible, we attempt to include in the name an indication of some distinguishing characteristic of the ontological concept (i.e., a characteristic distinguishing the concept from its sister-concepts). For example, VOLUNTARY-VISUAL-EVENT and INVOLUNTARY-VISUAL-EVENT indicate events that involve vision, with voluntary or involuntary participation, perhaps corresponding to the English words *look* and *see*, respectively.
- Consistently throughout the ontology, we use English words in one sense only as names of ontological concepts. We therefore provide definitions for all concepts, so that when the name of a concept (e.g. STORE) corresponds to a polysemous word, the intended meaning will be clear.

In addition to greater organization and understandability, the chief advantage of the taxonomic organization of the ontology is in the use of inheritance in specifying the attributes of concepts in the ontology without extreme redundancy (see, for example, Touretzky (1986) for discussion of such inheritance mechanisms). Although we make no claims as to the cognitive or psychological plausibility of the specific model or its organization, we do find that some sort of model is, in fact, necessary in order to apply our semantic analysis paradigm, as is described in subsequent chapters.

3.1.4 The Onomasticon

In addition to the ontology, which essentially stores information about Platonic ideals, natural kinds, or concepts, it is also useful to have a knowledge base of particular instantiations of those concepts. The extent of such a knowledge base is debatable; a guiding principle in determining the contents of such a knowledge base may be to include such instantiations which the speaker of an utterance (or the writer of a text) would reasonably expect the hearer (reader) to know. This expectation would naturally vary from context to context, domain to domain (although some core subset may be common to most contexts.)

We are introducing the term *onomasticon* to refer to a knowledge base of instantiated concepts, typically with names, generally statically (but possibly dynamically) acquired.[1] Instances produced earlier in a discourse don’t enter into the onomasticon, but remain in a context or discourse store.

Often the instantiations of this knowledge base will have names, and may be referred to as *named instances*. Typical named instances include instances of COUNTRY, CITY, HUMAN, for example, for Japan, Paris, and Abraham Lincoln, respectively. These names would necessarily be in

1. The Oxford English Dictionary defines onomasticon as “a vocabulary or alphabetic list of proper nouns, esp. of persons. Formerly used more widely of a vocabulary of names, or even of a general lexicon”

a particular language; however, these names are for the convenience of the knowledge base acquisition and maintenance process, just as the symbols attached to concepts in the ontology are just mnemonics for convenience of use. These named instances could be referenced in whatever language as appropriate. Thus such a knowledge base would have an entry for Germany, and this entry might be called GERMANY if it were convenient for the knowledge base maintainer to use English names; however, the lexicon for any language would reference that instantiation and give it that language's name, such as *Allemagne* in a French lexicon.

In addition to instantiations of entities, it may also be useful to encode, in a knowledge base, instantiations of events or other concepts from the ontology. The Battle of Gettysburg may be such an event that could be useful for some domains, and hence may be included in the static knowledge base for a particular application or domain.

Knowledge bases of instantiations of concepts may be either static or dynamic. Instantiations of countries and cities, for example, would fall into a static knowledge base, because this type of information would be obtained from gazetteers or from similar references. Instantiations may also be of a more dynamic nature, along the lines of what used to be called *instance memory* or *episodic memory* in cognitive science, for example, in Tulving (1985). Concepts which were instantiated by a system using the approach described here would be retained and archived if they appeared particularly useful. This dynamic acquisition of instances would typically be performed while transferring to a new domain.

Any of the onomasticon entries could be referenced from within the lexicon of any language. Just like a specific concept from the ontology would be referenced (see Section 4), a lexicon entry would reference an instantiation instead. Thus for a given language there would be lexicon entries for Japan, Paris, and John F. Kennedy, pointing to the appropriate instantiated concept from within the onomasticon, and with the appropriate name for that language forming the lexeme.

In addition to the onomasticon as a static knowledge source, the identification of any additional named instances (typically people, places, organizations, or products) can be accomplished by shallow Information Extraction techniques, namely *named-entity taggers*, such as those described in Sundheim (1995). As a recovery mechanism when a name is missing from the onomasticon and language lexicons, such named-entity taggers can provide a semantic type estimate for aid in disambiguation of the rest of the sentence, and in the MT context a transliteration mechanism would complete error recovery for unknown names.

3.2 The Lexicon

The lexicon for a given language is a collection of *superentries* (see Meyer and Steele (1990)) which are indexed by the citation form of the word (represented in the **`ORTH-FORM`** field). Within a superentry, individual lexemes are represented in the frame-based formalism described in Section 3.1.1. A superentry includes all the lexemes which have the same dictionary form, regardless of syntactic category, pronunciation, or sense. Thus, a given superentry might include any number of lexemes that are nouns, verbs, adjectives, etc.

Lexemes ("entries") inside a superentry are given names which are formatted using the character "+", followed by the citation form (the lemma), followed by "-" and an indication of the syntactic category (e.g., **v**, **n**, **adj**) of the entry and its sense number (for practical reasons, for languages that use non-Roman alphabets, the citation form is encoded to be renderable in Roman alpha-numeric characters; this encoding is invisible to the lexicographers). For example, **+eat-v2** introduces the entry for the second verbal sense of *eat*.

Proper names in the lexicon reference an entry in an onomasticon (the list of named instances, see Section 3.1.4) in their lexical-semantic description (see below), but otherwise are similar to other lexicon entries. For example, **+Paris-n1** might be the label for the lexical item *Paris* which names the city Paris, France. This arrangement allows language-independent world knowledge to be maintained independently of language-specific nomenclature (which, in turn, affects its phonology, morphology, syntactic behavior, etc.)

Each lexicon entry is comprised of a number of *zones* corresponding to the various levels of lexical information. The zones are **`CAT`** (syntactic category), **`ORTH`** (orthography -- abbreviations and variants), **`PHON`** (phonology), **`MORPH`** (morphological irregular forms, class or paradigm, and stem variants or "principle parts"), **`APPL`** (dialect or other sublanguage indicators), **`SYN-STRUC`** (indication of sentence- or phrase-level syntactic dependency, centrally including subcategorization), **`SEM-STRUC`** (lexical semantics, meaning representation), **`LEXICAL-RELATIONS`** (collocations, etc.), **`LEXICAL-RULES`** (listing of true positive and false positive lexical rules that appear to apply to the lexeme), and **`PRAGM`** (information related to pragmatics as well as stylistic factors). A special **`ANNOTATIONS`** zone contains ancillary information for the user or lexicographer, in addition to administrative information, such as modification audit trail, example sentences, printed dictionary definitions, cross-references (what other lexemes is this one referenced by), etc. Below is a fuller specification of the structure of the lexicon entry, starting with the superentry (such as **bark** which is then broken down into categories and senses, such as **+bark-v1**) and further specifying the zones and fields in each zone, in a BNF-like notation (bold text is used to identify the short forms of the zone/field names as used in the discussion):

```
<superentry> :=
     ORTHOGRAPHIC-FORM:  "form"
     ({syn-cat}: <lexeme> * ) *

<lexeme> :=
     CATEGORY: {syn-cat}
     ORTHOGRAPHY:
               VARIANTS: "variants"*
               ABBREVIATIONS: "abbs"*
     PHONOLOGY: "phonology"*
     MORPHOLOGY:
               IRREGULAR-FORMS: ("form"
                                 {irreg-form-name})*
               PARADIGM: {paradigm-name}
               STEM-VARIANTS: ("form" {variant-name})*
     ANNOTATIONS:
               DEFINITION: "definition in NL" *
               EXAMPLES: "example"*
               COMMENTS: "lexicographer comment"*
               TIME-STAMP: date-of-entry lexicog-id
               DATE-LAST-MODIFIED: date lexicog-id
               CROSS-REFERENCES: lexeme *
     APPLICABILITY:
```

```
                LOCALITY: "locale"*
                FIELD: "field"*
                LANGUAGE: "language"*
                CURRENCY: "era"*
SYNTACTIC-STRUCTURE:
                SYNTACTIC-STRUCTURE-CLASS: class
                SYNTACTIC-STRUCTURE-LOCAL: fs-pattern
SEMANTIC-STRUCTURE:
                SEMANTICS-CLASS: class
                LEXICAL-MAPPING: lex-sem-specification
                MEANING-PROCEDURE: meaning-specialist
LEXICAL-RELATIONS:
                PARADIGMATIC-RELS: ({p-r-type} lexeme)*
                SYNTAGMATIC-RELS: ({s-r-type}
                                f-struct | lexeme)*
LEXICAL-RULES:
                LR-CLASS: class *
                LR-LOCAL: (LR# (lexeme | OK | NO)) *
PRAGMATICS:
                STYLISTICS: ({FORMALITY, SIMPLICITY,
                            COLOR, FORCE, DIRECTNESS,
                            RESPECT} value) *
                ANALYSIS-TRIGGERS: trigger *
                GENERATION-TRIGGERS: trigger *
```

The **CAT**, **ORTH**, **MORPH**, and **SYN-STRUC** zones are used primarily during syntactic parsing stage (which in our paradigm also includes segmentation, tokenization, and morphological analysis). This stage precedes semantic analysis in the simplest implementation, but may be interleaved with semantic processing in future experiments. The **SYN-STRUC** zone, discussed in further detail in Section 3.4.2, specifies local syntactic context for the lexeme for use in syntactic parsing, but also plays a crucial role in establishing bindings in the syntax-semantics interface. The **APPL** zone provides information in analysis that may be used in preferring one word sense over another (depending on the expected or identified sublanguage), and in generation in selecting from among synonyms.

The **LEX-REL** zone, currently still in preliminary development, is intended to provide reference to primarily collocational information; each collocation is categorized, e.g., using Mel'chuk-style categories (Mel'chuk and Zholkovsky (1984), Apresjan *et al.* (1969)), and represented in a partially-specified f-structure (of the same style as the **SYN-STRUC** specification). Since collocations are compositional in meaning (have transparent "decoding" despite the idiosyncratic "encoding"), particularly when the word senses are identified, there is typically no need to represent the semantics or pragmatics of the collocations further; if there is need, the relation is instead represented by direct reference to another lexeme. The **PAR-RELS** slot of the **LEX-REL** zone is used to represent such relations as synonymy, antonymy, or hyponymy, but primarily only as an indexing convenience; these relations are primarily reflected by the relative ontological positions of the concepts used to define each lexeme.

As our primary current research interests center on issues in semantics (in particular, lexical semantics) the **SEM-STRUC** zone attracts central attention. Through this zone the lexicon connects with the ontology and the onomasticon, thus becoming the locus of the atomic links between lexical units in texts and the language-neutral text meaning representation, or TMR. The formalism for the lexical semantic specification in the **SEM-STRUC** zone (specifically, the **LEX-MAP** field) is discussed in detail in Section 4, while the utilization of that specification is discussed in subsequent sections. The **SEM-CLASS** field actually doesn't specify a specific lexical semantic representation for the lexeme, but presents a template for the semantic side of the syntax/semantics interface. For example, a class might be `EVENT-AGENT-THEME`, where an event would link the *AGENT* to `^$VAR1` and *THEME* to `^$VAR2`; then the **LEX-MAP** for the verb could just specify (**%ingest**), for example, and rely on the class to fill in the syntax-semantics interface. Details of lexical-semantic specification are found in Section 4.

Since the meaning of some lexemes is not representable by the instantiation of a **LEX-MAP** template, alternative mechanisms are provided: the **MEAN-PROC** in the **SEM-STRUC** zone, and the **TRIGGER** slots in the **PRAGM** zone. These mechanisms allow for the invocation of functions or procedures that modulate or modify the meaning representation of an utterance, in a manner somewhat akin to Word Expert Parser specialists in Small and Rieger (1982), but intended to be used as a last resort, typically on closed-class words, not as a mechanism to be used for most open-class words. The **MEAN-PROC** allows for functions that modulate the meaning representation, as in the case of the adverbial *very*, where the function intensifies the value of a scalar attribute towards one or the other extreme. The trigger mechanisms in the **PRAGM** zone allow for the invocation of procedures or microtheories that have a particular mission; the **A-TRIG** for *the*, for example, invokes a definite reference resolution mechanism during semantic analysis.

Specific details of the lexicon format used are available in Meyer *et al.* (1990); no philosophical or general adequacy claims are made about the format.

Among the central issues in current computational lexicography are "packing" information in the lexicons and facilitating acquisition. Two (connected) approaches have been followed. First, attempts have been made to cross-index information in the entries, for instance, by building lexicons as hierarchical structures, with a variety of features inherited from parents to children. Second, certain word senses might not be overtly listed in the lexicon as separate entries, instead, instructions are supplied of how to create such entries when an application program requires them. For various pragmatic reasons, some of these senses are generated not on an as-needed basis but at acquisition time (thus nullifying the space savings in favor of improved lexicon quality by reducing overgeneration).

3.2.1 Cross-Indexing and Inheritance

A variety of cross-indexing mechanisms can be used to minimize redundancy within the lexicon. Inheritance is one type of cross-indexing used, for example, to indicate that a particular verb is of syntactic class *basic-bitransitive*, thus avoiding the need for a syntactic specification or syntactic features to be specified locally in the corresponding entry: the information will be inherited from the specification in the definition of the class. Inheritance is used explicitly in our lexicon in the **MORPH** (paradigm), **SYN-STRUC**, and **LEX-RULES** zones. "Horizontal" cross-reference can be used to indicate that, say, the third and fourth verb sense of *eat* share the same **SYN-STRUC**, zone or that all verbal senses of *eat* share the same **PHON** and **MORPH**; this is accomplished by simple reference pointers in the underlying data structures.

3.2.2 Lexical Rules

In the interests of efficiency in knowledge acquisition and to capture generalizations about productive lexical alternations and derivations, lexical rules (LRs) are used to generate lexical entries dynamically from lexical entries encoded statically in the lexicon. In our model LRs can be used to cover a broad spectrum of phenomena, including syntactic alternations such as passivization and dative, regular non-metonymic and non-metaphoric meaning alternations (such as those described in Apresjan (1974) and Pustejovsky (1991)), as well as some productive derivational processes such as formation of deverbal nominals or deverbal adjectives. Thus the LRs in this paradigm include the phenomena covered by LRs in LFG in Bresnan (1982), but also many of the Lexical Inference Rules (LIRs) from Ostler and Atkins (1992), which necessarily include semantic shifts. When LRs are added to the lexicographer's arsenal, the lexicon as a whole becomes a list of (super)entries plus a list of LRs. The discussion below highlights some *a priori* restrictions on the scope and content of LRs.

LRs in our model apply to only one lexical entry at a time and, thus, do not cover phenomena which involve two or more senses (such as compounding in German and other languages). Also not covered (by the mechanism of LRs) is the treatment of metonymy, seen procedurally as the situation when there is a violation of selectional restrictions for an entire set of senses of two or more lexical units that have to be combined in a single semantic dependency structure. We delegate the treatment of metonymy and similar phenomena to the processing component of the application system: a dynamic knowledge source such as a semantic interpreter or a lexical selector in generation; however, the knowledge required for these processes is in fact encoded into the ontology's network of relations or links, as well as in weights for those links (see Section 5 for a description of ontological graph search – our central mechanism for carrying out semantic analysis, and Section 9 for a discussion of metonymy processing in our model).[1] A useful rule of thumb for deciding whether a phenomenon should be treated in a static knowledge source (e.g., through LRs) or in a dynamic knowledge source (a processor module) is whether the phenomenon is language-specific (go with LRs) or language-neutral (treat it using processing-related rules). Thus the scope of our LRs differs slightly from Ostler and Atkins' LIRs, where they capture both language-dependent and language-independent derivations in the LIRs, while we focus our LRs on language-dependent derivations; they do exclude alternations strictly based on pragmatics or world knowledge, as we do.

LRs consist of a left-hand side (LHS) which constrains the lexical entries to which the rule can apply and a right-hand side (RHS) which stipulates how the new lexical entry will differ from the original. Lexical entries which are produced by a LR are themselves eligible to match the LHS of an LR. Both sides of the LR can reference any zone of the lexical entry; typically the RHS modifies the local syntactic information and the lexical semantic specification (or at least the syntax-semantic interface). Often, however, the syntactic category, syntactic features, and orthography are affected as well (in derivational cases).

All of the lexicon zones are available to the LRs, both to the LHS for constraining the application of the rules, as well as to the RHS for modification as part of the alternation or derivation that the LR reflects. The syntactic category (i.e., the **`CAT`** zone) is often modified in derivational rules,

1. Some simple metonymies might be handled by LRs (with equivalent results) in a more economical manner than the semantic analysis processing provides, and thus may be "cached" by encoding them into LRs; this will be addressed by further experimentation.

e.g., in those LRs which produce nominal or adjectival forms from verbs. The syntactic features and syntactic structure (in the **SYN-STRUC**) of a lexeme would be affected in most LRs. Particularly in derivational LRs, the word form itself changes, thus the **ORTH**ography, **PHON**ology, and **MORPH**ology of the lexeme would change. The lexical semantic representation (**LEX-MAP**) can be used to constrain the application of rules in the LHS: in the passive rule (see above), an *AGENT* is required in the lexical semantic specification on the LHS. Regular alternations such as those described by Beth Levin and others, for example in Levin (1989, 1991), would be constrained to apply to only those lexemes with a particular concept (or descendant of that concept) as semantic head (e.g., LOAD-EVENT) in the **LEX-MAP**. Some LRs cause the semantic representation itself to change. In other cases, however, there is no actual semantic reflection of derivational LRs, because, for example, deverbal nouns and adjectives (such as *abuse* and *abusive*) are typically represented in the identical fashion as the base forms in this paradigm (perhaps only with a different syntax-semantics interface; see Raskin and Nirenburg (1995)). Ostler and Atkins require changes in the semantics in the RHS of their LIRs, but the nature of the semantic representation used in this paradigm (tending to diverge substantially in predicate/argument structure from that of the surface syntax) results in no semantic change for some derivations.

The simplest mechanism of rule triggering is to include in each lexicon entry an explicit list of applicable rules. LR application can be chained, so that the rule chains must be expanded, either statically, in the specification, or dynamically, at application time. This approach avoids any inappropriate application of the rules (overgeneration), though at the expense of tedious work at lexicon acquisition time. The other approach is to maintain a bank of LRs, and rely on the left-hand sides to constrain the application of the rules to only the appropriate cases; in practice, however, it is difficult to set up the constraints in such a way as to avoid over- and undergeneration. For example, it is difficult to constrain the LHS to select exactly the set of verbs to which *-able* derivation applies. As another example, to prevent the passivization of idioms such as *kick the bucket* (but allow it on *spill the beans*) it is necessary to set up constraints to block application in inappropriate cases; in this case, requiring both an *AGENT* and a *THEME* (or *BENEFICIARY*, etc.) in the lexical semantics appropriately constrains the passive rule and prevents overgeneration. Related mechanisms for restricting the application of LRs to avoid overgeneration, such as blocking and preemption, have reduced effectiveness in practical situations where the lexicon is incomplete or is under construction (because the form that is supposed to block or preempt may not have been entered yet).

The reliance on rule application at run-time (vs. listing in the lexicon) does not allow explicit ordering of word senses, a practice preferred by many lexicographers to indicate relative frequency or salience; this sort of information can be captured by other mechanisms (e.g., using frequency-of-occurrence statistics). This approach does, however, capture the paradigmatic generalization that is represented by the rule, and simplifies lexical acquisition.

The approach adopted in Mikrokosmos (although still under development) is a hybrid approach, as a compromise of linguistic generality, processing considerations, and acquisition considerations (the full space of related issues is addressed in Viegas (1996)). The LRs are written with the LHS attempting to constrain the forms to which the rule applies as tightly as possible (to include preemption in addition to constraints in **SYN-STRUC**s, **SEM-STRUC**s, etc.) At lexicon acquisition time, all applicable LRs are applied to the base form, producing full lexical entries for the derived forms. For any rules that successfully apply, the acquisition tool checks for the existence of the resulting orthographic form in a corpus; this only helps in the cases where a new dictionary form is created, and does not apply in the cases of meaning-only or subcategorization shifts. Any new senses with orthographic forms that do not appear in the corpus are summarily rejected; all the remaining senses are presented to the lexicographer for verification and/or augmentation. Additionally, the occurrences in the corpus are retained by the lexicographer as examples (in the **`ANNOTATIONS`** field). The lexicographer needs to review the corpus examples and determine the distribution of senses for that form, as is usual in lexicography. The base form maintains an inventory of LRs which apply (in the **`LR-LOCAL`** facet), along with either an indication of that the result was rejected by the lexicographer (‘`NO`’), or the lexeme name of the resulting form; for some LRs which are productive and fairly regular (such as passivization), instead of storing all the derived lexical entries, the LR fields in the base forms merely indicate ‘`OK`’, and the lexical entries are produced at run time on an as-needed basis. In syntactic parsing of texts (normal processing), for any word form in the text which is not found in the lexicon, all the LRs can be attempted to try to generate the novel form, as a recovery procedure; some rules (such as passivization) are explicitly invoked by the syntactic parsing processes. As the lexicon grows, more of the highly productive LRs will be restrained from application at acquisition time, and only applied at run time, for purposes of storage economy.

3.3 Text Meaning Representation Language

The goal of semantic analysis in the current model is to take a natural language utterance (or a set of natural language utterances) and to capture the meaning of that utterance in a machine-tractable formalism. TMR is the meaning representation language in which output is represented in our paradigm. A TMR expression captures the explicit and some of the implicit information of a natural language utterance using a well-defined language-independent formalism. In addition to the basic semantic content of the utterance, TMR attempts to capture the pragmatics of the utterance, including focus, textual relation, speaker attitudes, and stylistic factors.

A TMR expression is a network of case frames of various types, potentially including instantiated ontological concepts (often with additional slots in the case frame filled), entries from the onomasticon, a frame representing information about the context of the speech act or utterance, frames representing relations among other frames in the network, and frames representing attitudes that the speaker may have conveyed about elements of the content of the utterance. The TMR expression does not, however, capture linguistic knowledge in the form of syntax or any kind of lexical information such as word senses, subcategorization, etc. (although all of these sources might be used in the process of building the TMR); in general, there is no information in the TMR that directly identifies the input string, syntactic dependency structure, lexical elements, or even the source language. In this way, the TMR differs substantially from Sowa (1993), whose Conceptual Graphs blur linguistic language-related information with the underlying meaning or content; despite Sowa’s improperly generalized objections about the expressiveness of frame-based approaches, the TMR can handle more types of represented meaning and interrelationships than the CG formalism, partly because the augmentation of the underlying case-frame with the additional mechanisms discussed in the subsections below extends the expressiveness to higher orders, unlike the first-order nature of CG. In contrast, the frameworks that rely on a fundamentally first-order-logic-based approach to representing meaning, such as Dahlgren *et al.* (1989) or Charniak and Goldman (1988), would have an unmanageably complex set of interrelated propositions and axioms if they attempted to represent the depth of knowledge expressible in a TMR (even though the two systems might be formally equivalent), assuming they could resolve any remaining difficulties in soundness and completeness of the denotational semantics that would inevitably be encountered in switching to the necessary higher-order logic and in formalizing their logical system.

The term TMR refers to both the formal representation language specified below and specific representations of the meaning of texts using that language. In the discussions below, TMR will be used to refer to either the former or the latter, and context will clarify which is being referred to.

3.3.1 TMR Definition

The notation below defines the syntax of TMRs in a BNF-like notation. The notation `::=` is used to define the structure of frames; the notation `-->` identifies a rewrite rule or expansion. Explanations of many of the structures in the BNF below are found in subsequent sections.

```
<TMR>           ::=
     propositions:       <proposition> +
     speech-acts:        <speech-act> +
     stylistics:         <stylistic-factors> *
     relations:          (<text-relation> | <coreference>
                         | <temporal-relation>
                         | <quantifier-relation>
                         | <domain-relation> ) *

<proposition>   ::=
     head:               (<concept-instance> | <attitude> |
     <set>)
     aspect:             <aspect>
     time:               <time>*
     modality:           <modality>*
     attitude:           <attitude>*

<concept-instance>       ::=
     instance-of:        <<concept>>
                ; the frame is actually usually named by a gensym
                ; of the concept name
     [<<property-name>> :(<<concept>> | <concept-instance>
                          <<value>> | <set>)* ]*

                ;case roles and physical properties are among the most
                ;typical properties of concepts; all of those we
                ;expect to have been defined in the ontology

<<concept>>     -->      ONTOSUBTREE-OR(all)

                ;ONTOSUBTREE-OR is a function which returns a DISJUNCTIVE
                ;SET of all the elements in the ontological network
                ;rooted at its argument(s)
                ;Note that this function is not part of TMR, only of our
                ; description of it.
```

```
<<property-name>>-->     ONTOSUBTREE(property)

                    ;ONTOSUBTREE returns a single terminal element of the
                    ;subtree specified by its argument; in this case, returns
                    ;tree of all possible properties


<aspect>        ::=
     aspect-scope:        <<scope>>
     phase:               begin | continue | end
     duration:            momentary | prolonged
     telic:               <<boolean>>
     iteration:           <numerical-value> | multiple

                    ; the number of iterations can be explicitly stated (e.g.
                    ; "twice") or just known to be multiple (e.g. "John hopped
                    ; around on one foot").


<time>          ::=
     at:                  <<time-expression>>
     start:               <<time-expression>>
     end:                 <<time-expression>>
     duration:            <<numerical-value>> {unit}

<<time-expression>> -->  (< | > | >= | <= | ) YYMMDD

<attitude>      ::=
     attitude-type:       <<attitude-type>>
     attitude-value:      [0.0, 1.0]
     attitude-scope:      <<scope>>
     attributed-to:       <<attributed-to>>
     attitude-time:       <time>

<attitude-type>::=        evaluative | saliency

                    ;the number of attitude types may change

<<scope>>       -->       any TMR expression or set of such

<<attributed-to>> -->    ONTOSUBTREE-OR(intelligent-agent)

                    ;any instance of the ontological type intelligent-agent

<modality> ::=
     modality-type:       <<modality-type>>
     modality-value:      [0.0,1.0]
     modality-scope:      <<scope>>
```

```
<<modality-type>>:-->    epistemic | deontic | volitive
                         | potential

<speech-act>   ::=
     speech-act-type:    <<speech-act-type>>
     speech-act-scope:   <<scope>> ;;usually a proposition
     speaker:            <<speaker-hearer>>
     hearer:             <<speaker-hearer>>
     time:               <time>

<<speech-act-type>>:-->  statement | question | ...

<<speaker-hearer>> -->   *speaker* | *hearer*
                         | ONTOSUBTREE-OR(intelligent-agent)

<stylistic-factors>::=
     [ <<style-factor>>: [0.0,1.0] ]*

<<style-factor>>-->      formality | politeness | respect
                         | force | simplicity | color |
                         directness

<<value>>      -->       <<numerical-value>> | <<literal-value>>

<<numerical-value>>      -->       any numerical expression

<<literal-value>>        -->       "string"

<set>          ::=
     member-type:        <<concept>> | <concept-instance>
     cardinality:        (< | > | >= | <= | ) <<numerical-expr>>
     elements:           <concept-instance> *
     complete:           <<boolean>>
     excluding:          <<concept>> | <concept-instance>
     subset-of           <set>
     multiple:           <<boolean>>
     indeterminate:      <<boolean>>
     proper:             <<boolean>>

<<boolean>>    -->       true | false

<coreference>  ::= <concept-instance> <concept-instance>+

<time-relation>::=
     type:               after | during
     arg1:               <time>
     arg2:               <time>
     value:              [0.0, 1.0]
```

```
<text-relation>::=
     type:                <<text-relation-type>>
     arg1:                <<text-relation-argument>>
     arg2:                <<text-relation-argument>>

               ;text relations are non-ontological relations which
               ; reflect relevant text structure (eventually to include
               ; discourse relations)


<<text-relation-type>>-->particular | reformulation
                         | progression | conclusion

<<text-relation-argument>> --> <proposition> +

<domain-relation>::=
     ;; actually, in implementation the type is usually prepended to the object name
     type:                <<domain-relation-type>>
     arg1:                <<domain-relation-argument>>
     arg2:                <<domain-relation-argument>>

<<domain-relation-argument>>--> (<concept-instance>
                         | <attitude> | <domain-relation> )*

<<domain-relation-type>> --> ONTOSUBTREE(domain-relation)
```

This BNF specification glosses over the frame facet level of representation — a detailed discussion of the uses of the frame facets is found in Section 3.1.2 and summarized in Table 3A there. In short, all TMR slots have the `VALUE` facet filled in (the right-hand-side of the slot/filler pairs in the BNF actually refer to the `VALUE` facet of the slot). TMRs carry `DEFAULT` facets as specified in the ontology and/or lexical semantics, to be available for use if the VALUE facet is not filled in. Although the `SEM` and `RELAXABLE-TO` facets aren't meaningful in TMRs proper, sometimes they are carried along from the semantic analysis process just for human debugging/ readability purposes.

3.3.2 Propositional Content

Basic semantic content or meaning of an utterance (sometimes called the propositional content or) is represented in a TMR representation as a network of instantiated concepts from the ontology (or imported instances from the onomasticon), combined and constrained in various ways. The semantic analysis processes (see Section 7) crucially rely on lexical-semantic information (defined in Section 4) from the appropriate lexicon entries. To obtain the semantic content of complex structures with dependencies, information about the argument structure of lexical units (also stored in the lexicon) is used. This information relates not only to ambiguity resolution in "regular" compositional semantics (i.e., straightforward monotonic dependency structure building), but also the identification of idioms, treatment of metonymy and metaphor, and resolution of reference ambiguities.

In general, each instantiated concept in a TMR reflects an individual (e.g., thing or event) in the world or in the speakers' discourse model, whether the instantiation is produced from the ontology or retrieved from the onomasticon; however, when considering the TMRs for entire texts, this characterization must be amended to refer to each *mention* of individuals. Thus when a particular individual is referred to in various portions of the text, multiple instantiations reflect the multiple mentions; an explicit set of coreference structures track the relationship among those instantiations. The motivation for this approach (vs. referring to the same instantiation throughout the text) reflects the fact that as a text progresses, new attitudes or properties may become known about the individual; but in generation, it is appropriate to make these new properties known not all at once, at the first mention, but at the appropriate point in the text (i.e., mirroring the source). However, the coreferences do make the cumulative information available, if necessary, for lexical selection or morphology (e.g., gender) in generation.

As mentioned above, the basic semantic content (the propositional content) is obtained by instantiating relevant ontological concepts, based on the mappings between word senses and ontological concepts indicated in the corresponding lexicon entries. In order to carry out the mappings of complex structures with dependencies, information about the argument structure of lexical units (also stored in the lexicon) is used. This argument structure information is an integral part of the lexical semantic representation (as defined in Section 4), and is dependent on the syntax-semantics interface integrated into the lexical-syntactic information (see Section 3.4.2).

It is well known, however, that the intent of a text is typically not capturable by representing propositional content alone (in our case, just instantiating ontological concepts); what is additionally needed is the representation of pragmatic and discourse-related aspects of language, that is, speech acts, deictic references, speaker attitudes and intentions, relations among text units, the prior context, the physical context, etc. As most of the knowledge underlying realizations of these phenomena is not society-general, universal, or constant but is rather dependent on a particular cognitive agent (a particular speaker/hearer) in a particular speech situation and context, the pragmatic and discourse knowledge units are not included in the ontology (which is supposed to reflect, with some variability, a relatively static model of the world). The representation of this “meta-ontological” information is thus added to the representation of meaning proper to yield a representation of text meaning.

The structures for many of the nonpropositional (therefore non-ontological) components of text meaning are also derived from lexicon entries, where appropriate patterns are stored (see example lexicon entries below). Some of the most important non-ontological components of the TMR representation formalism are reviewed below (for more detailed discussion see Nirenburg and Defrise (1991)), specifically speaker attitudes, stylistic features, and rhetorical relations.

3.3.3 Attitudes and Modalities

A critical aspect of capturing the intent of a speaker in a meaning representation is rendering the *attitudes* that the speaker holds toward the objects or situations which are represented in the propositional (ontology-based) component of text meaning representation. The speaker may also convey attitudes about the speech act which in which the utterance was produced, about elements of the speech context, or even about other attitudes. Similarly, the speaker may convey events, certain relations (and sometimes other constructions) in a particular *modality.*

These attitudes and modalities are conveyed in TMR by a quintuple (either an `ATTITUDE` or a `MODALITY`) consisting of a `type`, a `value` in the interval *[0, 1]*, an `attributed-to` slot (identifying the person who holds the attitude, typically the speaker), a `scope` (identifying the entity towards which the attitude is expressed or the event etc. for which the modality is expressed), and a `time` (representing the absolute time at which the attitude was held). As is the case with all TMR constructs, attitudes and modalities may be either lexically triggered (i.e., explicitly specified in the **`LEX-MAP`** of a lexeme) or triggered by other (non-lexical) phenomena, such as syntax or morphology.

The following attitudes and modalities are among those used in TMR (for present purposes, the distinction between attitudes and modalities isn't relevant):

- `Epistemic`, ranging from *speaker does not believe that X* to *speaker believes that X*.
- `Evaluative`, ranging from *worst for the speaker* to *best for the speaker.*
- `Deontic`, ranging from *speaker believes that the possessor of the attitude must do X* to *speaker believes that the possessor of the attitude does not have to do X.*
- `Potential`, ranging from *the possessor of the attitude believes that X is not possible* to *the possessor of the attitude expects that X is possible*
- `Volitive`, ranging from *the possessor of the attitude does not desire that X* to *the possessor of the attitude desires that X*.
- `Salient`, ranging from *unimportant* to *very important*. This varies with the importance the user attaches to a text component, thus has some overlap with the notion of focus.

3.3.4 Stylistics

Even when the stylistic overtones or nuances of a lexical entry do not contribute directly to the propositional semantics of a text, they can still convey some element of meaning, whether it be in conveying attitudes, setting a mood, or using rhetorical devices such as irony. Thus we identify that the stylistics of a lexeme needs to be encoded in a lexicon entry, in addition to the lexical semantic information. In encoding lexicons for languages with rich social deictics, such as Japanese, the issue of stylistics becomes even more acute.

The TMR representation includes a set of style indicators which is a modification of the set of *pragmatic goals* from Hovy (1988). This set consists of six stylistic indicators: *formality, simplicity, color, force, directness,* and *respect*. In order to obtain this resulting TMR stylistic representation (essentially a set of overall values for the entire utterance, or multiple sets scoping over substrings of the utterance) it is necessary to label various lexical entries (including idioms, collocations, conventional utterances, etc.) with appropriate representations of values for these stylistic indicators. Values for these factors are represented as in the interval *[0,1]*, where *0.0* is low, *1.0* is high, and *0.5* represents a default, neutral value. In the semantic analysis process, the values are available for assisting in disambiguation (relying on expected values for the factors and utilizing the heuristic that typically the stylistics will be consistent across words in an utterance). The resulting values in the TMR representation help guide generation, etc.

Some examples of English lexical entries that might include style features are:

upside:	`formality - low`
	`color - high`

delicious:	`formality - somewhat high`
	`color - high`

great:	`formality - low`
one (pronominal sense):	`formality - high` `force - low`

3.3.5 Relations

Relational structures are used in TMRs to capture the relationships and connections between structures in the TMR, between real-world entities, elements of text, etc. Relations which represent real-world connections are defined in the ontology, whereas other relations (such as rhetorical relations, e.g., conjunction and contrast) refer to properties of text itself (and are reflected in the TMR) and are, therefore, not a part of the world model.

Formally, a TMR relation is a triple consisting of a *relation-type*, a set of *arguments* (either a set of TMR expressions, or the set *{first-i, second-i, third-i}*, *third-i* being an optional member), and a *value* (which is optional, but when present is a numerical value taken from the interval *[0.0, 1.0]*). The following types of relations are used in TMR (the difference in type face signals whether the relation is metaontological or ontological, as distinguished above):

- `TEXTUAL` Relations: such as rhetorical relations, e.g., conjunction and contrast, referring to properties of text itself
  - •`Particular` Text Relations. Relation connecting two elements of text where one is an example of the other
  - •`Reformulation` Text Relation. Relation connecting two elements of text similar in meaning, but expressed in different ways
  - •`Conclusion` Text Relation. Relation where a textual element marks the end of a segment of discourse
- `TEMPORAL` Relations: expressing a partial ordering between `TIME` structures, which are associated with `PROPOSITION`s, thus establishing (partial) temporal ordering of events.
- `COREFERENCE` Relations: identify that two instantiations in fact refer to the same real-world entity (although, possibly, at different time intervals)
- `QUANTIFIER` Relations: which are used for comparison of numerical quantities
- DOMAIN Relations: represent real-world connections (and, therefore, are instantiations of ontological relations) between objects or events.
  - •CAUSAL: Relations of dependence among events, states, and objects; can be either Volitional (the relation between a deliberate, intentional action of an intelligent agent, and its consequence) or Non-volitional (the relation between a non-intentional action or a state of an intelligent agent and its consequence. Subtypes: REASON, ENABLEMENT, PURPOSE, CONDITION
  - •CONJUNCTION Domain Relations. Relations among adjacent elements that are components of a larger textual element. Subtypes: ADDITION, ENUMERATION, CONTRAST, CONCESSION, COMPARISON
  - •PARTICULAR/REPRESENTATIVE Domain Relations. Relations which identify that one element is an example, or a special case, of the other element. Subtypes: PARTICULAR, REPRESENTATIVE
  - •ALTERNATION Domain Relations. Relations that are used in situations of choice, parallel to the logical connector "OR." Subtypes: INCLUSIVE-OR, EXCLUSIVE-OR

- ••TEMPORAL Domain Relations. Identify when one event (or object instance/snapshot) happened relative to another. Subtypes: AT, AFTER, DURING
- ••SPATIAL Domain Relations. Identify relations between objects and/or events in space. Subtypes: IN-FRONT-OF, ABOVE, ON, LEFT-OF, for example (exact inventory remains to be determined).

Recent inventories of relations, such as those in Hovy *et al.* (1992) and Hovy and Maier (1994) may allow us to restructure and complete our domain (and textual) relation inventory; our current inventory has been developed as necessary, based on examples from our corpora. More specific discussions of relation inventories for our model exist in Carlson *et al.* (1994), for example.

By definition of DOMAIN relations, the arguments need be instantiations of ontological concepts, for example, to indicate that a causal enablement relation exists between a particular contractual agreement and a sales event:

```
TMR:
       (CONTRACT23 ...)
       (SALES44 ...)
       (DOMAIN_ENABLEMENT46
                    ARG1:       CONTRACT23
                    ARG2:       SALES44)
```

COREFERENCE relations typically take as arguments instances of ontological concepts. The other relation types usually relate non-ontological TMR constructs such as ATTITUDEs, PROPOSITIONs, other RELATIONs, TIME objects, etc.

```
TMR:
       (PROPOSITION89
                    HEAD:       CONTRACT23 ...)
       (PROPOSITION94
                    HEAD:       TIE_UP69 ...)
       (TEXTUAL_PARTICULAR47
                    ARG1:       PROPOSITION89
                    ARG2:       TIE_UP69)
```

This textual relation identifies that a discourse relation exists between the portion of the input text that discusses the contractual agreement and the portion that discusses the joint venture, and that the discourse relation is of general/particular.

3.4 Syntax and Syntactic Parsing

As mentioned above, the syntactic component operates as a dynamic knowledge source, producing knowledge in the form of parse trees which is used as input for heuristics in the semantic analysis component. The discussions below will only sketch components of one possible model for a syntactic processing module and associated knowledge sources, as necessary for further exposition in this paper; for more detailed discussion of this particular syntactic parser, a bidirectional chart parser which dynamically builds grammar rules from the **SYN-STRUC**s of lexemes in the input, see Gibson (1990), Gibson (1991a), and Gibson (1991b), and for more detailed discussion of the position of syntactic information in the lexicon see Section 3.2 and Meyer *et al.* (1990).

As discussed in Section 3.3, the result of the analysis process is a TMR interlingual representation; a point that bears repeating here is that the TMR does not represent syntactic information about the input string. However, the approach taken here is consistent with the mainstream semantic analysis approaches in that it does involve syntactic analysis. The syntactic parse produced in the early stage of analysis is considered to be one type of knowledge produced by a dynamic knowledge source; this knowledge is used as input to the heuristics which help guide the two phases of the semantic analysis process. The syntactic parse identifies the lexemes corresponding to words, idioms, or morphemes in the input string, and eliminates those lexemes which do not meet basic syntactic constraints; this information is used by the *instantiation* process in semantic analysis (see Section 6.2 and Section 7.3). The syntactic parse structure is used by a heuristic (Heuristic VI) to help guide the application of the *combination* operator (see Section 6.3 and Section 7.4). In some combination processes (Section 6.3.2) the syntactic parse is a significant source of knowledge for the heuristic guiding the order of application.

The model presented here does not stipulate a specific syntactic theory, syntactic parser, or syntactic representation. There are only a few assumptions made about the syntactic representation and structure. A parse imbeds specific lexemes (entries from our lexicon) in the structure, typically as heads projecting structure; each lexeme that matches the orthographic, morphological, and syntactic constraints (including subcategorization frames) is imbedded in the parse structure (or in the forest of possible parse structures). Lexemes of any syntactic category can select for other constituents in this model, although this isn't obligatory. To handle conventional and idiomatic language, we allow heads to select for specific lexemes; these heads can then bear (in their lexical entries) the semantics of selected-for lexemes, phrasal constituents, or entire constructions. This is in line with the construction grammar discussed in Fillmore *et al*. (1988). Furthermore, we allow constructions or idioms to be broken into two or more meaning-bearing units, which allow us to handle idiomatic expressions such as *his goose is thoroughly cooked* or *she pulled many many strings to get that agreement signed* (see further discussion of this issue below).

3.4.1 The F-Structure

The syntactic structure used in the implementation of the model being presented here is a modification of a Lexical-Functional Grammar (LFG) f-structure representation (we will still refer to it as an f-structure). The traditional LFG f-structure is augmented by a ROOT identifier (akin to the labelling of a node in a tree structure); at each level, the ROOT identifier is followed by the word sense identifier (lexeme name) for the relevant word. The representation can be thought of as a list representation of a (possibly recursive) feature structure, where each attribute name is followed by either a symbol value or another (imbedded) f-structure. For example, the f-structure below is the preferred parse of the sentence *The old man ate a doughnut in the shop*.

```
F-STRUCTURE:
     ((ROOT +eat-v1)
       (MOOD DECL) (VOICE ACTIVE) (NUMBER S3)
       (CAT V) (TENSE PAST) (FORM FINITE)
       (SUBJ ((ROOT +man-n1)
              (NUMBER S3) (CAT N)
              (PROPER -) (COUNT +) (CASE NOM)
              (DET ((ROOT +THE-DET1) (CAT DET)))
              (MODS ((ROOT +old-adj1) (CAT ADJ)
                  (ATTRIBUTIVE + -))))
    (OBJ ((ROOT +donut-n1)
          (NUMBER S3) (CAT N) (PROPER -) (COUNT +)
          (DET ((ROOT +a-det1) (CAT DET)))))
    (PP-ADJUNCT ((ROOT +in-prep1)
                 (CAT PREP)
                 (OBJ ((ROOT +shop-n1)
                       (NUMBER S3) (CAT N)
                       (PROPER -) (COUNT +)
                       (DET ((ROOT +the-det1)
                             (CAT DET))))))))
```

The same structure may also be viewed in the (perhaps more familiar) typed feature structure matrix shown in Figure 3C.

3.4.2 Lexical Syntactic Specification

The contents of the **SYN-STRUC** zone of a lexicon entry is an indication of how the lexeme fits into parses of sentences. In addition, this zone provides the basis of the syntax-semantics interface. Thus a brief specification of this zone is necessary to present the foundation of the semantic analysis process, which relies on the syntax-semantics interface as one of the dynamic knowledge sources used in constructing a semantic representation (i.e., the TMR) for input text.

The information contained in the **SYN-STRUC** zone essentially amounts to an underspecified piece of an f-structure parse of a typical sentence using the lexeme (as specified in Section 3.4.1); this piece, called an *fs-pattern*, contains the lexeme in question, and may include information from, typically, one or two levels of structure above and/or below the current lexeme.

Since f-structures do not indicate linear order, the fs-pattern is essentially a dependency structure. In the simple case, the fs-pattern for a verb will indicate the arguments for which the verb subcategorizes. In LFG f-structures, all arguments (including subjects) are immediate children of the verb node, so the selection in the fs-pattern is for elements which are descendants of the current lexeme in the f-structure tree. We use the same mechanism for syntactic relationships other than arguments. So adjectives and prepositions, for example, select (in their respective fs-patterns) for the syntactic head which they modify (in addition, prepositions select for their arguments.)

In the fs-patterns, we place variables at the `ROOT` positions selected for by the lexeme in question, which is identified by the variable `$VAR0`; this allows the fs-patterns to be inherited (using the **SYN-S-CLASS** syntactic class mechanism described below). Subsequently numbered variables (`$VAR1`, `$VAR2`, ...) identify other nodes in the f-structure with which the current lexeme has syntactic or semantic dependencies. For example, the fs-pattern below is appropriate for a regular monotransitive verb:

```
SYN-STRUC:
      ((ROOT $VAR0)
       (SUBJ ((ROOT $VAR1) (CAT N)))
       (OBJ ((ROOT $VAR2) (CAT N))))
```

$$
{}_{\text{+EAT-V1}}\begin{bmatrix}
MOOD & DECL \\
VOICE & ACTIVE \\
NUMBER & S3 \\
CAT & V \\
TENSE & PAST \\
FORM & FINITE \\
SUBJ & {}_{\text{+MAN-N1}}\begin{bmatrix} NUMBER & S3 \\ CAT & N \\ PROPER & - \\ COUNT & + \\ CASE & NOM \\ DET & {}_{\text{+THE-DET1}}\begin{bmatrix} CAT & DET \end{bmatrix} \\ MODS & {}_{\text{+OLD-ADJ1}}\begin{bmatrix} CAT & ADJ \\ ATTRIBUTIVE & + \end{bmatrix} \end{bmatrix} \\
OBJ & {}_{\text{+DONUT-N1}}\begin{bmatrix} CAT & N \\ NUMBER & S3 \\ PROPER & - \\ COUNT & + \\ DET & {}_{\text{+A-DET1}}\begin{bmatrix} CAT & DET \end{bmatrix} \end{bmatrix} \\
PP-ADJUNCT & {}_{\text{+IN-PREP1}}\begin{bmatrix} CAT & PREP \\ OBJ & {}_{\text{+SHOP-N1}}\begin{bmatrix} CAT & N \\ NUMBER & S3 \\ PROPER & - \\ COUNT & + \\ DET & {}_{\text{+THE-DET1}}\begin{bmatrix} CAT & DET \end{bmatrix} \end{bmatrix} \end{bmatrix}
\end{bmatrix}
$$

Figure 3C. Matrix f-structure representation

Or, viewed as a feature structure:

$$
{}_{\boxed{0}}\begin{bmatrix} \text{SUBJ} & {}_{\boxed{1}}\begin{bmatrix} CAT & n \end{bmatrix} \\ \text{OBJ} & {}_{\boxed{2}}\begin{bmatrix} CAT & n \end{bmatrix} \end{bmatrix}
$$

The exact syntactic relationship of words in a sentence may vary due to syntactic transformations, valency changes, or movement rules; the variables support a level of indirection in the fs-patterns. Additional advantages of this mechanism include the ability to inherit fs-patterns from a hierarchy, as well as reducing the work in assigning correspondences between lexical functions and case roles.

In cases of lexicon entries for idioms, verbs with particles, non-compositional collocations, etc., the `ROOT` attribute in an fs-pattern may be followed by a specific lexeme instead of the variable. For example, the special sense of *kick* which defines the idiom *kick the bucket* will select for an `OBJ`ect with `ROOT` **+bucket-n1**, where **+bucket-n1** is a lexeme identifier for a standard sense of the word *bucket*. Additionally, in the fs-pattern, the attribute-value pair will be followed by the symbol `null-sem` as follows: `(ROOT` **+bucket-n1** `null-sem)` to indicate that this word sense does not contribute to the semantics of the phrase. In cases of idioms such as *spill the beans*, *spill* will select for an `OBJ`ect which will specify `(ROOT` **+beans-n3**`)`, meaning that this special sense of *beans* (meaning *information*) does contribute its meaning as an idiom chunk to the entire idiom. In both of these cases it is obligatory to specify the root, so the special sense in question will fail the syntactic parse (in analysis) if the selected root does not appear in the utterance. In generation, any special sense will get selected in the lexical selection process only if the meaning is appropriate. This approach to multi-word lexical entries resembles the Construction Grammar approach of Fillmore *et al.* (1988) or as discussed in Levin and Nirenburg (1994a) or used in Oflazer and Yilmaz (1996) and McRoy (1992), but augmented with the semantic chunk approach for internal structure for some idioms (which also, at least partially, addresses the idiom frozenness issue) which have internal semantic structure (unlike Dyer and Zernik (1986), for example, who insist on a single concept per phrasal entry).

The **SYN-STRUC** zone has two facets. If the word is syntactically regular, non-idiomatic, has no particles, etc., then the **SYN-S-CLASS** facet is used to indicate which fs-pattern to inherit from the class hierarchy of fs-patterns (see, e.g., Mitamura (1990) for an early description of this kind of mechanism). If none of the class fs-patterns are appropriate for the lexeme in question, an fs-pattern may be locally specified in the **SYN-S-LOCAL** facet; in fact, both a class and local information may be specified, and the two fs-patterns are unified.

In addition to specifying syntactic dependency structure, the fs-pattern also indicates an interaction with the meaning pattern from the **SEM-STRUC** zone (as described in Section 4), in that certain portions of the meaning pattern for a phrase or clause are regularly and compositionally determined by the semantics of the components (Principle of Compositionality); the structure of the resulting meaning pattern is determined not only by the semantic meaning patterns of each of the components, but also by their syntactic relationship in the f-structure.

3.4.3 Syntactic Processing

Semantic processing is initiated by invoking the appropriate module with the augmented f-structure representation of the text (one sentence at a time, currently) which the syntactic parser produces; the augmentation of the base LFG-like f-structure involves the binding of the variables (such as `$VAR1`), if any, found in the **SYN-STRUC** of each of the lexemes in the f-structure. For example, the **SYN-STRUC** for the transitive sense of *eat* is:

```
SYN-STRUC:
     ((ROOT $VAR0)        ;$VAR0  gets bound to lexeme +eat-v1
      (SUBJ ((ROOT $VAR1);$VAR1  gets bound to head lexeme
             (CAT n)))    ;this phrase is headed by N
      (OBJ ((ROOT $VAR2); $VAR2  gets bound to head lexeme
            (CAT n))))    ;this is also a noun phrase
```

This segment of an LFG f-structure-like pattern acts as a template in the syntactic parsing process, providing the subcategorization frame. The pattern indicates that the word *eat* is the head (root) of a piece of f-structure, and that it has two arguments. The structure of the formalism allows sentential modifiers or prepositional phrases, for example, but disallows additional arguments (this will become clearer below). So during parsing, the subject gets identified, and the f-structure piece after `SUBJ` gets unified, in effect, with the f-structure of the possible subject argument to the verb. This constrains the subject to be a noun phrase (i.e., an f-structure whose `ROOT` (meaning head) is of `CAT`egory noun). Similarly, the `ROOT` value itself gets unified with the `$VAR1`, thereby binding the `$VAR1` to be the lexeme head of the subject noun phrase f-structure. Thus if the sentence were *The dog ate the rabbit*, `$VAR1` would be unified with (bound to) the lexeme **+dog-n1** (which is the lexeme of the appropriate sense of *dog*.) Likewise, `$VAR2` would be bound to **+rabbit-n1**.

In some cases the **SYN-STRUC** of a lexeme does not subcategorize, in the usual sense, but selects where it may fit in an f-structure of a sentence. This is the case for prepositions, adjectives, adverbs, among others. The **SYN-STRUC** below is for the basic location sense of the preposition *in* (**+in-prep1**):

SYN-STRUC:
```
      ((ROOT $VAR1)
       (CAT N)
       (PP-ADJUNCT ((ROOT $VAR0)
                    (OBJ ((ROOT $VAR2)
                          (CAT N))))))
```

In this example, the lexeme in whose entry this pattern appears, namely **+in-prep1**, appears in the middle, not at the top, of the f-structure piece (`$VAR0` is always used to identify the lexeme to which the pattern belongs). In a sense, this lexeme "selects" down the f-structure tree for its argument (in this case an NP whose noun head gets bound to `$VAR2`), as well as "selecting" up the f-structure tree, thus identifying where this structure may attach itself too. In this case, the constraint on the `ROOT` which specifies the category to be a noun restricts this f-structure pattern from identifying non-nominal PP-adjuncts; this pattern will only unify if the attachment point is an NP, not a sentence or VP. Thus `$VAR1` gets bound to the noun head lexeme of the NP which is the attachment point of the PP.

This same formalism, by the way, allows the handling of verb-particle pairs and idioms without any change. Both idioms and verb-particle pairs are entered in the lexicon as special senses of the head of the expression, typically the verb. For example, the **SYN-STRUC** below is for the verb-particle pair *drop by*, in the sense of coming to visit:

SYN-STRUC:
```
      ((ROOT $VAR0)
       (SUBJ ((ROOT $VAR1) (CAT N)))
       (OBLIQUE1 ((ROOT +by-prep2)
                  (NULL-SEM +)
                  (CAT PREP)
                  (OBJ ((ROOT $VAR2) (CAT n))))))
```

Here the lexeme **+drop-v17** selects for a subject, whose head lexeme gets unified with `$VAR1`, as well as an oblique argument which is a prepositional phrase, whose `ROOT` head is the preposition

+by-prep2. The verb head discounts the semantic contribution of the particle (i.e., by the `NULL-SEM` structure). That preposition or particle has an object, and `$VAR2` gets bound to the lexical head of that noun phrase argument. Thus, as a result of syntactic processing, the variables `$VAR1` and `$VAR2` end up with the same bindings for the sentences *The girl dropped by the store* and *The girl visited the store*: `$VAR1` is bound to **+girl-n1**, and `$VAR2` to **+store-n1**, and the particle no longer plays a role.

In other words, all multi-word expressions whose meaning is conventional, idiosyncratic, idiomatic, or otherwise not immediately compositionally derivable from the literal meaning of the constituents are handled the same way, with a head which selects for the particular lexemes which constitute the expression, and with the head specifying the nature and content of the semantic contribution of each component lexeme.

4. Representation of Lexical Semantic Knowledge

The lexicon used in this discussion is based on work on the DIANA system (described in detail in Meyer *et al.* (1990)), subsequent work on the MIKROKOSMOS lexicon, and builds on a discussion focussing on some of the lexical semantic issues found in Onyshkevych and Nirenburg (1991, 1995).

As described in Section 3.3, the semantic representation used in TMR is a connected network of frame structures; the type of these frames can be identified as being either *ontological* or *non-ontological*. The non-ontological structures in the TMR representation arise from one of two sources: either lexical triggers (i.e., they are instantiated from the **`LEX-MAP`** definition in the lexical entries of the word sense of a word in the text), or from pragmatic knowledge sources (including discourse structure interpretation/expectation mechanisms). The non-ontological structures which arise from lexical triggers behave similarly to the ontological ones in the basic instantiation and structure-building process, and will be so treated below in the discussion of that process in Section 7. The ontological graph-search processes described in subsequent sections do not apply to these non-ontological structures; none of the discussions in Section 5, Section 8, or after apply to the latter type of non-ontological structures.

The lexical semantics of a lexical unit is typically represented in the **`LEX-MAP`** field of the **`SEM-STRUC`** zone of a lexical entry. In the simplest case, the **`LEX-MAP`** links the lexical unit with an ontological concept; thus, the essence of the lexical meaning is referring to an ontological concept. Viewed procedurally, the link in the **`LEX-MAP`** field is an instruction to the semantic analyzer to add an instance of the ontological concept in question to the nascent TMR. So, for example, one sense of the English word *dog* might be treated in our system as a link to the concept DOG in the ontology, or, in other words, a command to create an instance of it (e.g., **DOG34**). The meaning assignment mechanism works this simply only in the case of one-to-one mapping between word senses and ontological concepts, which is not necessarily the case for many lexical units, as is discussed below. More complex mappings are required for most lexical units in a realistic lexicon.

4.1 Issues in Lexical Semantic Representation

4.1.1 Elements of the Language of Lexical Semantic Representation

As mentioned in the introductory chapters above, our framework requires meaning representation, using a metalanguage — in our case, the lexical semantic specification language and the TMR. We find that, in order to provide a coherent computational semantic theory in addition to supporting practical application of that theory to real-world tasks (in our case, translation), that metalanguage needs to be defined. We concern ourselves with one aspect of the definition of this language here: the set of primitives; another aspect, the syntax and semantics of the language for specifying lexical meaning, is explored in this chapter below, while the syntax, semantics, and lexicon for the language for representing sentential/propositional meaning are treated in Section 3.3 above. More detailed discussion of this issue can be found in Onyshkevych and Nirenburg (1994), Levin and Nirenburg (1994), Nirenburg *et al.* (1995).

The status of primitives in many other approaches to lexical or computational semantics remains unclear. One group of computational frameworks, including Fass (1986, 1988), as well as MRDs for human use, relies on words of the language defining each other, namely lexical semantic specification via other words or word senses (only a small subset of MRDs, namely Procter *et* *al.* (1978) and Procter (1995), use a fixed inventory of defining terms, albeit in the same language and without word sense tags). Although this may support moderate word sense disambiguation, as described in Section 8.1, applications such as KBMT which require inference, resolution of non-literal relations, etc. need more of a lexical representation than an ambiguous circular definition using word senses.

Another group of practitioners of computational semantics use primitives of a metalanguage which is claimed to be something other than English or any other language, but, when pressed, they are unable to define the inventory or vocabulary of this metalanguage. We take, as an example, the generative lexicon work of Pustejovsky (1991, 1995), Pustejovsky and Bouillon (1995), or Buitelaar (1997). The 'qualia' structure introduced in this work is populated with supposedly metalanguage-based terms, such as physobj or read, but with no definition for those terms.

Dorr (1993) and colleagues rely on Jackendoff's Lexical Conceptual Structures, as discussed in Jackendoff (1983, 1988, 1990), which are, however, still somewhat reflective of syntactic information (predicate structure, for the most part), and are, thereby, somewhat language-dependent. These LCS constructions do allow some decomposition (see Section 4.1.2 below) by a small number of underdefined semantic primitives, used for limited (compared to the framework discussed in Section 8.1.2) selectional restrictions. As discussed in Wilks (1992a), Onyshkevych and Nirenburg (1994), or Arnold (1996), LCS is nowhere near expressive enough for full representation of meaning needed for KBMT, focussing mostly on verb/argument structure. Although the approach has traditionally relied on a limited number of primitives, the practitioners don't seem concerned about increasing the inventory and defining those primitives in a well-founded manner, but use "upper-case semantics", resulting in primitives that have no grounding, no definition relative to other primitives, and no semantics (such as the recent addition of a spate of "primitives" such as runningly, swimmingly, etc.) As explored in Levin and Nirenburg (1994b), as a result of this choice regarding lexical semantic specification (in addition to others), LCS are sometimes the same across languages, but cause a significant problem for MT when they're not. These and other problems arise when there are translation equivalents which have different lexical semantics, as partly recognized in Dorr (1994). This range of problems (including cases where an argument-taking predicate in one language corresponds with one which doesn't in another, like German *gern*) would be avoided by predefining a complete inventory of primitives for meaning representation, and the redefinition of LCS to be less reflective of the predicative structure of each language treated.

Having a defined inventory of ontological primitives, however, isn't sufficient for supporting interlingual MT — a syntax and semantics for the metalanguage, using the primitives, is required. McRoy (1992), for example, uses an ontology (or, rather, a taxonomy) of 1000 concepts for use by selectional restrictions, but not for use in defining the specific lexical semantics of each word sense. Although adequate for her purposes and experiments in word sense disambiguation, this structure wouldn't be sufficient for MT, since many dozens of word senses map to single concept, with no discrimination between them (e.g., the 65 siblings for *issue*, in the magazine sense, also include all other published documents).

4.1.2 Ontology vs. Lexicon Trade-Off: Granularity

As suggested in Section 3.1, there is no consensus in the semantics or knowledge representation fields about the granularity of an ontology or world model; the granularity decision has a profound impact on the lexical semantics zone in the lexicon. One view of the ontology is to have a one-to-one correspondence between every word-sense in the lexicon and a concept in the ontology. This *word-sense* view of the ontology, in addition to the obvious disadvantage of rampant proliferation of ontological concepts (related to what Hobbs (1985) calls *ontological promiscuity*), leads to problems in multilingual applications — often roughly comparable words in different languages do not "line up" the same way; this, in turn, leads to further proliferation of new concepts with every new language, as well as inaccurate lexical mappings, problems that have plagued transfer approaches to MT, as suggested by the categorization of divergences in MT in Dorr (1994). These and other problems make this approach impractical for those applications which require significant inferencing of any sort, as would be the case for MT with fine word-sense discrimination. Practitioners of this word-sense approach to KBMT include Knight and Luk (1994) and Farwell *et al.* (1993), among others.

Another well-known approach, which may be called the *decompositional* approach, utilizes a small restricted set of primitives which are combined or constrained in an attempt to render the meaning of any lexical unit. This approach leads to other difficulties in building large-scale world models and capturing shades of meaning; it is not clear that it is possible to derive a set of *a priori* primitives and a compositional formalism which would be expressively adequate to capture the meanings of all desired word senses. This gets even more complicated when insisting that each primitive be lexicalized in every world language (Wierzbicka (1992)). Additionally, this approach can yield enormous, unmaintainable lexicon entries for complex concepts; Wilks (1975a) required 14 primitives in a three-deep structure to represent *drink*, and Small and Rieger (1982) required pages of code for each lexical entry. In the Conceptual Dependency paradigm of Schank (see Schank and Abelson (1977), Schank (1975)), the number of primitives for representing events numbered in the teens, resulting in large complexes for simple events, not unlike Wilks (1975a).

The approach taken in the model adopted here lies somewhere in between the word-sense and the decompositional approaches, as expressed in Nirenburg and Goodman (1990):

> "Viewed as an object, developed in a concrete project, an interlingua should be judged by the quality of the translations that it supports between all the languages for which the corresponding SL-interlingua and interlingua-TL dictionaries have been built. As a process, its success should be judged in terms of the ease with which new concepts can be added to it and existing concepts modified in view of new textual evidence (either from new languages or from those already treated in the system.)" (p. 9).

The notion of "completeness as proof of feasibility for interlinguae" (p. 10) is rejected in the design of the ontology; the ontology is not determined *a priori*, but rather, is updated and revised as new lexemes are entered into the lexicon, as new cross-linguistic evidence of shared concepts arises, and as the domain of the ontology is shifted. The TMR therefore reflects this decision as to the scope of the ontology. We do agree, in principle, with Wilks (1975a):

> It is a hypothesis of this work that we can build up a finite but useful inventory of bare templates adequate for the analysis of ordinary language: a list that can be interpreted as the messages that people want to convey at some fairly high level of granularity".

However, we have a different model of the nature and number of the elements that enter into this inventory; while Wilks used an inventory of 70 simple primitives and an inventory of unspecified size of "bare templates" (essentially very sketchy scripts, such as `MAN GIVE THING`) that represent primitive events, we build an ontology, consisting of under ten thousand frames that we use compositionally.

Sowa (1993), Dahlgren *et al.* (1989), and even Lakoff (1987) object to an effort to decompose meaning using an inventory of primitives, citing essentially Wittgensteinian arguments against Platonic categories, under the assumption that decompositional efforts such as Schank and Abelson (1973), Jackendoff (1988), or the work described here needed to concern itself with essentially extensional concerns of exact specification of class membership for real-world entities. However, even without recourse to prototypes, as described in Rosch *et al.* (1976), Givón (1986) and others, decompositional approaches to computational or practical lexical semantics are able to continue using discrete semantic categories.[1] The reason for this lies in the goal of the enterprise: many of these decompositional approaches to computational semantics are only concerned with representing the intended meaning of an utterance, and not with truth conditions or categorization of real world entities (see the first desideratum in Section 1.3). As discussed in Section 1.2, this corresponds to the general approach of experientialist cognition of Lakoff (1987), among others, that we implicitly adopt. Thus the primitives used in decompositional meaning representation correspond to choices made by the human speaker of the utterance about class membership. Thus if a speaker chooses to call his pet with one wolf parent, only three legs, and no fur a *dog*, then for purposes of communication it is exactly a dog, even using Platonic categories. Thus any objections on the basis of category membership (vs. gradient), necessary and sufficient conditions, defining features, or truth-conditions are not relevant, because of the nature of our enterprise.

Dahlgren *et al.* (1989) cite further objections to lexical meaning representation via ontology-based decomposition, in that not all categories can be decomposed into primitives. In fact, we benefit from any such cases in that those categories are then natural candidates for elements of the ontology. As mentioned above, the ontology serves to provide exactly the level of granularity of description of meaning necessary for differential representation of meaning over the languages of interest.

Our ontology serves multiple functions, as suggested in Mahesh and Nirenburg (1995). The primary purpose is to define the semantic primitives used to define lexical meaning and, from there, sentential/discourse meaning. Each concept in the ontology can also serve as a selectional constraint on a concept; this alone does not drive the addition of concepts (unlike Dahlgren *et al.* (1989), who build ontologies, despite the above objections, on the basis of the needs of (and for use by) selectional restrictions), but merely allows the full generality of reusing the same resource for both primitives and constraints. Our ontology is designed to be language-independent, but not necessarily in an absolute sense, but relative to the languages under investigation at any given time; in fact, to add an English lexicon after acquiring Spanish, we needed to change or augment less than 0.2 of 1% of the ontology.

4.1.3 Ontology vs. Lexicon Trade-Off: Enumerability of Senses

Our approach to lexicography assumes that some substantial degree of enumeration of senses has taken place in the lexicon acquisition process. Directly related to the issue of the existence and the cardinality of primitives is the question of whether or not a word can only have only one sense, as in Wierzbicka (1992), a finite and enumerable number of senses (our approach), or an infinite number of senses that arise contextual and cannot be predicted, such as what Pustejovsky (1991)

1. Although we find no purpose in entering the debate, at this point, on the nature of our concepts (Platonic vs. Wittgensteinian vs. prototypical, etc.), we find it instructive that Jackendoff (1988) believes that natural language types are “assembled from a finite innate set of primitives and principles of combination - in other words, a decompositional view of word meaning”, since “a type without internal structure cannot be compared with novel tokens to yield categorization judgements”.

claims is required.

Sowa (1993) suggests that “a unified semantic basis along classical lines is not possible for any natural language. Instead of assigning a single meaning or even a fixed set of meanings to each word, a theory of semantics must permit an open-ended number of meanings for each word.” This follows, he claims, from the Wittgensteinian notion of language games, where the meaning of a word is determined by its use. Additionally, Nunberg (1978) suggests that there is an infinite number of contexts that can generate referring functions that affect the meaning of a word. Gibbs (1993) suggests that “understanding contextual expressions involving metonymy requires that a process of *sense creation* must operate to supplement ordinary *sense selection”*. This line of reasoning, however, appears to be conflating the notion of word sense and meaning of a word in context, which would follow from applying the Principle of Compositionality strictly. Approaches such as Buitelaar (1997), Ostler and Atkins (1992), or Pustejovsky (1991, 1995) also don’t seem to make this distinction between senses of a word and meaning of a word in context, and argue that sense enumeration is impossible; they rely on contextual rules, interacting with pre-specified lexical information in the otherwise underspecified lexical semantic representations, to generate new *senses* dynamically in a given context. Nunberg (1978) effectively suggests that pragmatic contextual schemata generate meanings to conform to the context.

But since such approaches have access to only a limited range of such contextual mechanisms, and Nunberg (1978) demonstrates that an inventory of all possible contexts isn’t possible, what these supposedly underspecified approaches actually achieve is merely economy of disk storage: the limited set of contexts that these approaches can capture (and which are enumerable in a computational application) could be applied iteratively, using the sense refinement mechanisms, in an *a priori* method, resulting in a full enumeration of all senses that their mechanisms are able to produce, as discussed in Viegas *et al.* (1997) or Nirenburg and Raskin (1996), who show that “A good enumerative lexicon can be seen as at least weakly — and, in fact, strongly — equivalent (see Chomsky 1965:60) to a generative lexicon after all the lexical rules have fired”.

Our approach does involve enumeration of basic senses (but requires far fewer sense distinctions than an MRD), and is coupled with a contextual meaning refinement mechanism at the propositional level (as opposed to sense refinement or sense creation at the word level)[1]. This meaning refinement is achieved by the application of all available constraints, resulting in inference of appropriate expected meanings (resulting in the addition of additional concepts or links to a TMR; see Section 9.3), in addition to the information made explicit in the text; this is addressed through an essentially abductive process, namely the SDS-building process outlined in Section 7 and Section 5. Since our approach encourages compositionality in lexical semantic representation and in propositional meaning representation, we can represent novel uses for words in context, in effect, by either combining or further constraining meaning elements to form a meaning complex not represented by any one lexical entry; for example, metonymy introduces new entities into a TMR that are related to mentioned entities through specific relations.

Yamanashi (1987) effectively adopts a similar position by suggesting enumeration of core

1. Strictly speaking, we reject the assumption in Wittgenstein (1921) that “when translating one language into another, we do not proceed by translating each *proposition* of one into a *proposition* of the other, but merely by translating the constituents of propositions”. Although we certainly make extensive use of the constituents of propositions, we also introduce elements of meaning into the target language proposition that were conveyed by the proposition as a whole, but not by any individual lexical elements.

senses only, then dynamic implication processes to represent sense coercion in cases of metonymy or other mechanisms. On the other hand, Sowa (1993), who relies on the Wittgensteinian notion of language games (see Wittgenstein (1953)) to motivate his approach, assigns to each word a finite number of lexical patterns "that determine the rules that are common to all the language games that use the word" (thus are not truly dynamic, since they are enumerated, thus only offer economy of storage). These are linked, however, to conceptual patterns, which are language independent; in his model, the meaning component can be dynamically embedded in a meaning complex to represent shifts of meaning in a given context.

In short, the approach we adopt involves enumeration of all core (literal) senses of a word, and then use of inference mechanisms, applied to semantic expectations encoded in the lexicon and ontology, in addition to the expectations deriving from the model of communication, to produce meaning complexes to capture the combined propositional meaning. However, this does not involve either the coercion of a word sense into a new meaning nor the generation of new senses, only the representation of the overall meaning of an utterance by means of inferred meaning elements in addition to the sum of lexical semantic meanings.

4.1.4 Metalanguage for Lexical Semantic Representation

Apresjan (1973) enumerates a set of conditions which he feels are necessary for any metalanguage used for defining or representing semantic knowledge. His basic concern is for lack of ambiguity in the set of semantic primitives: "each word in the vocabulary of the semantic language should express exactly one meaning", and "in the dictionary of the semantic language there should be neither synonymy, nor homonymy of the names of the meanings"; the lexical semantic specification language defined below certainly conforms to this condition. However, Apresjan assumes a one-to-one correspondence between words in the language and defining terms in the metalanguage: "each meaning should be expressed by exactly one word of the semantic language, irrespective of the definition in which it occurs (meanings and their names should be in a one-to-one correspondence)"; while this approach is appropriate for frameworks such as Farwell *et al.* (1993), it is rejected here because of concerns regarding fine granularity discussed above. Other concerns that he has regarding the metalanguage stem from this decision, and since the approach described below makes use of compositional representation of meaning in lexicon entries, his other concerns regarding syntactic and semantic structure being bearers of meanings are addressed. There is one issue that he raises and we agree with, in principle, but have difficulty conforming to, and that is canonicality of representation; because of the expressiveness of the metalanguage, we find that there are often multiple ways of representing lexical meaning, never mind sentential meaning, which could only be avoided at great expense and loss of perspicuity in the notation.

4.2 Sense Mapping

It is possible to find, in some cases, one-to-one correspondences between the meaning of a lexical unit and concept, as in the simple example of *dog*. Unfortunately, in the majority of cases, this is not possible without developing the problems discussed above in relation to the "word-sense" approach. Therefore, mappings are allowed to the most appropriate concept, that is, to the most specific concept that is still more general than (i.e., that subsumes) the meaning of the lexeme in question. Once the most directly corresponding concept is determined, additional information may be specified, thus constraining the primary concept, within the lexical semantic specification in the lexicon entry. This modified mapping may either add information to the concept as it is specified in the ontology, override certain constraints (e.g., a selectional restriction), or indicate relationships with other concepts expected in the sentence. The subsections below deal with these two basic cases: simple one-to-one mapping, which we are calling *univocal* mapping, and then complex, or *constrained* mapping (see Meyer *et al.* (1990)).

4.2.1 Univocal Mapping

As illustrated above with *dog*, this is the simplest kind of mapping, used when there is a one-to-one correspondence between a lexeme and an ontological concept. In the "word-sense" approach to ontology granularity, the bulk of all lexical semantic specifications would be of this type (modulo the semantic dependency structure hooks discussed below). However, as stated, the approach here is somewhere between the "word-sense" and the "decompositional" approaches, so this type of mapping ends up being the case in less than a majority of all entries.

The univocal mapping of exactly one concept to lexeme is utilized when the concept denoted by the lexeme is rather *universal* (essentially, universal has come to mean "common to the languages we, or our informants, know"). As additional languages are treated and cross-cultural concepts come to be reflected in the ontology, the share of univocal entries may increase or decrease. Examples of universal concepts might include the meaning of **+die-v1** (in the most literal sense of "cease to live"), natural kinds such as *tree*, *dog*, artifacts or terms in technical sublanguages, etc. Clearly, when constructing a practical lexicon and ontology, these universal concepts are derived somewhat intuitively, and may reflect the pragmatics of the textual domain in question; for example, technical domains tend to have a high percentage of such concepts.

In a terminological lexicon (i.e., a lexicon for the nomenclature of an expert domain, e.g., the sublanguage of a particular scientific or technical field), the share of univocal mappings will increase, reflecting the higher tendency for terminological nomenclature which corresponds to conceptual objects or objects in the onomasticon precisely (e.g., chemical compounds, machinery, electronic components, etc.).

Our notation for a univocal lexical semantic mapping is straightforward. Thus, the primary noun sense of the word *dog*: **+dog-n1** will have the following **SEM-STRUC** zone:

```
SEM-STRUC:
      LEX-MAP:
                  (%DOG)
```

The "%" indicates an ontological concept that is to be instantiated when the meaning in question is included in the overall semantic dependency structure of a sentence in which *dog* appears. Note that the name of the concept from the ontology (or onomasticon) need not be the same as that of the lexeme in question. A univocal mapping between lexeme and ontological concept merely implies that all constraints on an ontological concept (i.e., all information provided within the frame for that concept in the ontology) are consistent with the meaning of the lexeme.

4.2.2 Constrained Mapping

As mentioned above, in many cases the univocal mappings are not possible, so a complex mapping must be undertaken. Complex mappings may either involve multiple structures in the mapping (for example, a mapping to a concept is often accompanied by an ATTITUDE structure), or a set of constraints added to a mapping to a concept from the ontology (both types of complexity may also be found together in a lexical semantic mapping). In a constrained mapping, linkages are allowed to the most appropriate concept, that is, to the most specific concept that is still more general than (i.e., that subsumes) the meaning of the lexeme in question. Once the most directly corresponding concept is determined, additional information may be specified, thus constraining the primary concept, *within the lexical semantic specification in the lexicon entry.* This modified mapping may either add information to the concept as it is specified in the ontology, override certain constraints (e.g., selectional restrictions), or indicate relationships with other concepts in the sentence.

Once the "closest" concept is determined, constraints and further information (including possible reference to other concepts from the ontology) are recorded in the appropriate slots in the lexicon entry. The facet facility of FRAMEKIT and FRAMEPAC is invoked to encode values, constraints on concepts in a constrained mapping of a semantic specification, and other information; Table 3A, "Head and Facet Use in Ontology, TMR, and Lexical Semantics," on page 31 identifies the facets that are available to the lexical semantic specification — note that they are all used.

The following example illustrates a simple case of lexical semantic mapping, (the lexeme is the lexeme **+eat-v1**, the primary sense of *eat*). The **`SYN-STRUC`** lexicon zone contains the lexical syntactic specification of the lexical entry, in which the subcategorization pattern of the verb is described:

`SYN-STRUC:`

```
                    ((ROOT $VAR0)          ;$VAR0  gets bound to +eat-v1
                     (SUBJ ((ROOT $VAR1)$VAR1  gets bound to head lexeme
                           (CAT n)))       ;whose lexical category is N
                     (OBJ ((ROOT $VAR2)    ;$VAR2  gets bound to head lexeme
                          (CAT n))))))     ;this is also a noun phrase
```

During analysis, the variables `$VAR1` and `$VAR2` are initially bound to "placeholders" for the lexical semantics of the subject and object of the verb, respectively. Once the lexical semantics of those syntactic roles is determined, the semantic composition process gets under way. If this process is successful, a semantic representation for a higher-level text component is produced. The **`SEM-STRUC`** zone of the lexicon entry for **+eat-v1** contains linking information as well as selectional restrictions, constraints on the properties of the meanings of the verb's syntactic arguments:

`SEM-STRUC:`

`LEX-MAP:`

```
                    (%ingest                         ; +eat-v1  maps into %ingest
                       (AGENT  (VALUE ^$VAR1)        ; subject maps into AGENT
                               (SEM *ANIMAL))        ; filler should be a descendent
                                                     ; of ontological concept *ANIMAL
                       (THEME  (VALUE ^$VAR2)        ; object maps into THEME
                               (SEM *INGESTIBLE)
                               (RELAXABLE-TO *PHYSICAL-OBJECT))))))
                  ;theme's meaning should be a descendent of  *INGESTIBLE
                  ;or at least of a *PHYSICAL-OBJECT
```

This structure can also be represented as a feature structure matrix:

$$
{}_{\text{ingest}}\begin{bmatrix}
AGENT & \begin{bmatrix} VALUE & \hat{}\,\boxed{1} \\ SEM & \text{*animal} \end{bmatrix} \\
THEME & \begin{bmatrix} VALUE & \hat{}\,\boxed{2} \\ SEM & \text{*ingestible} \\ RELAXABL\bar{E}-TO & \text{*physical-object} \end{bmatrix}
\end{bmatrix}
$$

Traditionally, selectional restrictions are defined in terms of a small fixed set of concepts or features; we have found that it is often useful to use arbitrary concepts from the ontology as “selectional restrictions”. These constraints are represented in the SEM facet, and can be arbitrary concepts from the ontology:

+taxi-v1 (sense of ‘move-on-surface’, said only of aircraft, e.g., *The plane taxied to the terminal*; *The pontoon plane taxied to the end of the lake*)

```
SEM-STRUC:
     LEX-MAP:
               (%MOVE-ON-SURFACE
                         (THEME
                                   (SEM *AIRCRAFT)
                                   (RELAXABLE-TO *VEHICLE))))
```

Note that there may also be “second-order” constraints (i.e., constraints on constraints):

+jet-v1(literal sense of 'to travel by jet', e.g., *The presidential candidate spent most of the year jetting across the country from one campaign rally to another*

```
SEM-STRUC:
     LEX-MAP:
               (%MOVE
                         (THEME
                                   (SEM *AIRCRAFT))
                         (PROPELLED-BY
                                   (VALUE %JET-ENGINE))))))
```

Thus we see that semantic constraints in this approach can be any arbitrary concept, constrained concept, or even set of concepts from the ontology; this substantially extends the traditional notion of selectional restriction to more fully utilize the knowledge available (from the ontology) for disambiguation.

It is not expected that the meaning of verbs will always be a link to an ontological concept of type EVENT; or that meanings of nouns will uniformly be descendents of the ontological concept OBJECT. There is a great deal of variance in the correspondences between ontological subtrees and parts of speech. For example, many adjectives and nouns (such as *abusive* or *destruction* in English) may be represented as events, whereas many verbs map to attitudes or properties (e.g., *own* or *reek*).

4.2.3 Non-Propositional Mapping

In addition to the direct or modified mapping into ontological concepts as outlined above, four other scenarios can occur in lexical semantic definitions (and represented in **SEM-STRUC** zones of corresponding lexicon entries), either in conjunction with a propositional mapping (and/or each other) or without such a mapping.

- The first case involves situations where the meaning of a lexeme corresponds not to a concept, but to a particular filler of a slot defined in another concept; for example, the basic attributive sense of the adjective *hot* maps to a particular value of the *TEMPERATURE* slot (property) of the meaning of the noun it modifies. In some cases, the semantics of the lexeme indicate the name of the property which connects the meaning of two other lexemes. For example, the locative sense of *in* suggests *LOCATION* as the property on which the meanings of the prepositional object and its attachment point are linked; thus, the meaning of *in* within the phrase *the dog in the park* is that the meaning of *the park* fills the *LOCATION* slot of the meaning of *the dog*. Many syntactic morphemes (including many case markings) exhibit this kind of semantic behavior.
- The second case involves mapping to TMR constructs which are non-propositional, hence non-ontological, in nature — speaker attitudes, stylistic factors, etc. The representation of this "metaontological" information is thus added to the representation of propositional meaning to yield a full TMR. As is the case with propositional information, lexical entries may specify which specific constructs those entries trigger as contributions to TMRs of entire texts. The example below illustrates both of the cases mentioned above. The lexical semantics of the lexeme **+delicious-adj1** contains the following two structures:

```
SEM-STRUC:
    LEX-MAP:
           (^$var1
                     (INSTANCE-OF (SEM (value *INGESTIBLE))))

           (ATTITUDE
                     (type (value evaluative))
                     (attitude-value (value 0.8))
                     (scope (value ^$VAR1))
                     (attributed-to (value *speaker*))))))
```

The first construct places a semantic constraint (i.e., must be a descendent of *INGESTIBLE) on the meaning of what the adjective modifies (referred to by the variable `^$VAR1`). The evaluative `ATTITUDE` scopes over this same meaning. The attitude is attributed to the speaker, which is a default value.

An outstanding issue in this approach to mapping is identifying the particular feature or attribute of the scoped object that is being addressed; it is not the general existence of a food item that is being evaluated highly by *delicious*, but the taste of that food item. The general solution to this issue is to scope over a particular attribute, not the entire object (e.g., `(scope (value ^$VAR1.TASTE))`.

Note that lexicalizing such information explicitly contrasts with the general attitude taken in frameworks such as Dyer and Zernik (1986), who rely on the success or failure of goals and plans in the text at large to provide evidence for emotions and attitudes. While we could make use of such a mechanism (although we have no current microtheory to handle goals and plans), we also make extensive use of lexicalized information about goals and plans as well.

- The third case involves special treatment, different from the usual instantiation/combination processing. For example, the meaning of the definite article *the* in English, at least in one of its senses, involves reviewing the discourse or deictic contexts for entities of a particular semantic type. The meaning of that sense of the article would not involve the instantiation of any new concepts, but will rather serve as a clue for the identification of previously-instantiated concepts for reference.
- The fourth case addresses the lexicalization of information which is typically expressed through morphological or syntactic means. For example, certain verbs contain inherent aspectual markers, or are themselves explicit aspectual markers (such as *begin)*. Thus the lexical semantic specification of these verbs consist primarily (or entirely) of a mapping to an ASPECT structure, but no EVENT concept. Unlike Dahlgren *et al.* (1989), aspect is not a source of distinction within the ontology, but is represented by a meta-ontological mechanism that is orthogonal to the EVENT hierarchy in the ontology. Thus a TMR may have an ASPECT structure, even if there is no EVENT in the TMR.

4.3 Mapping / Category Correlation

Although, at first glance, it appears that there is substantial parallelism between syntactic categories and the general high-level ontological partition into which the corresponding head concept mapping falls, this apparent correlation is very frequently violated and therefor unreliable; the syntactic categories verbs, nouns, and adjectives or prepositions only sometimes map onto concepts in the subtrees headed by EVENT, OBJECT and PROPERTY (the latter being partitioned into ATTRIBUTE and RELATION) respectively. Some of the points below highlight the many types of exceptions to this generalization.

- Equivalent lexical semantic mapping of verbs and deverbal nouns (or nouns and verbalizations) are frequent; since nouns and verbs select for different (subcategorization) frames, the syntax/semantic mapping will be different, however. For example, the lexical semantic specification for both *destroy* and *destruction* are headed by the same concept from the EVENT portion of the ontology. Further discussion of this set of issues can be found in Carlson *et al*. (1994).
- Some verbs (for example, in *The rose smells sweet*), and nouns (such as the object in *The rose has a sweet smell*) map into attributes of other concepts (the semantic mapping of *rose*, in these examples), not EVENTs or OBJECTs.
- Many adjectives map into ATTITUDEs, not *ATTRIBUTE*s (such as *delicious*).
- Since the ontology EVENT subtree avoids stative verbs, such verbs tend to map into *RELATION*s or *ATTRIBUTE*s of OBJECTs. *Own* or *possess* would be mapped onto the relation *OWNER-OF*, for example, and *contain* maps onto the relation *CONTAINS*.

See Meyer *et al*. (1990) and Meyer and Steele (1990) for further discussion of this set of issues.

5. The Underlying Heuristic: Ontological Graph Search

Subsequent sections define and illustrate the architecture of the overall semantic analysis process, namely the state-space search. We precede those discussions, however, with an explanation of a major strength of the approach, namely, that within the framework of that architecture, essentially all facets of core semantic analysis are guided by one central heuristic, the ontological graph search mechanism (Heuristic I). This graph search mechanism applies to usual selectional-restriction satisfaction cases as well as to cases requiring some relaxation of tight constraints (see Section 7), thus it centrally supports word sense disambiguation (WSD), as described in detail in Section 8. However, the method also applies to linguistic constructions which are typically troublesome for semantic analysis approaches: the same mechanism applies to cases of metonymy resolution (see Section 9), as well as some cases of reference resolution. With some additional mechanisms to handle the bracketing problem, the same mechanism may be applicable to the issue of resolving noun-noun compounding (see speculation on this in Section 11.1). The nature of the control architecture, the state-space search, and the heuristic (the ontological graph search) enable these constructions to be addressed within the same framework, without separate modules or special treatment.

In brief, the heuristic attempts to find the best match between *i)* semantic constraints on case roles or relations of a concept and *ii)* the candidate fillers for those roles or relations. As described in Section 3.3, any concept in the ontology may be instantiated and included in a TMR representation; all instantiated concepts are linked to other concepts through roles or relations. Attempting to satisfy constraints on roles or relations is the central mechanism of the WSD and semantic dependency structure (SDS) building activities of semantic analysis.

All concepts have relations (including arguments or case-roles in the case of events and attributes in the case of entities). Each relation may have a constraint or a set of constraints on what concepts (or complexes of concepts) may fill that slot. (In fact, relations in the ontology have up to three levels of selectional constraints: an overall constraint, an expected (default) filler that meets the constraint, and a limit to allowed relaxation of the constraint.) The constraints are themselves also concepts from the ontology (or Boolean combinations of concepts).

Thus by identifying the "best" path in the ontology between a constraining concept and the candidate filler, it is possible to determine the degree to which the candidate filler satisfies the constraint. The heuristic attempts to find the best path, i.e., the shortest distance (if arc weights were thought of as having some inverse relationship to arc distance), between the constraint and the candidate in the ontology graph.

Heuristic I. Semantic constraint satisfaction and relaxation corresponds to finding the best path over weighted arcs in an ontological graph from the candidate filler to the constraint.

Section 5.1 presents the algorithm for the ontological graph search *per se*, Section 5.2 focuses on the arc weights, and Section 5.3 explores how the arc weights are acquired. Subsequent chapters apply this heuristic specifically to semantic analysis.

5.1 The Graph Search

Because natural language use is not literal or precise (because of vagueness, metonymy, etc.), we often need to relax constraints because strict semantic restrictions do not apply as expected; however, relaxing or discarding semantic constraints unrestrictedly would result in egregious proliferation of readings in semantic analysis (and no success at word sense disambiguation). On the other hand, limiting the relaxation of constraints results in lack of successful analysis of non-literal input.

The approach taken in Fass (1986), (1988), or (1991), for example, is to have a limited set of patterns over an ontology-like hierarchy in identifying where semantic constraint relaxation may be licensed. We find that having a fixed set of patterns is too restrictive to account for a wide variety of language constructs; the richness of the ontology in the current model allows paths of virtually unrestricted topologies between constraint and filler. Thus this model allows a path of any topology (instead of one of several topologies as in Fass) over the ontology, so long as the arcs in the path collectively are deemed most appropriate.

5.1.1 The Ontology as a Graph

The premise of our heuristic is that some number of semantic relations between the source concept (the candidate filler) and target concept (the constraint) must exist, and the task of the semantic constraint mechanism is to find the most plausible relation or combination of relations. The relation may be too remote for the purpose at hand, hence a cost threshold; if no relation is found within the cost threshold, the ontological graph search algorithm, as a heuristic, identifies to the overall control architecture of the system (the state-space search) that there is no appropriate semantic relation between the two concepts, hence the interpretation or reading which is being investigated must be considered implausible and thus rejected.

In our method, controlled constraint satisfaction is managed by considering all relations, not just *IS-A* arcs, and by levying a cost for traversing any of those relations, especially non-taxonomic ones. We treat the ontology as a directed (possibly cyclic) graph, with concepts as nodes and relations as arcs. Thus constraint satisfaction is treated as a cheapest path problem, between the candidate concept node and the constraint nodes; the best path thus reflects the most likely underlying semantic relation, whether it be metonymic or literal.

In the easiest case, the selectional constraints on the correct set of senses are all satisfied, and are violated for incorrect combinations of senses. Satisfied selectional constraints appear in the method as a simple taxonomic path over the *IS-A* hierarchy between the candidate concept and the constraint.

But other slots define arcs as well. For example, an *INGEST concept may have an *AGENT* slot which is to be filled (in the VALUE facet) with the concept instantiation identifying the eater, and an *THEME* slot which gets filled with the instantiated concept representing what is being eaten. Similarly, a concept such as *AUTOMOBILE would have slots (either locally or inherited) labelled *HAS-AS-PARTS* and *OWNER*, for example, which also would get filled with the names of concepts, either instantiations or from the ontology.

The SEM facet (see Table 3A, “Head and Facet Use in Ontology, TMR, and Lexical Semantics,” on page 31 for discussion of the facet definitions) is the most prominently used facet in the ontological graph search algorithm, because it contains a concept (or a Boolean combination of concepts) which acts as the semantic constraint on possible fillers of the slot (which would be inserted into the VALUE facet). Taking the example from above, *INGEST would have, in the *AGENT* slot, a SEM facet whose filler is *ANIMATE. Viewing the ontology as a graph, there exists an arc labeled *AGENT* from *EAT to *ANIMATE.[1] The slots which are inherited by a concept from parent concepts do not correspond to direct arcs in the graph. The reason why it is valuable to include the

VALUE facets of slots is to access fillers which are either defined values in the ontology (which is where the vertical hierarchical relations defining the ontology as a network or tree actually reside, specifically in *IS-A* and *SUBCLASSES* slots) or to access instantiations (accessed through the VALUE facet of the *INSTANCES* slot) and their internal structure (for example, for reference resolution).

When we are discussing the ontology as a graph, we will treat the fillers of the VALUE facets (on *IS-A* and *SUBCLASSES* slots) and fillers of the SEM facets (on all other slots) equally. In diagrams of the ontology (such as Figure 5A below) there is no distinction between the contents of these facets; in figure of lexical semantic specifications or TMRs, on the other hand (such as Figure 2E and Figure 2G, respectively), a vertical arc (labelled with the slot name) connects the concept with a node (labelled with the filler of the VALUE facet), while other facet fillers are identified by labels on nodes connected by labelled horizontal arcs.

We also treat the fillers of the DEFAULT facet. In fact, although these facets aren't as frequent as SEM facets, they contain more informative constraints, in that they reflect the typical fillers of that slot. For that reason, we make the arc cost cheaper (by using an exponent of 0.5) to suggest that the relation to a concept in a DEFAULT facet is more plausible (hence cheaper) than the relation to the concept in the corresponding SEM.

In addition to the actual ontology itself, "ontological knowledge" in the current context includes instantiations of concepts from the ontology, as mentioned above. At any point in analysis, there may be "available" concepts instantiated by the current or previous sentences (the tenure of past concepts is controlled by other microtheories, specifically discourse processing mechanisms). The ontological concept which is used as a "template" for instantiation acquires a pointer to the instantiation (via the frame instantiation mechanisms) in the *INSTANCES* slot, which satisfies the criterion for an arc (above). The graph being traversed by the ontological graph search algorithm is therefore populated with instantiations as well as ontological concepts. This allows entities instantiated elsewhere to be identified in the graph search process described below; in particular, this allows the reference resolution mechanisms to access instantiations from elsewhere in the discourse structure.

In the ontological graph search, arcs that are traversed impose a cost; the cost will typically be discussed below as a weight on an arc. Many graph search approaches simply use the same unit cost for each arch; that approach was discarded as inadequate, as discussed below in Section 8.1. The simplest effective model of arc costs that we used in our experiments is to consider them multipliers within *[0.0, 1.0]*, so an arc cost of *1.0* would result in no cost being assessed for a path traversing the arc, whereas an arc cost of *0.0* would indicate an infinitely expensive arc, that is, one that is not allowed in the desired path. The optimal path is thus the maximal one, i.e., closest to *1.0*. Section 5.2 below contains a discussion of the mechanisms and approaches to assigning costs to individual arcs traversed by the ontological graph search.

Figure 5A presents an (unrealistic and oversimplified) example of a segment of an ontology, with the arcs labelled. Note, however, that all arcs have an inverse:

1. many of the examples here are for illustrative purposes only, and do not necessarily reflect the current ontology

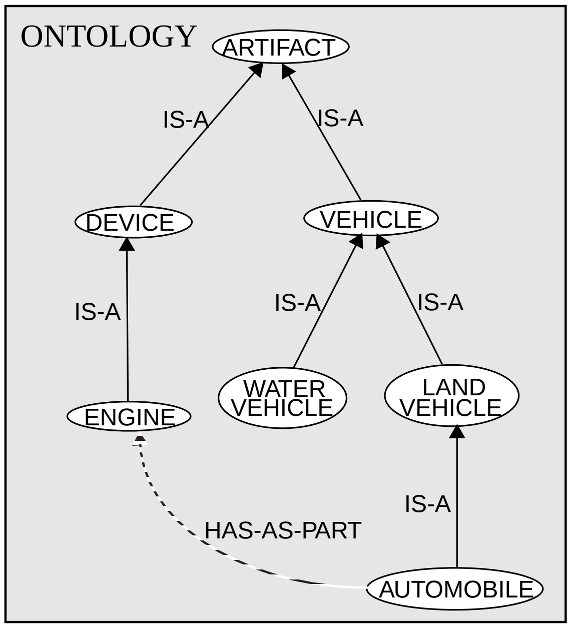


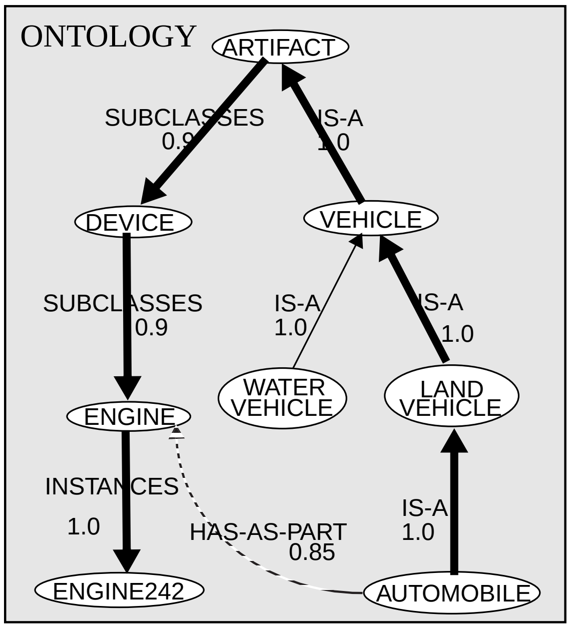
Figure 5A. Toy illustration of arcs.

SLOT	INVERSE
IS-A	*SUBCLASSES*
SUBCLASSES	*IS-A*
HAS-AS-PART	*PART-OF*
PART-OF	*HAS-AS-PART*
INSTANCES	*INSTANCE-OF*
. . .	

The inverse arcs are not indicated in the figure; the arc label will be associated with the direction of the arc. In the mechanism that provides a weight for each arc described in Section 5.2, the weight returned for an arc is often different from the weight returned for its inverse. The reason for that will become clear from the examples below.

Figure 5B illustrates two paths over the same ontology, along with indications of sample arc weights (not necessarily those in actual use) that might be returned for each particular arc for the indicated direction, using the simplest table-lookup arc-weight mechanism for a simple constraint-satisfaction case. If an example were *Billy drove his V8 at 60 m.p.h. down Main Street*, there would be a need to fill some slot for the event concept with the automobile (assume, for the sake of example, that the sense of *drive* being used only allowed cars) being driven; *V8* would be identified (maybe after discarding vegetable juices as having unacceptably low preference) as a type of engine (possibly using the onomasticon knowledge source). The ontological graph search mechanism would be invoked in an attempt to identify whether ENGINE242 meets the constraint of *AUTOMOBILE. The path through the *ARTIFACT node would have a preference of *0.81*, as illustrated in the left view in Figure 5B, whereas the path over the *HAS-AS-PART* arc (right view) would have a better preference metric of *0.85* so would be preferred.

Note that this approach does not correspond to semantic distance or semantic relatedness approaches (typically over semantic networks) that have been used in word sense disambiguation ef-



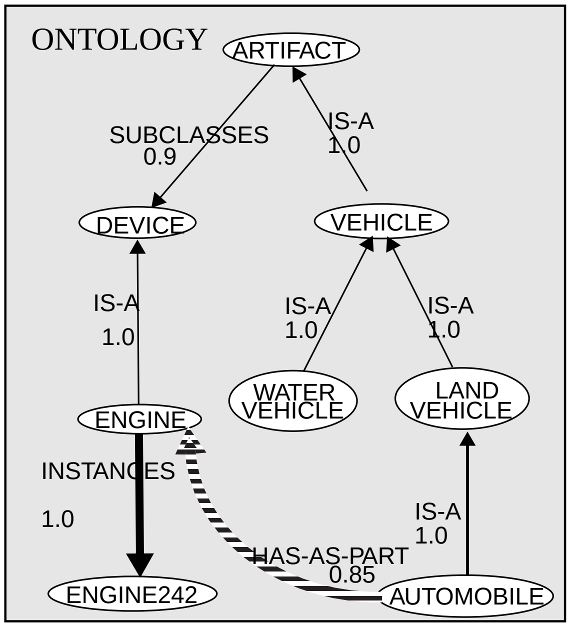


Figure 5B. Two views of example of Ontology, with paths identified with bold arrows

forts; see a detailed discussion of this issue in Section 8 of this distinction.

5.1.2 Search Algorithms

One way of viewing the same cheapest path graph search is as a shortest path problem, where the lengths of the arcs are set to the absolute value of the natural logarithm of the (multiplier) weight:

$$\text{arc-length} = -ln(\text{arc-weight}) \qquad \text{Equation (1)}$$

Since all of the weights are in *(0.0, 1.0]*, all of the natural logarithms are nonpositive, thus allowing minimization of the sums of the absolute values to yield shortest paths (arcs with weights *0.0* are not considered). Performing a standard shortest path search on the converted costs, adding the arc costs, will identify the same path as finding the path with the best (i.e., closest to *1.0*) cost. Well-known complexity results on the shortest-path problem (directed or undirected arcs, nonnegative lengths) identify it as solvable in polynomial time, as discussed below in detail in Section 5.1.3.[1] Because of this isomorphism, throughout this discussion, *best cost* or *cheapest cost* path will be used interchangeably with *shortest path* or *best path*; the figures and text will continue to treat the search as a best-cost problem, particularly since drawing examples with variable-length arcs makes for very confusing topologies, where possible in two dimensions. Figure 5C illustrates what type of topology this perspective on the ontological graph would produce. The nodes that are touching have paths of length *0.0* (equivalent to arc weights of *1.0*), but the graph is perverted by the fact that the nodes do not have null size. The non-zero-length paths have indications of the path length, as computed by Equation 1. If many more non-zero paths were to be added to Figure 5C, it would probably be impossible to render in two dimensions.

The search is invoked with a set of parameters including a *search threshold*, the *source node*

1. See Garey and Johnson (1979).

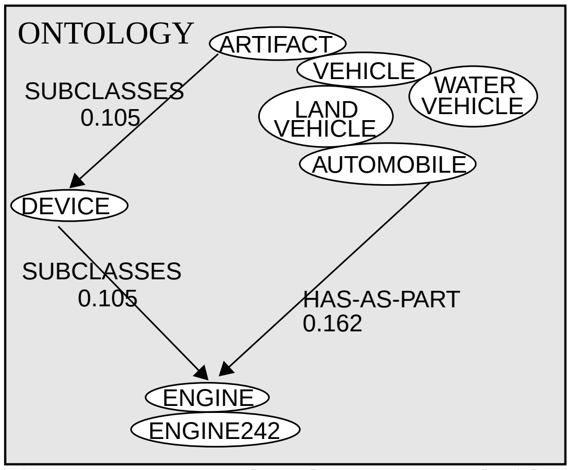


Figure 5C. Same ontology, but represented with different arc lengths for shortest-path search

of the search (typically, the potential filler), the *target node(s)* of the search (typically, the constraints), and an *arc weight determination mechanism*. The search threshold is used to terminate the search by indicating a bound on the acceptable total cost of a path returned by the search.

Another parameter which is given to the search program is the number of allowed answers (i.e., paths resulting from the search); in some cases only the one single best path (or *k* best paths) is required, while in other situations all of the possible paths above the cost threshold need to be returned (to a large extent, this depends on the overall control structure, and whether it follows a backtracking or a multiple-parallel-hypothesis model). The setting of the cost threshold (a matter for empirical exploration) is a function of numerous parameters such as computational power available and the expected literalness of the text (see Section 9 for discussion of metonymy and non-literal text). Given that the overall semantic analysis strategy is a state-space search, the preference of the reading that invoked the semantic constraint check is also a factor; if a reading has low preference already, then the cost threshold for a graph search will be set appropriately very high (no point in pursuing an unlikely interpretation for a reading which is already unlikely). When the search is done in an approximately best-first manner, the fact that the shared data structure scheme could allow other readings to benefit from the same search and possible combination process is not a problem (since the other readings would have a lower preferences anyway, or they would have been attempted already.)

Any of a number of graph search algorithms could be utilized for the purpose of identifying the cheapest path from source node to target node. A best-first heuristic search algorithm is sketched here for illustration. The search proceeds by taking the most promising node, as identified by a heuristic (where “most promising” may be simply as identified by Equation 2 below, i.e., lowest combined cost of path-so-far and cheapest next arc), and examining the node to which the identified arc (from the node at the head of the path) leads. If that new node is the target node, then a shortest path is identified. If at any point a path cost exceeds the cost threshold, it is removed from consideration, because costs cannot improve (i.e., multipliers cannot exceed *1.0*). Details of basic best-first search techniques and shortest path algorithms are well-described in the literature, and will not be presented any further here (see Nilsson (1980), Pearl (1984), van Leeuwen (1990), etc. for example discussions.) The specific heuristic and a comparison of this approach to some of the standard best-path algorithms are discussed in more detail in Section 5.1.3 below.

Other shortest path search algorithms that have been implemented for the application being described include an iterative deepening A*, as described in Korf (1988), and a best-first bidirectional search (akin to the ones described in de Champeaux and Sint (1977) or Nilsson (1980), or a bidirectional version of best-first heuristic searches described in Pearl (1984)), using Heuristic II as the search heuristic:

Heuristic II. Use Equation (2) as the search heuristic in the graph search over the ontology.

where the equation referenced appears as:

$$\mathit{cost}(\text{current-node}) + \mathit{min}(\mathit{weight}(\text{arc}[\text{current-node}, \text{node}_i]) + \mathit{cost}(\text{node}_i)) \quad \text{for all relevant } i \qquad \text{Equation (2)}$$

This heuristic computes scores for nodes by identifying the cheapest (i.e., lowest-scoring) sum of node cost and weight of arc to that node from the current node, over all other nodes in the ontology graph. In practice, the minimum is taken over only those nodes with explicit arcs leading to them, since undefined arcs have infinite weights. The best-first search algorithm uses the node with the cheapest cost as the next node to be explored.

5.1.3 Computational Complexity

As mentioned above, any number of single-source shortest path algorithms could be invoked for this purpose. The first of the algorithms described above in Section 5.1.2 was used most often in the experiments described here. The algorithm can be viewed as an A* version of Dijkstra's well-known Single-Source Shortest Path algorithm (Dijkstra (1959), Aho *et al*. (1974), etc.), optimized by using a (heap structure-based) priority queue and an association list (these two optimizations are similar to those described in Gibbons (1985), using code similar to Sedgewick (1983) or Budd (1994)). The algorithm is implemented with an A* flavor, where the heuristic h^* is a weak one: the combined cost of the cheapest next arc out and current node cost, as described above in Section 5.1.2. So $g(n)\ h^*(n)$ is, in fact, the next cheapest path. Admissibility holds, since the $h^*(n)$ returns a cost that is always ≤ the cost of the path to the goal node from node n.

A number of factors led to the experimental selection of the algorithm used, of which computational complexity order was not the main one, because other factors overwhelmed the complexity order in the average case. But we will discuss the complexity of the algorithm for purposes of comparison and illustration.

The upper bound for graph search algorithms is the Shortest Weight-Constrained Path problem, defined in Garey and Johnson (1979) as:

> Graph $G=(V,E)$, length $l(e) \in Z^+$, and weight $w(e) \in Z^+$ for each $e \in E$, specified vertices $s,t \in V$, positive integers K, W. Is there a simple path from s to t with total weight W or less and total length K or less?

Since we only use an arc weight, but no arc length (or *vice versa*, see discussion above), hence only W or K but not both (so either weights or lengths are all equal), and only positive lengths and weights, we can compute our results in polynomial time, unlike this NP-Complete problem. A number of other constraints on the problem definition, as well as constraints based on the actual data, end up reducing the complexity of the algorithm used to solve this problem substantially.

One such constraint or factor that affects the selection of the algorithms is the sparseness of edges. Algorithms such as those given in Wagner (1976) are optimized for sparse graphs (but may have other constraints that make them inappropriate). The ontology used in the experiments described here is far from being fully connected; viewed as a graph G, the number of nodes n (or $|N|$) is *4742*, while the number of edges $|E|$ is about *14,861*. So we see that $|E| << n^2$ (which is *22,486,564*). In fact, $|E| << n^2/log_2n$ (about 1.8 million; the importance of this will become apparent below).

The ontology graph does not have negative edges at all; in the multiplier approach, this translates to having no factors greater than *1.0*. While most of the efficient algorithms can handle negative edges, none of them can handle graphs with negative cycles efficiently. We are assured of no negative cycles. Other efficient algorithms are rejected because of weights on arcs at all (vs. arcs all having weight *1.0*), or because the set of allowable weights does not consist solely of a small number of small integers (Wagner (1976)).

In the algorithm used here, the "outer loop" effectively considers each edge (at most) once, by exploring each node multiple times until no unexplored edges are found, contributing a factor to the complexity of $|E|$. Inside the loop, the priority queue mechanism[1] (using a binary heap structure) provides the next node at a cost of log_2n, and at a cost of log_2n for adding a node to the queue. Thus the overall complexity of the algorithm is thus $O(|E|\ log_2n)$.

An additional term B (for the maximum branching factor) was ignored above, because it is bounded by a small constant, in practice, not by log_2n as it would be in the general case (by the complexity of an heap priority queue): while generating a new node in the A* search, the "out" arcs are assembled and sorted by cost. In practice, the maximum branching factor for any node in the real ontology is 21, with average of about 2.5. That the graph is not fully connected, but only very sparsely, isn't an arbitrary assumption, but reflects a fundamental design decision of the ontology. All edges in the ontology (except *IS-A* and its inverse) are binary relations that are defined as concepts in the ontology themselves. The theoretical maximum number of edges for a node is limited by the number of relations in the relation portion of the ontology (357 in the actual case). Furthermore, many of the relations are mutually exclusive, so we end up with a severely constrained number for the even worst case. For this reason, and because $n<<|E|$ and $B<<n$, the term $nlog_2B$ is dropped from the overall complexity $O(|E|\ log_2n)$, because $|E|\ log_2n+nlog_2B << |E|\ log_2n+|E|\ log_2n = 2|E|\ log_2n$ and constant terms are dropped from the complexity measure.

The above complexity was calculated for the static arc weight assignment mechanism, described in Section 5.2.1. An additional constant factor affects the complexity for the transition-table-based arc weight mechanism described in Section 5.2.3. In the latter case, the out-arcs for each node may be considered more than once, since the path-so-far for the node may result in various states. Thus, an additional constant factor, $|S|$ accounts for the number of possible states (no more than 20-some for the experiments described here).

For a number of reasons, the search does not run to worst case. The ontology is not a randomly-connected graph, and the searches do not, in fact, tend to reflect random source and target nodes. Furthermore, by using a threshold value to prune expensive paths, the search is con-

1. In the average successful case, the entire graph-search program, including start-up overhead, takes about 40% of the total CPU time when using the priority queue mechanism, as compared to total CPU time when using a sorted list; in cases of search failure, the priority queue implementation takes less than 10% of the CPU time of the sorted list time.

strained, and most edges and nodes do not, in fact, get explored. In the experiments run, the number of nodes explored in successful runs of the algorithm (i.e., ones where a constraint is satisfied or a metonymy is found) averages below 100, and rarely exceeds 300. For searches where no path is found above a threshold of *0.5*, the number of nodes rarely exceeds 1000.

Although a more efficient algorithm, at the limit, could be constructed or selected, it was not useful to do so here. Because the typical runs of the algorithm only explore a small fraction of nodes and arcs, as described above, any algorithm which requires an overhead (for initialization of costs) of size n or $|E|$ was rejected. The best case for a number of other algorithms, including Dijkstra's, is *2n*, while in the algorithm described here it is *B*.

5.1.4 Ontology Search as Abduction

Another way to consider the ontology graph search problem is as cost-based abductive inference problem, as described, for example, in Hobbs *et al*. (1988) or Charniak and Shimony (1994). In this view, the constraints and candidate fillers are the observations that abduction is supposed to explain by constructing a proof (the path from the latter to the former). We assign a zero cost to ontology nodes and treat them as assumptions. The arcs can be considered to be abduction rules with two conjuncts (the from node and the to node), and the arc cost thus becomes the rule cost.

The point of identifying this isomorphism is the discussion of problems and availability of techniques in this other paradigm that may be of use in our context. Charniak and Shimony (1990) identify a fault in the Hobbs-Stickel model of cost-based abduction (Hobbs *et al*. (1988)) in that the costs are essentially pulled out of a hat, and they suggest a probabilistic semantics for the weights that could be used in a discovery procedure. The graph search algorithm described above also relies on a set of weights; the Charniak and Shimony probabilistic semantics could be extended to address the semantics of both of the weight-assignment mechanisms described below in Section 5.2.1 and Section 5.2.3. Charniak and Husain (1991) describe an admissible heuristic for the graph search that is more informative than just best-path-so-far; their heuristic is richer than what is used in out work, but the essential element is the same, namely rely on passing back costs of nodes subsequent to the one being explored. Both the cost-based abduction efforts described above rely on more complex problem sets than required in the current context (including three-valued logics, AND/OR DAGs) so have complexity far worse than what we encounter.

Another area that may merit exploration in future work is a linear constraint satisfaction approach to cost-based abduction, described in Santos (1994). Because of the problem of finding good admissible graph search heuristics, he approaches the cost-based abduction problem by using linear programming optimization tools.

5.2 Determination of Arc Weights

The arc costs are the locus for heuristic knowledge for the search, thus are the core of the entire approach being presented here. The approach relies on the premise that an arc identifies a semantic relationship between the source and the target of the arc, and that the semantic relationships identified by various arcs are not uniform. Indeed, the label on the arc identifies the nature of that relationship. Some relationships are more salient or integral than others; these relationships result in arcs with cheaper cost (i.e., higher values) than the arcs identifying relationships which are more peripheral or incidental (which will be more expensive, i.e., higher costs). Heuristic III identifies this central premise.

Heuristic III. The weight on an arc from node A to node B in the ontology is proportional to the semantic affinity that node A has for node B.

Some of the arcs identify obviously central semantic relationships, such as *IS-A* arcs and the arcs which define metonymic relations (as discussed in Section 9.)

5.2.1 Static Arc Weight Assignment

The simplest mechanism for identifying the weight of an arc is a simple look-up table:

IS-A :	1.0
INSTANCE-OF :	0.99
SUBCLASSES :	0.9
PART-OF :	0.85
...	

Given a particular state in the ontological graph search, the search control structure will request the weight of an arc with a given label; the arc weight mechanism simply returns the value identified with that arc label in the table. In this model, each time an arc type occurs in the ontology, it is assigned the same cost. The acquisition of the weights for the table is discussed in Section 5.3 below. In the case of a bidirectional search, the arc weight mechanism is passed a flag which indicates which frontier is being explored: the one arising from the source node (the candidate filler) or the one arising from the target node (the constraint). The arc weight mechanism then utilizes the appropriate look-up table; the table for the target node frontier is derived from the source node table (used for the basic best-first search described above) by replacing the arc labels with the label of the inverse relationship:

SUBCLASSES :	1.0
INSTANCES :	0.99
IS-A :	0.9
HAS-AS-PARTS :	0.85
...	

Regardless of the viewing direction, the two tables identify that the path from a potential filler up the tree (up only *IS-A* links) to the constraining concept as an ancestor results in a no-penalty (i.e., weight *1.0*) path, thus is to be preferred. For example, the direct path from **BANANA33** to *INGESTIBLE (which would be the preferred path in *the man ate a banana*), would traverse an *INSTANCE-OF* arc from **BANANA33** to *BANANA (at a cost of *0.99*), then up an *IS-A* link to *TROPICAL-FRUIT at cost *1.0*, another *IS-A* link to *FRUIT, and so on, until *INGESTIBLE is reached, at combined path cost of *0.99 * 1.0 * 1.0 *... * 1.0 = 0.99*.

In cases of relaxation of constraints because of unusual circumstances (such as babies eating pennies or pebbles), the path from the potential filler to the constraint may need to change direction and traverse the ontology in the other direction (i.e., down a *SUBCLASSES* arc) to the constraint. For example, as illustrated in Figure 5D, in *the baby ate a penny*, the path from **PENNY33** to *PENNY (across an *INSTANCE-OF* arc at cost *0.99*) would then traverse up the tree over no-penalty *IS-A* links to *ARTIFACT, then down a few SUBCLASSES slots (i.e., inverse direction of *IS-A*) to the constraint, *INGESTIBLE. In this example, the concept *INGESTIBLE has two *IS-A* links leading from it: to *NATURAL-OBJECT to account for things such as bananas, and to *ARTIFACT for such things as Twinkies (pastries), as well as covering many food products somewhere in between.

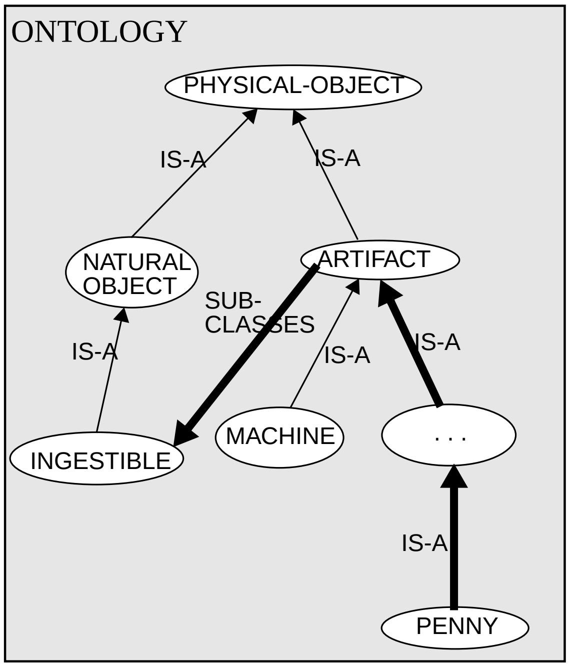


Figure 5D. Path over simplified ontology from *PENNY **to** *INGESTIBLE

As illustrated in Figure 5B, the path that is traversed in cases of metonymic relations (such as *PART-OF*) is computed, resulting a weight (returned from the look-up table, since the path is of unit length) of *0.85*. Various metonymic relations, as well as other relations given as examples in the text below, have similar costs, which are acceptable, but are less preferred to straight constraint satisfaction over *IS-A* links or minimal constraint relaxation. The issue of determining the actual weights to use is a difficult one, and is touched upon in Section 5.3 and other sections below.

The arc weights, in essence, encode various types of rules about the behavior of semantic relations. One might be tempted to try to categorize arcs and their associated weights as metonymic arcs, search-space-reduction arcs, N-N compound arcs, and so on. On the other hand, arcs and their weights reflect semantic relations, and there have been observations in the literature about overlap between the relations that exist for derivational word formation (as reflected in Lexical Rules), in N-N compounds, in regular polysemy, in collocations, and in metonymy (usually these observations are on specific pairs from the above list). There are inventories in the literature for each these phenomena (e.g., Mel'chuk and Zholkovsky (1984), Lakoff and Johnson (1980), Apresjan (1974), Stern (1965), Ostler and Atkins (1992), among others); those inventories can be exhaustively reflected by sets of relations in the ontology. Additional useful arcs are defined in the ontology based on ontology acquisition guidelines, lexicon acquisition experience, or empirical evidence. Thus it seems doubtful that it would be possible to partition the set of arcs with their associated weights.

5.2.2 Dynamic Arc Weight Assignment

In cases where this *static* arc weight assignment is insufficient, it is possible to define a *dynamic* arc weight assignment scheme, where the weight given to an arc is sensitive to the context of prior arcs in the path; thus it is possible to have a weight of *0.9* on the first occurrence of arc FOO in a given path, and weights of *0.2* on subsequent occurrences, for example. Such a mechanism is passed the entire path, in addition to the parameters passed to the basic static arc-weight mechanism described above.

This mechanism restricts the number of times that a particular relation can be traversed in one path. For example, in the case of the metonymic relation of Container-for-Contents, the ontological relation (i.e., slot or arc) that is traversed might be *CONTENTS* on concepts with *CONTAINER as an ancestor. However, in identifying this metonymic relation when talking about picking up soda in a store, we only want to traverse one *CONTAINED-IN* arc once to the container (some sort of *VESSEL-FOR-LIQUIDS) and stop there, instead of traversing another *CONTAINED-IN* arc to the store (which *IS-A* *BUILDING, which *IS-A* *CONTAINER because it can contain things like soda cans). Thus in our dynamic arc weight determination mechanism, once an arc such as *CONTAINED-IN* is traversed at a low cost (say *0.85*), any subsequent traversals become substantially more expensive (say *0.5*). In this case the second transition of that particular arc, although identifying a valid relation, is not desirable in the given context.

In other cases, the second transition is in fact incorrect in principle, not just inappropriate in context — any non-transitive relation cannot be chained in such a way (unless direct evidence for each arc appears in the source). A general example of such a relation is the “is friends with” relation, where even though A is-friends-with B, and B with C, it is incorrect to believe that A is friends with C unless there is evidence to support such a belief. Relations in the ontology which fall in this category include such relations as *USED-FOR-MANUFACTURING* (if system X is for manufacturing machine Y, and machine Y is for manufacturing widget Z, X does not manufacture Z), or SUBSISTS-ON (if *SPARROW-HAWK *SUBSISTS-ON* *SPARROW, and *SPARROW *SUBSISTS-ON* *INSECT, it is not appropriate to infer that *SPARROW-HAWK *SUBSISTS-ON* *INSECT). These sorts of relations exist as slots on entities, and could define the relations that exist in cases of Noun-Noun compounding, for example. The point of the dynamic weight mechanism is to avoid paths with such multiple arcs when not appropriate.

5.2.3 Transition Table Arc Weight Assignment

An implementation of the dynamic arc weight assignment mechanism which also provides clean control is arc weight assignment by a *transition table* mechanism. By using a general dynamic weight assignment mechanism, monotonicity in the graph search paths can be ensured. In addition to the problematic cases described in Section 5.2.2 above, this mechanism avoids excessive change of gradient. For example, after a series of *IS-A* arc traversals, and a *SUBCLASS* traversal, this mechanism would block resuming *IS-A* traversals (or at least at the low arc cost). For example, in Figure 5D, this mechanism would introduce high cost if, at the end of the path highlighted in bold, the *IS-A* arc from *INGESTIBLE to *NATURAL-OBJECT were attempted (assuming a different destination node). This would be accomplished by remaining in one state while the arcs are of one gradient (up, via the *IS-A* links), would transition to a different state upon a *SUBCLASSES* arc (i.e., the change in gradient), from which a transition over an *IS-A* arc would be very expensive.

The transition table consists of rows, labelled by state numbers, and columns, labelled by one or more ontology arcs. A given entry in the table is an ordered pair, <*cost, next_state*>, where cost is the associated return value that reflects the cost of the input arc. Each node on the frontier of the graph search, that is, the last node of each path being explored, would also store the new state in

Table 5E. Fragment of Example Arc-Cost Determination Transition Table

	IS-A	*INSTANCE-OF NAMED-INST.-OF ELEMENT*	*SUBCLASSES*	*CONTAINS HAS-MEMBER*	*PART_OF MEMBER-OF*	...
A	0.999/B	1.0/A	0.85/C	0.9/D	0.9/E	
B	0.999/B	[a]	0.65/C	0.9/D	0.9/E	
C	0.65/B		0.9/C	0.9G	0.9/F	
D	0.999/D		0.85/G	0.9/D	0.3/D	
E	0.999/E		0.85/F	0.3/E	0.9/E	
F	0.3/E		0.85/F	0.3/F	0.9/F	
...						

a. **Note**: Unmarked cells = Default cost (0.4) / same slot

the transition table. A request for an arc cost would be handled by finding the appropriately labelled column for the input arc, given the beginning state. For example, in Table 5E, the path below results in the following chain of transitions and cumulative weight:

```
(-, 1.0)        --INSTANCE-OF-->     1.0/A
(A, 1.0)        --IS-A-->            0.999/B
(B, 0.999)      --IS-A-->            0.999/B
(B, 0.998)      --MEMBER-OF-->       0.9/E
(E, 0.898)      --SUBCLASSES-->      0.85/F
(F, 0.763)      --HAS-MEMBER-->      0.3/F
(F, 0.229)      --IS-A-->            0.3/E
(E, 0.069)
```

This rather unlikely path is correctly scored low for reversing the gradient on the *MEMBER-OF* arcs, and lowered for resuming the gradient of *IS-A* arcs after changing directions to *SUBCLASSES*. Although this convoluted path would not likely be chosen, regardless, the transition table mechanism produces a very low score of *0.069*, as opposed to *0.64*, which would have been produced by the static mechanism (straight look up table as illustrated in Section 5.2 above).

The table given above is for illustration only; the discussion in Section 9.8 below and the accompanying table illustrate one of the transition tables that was built to handle both the general word-sense disambiguation problem and metonymy.

5.3 Acquisition of the Arc Weights

Regardless of the mechanism used by the search to return the arc weights, the knowledge base of weights and the algorithm of picking or calculating them are of critical importance in the graph search and thus need to be meticulously derived. Since weights are used to eliminate or prefer paths through the ontology graph, inappropriate weights can defeat the purpose of the ontology graph search — identifying the most appropriate semantic relation between the candidate filler and the constraining concept.

5.3.1 Manual Arc Weight Acquisition

In the initial stages of this effort, the weight tables used by both the static and dynamic mechanisms (Section 5.2) were manually built. The first step in the manual process involved collecting an inventory of arcs that are known to require special weights:

• Hierarchical links (*IS-A* and *SUBCLASSES*)

• Instantiation links (*INSTANCE-OF*, *NAMED-INSTANCE-OF*, and their inverses)

• Common metonymic links (see Section 9.6)

In addition, a default category was created to handle all other cases of relaxation and metonymy that were not covered by the inventories above, which are not expected to be complete. For each of these, a weight was determined by manual examination of a (small) number of sentences, augmented by expectation of certain sets of metonymic and other non-literal/unexpected expressions in text. This initial set of weights, although built essentially by introspection, was sufficient as starting point for experimentation, and supported a non-trivial amount of word-sense disambiguation and metonymy resolution (see Section 9 and Section 8 for evaluation results).

5.3.2 Automated Learning of Arc Weights by Simulated Annealing

Despite the (modest) success of the manually-determined arc weight set, clearly a method was needed to optimize this process and to automate it for different languages, domains, and text types. A variety of approaches could be used for this process, including machine learning, genetic algorithms, connectionism, and combinatorial or non-linear optimization techniques. During the course of work described here, experiments using simulated annealing were sufficiently successful that other approaches were abandoned.

Given an inventory of the arcs that need special weights (such as the list outlined in Section 5.3.1 above), the task of any of these training approaches is to find the best weight in *[0.0, 1.0]* for each arc such that the highest possible total score is achieved across the entire training/test data set. The individual scores reflect whether or not the ontology graph search, using the set of weights, returns the exactly correct best path. This process can be viewed as an optimization process over *p* parameters, where *p* is the number of arcs (including a “wildcard” arc) receiving special weights.

The training set consists of a number of problem statements coupled with the desired solutions. The problem statements consist of an ontology concept (representing the semantic head of a putative filler of an argument position) and a set of constraints (representing the extended selectional restrictions or slot constraints on the argument position that the putative filler is attempting to fill). The desired solution is a path from the filler to one of the constraining concepts; these paths are manually produced, reflecting the nature of the relationship between the filler and the constraints (such as a metonymic relationship). For example, if a putative filler is an ORGANIZATION, and it is acting in an *AGENT* position, which requires a HUMAN (the set of constraints is just the unary set containing this one concept), the correct path is `((ORGANIZATION -) (HUMAN`

HEADED-BY)), meaning that the organization is actually being used as a metonym for the humans that control it.

The entire training set is compiled from a variety of sources. For each case of metonymy described in Section 9.6, there is at least one training set data element reflecting the fillers and constraints, along with the associated metonymic arcs (Section 9.7) reflected in the correct path. Most of the training examples, however, are collected from examining corpora. An examination of the correct word senses for a text reveals any violations of the literal constraints on argument positions; each such example is then entered into data set, along with a manually-produced correct path. A slightly different process of building the training set is needed for word sense disambiguation, because there not only is the best path important, but also the weight of the best path for the correct set of word senses needs to be better than the weight of the best path for combinations of incorrect word senses.

Simulated annealing is an approximation algorithm for non-linear optimization of continuous parameters, or for combinatorial optimization of discrete variables. Based on work by Metropolis *et al.* (1953), the algorithm was suggested, among others, by Kirkpatrick *et al.* (1983) as "efficient techniques for finding minimum or maximum values of a function of very many independent variables", since which time it has become popular for a range of optimization problems (see, for example, van Laarhoven and Aarts (1987) or Davis (1987)). The underlying metaphor for simulated annealing is a metallurgical process where metal is heated to the melting point then cooled very gradually, allowing optimal (low-energy) alignment of molecules. As as approximation algorithm for optimization, it builds on traditional refinement techniques, but avoids the risk of settling in a local minimum solution, from which iterative refinement cannot escape. By introducing randomization techniques, locally optimal but globally suboptimal energy states can be avoided.

Simulated annealing has not been used extensively in natural language applications. One notable exception is word sense disambiguation work by Wilks *et al.* (1992) or Cowie *et al.* (1992), where simulated annealing is used to approximate the optimal combination of selected word senses, using a Lesk (1986)-like approach of optimizing the intersection of words in LDOCE definitions for each token in the input sentence. Another application of simulated annealing to NLP is work by Sampson (1986) and Selman and Hirst (1987), involving simulated annealing to optimize parsing. Unlike these two efforts, the work described here involves using simulated annealing not at run time, but for training purposes, namely acquiring a knowledge source for use by another algorithm; word sense disambiguation in our approach is primarily based the lexical semantic specifications of the words and the ontology, but the arc weight knowledge source, acquired by simulated annealing, informs the use of those two knowledge sources.

The version of simulated annealing applied here, using public-domain code from the Naval Postgraduate Institute described in Carter (1995), is a straightforward application of the Kirkpatrick *et al.* (1983) algorithm. An "annealing schedule" is set up, where the parameters are first "melted" (essentially randomized) into a high-energy equilibrium, then annealing proceeds. Essentially an iterative refinement process, the annealing involves a number of sub-steps, iterating with the temperature parameter T decreasing at a Cauchy rate (at iteration i) of $T_i = T_{i-1}/(1+0.1i)$:

- Perturbing each parameter at a rate proportional to T
- Measuring the energy state E of the resulting system; in this context, E reflects 1 minus the fraction of paths correctly found by the ontology search over the training data. For example, if 99 out of 100 paths in the training set were correctly found, the energy state E

would be *1.0 - 99/100 = 0.01*.

- If the new parameter set results in a lower energy state (better scores), the parameter changes are accepted.
- If the energy is higher (the scores are worse), the new state may still be accepted, with probability $P(\Delta E) = exp(-\Delta E/k_b T)$. The constant k_b is the Boltzmann constant, and was set to *1.0* for these experiments. So if *rand[0.0, 1.0]* is less than $P(\Delta E)$, the new worse configuration is accepted. This equation simulates the Boltzmann distribution.

The iterations terminate when either an equilibrium is reached (a certain number of iterations without any change in energy state) or a maximum number of iterations are accomplished (here, 400).

5.3.3 Success of Training

The simulated annealing was able to train successfully on the training data used, using a Boltzmann constant of *1.0* and a learning rate of *0.5* when training for 20 parameters, with a maximum cap of *800* cooling iterations, with a cap of *20* adjustments per parameter per cooling iteration. At this setting, the system usually converged to within 95% correctness on the training data. In cases where 100% was not achieved, it was usually pretty easy to manually adjust one parameter to achieve 100%; this one parameter would be stuck in a local minimum very far from the value that achieved success (for example, the parameter might get stuck below *0.1*). This little bit of manual intervention was not completely necessary; with more iterations and a slower annealing schedule, the system was able to achieve 100% precision on the training data. The annealing schedule mentioned above reflected the best trade-off of precision vs. wall-clock time required for the training run.

Below is an example of the results of a successful simulated annealing training run. The inventory of arcs is manually determined (as described above). The last arc is a wildcard, and covers all arcs not inventoried.

```
IS-A                     0.957719
SUBCLASSES               0.771272
HAS-MEMBER               0.931212
PRODUCER-OF              0.447723
PRODUCED-BY              0.876912
INSTRUMENT-OF            0.809355
REPRESENTS               0.796348
HEADED-BY                0.998862
LOCATION                 0.8062
OWNED-BY                 0.754007
MADE-OF                  0.896587
ACTIVITY-FOR-ROLE        0.899341
NAME-OF                  0.971266
AGENT                    0.65
THEME                    0.773466
THEME-OF                 0.609619
SOURCE                   0.799949
AREA-OF-ACTIVITY         0.88442
```

CONTAINS	0.99894
CONVEYS	0.83
ELEMENT	0.978201
HAS-AS-PART	0.969287
LOCATION-OF	0.683668
SYMBOL-OF	0.90624
INSTANCE-OF	1.0
NAMED-INSTANCE-OF	1.0
*	0.695308

Some arcs end up with fairly small weights (below *0.6*), indicating that there weren't any examples of paths requiring that particular arc in the training data used for the training run that resulted in this data set. The results for SDS-building, disambiguation, and metonymy resolution that were achieved using this training method are discussed in Section 8.3 and Section 9.9.

5.4 Issues involving the Ontology Search Algorithm

The Artificial Intelligence search literature suggests a variety of algorithms for traversing a search space, but they rely on having a good heuristic for choosing one path or frontier in the search space over the other. Using Heuristic II in selecting the next node/path to explore is fairly straightforward, and it worked adequately, if inefficiently, in early implementations of this approach. However, if a better heuristic could be found, the ontology search process could be substantially improved.

Thus the focus in improving the search becomes finding some way of determining how "warm" a particular node on the search frontier is to the goal node, i.e., the constraint. Topological closeness over the ontological hierarchy would only help us, in the base case, if all links between nodes were of the same cost, and the ontology could be evaluated as a two-dimensional space over which distance could be easily calculated. However, the variety of relations, the various costs on arcs, and the denseness of graph connections rule out such simple heuristics.

The use of a dynamic arc weight determination algorithm, as in the model in Section 5.2.2, significantly complicates the work involved in the heuristic; entire paths need to be assessed in order to identify a single arc cost, as opposed to just simple lookup. That is why in the transition table approach from Section 5.2.3, a state variable is used to simplify the work in calculating the arc cost. Furthermore, if a bi-directional search is being used, the dynamic arc weight mechanisms complicate matters even further, since goal-frontier path costs aren't absolute, but are subject to change when they meet up with source-frontier paths.

There are some questions that, if answerable at reasonable cost, could contribute to an informative heuristic. If it can be determined that some regions of the ontology are unlikely to be traversed in a successful path between specified nodes, some paths can be determined less likely than their cost suggests; for example, if it turned out to be the case that a path where both end nodes (i.e., source and goal nodes) are *PHYSICAL-OBJECTs is unlikely to cross through the *MENTAL-OBJECT ontology subtree, certain paths could be down-graded in likelihood. Similarly, there might be arcs which are unlikely to appear in successful paths involving certain other arcs, which could either contribute to the heuristic or to the weight determination mechanisms.

Another potential contributing factor to developing a good heuristic is the SALIENCE facet in ontological concepts. It is reasonable to assume that slots with higher salience are more likely to define a semantic relation between two concepts than low salience slots, thus helping prefer possible paths over others.

Each of these ideas could contribute to the development of a good heuristic; experimentation with possible heuristics and review of appropriate paths (perhaps manually selected) that account for the semantic relation between concepts, based on a corpus, will be involved in developing such a heuristic.

The number of returned paths is an issue that could increase the effectiveness of the algorithm. If the overall control strategy of the semantic analyzer supported multiple parallel hypotheses, the algorithm, described here could easily be parametrized to return the *k* best paths for any search. In the multiple microtheory model of processing, as briefly described in Section 1.4, various other microtheories would contribute to the structure and preference values of the alternative hypotheses, instead of relying on the just the one dynamic knowledge source, described here, to perform complete semantic disambiguation.

6. Semantic Dependency Structure Building as a Search Problem

The semantic interpretation process (specifically, the process of building a *semantic dependency structur*e or SDS) is treated as a somewhat abstracted state-space search in this section. In this view, each state represents a possible reading being pursued, where a reading consists of a) the partial (or complete) TMR configuration, b) the remaining lexemes not yet processed, and c) the cumulative *preference* cost for the reading to date (see below for definition). The goal of the search is to find the minimum-cost state which satisfies final well-formedness conditions (i.e., no remaining lexemes, and connectedness of all structures in the TMR configuration). The operators for traversing the search space are a) *instantiation* (of concepts or non-ontological TMR structures) as prompted by lexemes, discussed in Section 6.2, and b) *combination* of these structures to form appropriately connected structures, as discussed in Section 6.3. The application of the combination operator is guided by a set of heuristics, of which the central one is the ontological graph search operation (see Section 5.1 for discussion).

In general, the goal of the SDS-building process is to find the most appropriate semantic interpretation of the input text. Candidate readings are ranked according to their *preference* value, a cumulative measure of evidence or likelihood that is used to order competing interpretations (the use of this term is different from its familiar meaning introduced by Wilks (1975b)). If the assignment of preferences by the search process is appropriate, then the interpretation with the highest preference value at the end of processing should indeed be the one which human translators would choose. Preference values are used by the search heuristic both for pruning paths with low preferences, and for guiding a best-first search method (see Section 6.5). The preference in the current implementation is a value in the interval *[0.0, 1.0]*, with adjustments to the preference made by a multiplier (or by root-mean-square in the MIKROKOSMOS implementation, described in Beale (1997)).

The discussions below will discuss the SDS-building process as a search, considering the search space, the search operators, etc. from a generalized or abstracted perspective, while detailed discussions of the application of the combination and instantiation operators as semantic interpretation (SDS building) can be found in Section 7. Discussion of efficiency in the search is delayed until Section 7.

6.1 Search States

The search space consists of states σ which are triples: (T, ι, Π). Π is the cumulative preference (accumulated over the path-so-far) score or cost for the path through the search space represented by the current state, in the interval *[0.0, 1.0]*, with *1.0* being the most preferable and *0.0* a rejected reading. The score is calculated by applying a multiplier at each branch in the search tree (i.e., the application of one of the two operators), where the multiplier is less than *1.0* if the application of the operator is less than preferred, and equal to *1.0* if the operator's effect is very probable or appropriate (examples throughout the text below).

The variable ι represents a structure which contains the input lexemes which have not yet undergone the instantiation process — this is the remaining input string. In practice, this structure is carried not in list or string form, but in the LFG f-structure form which the syntactic parser produces. In representing a state in the search space, either the symbol ι will be used, or a string such as $\iota_3\iota_4\iota_5\iota_6\iota_7$ may be used to discriminate the individual lexemes of which ι is comprised.

The T is a TMR configuration, consisting of zero or more unconnected or partially connected structures. Each structure is in the form of a directed graph, where the nodes of the graph represent frame structures. The structures may be either ontological or non-ontological, as defined in Section 3.3. Both the ontological and the non-ontological structures in the text arise from one of two sources: either lexical triggers from one of the lexemes in the input structure ι (i.e., they are instantiated from the **`LEX-MAP`** definition in the **`SEM-STRUC`** zone of the lexical entries of the word sense of a word in the text), or from other knowledge sources (including discourse structure interpretation/expectation mechanisms) or processing specialists (microtheories).[1] Thus an application of the instantiation operator produces an additional graph (consisting of one or more nodes), and the combination operator either adds arcs between nodes already in the same graph, or links together two or more graphs into one by the addition of one or more arcs or the replacement of a node in one graph by a node from another graph, thereby connecting them. In representing a state in the search space, either the symbol T will be used to designate a TMR representation, or a set notation, such as $\{T_7, T_8\}$, may be used to identify the structures which comprise the TMR T.

6.2 The Instantiation Operator

The discussion of the instantiation operator as it applies to semantic analysis (SDS building) is found in Section 7.3; this section treats instantiation as an operator over the search space. As guided by a traversal of the remaining input ι for the current state, the instantiation process identifies the next appropriate lexeme to undergo the instantiation process, removes it from the remaining input ι, and then retrieves the contents of the **`LEX-MAP`** zone of the lexeme's lexical entry (as specified in Section 4). The contents of that zone will be a combination of zero or more of each of three types of data structures.

- *Ontology Nodes* (Ontological concepts). A concept can be thought of as a node from the ontology (as defined in Section 3.1). Considering the *IS-A* arcs alone, the graph is a directed acyclic graph, specifically a tangled tree. Since each node is actually a frame, the tangled tree provides for (multiple) inheritance of frame slots. Additionally, however, there are arcs with numerous other labels interconnecting nodes in this graph; these arcs identify other relations that may exist between the two concepts, and correspond to slot names; the nodes reached by the arcs correspond to fillers of slots. Figure 6A illustrates what a (small) ontology may look like.[2] A **`LEX-MAP`** reference to ontological nodes may appear as illustrated in Figure 6B, which illustrates a template for instantiating a TMR fragment, as might be found in the lexical entry for a particular lexeme. Note that information may be added (locally) to what was presented in the ontology in Figure 6A. The arc on the left identifies a *COMPOSED-OF* relation to a node which has an unknown ontological type, that is, it is unknown at this point what the filler of the slot will be, since that information

1. The non-ontological structures which arise from lexical triggers behave similarly to the ontological ones in the search (i.e., the basic instantiation and combination process), and will be so treated below in the appropriate discussion in Section 7.
2. These figures will use the following symbols: Y to designate nodes from the ontology (with an arbitrary letter as a node identifier or name), ● to designate instances of nodes from the ontology (typically followed by an identifier such as **D3**), □FOO to designate non-ontological TMR construct nodes, ■ to designate non-ontological TMR node instances (also followed by an identifier, typically of the form **FU7**), and △ to indicate as-yet unresolved $VARs, i.e., links to/from unspecified nodes. Additionally, the following conventions for arrows will be used: solid arrows ⟶ for *IS-A* links, grayed ⟶ for *INSTANCE-OF* links, and broken - - - -▸ for other ontological arcs.

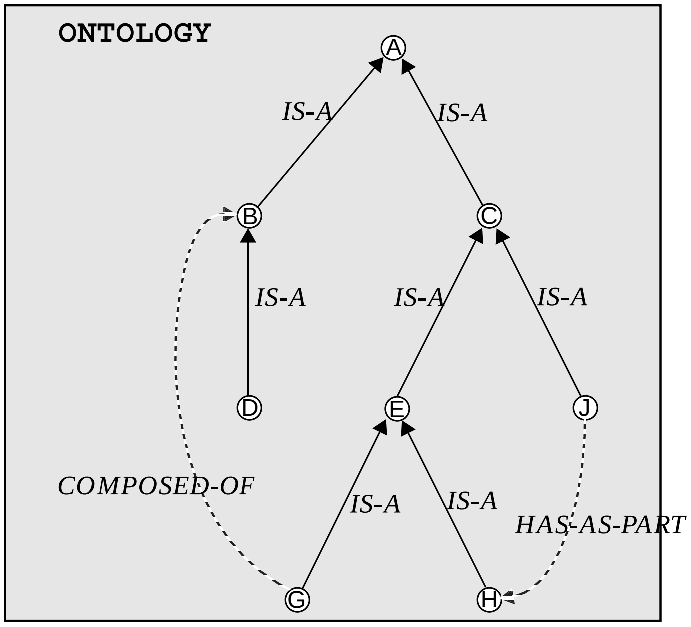


Figure 6A. Example Ontology

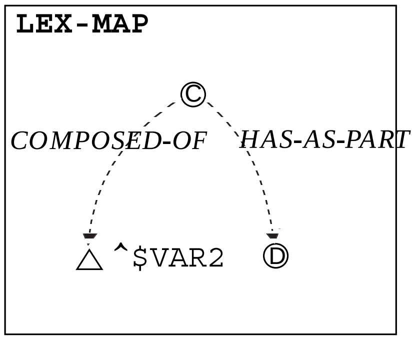


Figure 6B. `LEX-MAP` reference to ontological node

would be specified in the **`LEX-MAP`** of some other lexeme. So in this **`LEX-MAP`**, the `^$VAR2` acts as a place-holder until the combination process (below) replaces it appropriately.

- *Ontology Arcs*. In some cases, instead of calling for the instantiation of a concept (or non-ontological TMR construct), the **`LEX-MAP`** template or specification calls for the instantiation of an arc alone, or an arc with a termination node, but without a specification of the origination node. Note that in Figure 6C, there is no origination node, just the place-holder `^$VAR1`.
- *Non-ontological Nodes*. As described in Section 3.3, the representation language TMR includes structures which are not derived from the ontology. These structures also correspond to nodes, which, however, are not part of the ontology graph (Figure 6D).

The contents of **`LEX-MAP`**, once retrieved from the lexical entry, then undergo the instantiation proper. The process of instantiation involves producing an *instance* of each of the nodes in the

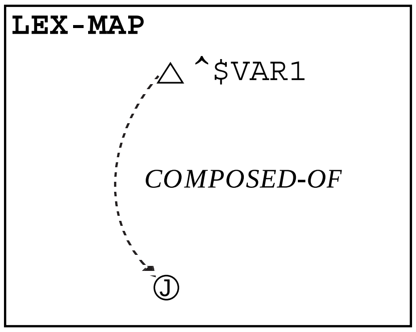


Figure 6C. Origination-node-less `LEX-MAP` specification

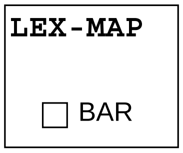


Figure 6D. Non-ontological node reference

retrieved combination of structures. An instance is a structure with a unique identifier which represents a specific single case of the node; the concept in the ontology essentially acts as a template, which can be viewed as a type definition for producing another node which represents a particular instance of the concept in the current discussion. These instances are identified by concatenating a unique number to the name of the concept from which the instance was produced: **FOO3**, **G89**, **A1**, etc.

If the **`LEX-MAP`** of a lexeme ι_3 contained the contents of Figure 6B, Figure 6C, and Figure 6D, then the result of instantiation would produce instances (for example) **C2**, **D4**, **J7**, and **BAR3**, represented in Figure 6E.

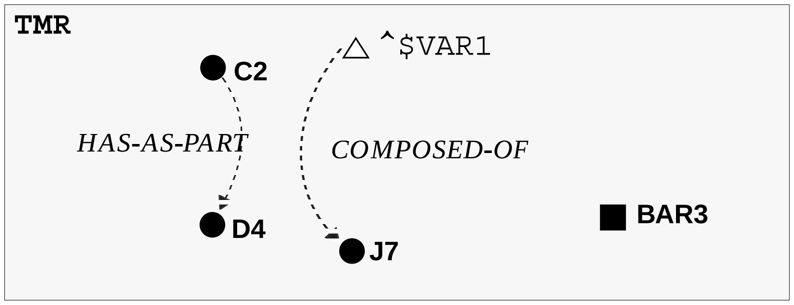


Figure 6E. Example of instance structures produced by instantiation.

The process of instantiation results in the addition of an (as-yet) unconnected graph to the TMR representation graph. So if a search-space state representation is:

$(\{T_2, T_3\}, \iota_3\iota_4\iota_5\iota_6\iota_7, \Pi)$

(where $\{T_2, T_3\}$ is a set notation representation of the elements of the TMR T, and $\iota_3\iota_4\iota_5\iota_6\iota_7$ is the remaining input ι represented as a concatenation of its components, i.e., words or morphemes) then the application of the instantiation operator may produce

$(\{T_2, T_3, T_4, T_5, T_6\}, \iota_4\iota_5\iota_6\iota_7, \kappa\Pi)$

with the changes to the TMR represented in Figure 6F.

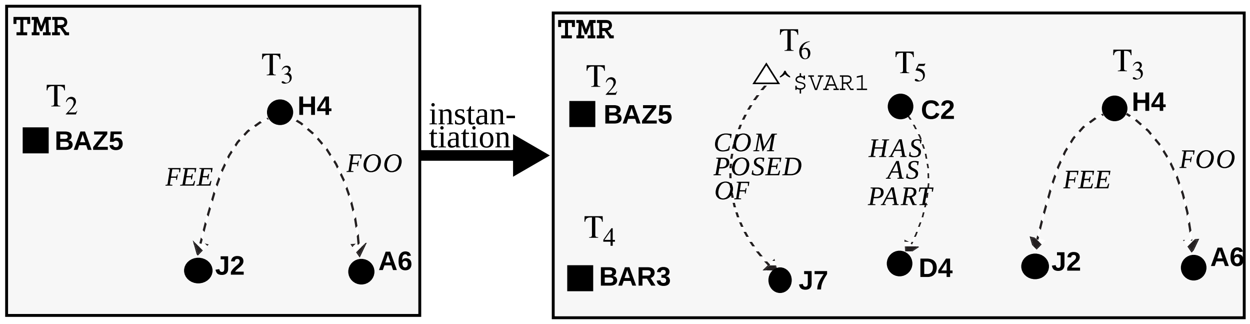


Figure 6F. Example of instantiation operator with input and output TMR forms.

Those nodes in the TMR representation which were instantiated from ontological concepts have *INSTANCE-OF* links back to the nodes from which they were instantiated. These arcs are "para-ontological", and essentially provide the link from a TMR representation to the ontology, as depicted in Figure 6G. The ontology and the TMR are depicted as different planes, and the *IN-*

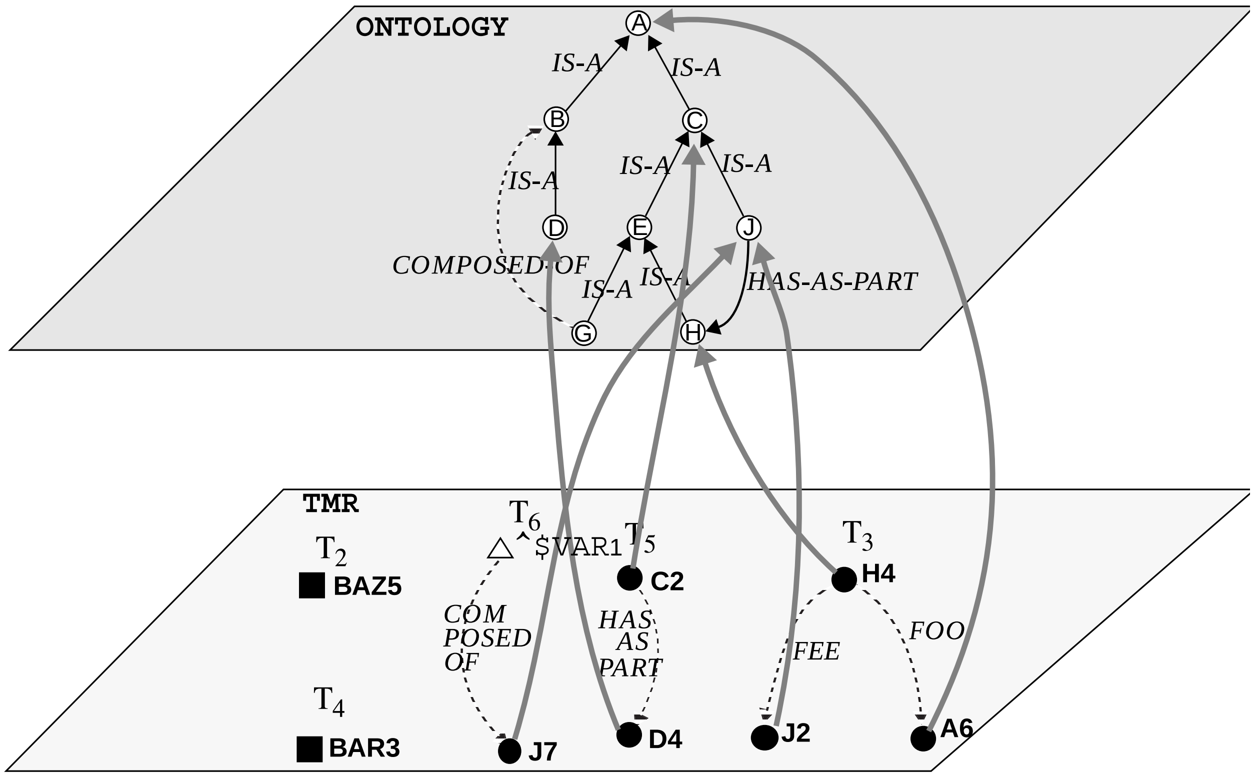


Figure 6G. Interrelationship of TMR structure and the ontology where concepts are defined (shaded arcs are *INSTANCE-OF* links).

STANCE-OF links as directed arcs between nodes on the two planes.

The instantiation operator always succeeds, and either increases the cardinality of the set T or it doesn't affect it (if the particular ι_i to which the operator was applied had a null **LEX-SEM**). The operator can no longer be applied to a search-state representation (T, ι, Π) when ι = ε (empty string). An application of the instantiation operator produces at most one arc in the state-space search tree. In other words, the application of this operator is deterministic, in that it can only produce one result on a given lexeme, regardless of the context.

6.3 Combination Operator

The application of the combination operator takes two TMR fragments (either the result of instantiation or the product of a previous application of the combination operator), and attempts to produce a single, larger TMR fragment. This is the core of the SDS-building process, as discussed in Section 7.4.

There are preconditions on the application of this operator which are described below. The operator takes as input any two TMR fragments, call them T_i and T_j (such as those illustrated in Figure 6F). The two subsections below discuss the two conditions which trigger the application of this operator, thereby identifying how the two input TMR fragments are identified, along with (possibly) an indication of the instance and slot through which the two fragments are to be combined. This operator does not affect the element ι of the search-space state; however, it may modify the value of Π. (See the discussions in the subsections below for discussion of the effects on T.) The application of this operator may fail, thereby terminating that path in the search space.

6.3.1 Triggered by Syntax/Semantics Interface

The locus, and therefore the two input TMR fragments, can be explicitly identified by the syntax/semantics interface: the appearance of a `^$VAR` in one of the elements of T licenses the application of the combination operator. Essentially, the combination operator's job is to replace occurrences of `^$VAR` with elements from the set T. The set T may be reduced in cardinality, with a consequent modification (typically an increase) in the size/structure of one or more elements of the set T. The operator essentially replaces a `^$VAR` in one element of T, say T_i, with a node (typically the `ROOT`) of another element of T, say T_j. Therefore T_j is imbedded in T_i, and is no longer unconnected, so is no longer a separate element of T (in Figure 6H below, T_7 replaces the `^$VAR` of T_8).

The application of this operator is guided by a heuristic (essentially the syntax-semantics interface) which identifies the TMR structures to which the `^$VAR` pointers are bound.

Heuristic IV. The syntax/semantics interface identifies the correlation between elements in the syntactic argument structure and semantic dependency structure by means of parallel variable structures.

In the syntactic parsing process, (locally) prior to the initiation of the state-space search, the `$VAR` are bound to other lexemes according to the **SYN-STRUC**s (see Section 3.4.2). In the example implementation of this approach, a table is kept with pointers from each occurrence of the `$VAR` to their syntactic references. The ^ operator then "dereferences" that binding, retrieving an element ι_i, and identifies which element of T, say T_j, was produced by the application of the instantiation operator to that symbol ι_i. Section 7.4 below provides examples of this syntax-semantics interface and the way it selects the meaning structures (TMR fragments) to be combined.

There is a precondition on the application of this operator. This process cannot apply if the element of T which is to be imbedded, say T_i, still has any `^$VAR`s which haven't been resolved yet (via the combination operator). Essentially, this enforces an element of bottom-up processing on the semantic analysis process; depending on the semantic dependencies, this may not necessarily correspond to processing lexemes bottom-up in the syntactic parse tree, however. Care is taken to avoid dependency loops in processing scheduling if a series of two or more elements of T have interdependent references (i.e., the resulting structure T has cycles).

In Figure 6H, the TMR representation on the left may represent the T from the search-space

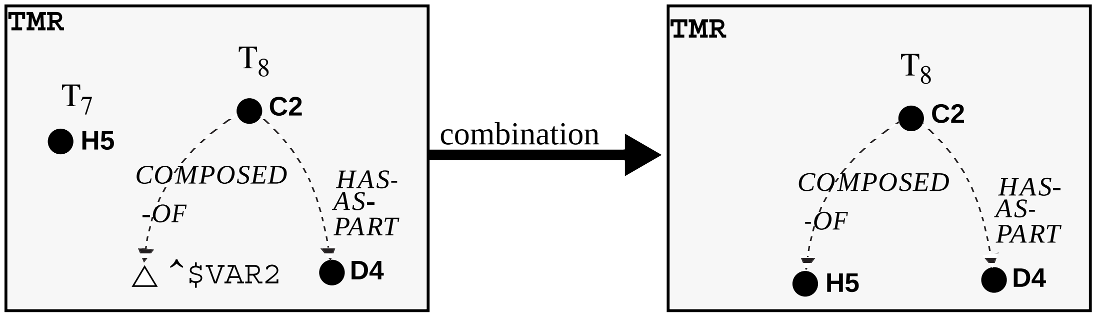


Figure 6H. Example of combination operator with input and output TMR forms.

state representation

$(\{T_7, T_8\}, \iota_3\iota_4\iota_5\iota_6\iota_7, \Pi)$

The application of the combination operator may produce the state

$(\{T_8\}, \iota_3\iota_4\iota_5\iota_6\iota_7, \kappa\Pi)$

with the TMR represented as on the right in Figure 6H. Notice also that the input string is not affected, and that the preference factor Π is affected.

In cases where the ^$VAR appears at the source of an arc in the **LEX-MAP**, the concept instance inserted into the variable position becomes the head of the structure. This situation arises when the meaning of a lexeme is an attribute value of some other lexeme, or a relation between two other lexemes (see Section 4 for examples of such **LEX-MAP** representations, and Section 7.4 for examples of how this contributes to the SDS-building process). Figure 6I illustrates the change

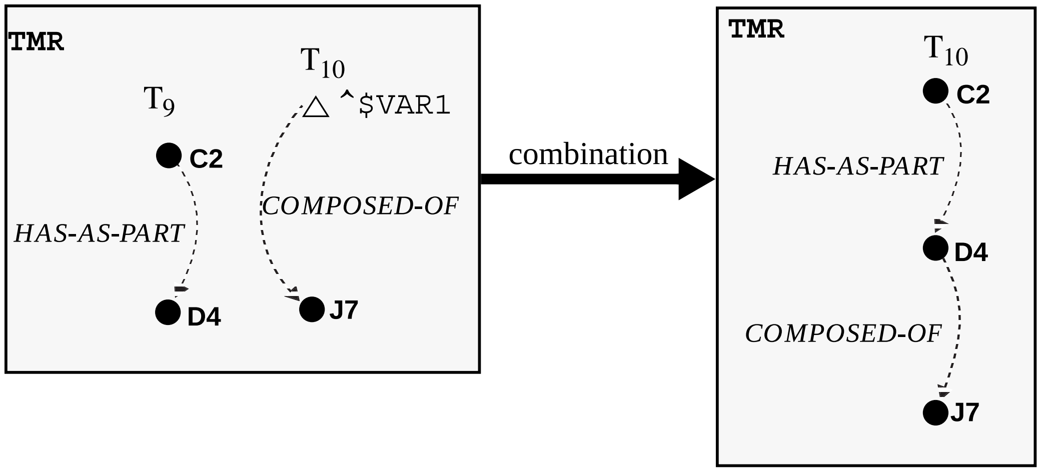


Figure 6I. Example of combination operator, in the case of an origination-node-less structure.

in the TMR which results from the successful application of the combination operator. In this example, the `^$VAR1` is resolved by the combination operator to be the node **D4**. Thus the arc which had originated from `^$VAR1` now originates at **D4**. Note that the resulting `TMR` representation has one structure.

The heart of combination procedure is determining the factor which affects the cost or preference term Π of the search state. This is accomplished by the other major search process in the semantic analysis procedure, and that is the ontological graph search process, described in Section 5.1. This procedure directly affects the combination operator, in that if the ontological graph search process returns a preference factor of *0.0*, the combination is disallowed and the application of the combination operator fails. There is a threshold factor which may be set which also causes the application of the combination operator to fail if the resulting preference factor (e.g., .9Π) falls below that threshold. This effectively terminates or prunes some paths in the search space traversed by means of the combination and instantiation operators.

The ontological graph search attempts to find the lowest-cost (i.e., highest preference) match between the structure being inserted in the stead if the `^$VAR` and a) the constraints that the receiving structure may have on the legal filler of the slot in the case illustrated in Figure 6H, or b) the constraints that the node which is replacing the `^$VAR1` may have on its substructure (i.e., slots and fillers), in the case illustrated in Figure 6I. For discussion of the constraint satisfaction search process see Section 5.

6.3.2 Licensed by Underlying Syntactic Cues

There are cases where the semantic combination operator is triggered to apply in the absence of the overt trigger `^$VAR` from the syntax-semantics interface. The application of the operator may be licensed by a particular syntactic construction in the parse tree from which the lexeme sequence ι and the `$VAR` bindings stem. The clearest and most prevalent syntactic construction occurring in English which licenses this application of the combination operator in this manner is the Noun-Noun compound.

Essentially, after instantiating the lexemes in question, this case corresponds to a TMR structure where the origination and termination nodes of an arc are known, but the arc label is unknown (in terms of frame structure, this corresponds to the frame name and the filler being known, but the slot name is not.) Figure 6J illustrates this case. Although the two TMR structures (T_{12} and

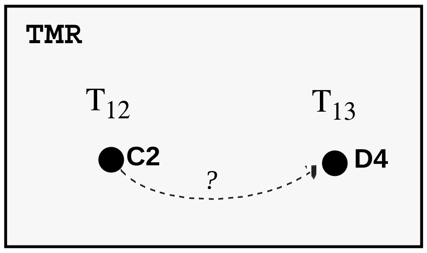


Figure 6J. Example TMR with unlabeled arc.

T_{13}) are shown linked by an arc, since the arc label is unknown, the two structures are still considered to be distinct.

That the two concepts are related is inferred from the fact that they participate in a syntactic relationship which implies such a semantic relation. In the case of English, furthermore, we know that in most cases N-N compounds are pair-wise right-headed (the few that are left-headed are fixed expressions and are therefore entered into the lexicon intact, eliminating the need for this process); this translates, in the semantic dependency structure, to the selection of the right element's meaning representation as the head, and the left element's meaning representation as the filler of the (as-yet unknown) slot. Thus in the example in Figure 6J, if the compound is $\iota_6\iota_7$, ι_7 produced T_{12} and ι_6 T_{13}. This issue is independent of the bracketing question for compounds of length greater than two.

The same ontological graph search process described above is also central to this process, and it provides even more critical information: the name of the arc relating the two concepts. Additionally, as before, it provides us with the cost (thus the factor of Π). Figure 6K illustrates how the

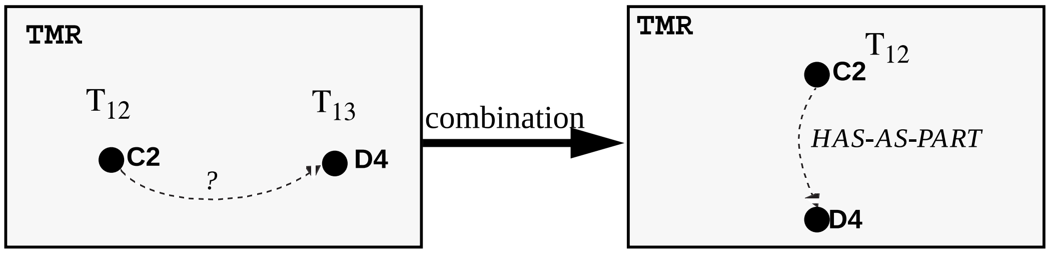


Figure 6K. Example of combination operator, in case of unlabeled arc.

application of the combination operator might affect the TMR structure of

$(\{T_{12}, T_{13}\}, \iota_3\iota_4\iota_5\iota_6\iota_7, \Pi)$

to the TMR structure of the search-space state representation

$(\{T_{12}\}, \iota_3\iota_4\iota_5\iota_6\iota_7, \kappa\Pi)$.

The ontological graph search returns a path from the filler to the constraining concept, which is the meaning representation (TMR fragment) for the right element of the compound. The inverse of the last non-hierarchical (i.e., not *IS-A*, *INSTANCE-OF*, *INSTANCES*, or *SUBCLASSES*) arc in the path identifies the slot name. For example (using Figure 6K), if the returned path is something like

```
(D4   INSTANCE-OF  -->     D)
(D    IS-A         -->     H)
(H    PART-OF      -->     B)
(B    IS-A         -->     C)
(C    INSTANCES    -->     C2)
```

then the slot that relates the two is *HAS-AS-PART*, since it is the inverse of *PART-OF*. The ontological graph search is invoked with a slightly different set of weights for the weight determination mechanism; see Section 11.1 for some speculation on the N-N compounding issue and processing strategies.

6.4 Search Parameters

The search described may begin from the initial state σ_0

$(\{\}, \iota_1\iota_2\iota_3\iota_4\iota_5\iota_6\iota_7, 1.0)$

in the case where the input string consisted of seven lexemes. A final state σ_{final} of the search may be

(T, ε, Π)

where $T \neq \{\}$, ε is the empty string, and Π is a preference cost where $\Pi > 0.0$ necessarily. The TMR T will, in fact, have cardinality 1: $|T| = 1$[1]. These conditions are the final well-formedness conditions.

A search path may terminate at a state σ_{allow} where $|T| > 1$, $\Pi > 0$, and $\iota = \varepsilon$, and the state may be termed *allowable*; an allowable state is one to which the combination operator cannot apply successfully (the instantiation operator can not apply because $\iota=\varepsilon$). Allowable states are considered only if there are no final states, in which cases the best allowable state (where best means greatest P and *min*($|T(\sigma_{allow})|$), that is, the smallest number of unconnected structures in the TMR T. The allowable states would be passed to processes outside of the state-space search algorithm for further processing (i.e., ill-formed input handlers or other microtheories).

State well-formedness conditions also include $\Pi \geq$ *threshold*, where the threshold may be set anywhere in the interval *(0,1]*.

6.5 Search Flow and Control Structure

The overall search space traversal process is illustrated in Figure 6L below. Note that there is no explicit ordering of application of the operators. The combination and instantiation operators are free to apply to any state which meets the preconditions of that operator, described in the sections above. If the preconditions are met, the application of the instantiation operator is always successful, and results in the current state originating a single arc to a single next state, whereas the application of the combination operator may fail, or may result in one arc (or more than one arcs) to a next state, as exemplified by Figure 6M. Of course, any given implementation will impose constraints on the traversal of the search space, namely the overall control structure.

In the diagram, the blocked application of operators (where preconditions are not met) is not represented. Also, this diagram only represents possible paths to the final preferred paths, therefore does not represent proliferation of next states as a result of the combination operator (or the instantiation operator, in cases where lexical ambiguity is not resolvable at a syntactic level); Figure 6O and Section 6.5.2 illustrate the proliferation of search paths. In Figure 6M, however, notice that even though there are multiple possible paths in the search space, they all reach the same final state. In other examples, where certain paths are pruned because of thresholds being exceeded, the judicious selection of paths (i.e., node expansion) can result in different efficiencies in the search. The search heuristic attempts to chose a best-first path through the search space, otherwise it is guided by a bottom-up left-to-right approach (which best matches the preconditions); in this case, the left-most path in the search space is the one that would be followed by the implementation of this approach. All the paths shown in Figure 6M are equally efficient, because all non-successful paths are not shown.

6.5.1 Flow of Control

The overall control structure of the state space search is open. In fact, what is described above is more of a theoretical description of the main elements of SDS-building than a description of a

1. The notation |T| is being used to indicate the cardinality of (or number of disjoint graphs in) T

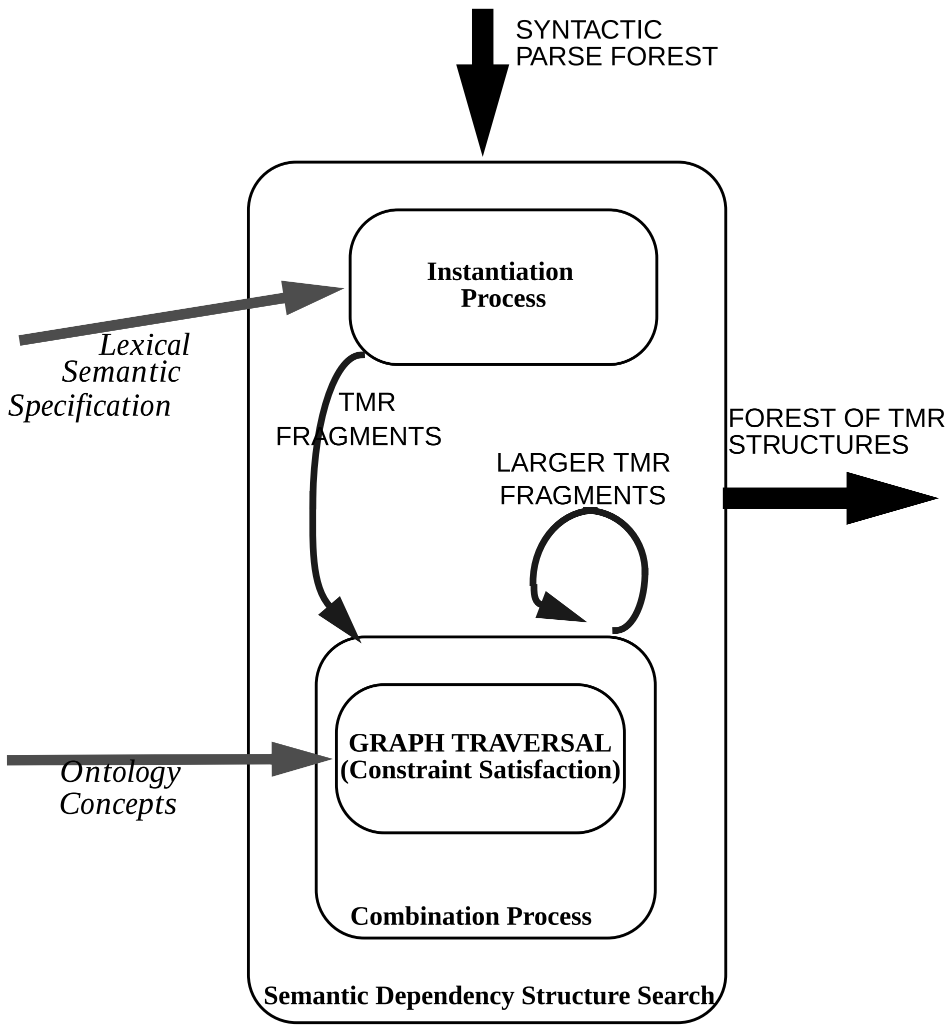


Figure 6L. Data flow and process interaction in the SDS-building search

specific method. Above, we described the initial state, search operators, and the halting conditions, and leave it to particular implementations to determine appropriate control structures and strategies. The overall framework presented here does, however, impose some constraints on the control flow. Due to the density of lexical information and the use of syntactic structure to inform binding of the variables, a certain element of bottom-up (or data-driven) processing is inevitable. On the other hand, other top-down expectations are introduced by linguistic assumptions and assumptions about human communication; these assumptions informed the definition of the TMR format and introduced the expectation of a proposition, of aspectual information, of stylistic information, etc. for each sentence in the surface structure. Any implementation would necessarily reflect those two constraints.

The first implementation for the framework described here, called the DIANA project at CMU/CMT, involves a blackboard-based architecture and a recursive descent through the syntactic parse tree. Each possible instantiation or combination opportunity is represented by a process, which is entered into the process queue of the blackboard control structure. Each process has preconditions, mirroring the preconditions on the application of the two operators, as outlined above. If the scheduling of the process queue were non-deterministic, any of the paths in Figure 6M

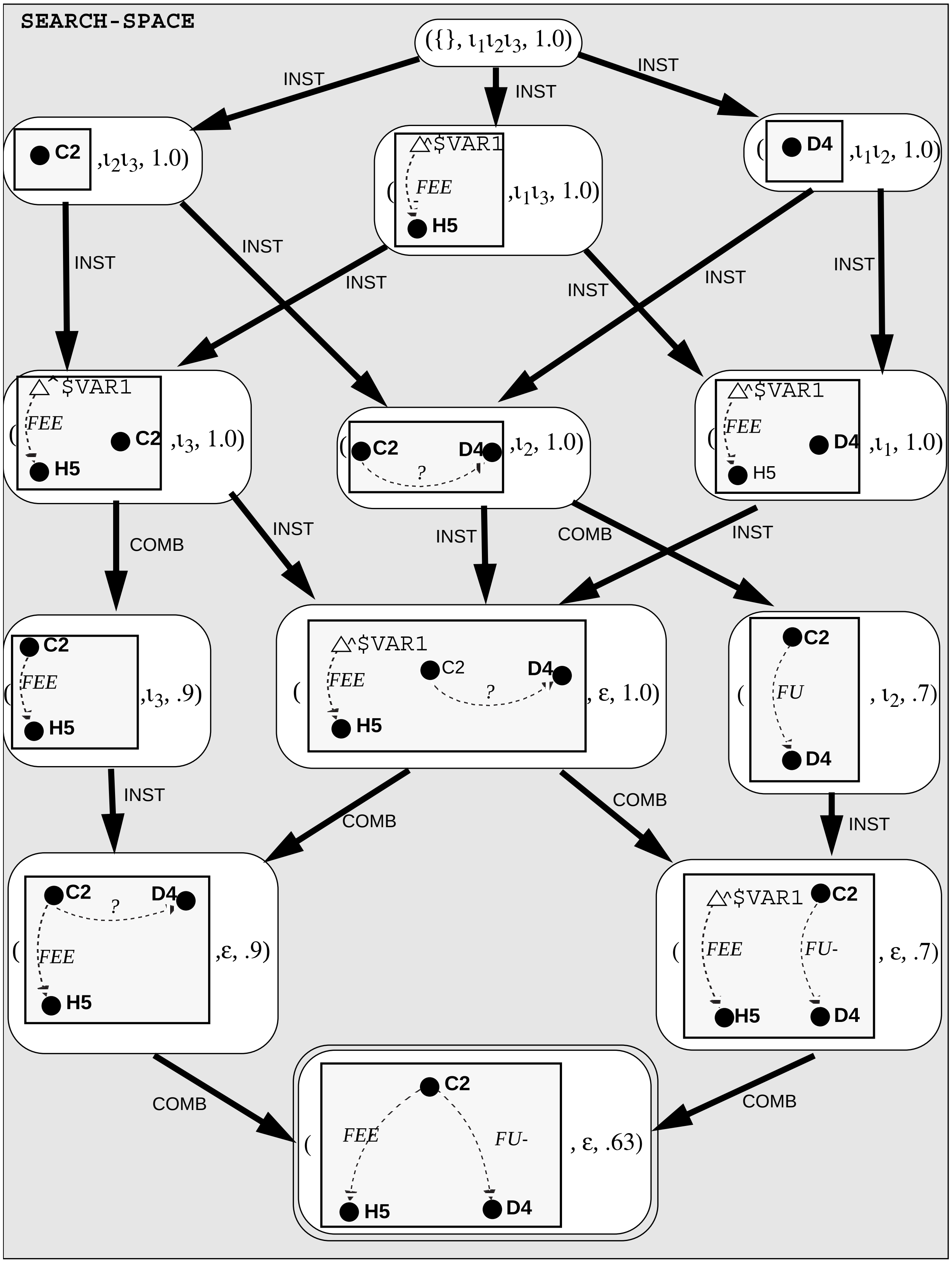

SEARCH-SPACE
({}, ι1ι2ι3, 1.0)
INST
C2 ,ι2ι3, 1.0)
^$VAR1 FEE H5 ,ι1ι3, 1.0)
D4 ,ι1ι2, 1.0)
^$VAR1 FEE C2 H5 ,ι3, 1.0)
C2 D4 ? ,ι2, 1.0)
^$VAR1 FEE D4 H5 ,ι1, 1.0)
COMB
C2 FEE H5 ,ι3, .9)
^$VAR1 C2 D4 FEE ? H5 , ε, 1.0)
C2 FU D4 , ι2, .7)
C2 D4 ? FEE H5 ,ε, .9)
^$VAR1 C2 FEE FU- H5 D4 , ε, .7)
C2 FEE FU- H5 D4 , ε, .63)

Figure 6M. Example search space for SDS-building, showing multiple possible paths to the goal state.

would be possible. This strategy produced multiple possible TMRs, each with an associated preference value ranking them.

The second implementation of this framework was part of the MIKROKOSMOS project at NMSU/CRL and DoD. The semantic analyzer was implemented using a branch-and-bound/constraint-satisfaction/solution-synthesis approach, as described in Beale (1997). This implementation focussed significantly on the problem of the very large search space defined by the framework, and produced one best TMR reading for each input, also with a preference value.

6.5.2 Proliferation of Search Paths

The application of the combination operator in cases such as N-N resolution (but actually in most other cases as well), the ontological graph search process may suggest a number of possible arcs, with different costs or preferences. This may result in a number of different resulting TMRs T and costs Π; this would spawn multiple nodes and paths in the state-space search process. Fig-

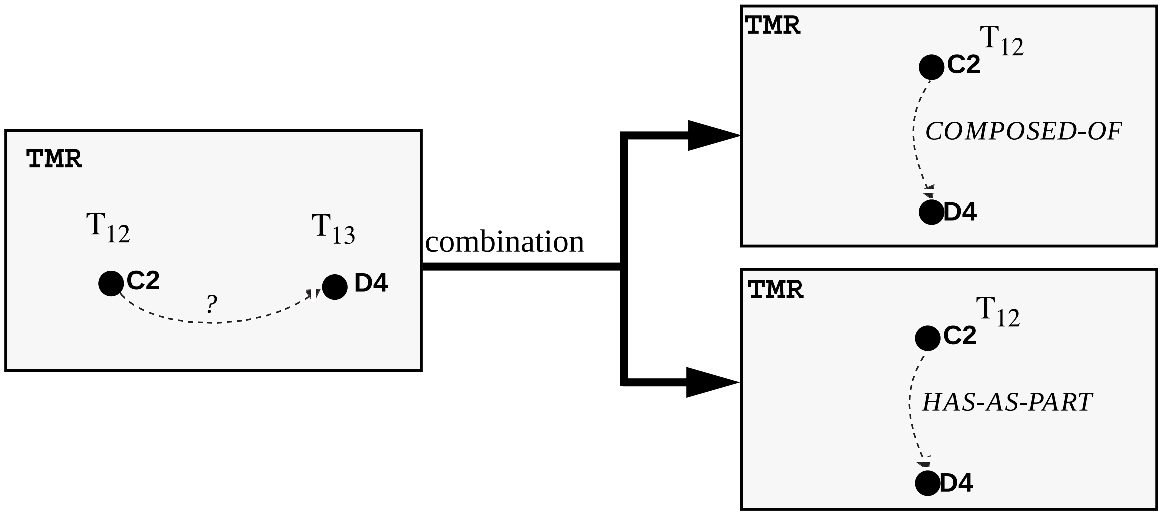


Figure 6N. Example of combination operator, in case where alternate TMRs result from identifying arc label.

ure 6N illustrates a possible result of the application of the combination operator to the TMR state represented in Figure 6J. If the initial search-space state representation were

$(\{T_{12}, T_{13}\}, \iota_3\iota_4\iota_5\iota_6\iota_7, \Pi)$

the resulting state representations might be

$(\{T_{12}\}, \iota_3\iota_4\iota_5\iota_6\iota_7, .9\Pi)$ and $(\{T_{12}\}, \iota_3\iota_4\iota_5\iota_6\iota_7, .85\Pi)$.

Note that the two structures T_{12} would be different in the two states. Typically a threshold would restrict the maximum number of paths which the ontological graph search process may return (this operates in addition to the preference factor threshold which also constrains the number of paths returned in that the paths are ordered from best to worst); otherwise the proliferation of states in the search space would be excessive and unproductive.

In some cases of the application of the combination operator, the ontological graph search operator may indicate the instantiation of additional nodes or that the arc whose endpoint was being resolved be replaced by a series of arcs. One such case might occur in the handling of metonymy; see Section 9 for detailed discussion of this process; for example, if the ontological graph search returns a path such as

```
(D4   INSTANCE-OF  -->   D)
(D    IS-A         -->   H)
(H    PART-OF      -->   B)
(B    IS-A         -->   C)
```

where C is the constraint on slot *FU* of instance **G1**, the control structure may call for the instantiation of B, resulting in a TMR such as:

```
TMR:
     D4:
               PART-OF:       B3
     B3:
               HAS-AS-PART:   D4
     G1:
               FU:            B3
```

The example in Figure 6O illustrates a small example of the usual scenario in traversing the search space — many paths are suggested by the instantiation and combination operators, only to be pruned by the threshold mechanism in the ontology graph search (i.e., constraints cannot be satisfied), or in the overall control flow, where the cumulative preference of a path falls below a threshold value.

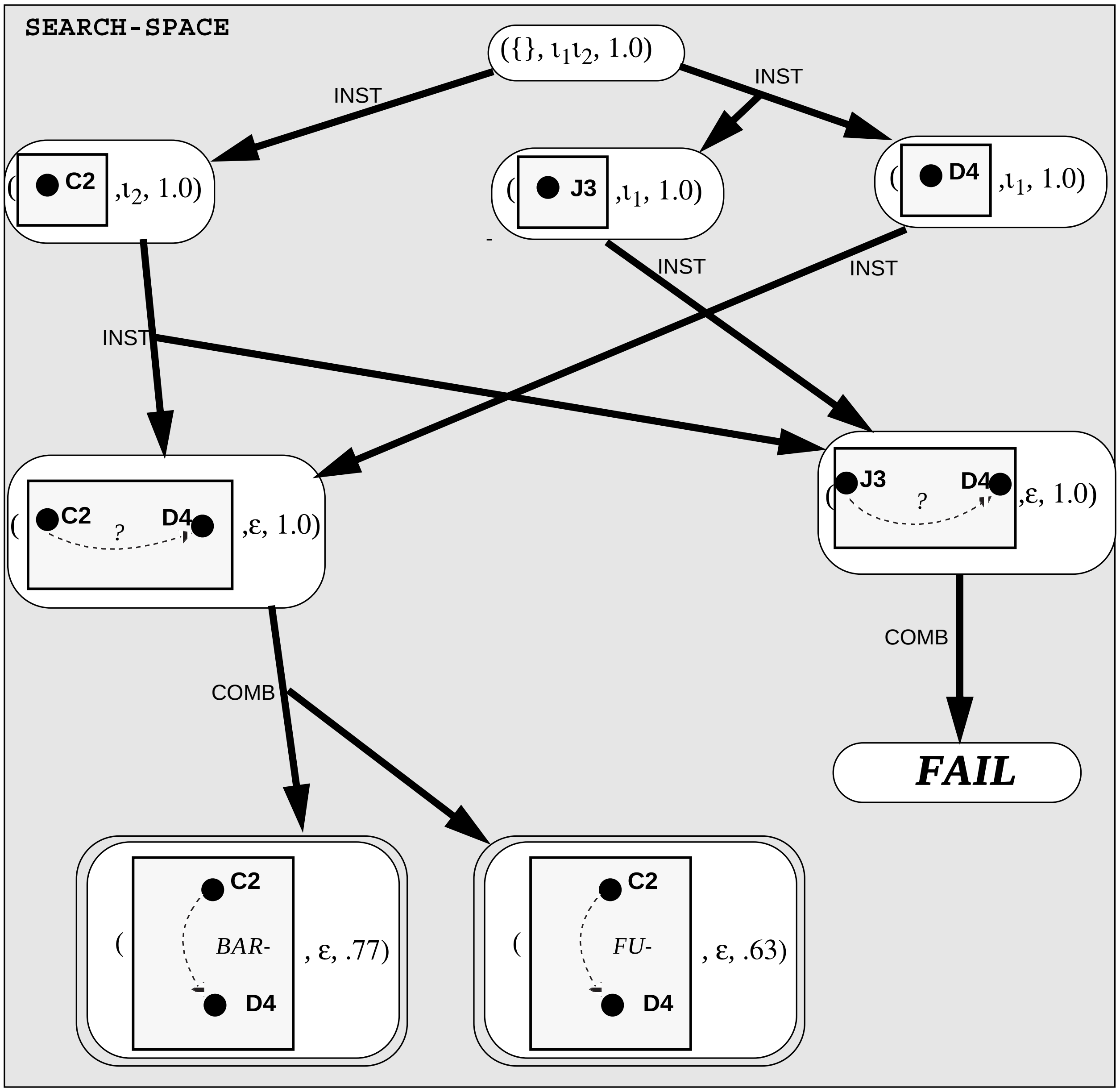


Figure 6O. Example of search space, showing multiple paths, both successful (reaching a state marked by double circles) and unsuccessful (which reach a FAIL state).

7. Basic Semantic Analysis as a Search Problem

The process of building the SDS (semantic dependency structure), which represents the semantic meaning of the text in the TMR formalism, is a state-space search process as described somewhat abstractly in Section 6, and discussed more specifically in terms of semantic analysis in this section. After an initial higher-level discussion and framing of the semantic analysis process as a search problem (Section 7.1 and Section 7.2), each of the two search operators are discussed in turn (in Section 7.3 and Section 7.4). Further subsections discuss the parameters of this search process as applied to semantic analysis (Section 7.5.1).

For the purpose of exposition, the discussion of the semantic processing mechanism in the early sub-sections initially proceeds on the (almost universally incorrect) assumption that there is no lexical, syntactic, semantic, or referential ambiguity found during processing; treatment of ambiguity, which is the focus of this approach, is discussed in later subsections (Section 7.5) and Section 8.

7.1 The Semantic Analysis Process

Figure 7A below illustrates the overall flow of the semantic analysis process. This process is initiated by the input of an f-structure tree, which is augmented with a `ROOT` node indicating the specific lexemes from the lexicon which satisfy all orthographic (**`ORTH-FORM`** or **`ORTH`** zones), categorial (**`CAT`** zone), and syntactic constraints (from **`SYN-STRUC`** zones). As indicated in the figure, the syntactic parser uses the lexical syntactic specification in the lexical entries, from the **`SYN-STRUC`** zone. The information that this zone provides enables the parser to perform two functions in addition to standard syntactic parsing: by unifying the sometimes extensive local syntactic constraints in the **`SYN-STRUC`** with the parse tree, the parser is able to eliminate many lexemes which don't meet simple valence subcategorization constraints. Secondly, the parser establishes the syntax/semantics interface by binding the variables that are specified in the **`SYN-STRUC`** with the values in the `ROOT` nodes of the specified structures.

When multiple parses are found by the parser, instead of returning an inventory of all such parses, the parser can return a packed forest of such parses, which shares common fragments of the parse trees.

This dynamic knowledge source (the f-structure) is utilized by both of the operators in the search process (combination and instantiation, described below).

Heuristic V. Semantic Dependency Structure Building is accomplished by iterative application of the Instantiation and Combination operators.

The instantiation operator takes as input that string of lexemes found at the `ROOT` nodes of the f-structure. In the combination operator, this dynamic knowledge source feeds the primary data-flow heuristic (Heuristic VI below), by indicating to the operator what is to be combined with what, as the SDS-building process traverses that parse tree. For these two reasons, the SDS-building process is not initiated until after the syntactic parse has finished; the interleaving of the syntactic parsing and the SDS building could be easily managed, if efficiency considerations indicated that such an approach would be prudent.

Given an f-structure, a recursive descent process (using either a blackboard for control, in the DIANA implementation, or a branch and bound mechanism (described in Beale (1997) in the MIKROKOSMOS implementation) proceeds by first taking the head lexeme(s) at each level, as

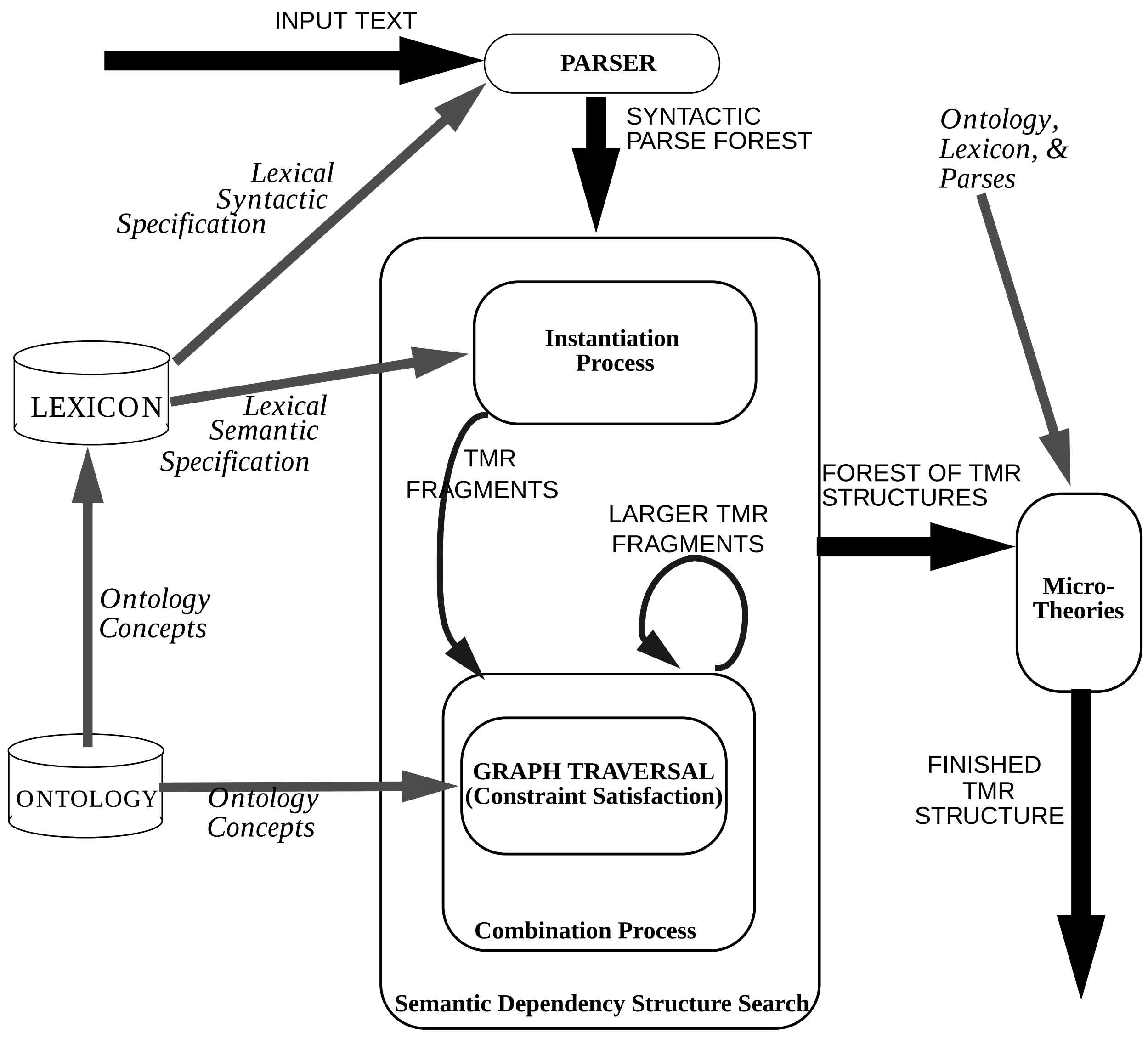


Figure 7A. Data Flow in Semantic Analysis

identified by the value of the `ROOT` feature in each phrase or maximal projection. The **`SEM-STRUC`** and **`PRAGM`** zones of the lexeme's entry are retrieved and instantiated, as described in detail in Section 7.3 below. All of the instantiated fragments are combined together by the combination operator (as described in Section 7.4); this process forms the initial TMR structure, representing the essence of the compositional meaning (as well as some of the lexically-triggered non-compositional meaning) of the utterance.

Heuristic VI. The semantic analysis process proceeds by recursive descent down the syntactic parse tree.

The **`PRAGM`** zone yields two items of information at this stage of processing. The **`STYL`**s of the lexeme are added to the cumulative pragmatic factor representation for the entire text, as represented in a `STYLISTICS` frame for the entire `PROPOSITION` (or, perhaps, at a lower granularity). The **`PRAGM`** zone also contains special-purpose procedural triggers which may be needed in analysis or generation; an example of an analysis procedural trigger would appear in a sense of *the*, as a reference-resolution procedure call.

The rest of this section and following sections will consider the state-space search and its operators, but this time with a focus on the application of the search and its operators on the semantics of the search space, which is the semantic analysis process.

7.2 Semantic Analysis States

The state-space search is conducted over the space of structurally possible combinations of the lexical semantics of feasible lexemes (i.e., polysemes, homographs, and morphologically or syntactically ambiguous forms). The representation of the explored states of this search are represented as different possible overall "readings" or semantic interpretations of the text, each with a preference or measure of relative plausibility of that reading; see Section 7.5.2 for a discussion of the ambiguity representation and resolution mechanisms.

The expansion of a given state typically corresponds to a call to the instantiation or combination operators as outlined in Section 7.3 and Section 7.4. The combination process essentially corresponds to linking two (or more) concept instantiations (rather, the structures representing them) in some way. The manner of combination is dictated by the syntax (as specified above) and/or the lexical semantics, and essentially corresponds to making a structural dependency link. In the meaning representation, this kind of link is indicated by having one object (the *subordinate* in the dependency relation) be the filler of a slot and facet of the other structure (the *superordinate* structure). Typically the superordinate concept will have some indication of semantic constraints on the possible fillers for the slot and facet — either (a) locally defined in the ontological definition of that superordinate concept; (b) inherited from concepts higher than the superordinate concept in the ontology; or (c) defined in the lexical semantic specification of the lexeme in question. The subordinate concept may also have semantic constraints on the superordinate concepts with which it may stand in a dependency relationship. Either way, the application of semantic constraints onto the dependency structure building is the heart of the semantic analysis process, as it prefers semantically more plausible matches and avoids semantic violations where possible.

Each state in the state-space is a complete or partial semantic interpretation of the text or portions of the text, and each state has a relative plausibility metric (the preference) which reflects the cumulative effects of heuristics which applied in the analysis process, including constraint relaxation, stylistic matches, expectations satisfied or violated, frequency statistics, etc. The search roughly corresponds to a best-first search over the state-spaces; because of the shared data structures utilized in the state representations, a the processing of a leading state may also affect other sibling nodes (i.e., expand them by providing new information or split them into new readings).

7.3 The Instantiation Process

The **`SEM-STRUC`** zone reflects the lexical semantics of the lexeme, as described in Section 4. The lexical semantic specification in the **`LEX-MAP`** facet of the zone undergoes an *instantiation* process. As described in the more abstract discussion in Section 6.2, instantiation is the process of producing an *instance* of a concept. In terms of the SDS, an instantiation reflects a particular entity or event in the model of discourse, or, in some cases, a particular mention of such an entity or event.

Heuristic VII. The instantiation operator produces a meaning representation for a specific or generic instance of an entity, event, etc., as specified in the `SEM-STRUC` zone of a particular lexeme; these fragmentary meaning representations constitute fundamental building blocks of the TMR.

Any non-ontological structure undergoes straightforward instantiation. For example the lexical semantic specification of the lexeme **+delicious-adj1** in the **`LEX-MAP`** field of the **`SEM-STRUC`** zone contains the following two structures:

```
LEX-MAP:
     (^$VAR1
               (INSTANCE-OF
                         (SEM *INGESTIBLE)))
     (ATTITUDE
               (TYPE (VALUE EVALUATIVE))
               (ATTITUDE-VALUE (VALUE 0.8))
               (SCOPE (VALUE ^$VAR1))
               (ATTRIBUTED-TO (VALUE *SPEAKER*)))
```

In this example, the second expression, the `ATTITUDE`, is a non-ontological structure which conveys the attitude of the speaker of the text (as represented by the representation `*SPEAKER*`). The `^$VAR1` is an indication of how semantic dependency structure is to be built up (see Section 7.4 below). In the example above, the first structure does not have a structural head at this point, i.e., the structure is rooted at[1] `^$VAR1`, since the particular ontological or non-ontological concept or frame name is not known yet. This first (underspecified) structure passes through the instantiation process intact, since there is no request for the instantiation of a particular concept or TMR structure. The second expression, however, calls for the instantiation of a specific `ATTITUDE` structure.

The results of passing the lexical semantics form above through the instantiation process may be of the following form:

```
TMR:
     (^$VAR1                                    ; not affected
      (INSTANCE-OF (SEM (VALUE *INGESTIBLE))))
     (ATTITUDE647                               ;One-up number for uniqueness
       (TYPE (VALUE EVALUATIVE))
       (ATTITUDE-VALUE (VALUE 0.8))
       (SCOPE (VALUE ^$VAR1))
       (ATTRIBUTED-TO (VALUE *SPEAKER*)))
```

Any structure where the head type is known, whether ontological or non-ontological, will result in an instance; the only expression in a **`SYN-STRUC`** which does not result in an instantiation is the unspecified head structure, that is, an expression where the head is a `$VAR`. The **ATTITUDE647** is a frame structure which is an "entity", and is available to other processes (via the blackboard structures). The numeric suffix on the frame name is generated merely to distinguish this "entity" (or instantiation of the frame structure) from other instantiations; the number is generated to insure

1. "rooted at" is used to mean that if the structure were to be viewed as a tree, the root node of the tree -- as reflected by the frame name when the structure is viewed as a frame

uniqueness and is not meaningful of itself.

In cases where the lexical semantics indicates that an ontological object is to be instantiated, there may be other interacting processes which affect the instantiation process. For example, the **LEX-MAP** zone for the lexeme **+doughnut-n1** may appear as:

```
LEX-MAP:
                    (%DOUGHNUT)
```

The % sign here indicates that the following symbol identifies an ontological concept to be instantiated. In the basic default case, the instantiation process will retrieve the *DOUGHNUT concept from the ontology and instantiate a copy of it (perhaps producing a frame by the name of **DOUGHNUT648** on the blackboard).

In other cases, however, pragmatics processing triggered by other relevant lexical units in the sentence may either block this instantiation process or may initiate reference resolution (or other pragmatics processing) which identifies which instantiations are coreferential and merges them as necessary. For example, consider the text *The man bought a doughnut and three danishes. The doughnut was soggy.* The second sentence's reference to the doughnut ultimately shouldn't result in another doughnut concept instance, but should identify the entity or instantiation from the first sentence. The correct sense of the word *the* will have a trigger in the **PRAGM** zone which performs the reference resolution.

The approach being taken here is that, for the purpose of preserving the presentation order of information, multiple instances will in fact be created for multiple mentions of the same entity in multiple sentences, but that these instantiations will be coreferenced by the COREFERENCE-RELATION mechanism. If we just had the instantiation process invoke the reference resolution mechanism, which would return the instance from a previous mention, we wouldn't be able to distinguish the presentation order; for example, *The man bought a doughnut and three danishes. The doughnut was soggy.* would be indistinguishable from *The man bought a soggy doughnut and three danishes.* Thus, in a somewhat arbitrary distinction, the instantiation operator produces a new instance for multiple mentions in discrete sentences, but it returns the same instance for multiple mentions in the same sentence or clause (on the assumption that presentation order within a sentence or proposition will be identified by such mechanisms as FOCUS).

For cases of lexical semantic specifications which require the instantiation of an ontological concept with further constraints or information (which occurs more often than the univocal mapping case), the process is similar. For the lexical entry **+visit-v1**, for example, the **SEM-STRUC** specification may appear as:

```
SEM-STRUC:
        (%VISIT
            (AGENT  (VALUE ^$VAR1))
            (THEME  (VALUE ^$VAR2)
                    (SEM *BUILDING)
                    (RELAXABLE-TO *PLACE)))
```

The instantiation of this pattern will result in an concept entity instance (e.g., **VISIT670**), which will have all the other information from the pattern included in the frame. This information may be adding constraints or knowledge to the *VISIT concept from the ontology, or may be overriding constraints or information from the ontology, as described in Section 4.

As described in Section 4, the lexical semantics of a lexeme may appear as a **MEAN-PAT** instead of the usual notation in the **LEX-MAP** facet of the **SEM-STRUC** zone; these might or might not result in the instantiation of any entities.

7.4 The Combination Process

Once a lexeme's lexical semantic specification undergoes the instantiation process, it is available to participate in the combination process. The instantiation-combination process is a recursive one, as mentioned above, thus is finished in cases where the lexeme's syntax and lexical semantics are unitary, as is the case in the **+doughnut-n1** example above (there are no semantic dependencies indicated in the lexical semantics of the word). In other words, the combination operator does not apply if there are no dependencies indicated by `^$VAR` within the structure or by syntactic cues.

In more complex cases, the lexical semantics will indicate the pattern of semantic dependencies. Although it is not the case that semantic and syntactic dependencies are necessarily in parallel, at the level of lexical semantics and semantic dependency structure building the semantic dependencies are triggered, in some fashion, by syntactic dependencies, as reflected in the f-structure representation of a sentence. Take an example sentence *The man ate the doughnut*. For now, we shall ignore issues of tense, aspect, definiteness (i.e., the determiners), lexical ambiguity, referential ambiguity, etc. A simplified form of the f-structure for the sentence would appear as:

```
F-STRUCTURE:
      ((ROOT +eat-v1)                       ;$VAR0 gets bound to +eat-v1
       (SUBJ ((ROOT +man-n1)                ;$VAR1 gets bound to +man-n1
               (CAT n)))
       (OBJ ((ROOT +doughnut-n1)            ;$VAR2 gets bound to +doughnut-n1
              (CAT n))))
```

In comparing this structure with the **SYN-STRUC** given for the lexeme **+eat-v1** in Section 3.4, the most evident difference is that the `$VAR` forms have been replaced by actual lexemes. During the syntactic parsing process, these "variables" are set to reference the `ROOT` position in the f-structure parse where the lexemes for the arguments (and other selected-for elements) are identified. Thus along with the f-structure, the syntactic parser also produces a table for each lexeme in the entire f-structure, reflecting the variable bindings for `$VAR1` on up, if any; actually, to avoid confusion if a lexeme appears twice or more in a sentence, the table contains pointers to other locations in the f-structure tree rather than lexeme names.[1] Thus, for example, the f-structure above would actually appear as:

```
F-STRUCTURE:
      ((#11:ROOT +eat-v1)
       (SUBJ ((#12:ROOT +man-n1)
               (CAT n)))
       (OBJ ((#13:ROOT +donut-n1)
              (CAT N)
              (DET ((#14:ROOT +a-det1) (CAT DET)))))
```

1. The actual mechanism for this is slightly different in the MIKROKOSMOS implementation, and is described in Beale (1997).

The binding table would appear as:

```
#11:            VAR1: #12
                VAR2: #13
#14:            VAR1: #13
```

This table lists those `ROOT` positions whose lexemes define variable bindings (i.e., select for something); here, only the lexemes at the `ROOT` position indexed by `#11` (i.e., **+eat-v1**) and `#14` (i.e., **+a-det1**) select for anything. After each such listing, all the variables mentioned in the **SYN-STRUC** of those lexemes are listed, along with what index position they are bound to. For example, in the table above, the `VAR1` for index `#11` (**+eat-v1**) is bound to the lexeme at index position `#12` (namely, **+man-n1**).

In processing the semantics of the above example, the instantiation-combination process is invoked, using the f-structure to guide the order of application of the cycle. The instantiation process is called on the `ROOT`, then a recursive call is made to the instantiation-combination process on each branch of the f-structure (in this case, on the `SUBJ` and `OBJ` structures). Instantiating the `ROOT` lexeme produces a structure of the following form

```
TMR:
     (INGEST661
               (AGENT (VALUE ^$VAR1))
               (THEME (VALUE ^$VAR2)))
```

(The details of the lexical semantic representation are glossed over here for brevity.) The instantiation-combination process, called on the branches of the f-structure, return the objects **MAN663** and **DOUGHNUT664** (ignoring details of definite article meaning, etc.) The combination process then traverses the instantiated lexical semantic structure starting from the root form. The first (and only in this call) object passed to the instantiation process is **INGEST661**. Since the object itself is an instantiation, a traversal of its slots begins. The *AGENT* slot has a `VALUE` facet with a variable as a filler: `^$VAR1`. The variable itself is bound to the lexeme in the `SUBJECT` position in the f-structure; the effect of the caret operator is to identify the “intension” of the reference. In other words, the caret operator, when invoked on the variable, returns the results of the instantiation-combination processing on the `SUBJECT` branch of the f-structure, i.e., a list containing only **MAN663**. The combination process then searches this list, and retrieves the semantic “head”: the head is the first occurrence of an ontological concept instantiation, or if none, the first non-ontological structure instantiation. The combination process then checks to see whether any semantic constraints are satisfied (see Section 8.2) and inserts the concept into the place of the variable, yielding:

```
TMR:
     (INGEST661
               (AGENT (VALUE MAN663))
               (THEME (VALUE ^$VAR2)))
```

Since the data structures here are frames, the value **MAN663** is actually the name of (i.e., a pointer to) another frame. A similar process is then invoked on the *THEME* slot of the frame. The resulting meaning representation for the text will include the following objects:

```
TMR:
      (INGEST661
                  (INSTANCE-OF (VALUE *INGEST))
                  (AGENT (VALUE MAN663))
                  (THEME (VALUE DOUGHNUT664)))
      (MAN663
                  (INSTANCE-OF (VALUE *MAN)))
      (DOUGHNUT664
                  (INSTANCE-OF (VALUE *DOUGHNUT)))
```

The *INSTANCE-OF* slot (often left out because the frame name implies the contents of this slot) merely represents the ontological concept from which the structure was instantiated.

A similar process is invoked on non-ontological structures which may be instantiated based on the lexical semantics of a lexeme. For example, consider the phrase *delicious doughnut*, which may result in an f-structure such as:

```
F-STRUCTURE:
      ((ROOT +doughnut-n1)
       (CAT n)
       (MODS ((ROOT +delicious-adj1)
              (CAT adj))))
```

The instantiation-combination procedure first instantiates the ROOT of the structure, yielding DOUGHNUT642, then traverses the branches of the structure. Note, however, that the lexical semantics of the semantic head here (i.e., **+doughnut-n1**) does not have any indications of how to combine any other concepts with that head — that information will be extracted from the lexical semantics of the “subordinate” concepts (although this process could result in the head concept becoming subordinate and v.v.) The only branch of the structure (other than ROOT) which yields any semantics in this process is the adjective modifier *delicious*. The **SYN-STRUC** of the lexeme **+delicious-adj1** appears as:

```
SYN-STRUC:
      ((ROOT $VAR1)
       (CAT N)
       (MODS ((ROOT $VAR0)
              (CAT ADJ))))
```

Note that (not coincidentally) this is similar to the f-structure produced for the phrase. The variable bindings for the lexeme **+doughnut-n1** are empty (we ignore the $VAR0 binding which points to the lexeme we are looking at), and the variable bindings for the adjective are: VAR1: **+doughnut-n1**. The instantiated structures below would result from calling the instantiation process on the lexical semantics of **+delicious-adj1**:

```
TMR:
     (^$VAR1
               (INSTANCE-OF (SEM (VALUE *INGESTIBLE))))
     (ATTITUDE647
               (TYPE (VALUE EVALUATIVE))
               (ATTITUDE-VALUE (VALUE 0.8))
               (SCOPE (VALUE ^$VAR1))
               (ATTRIBUTED-TO (VALUE *SPEAKER*)))
```

Now the combination process is called on the list of returned structures. The first structure is essentially a constraint on what the semantic head (i.e., the concept which is associated with what the adjective modifies) can be ontologically — the semantic constraint checking (again, see Section 8.2) verifies that **DOUGHNUT642** is of type *INGESTIBLE, which it is. In this case no new information is added; however, the entire process would have failed if the semantic constraint checking failed outright. The second structure returned from instantiating the lexical semantics of **+delicious-adj1** is the **ATTITUDE647** structure. Here the combination process (as in the above example) inserts the head concept resulting from instantiating (and, had it been relevant, recursively calling the instantiation-combination procedure on it) the lexeme bound to `$VAR1` (i.e., **DOUGHNUT642**) into the place of the variable-operator sequence `^$VAR1`. The `ATTRIBUTED-TO` slot takes on as `VALUE` a pointer to the actual frame which has been instantiated to represent the speech act representation data structure in this processing example pass. The resulting data structures for the phrase (stripped to show only relevant information) ends up as:

```
TMR:
     (DOUGHNUT642)
     (ATTITUDE647
       (TYPE (VALUE EVALUATIVE))
       (ATTITUDE-VALUE (VALUE 0.8))
       (SCOPE (VALUE DOUGHNUT642))
       (ATTRIBUTED-TO (VALUE *SPEAKER*)))
```

In cases such as the phrase *the man in the store*, the syntactic and semantic representations fit the description languages as informally defined here, and the combination process proceeds along the same algorithm; a brief overview of the processing of this phrase shows no new mechanisms, only a more general (but perhaps less perspicuous) application of the above processes. As shown in Section 3.4 above, the **SYN-STRUC** for the preposition *in* is:

```
SYN-STRUC:
     ((ROOT $VAR1)
      (CAT N)
      (PP-ADJUNCT ((ROOT $VAR0)
                   (OBJ ((ROOT $VAR2)
                         (CAT N))))))
```

Note that the `OBJECT` of the preposition is bound to `$VAR2`, and the noun phrase head to which the PP is attached to `$VAR1`. The semantics for the basic physical-object location sense lexeme appears as:

```
LEX-MAP:
      (^$VAR1
                  (INSTANCE-OF (SEM *OBJECT))
                  (LOCATION   (VALUE ^$VAR2)
                              (SEM *PHYSICAL-OBJECT)))
```

Here the INSTANCE-OF slot serves to constrain the object in question. The SEM constraint on the *LOCATION* slot restricts locations to be of the *PHYSICAL-OBJECT ontological type (with these restrictions, however, the process of relaxation may apply — see Section 8.2 and Section 9.) The instantiation process yields the following structures:

```
TMR:
      (MAN721
         (INSTANCE-OF (VALUE *MAN)))
      (STORE722
         (INSTANCE-OF (VALUE *STORE)))
      (^$VAR1
         (INSTANCE-OF (SEM *OBJECT))
         (LOCATION (VALUE ^$VAR2)
                   (SEM *PHYSICAL-OBJECT)))
```

The combination process produces the following structures:

```
TMR:
      (MAN721
         (INSTANCE-OF (VALUE *MAN))
         (LOCATION (VALUE STORE722)
                   (SEM *PHYSICAL-OBJECT)))
      (STORE722
         (INSTANCE-OF (VALUE *STORE)))
```

Notice that the combination process, via unification, combined (*INSTANCE-OF* (VALUE *MAN)) (as it appeared in **MAN721**) and (*INSTANCE-OF* (SEM *OBJECT)) (as it appears in the semantics of the lexeme for *in*) to the most informative (i.e., most specific) of the two, that is, (*INSTANCE-OF* (VALUE *MAN)) in the usual manner of unification; this unification process is part of the constraint satisfaction (and constraint relaxation) process described in Section 8.2 (which utilizes a mechanism introduced in Section 5.1.) Since it is "unheaded", the lexical semantic representation for *in* does not appear as a separate data structure in the final combined form; it only serves to combine other semantic structures in appropriate ways. In some cases these "unheaded" structures do add information *per se* — many attributive adjectives in English would have lexical semantic representations of this nature.

The combination process is typically guided by the lexical semantic representations of the lexemes in the text. In some cases, however, there is no predefined or expected semantic structure which can be identified as lexicalized. In the case of Noun-Noun compounding in English, each of the nouns involved may have simple lexical semantic structure, with no overt indication of how to combine them; some speculation on the application of the instantiation-combination process to this construct is found in Section 11.1.

Heuristic VIII. The Combination operator attempts to combine TMR fragments according to either expectations from the syntax/semantics interface or as indicated by syntactic clues; the success of the operator is contingent on the successful application of the constraint satisfaction check, as embodied by the ontological graph search mechanism.

7.5 Semantic Analysis Flow and Control

The control flow of semantic analysis process is left open to any of a variety of architectural control structures; although the choice of control structure affects efficiency substantially, it can also affect the final outcome in a complete model, specifically in committing to either a backtracking approach vs. a multiple-parallel-hypotheses approach. We implemented an agenda-driven blackboard system in the DIANA effort, and a branch-and-bound implementation for the MIKROKOSMOS effort is described in Beale (1997). In the blackboard version of the architecture, each instantiation and combination operator is a separate process on the agenda. The example illustrated in Figure 7B demonstrates a very simple example of a traversal through the search space; note that this example assumes a list of lexemes (vs. lexemes embedded within an f-structure), no lexical ambiguity, and no search paths that are attempted then abandoned. Essentially, all the figure shows is that there are multiple possible control flows that can lead to the same final state; the agenda handler in the blackboard approach or the control flow mechanism for other control architectures have the liberty to traverse any of the possible paths through the search space, because the preconditions on the application of the operators are well-defined.

7.5.1 Parameters of Semantic Analysis

The main parameter which controls the selection of the next node to explore from the search threshold is the preference value (assuming any sort of best-first or derivative search algorithm). A number of threshold values can be set to prune search paths. First, within the ontological graph search itself, a threshold can be set to quit if there are no possible paths at or above the threshold value; the search will return a FAIL state to specify that the constraint cannot be met. In the state-space search, if the combined preference for the node at the tip of the path dips below a different threshold value (or if a specific operator fails), then that path is pruned.

7.5.2 Ambiguity Representation

In actual semantic processing, there may be numerous intermediate and final states; it may be possible to represent the various states in a shared packed forest of TMR representations. Additionally, by sharing portions of the TMR structure between states in the search space, it is possible to expedite the search process by caching the results of a particular instantiation or combination that was already calculated in a previously evaluated state.

The search process may produce any number of final states, each reflecting a somewhat different semantic analysis through a different TMR structure. Each such representation would have an associated preference value. Any of these TMR alternatives can be modified, or the preference adjusted, by one or more of the microtheory processing modules that can operate on the TMR after it is produced by the SDS-building process. The assumption taken in this approach is that between the constraint satisfaction process and the microtheories which consider context and other knowledge sources, one TMR would have a better preference value than the competing readings.

Thus, ambiguity is represented as a disjunction of possible TMRs (or fragments of TMRs).

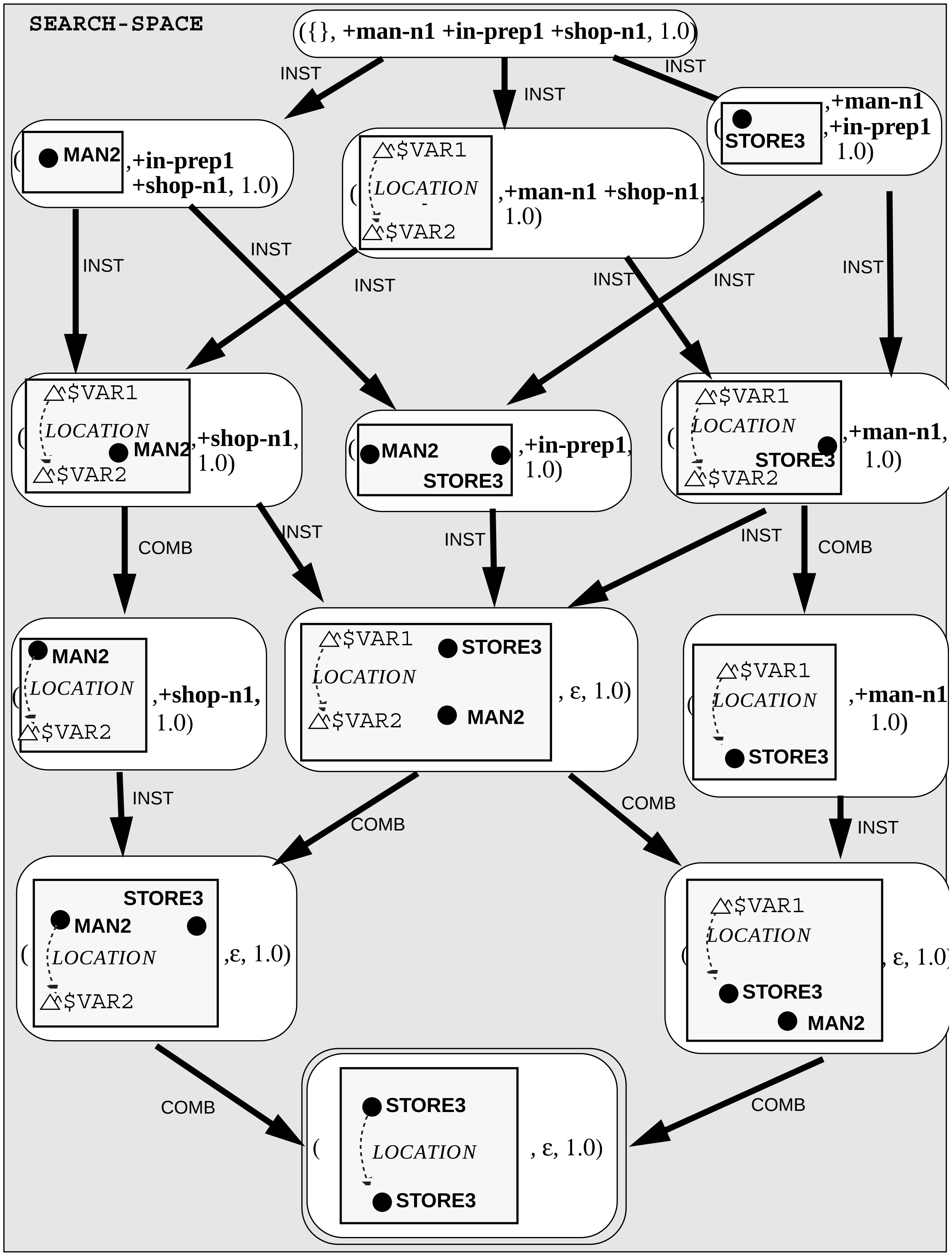

SEARCH-SPACE
({}, +man-n1 +in-prep1 +shop-n1, 1.0)
INST
MAN2 ,+in-prep1 +shop-n1, 1.0)
$VAR1 LOCATION $VAR2 ,+man-n1 +shop-n1, 1.0)
STORE3 ,+man-n1 ,+in-prep1 1.0)
$VAR1 LOCATION MAN2 $VAR2 ,+shop-n1, 1.0)
MAN2 STORE3 ,+in-prep1, 1.0)
$VAR1 LOCATION STORE3 $VAR2 ,+man-n1, 1.0)
COMB
MAN2 LOCATION $VAR2 ,+shop-n1, 1.0)
$VAR1 STORE3 LOCATION $VAR2 MAN2 , ε, 1.0)
$VAR1 LOCATION STORE3 ,+man-n1 1.0)
STORE3 MAN2 LOCATION $VAR2 ,ε, 1.0)
$VAR1 LOCATION STORE3 MAN2 , ε, 1.0)
STORE3 LOCATION STORE3 , ε, 1.0)

Figure 7B. Search states for "*the man in the shop*", assuming no ambiguity or abandoned search paths

Vagueness, on the other hand, is represented in the TMR structure by instantiating an appropriately vague concept from the ontology.

7.5.3 The Introduction of Additional Nodes

In some cases, when the application of the combination operator invokes the ontological graph search, it may indicate the instantiation of additional nodes is necessary, or that the arc whose endpoint was being resolved be replaced by a series of arcs. One such case might occur in the handling of metonymy; see Section 9 for detailed discussion of this process. As a brief example, in a Composer-for-Composed metonymy (e.g., *he played Bach*), where the putative interpretation would be "he played some music composed by Bach", an additional node would be instantiated in the path between the instantiation of *PERFORM-MUSIC concept and the $J-S-BACH instance (from the onomasticon):

```
TMR:
    %PERFORM-MUSIC23:
                (THEME              %MUSIC-PIECE44)
    %MUSIC-PIECE44:
                (COMPOSED-BY        $J-S-BACH)
    $J-S-BACH:
                (INSTANCE-OF        *COMPOSER)
```

In some cases, the additional concept that is instantiated results in a specific instance, while in other cases a generic instance is required. Further discussion of these issues is found in Section 9 below.

7.5.4 Combination of Evidence from Multiple Sources

Since the framework described above produces multiple preferences measures (one from each constraint, in addition to other microtheories, such as a domain model, phrasal/idiom preference mechanism, frequency information, and so on), the issue arises of combination of evidence. Syntactic sources of evidence (syntactic category and subcategorization frames) in both our implementations act to prune out senses or readings that aren't compliant with the constraint. But all the other sources of disambiguating and SDS-building knowledge merely calculate a preference in *[0.0, 1.0]*.

A range of possible ways of combining evidence could be applied to producing an overall result, as discussed in Jones and Onyshkevych (1997). For example, McRoy (1992) accumulates preferences additively. She uses a "specificity"-based measure (in *[-10, 10]*) for weighting each individual preference, then adds the preferences. The more specific a constraint that is met, the higher the preference (closer to *+10*). But the inverse holds for failed constraints, i.e., a failed weak constraint gives *-10*. Harley and Glennon (1997) also use additive weights, as do Alshawi and Carter (1994).

In our model, however, the number of constraints that apply to competing word senses can vary significantly, and an additive model would prefer the sense with the most constraints. Additionally, we found that some failures of constraints should absolutely eliminate a sense. For these reasons, it was most convenient to use a multiplicative approach for combining preferences in the DIANA model. A root-mean-square approach was adopted in the MIKROKOSMOS implementation (sum the squares of the individual scores, produce an average of the squared sums, and compare the square roots of the averaged sums for competing hypotheses). This approach was found to avoid some of the penalties imposed by higher numbers of constraints. We aren't entirely satisfied with this approach, however, and a subject for further research is exploring various competing models of combination of evidence.

8. Application of the SDS-Building Mechanism: Word Sense Disambiguation

The core application of the SDS-building mechanism for semantic analysis (as discussed in Section 7), using the ontological graph search mechanism for semantic constraint satisfaction (see Section 5), involves selection from among many potential semantic representations (TMRs) in the search space. Perhaps the most important process in traversing that search space is Word Sense Disambiguation (WSD), or the selection of the correct sense of each word that appears in the input (assuming, as we do, that an enumeration approach is taken to lexicon construction, as discussed in Section 4.1.2). This section uses the WSD approach to differentiate the overall framework from related work, and focuses on issues involving the application of the SDS-building and semantic constraint satisfaction process to WSD.

8.1 Background and Related Work

Except for very limited situations where raw **word sense frequency** can be used, such as in very restricted, fixed domain settings, all approaches to WSD make use of the words in a sentence to mutually disambiguate each other. The distinctions between the various approaches, however, lie in the source and type of knowledge that is made available by the lexical units in the sentence. We restructure the traditional top-level distinction between paradigmatic and syntagmatic approaches somewhat in order to differentiate the framework described in this document from other, perhaps superficially similar, approaches in the literature. We identify paradigmatic approaches as being based on knowledge of concepts and semantic relations among those concepts. Thus the knowledge base used by the WSD algorithm generally identifies the relations among word *senses* or, preferably, among the semantic primitives used to define those senses (often only through an IS-A hierarchy, but also sometimes using other semantic relations). In contrast, what we are calling syntagmatic approaches are based on expectations about how words are used in structure and what other words they appear with in an input sentence. Thus, the knowledge base allows the algorithm to relate each word to other words that appear in some linguistic relation to the word in question. This distinction should be taken as a caricature for expository purposes, and it will become clear that the distinction often becomes fuzzy.

8.1.1 Related Paradigmatic Approaches

Tversky (1977) popularized perhaps the simplest approach to comparing word senses, via comparing **features** of a word's senses with the features of the senses of another word in the sentence. By picking the subset of senses (one per word) that was maximally similar, sentence-level WSD can be achieved. This requires that all word senses have a set of predefined features or properties (at great acquisition expense, usually), and it assumes that sharing features (or, rather, feature values) signifies *semantic relatedness* or *conceptual similarity*, and that words in a sentence tend to be conceptually similar (two assumptions that still need validation). Other work, such as Waltz and Pollack (1985), also use this fundamental feature comparison approach, typically in conjunction with other mechanisms described below.

The canonical paradigmatic approach uses a relational structure of some sort, whether a **semantic network** stemming from the work of Quillian (1968), or a taxonomic hierarchy of word senses or concepts, such as WordNet (Miller *et al.* (1993)), or an ontology, such as Knight and Luk (1994) or our ontology, described in Section 3.1.3 above. Each sense of words in the input sentence typically correlates with elements of the tree or network, typically one or more nodes per word sense. These nodes can either represent the word senses directly (as in Quillian (1968), much of the work in Evens (1988), or in Charniak (1985, 1986), or the nodes may represent concepts or primitives that are used to construct meaning representations for word senses, as in our approach (see Section 4).

One way to utilize such relational structures for WSD is to consider the *distance* in the relational structure between nodes. Rada *et al.* (1989) define a **semantic distance** metric on a semantic net, using only the IS-A links defined in the network. The basic assumption of this approach is that the semantic net is organized according to similarity (again, an assumption that needs validation). By considering all senses of all words in an input sentence pair-wise, they are able to pick the set of senses that has the minimum semantic distance, therefor, supposedly, are most similar. Distance is computed by simply counting the minimum number of links or edges that need to be traversed over the tree structure between two word senses. In fact, Rada *et al.* (1989) also conducted human experiments that show that path length is reasonable measure of semantic similarity. Their distance metric approach did not fare well when they considered other sorts of relations, not just IS-A; we speculate that results would be more interesting if using the more elaborate mechanisms for weighting edges that is described in Section 5 (even though we use that mechanism to compute constraint-to-filler paths, not sense-to-sense paths).

Rada compares the semantic distance metric to **semantic relatedness**, which reflects all the interconnections between concepts (see above). So “Because ‘is-a’ relations are based on similarity between defining features, we hypothesize that when only is-a relations are used in semantic nets, semantic relatedness and semantic distance is equivalent” (Rada *et al.* (1989)).

Agirre and Rigau (1995) and Agirre and Rigau (1996) also use the Rada semantic distance approach, using WordNet as the taxonomy. Their distance measure factors in the depth of hierarchy (deeper concepts are closer together, so shorter distance) and density (the more siblings at a particular node, the closer they are). This approach does best on distinguishing homography, not the fine senses in WordNet. Not surprisingly, they intend to combine their method with other approaches to do real WSD, as they don’t think their approach can solve the whole WSD problem, although they do believe that they could do much better with a richer ontology. Our approach, in fact, provides supporting evidence for the utility of a distance-related metric on a richer ontology for WSD, although there are important distinctions between our ontology graph search and this simplistic distance measure, stemming from the fact that we use the graph distance mechanism to compute constraint-to-filler paths, not similarity of word senses over the taxonomy.

Our approach also bears superficial similarity to a group of methods for calculating paths over a relational structure using *spreading activation*. We use the distinction in Waltz and Pollack (1985) between two kinds of spreading activation: digital (also known as *marker passing*), and analog.

In the **marker passing** approaches, such as Quillian (1968), Norvig (1989), Fahlman (1982), Hirst (1987), and others, the idea is to start paths at all senses of words in the input sentence (somewhere in the network), and to propagate the “markers” until they collide, identifying proximity, relatedness, or making other inferences. These networks typically have a much richer set of links between nodes than the semantic distance taxonomies; however, this approach, as a class, suffers from the problem of overgeneration of false paths (Norvig (1989) finds only one good path per 10). To overcome this overgeneration problem, Fahlman (1982) types his network connections and his markers, allowing only certain markers to pass in certain links, in addition to preventing certain combinations of markers to coexist. Norvig has the marker strengths decay at each step (to prevent over-long paths), and uses a finite state machine to post-filter his paths to eliminate obviously bad paths and to prevent combinatorial explosion of the search space.

This decay mechanism (called a *gradient*) is one of many typically used in the "analog" approach to prevent search space explosion. In this type of **spreading activation,** nodes in the network are initially *activated* to reflect possible senses of the words in the sentence. This activation then spreads over (weighted) links to neighboring nodes until equilibrium is reached, and certain senses are activated, reflecting the high level of activation they receive from their neighborhood. This algorithm, too, suffers from overgeneration (in the form of too many nodes becoming active), so a range of mechanisms is used to restrict the search space. Collins and Loftus (1975) use "criteriality" to identify importance of links to their originating nodes, and also filter resulting paths based on the surface syntax of the sentence (since their net consisted of words from the sentence itself). Charniak (1985, 1986) does path-checking over all found paths for the most "reasonable" path (as Hirst (1987) does), and divides the activation energy (or the marker weight) by the branching factor at each node, in addition to an exponential time decay of activation energy (for him, only one of 20 paths was good). Granger *et al.* (1984) too filter out potential paths before trying, based on "parsimony, cohesion, and specificity". Waltz and Pollack (1985) used a mechanism called *lateral inhibition* to reduce over-activation, and used a vector of 1000 micro-features for each node, causing a serious knowledge acquisition burden. As Rada *et al.* (1989) note, their semantic distance metric could be considered a special case of spreading activation, if only IS-A links are allowed, with only positive activation (no inhibition etc.)

Of these approaches, only Charniak (1985, 1986) actually bears some similarity to our framework, in that he checks for paths between selectional restrictions and potential fillers of those roles (among other things); given a path, he looks for elements of that path to provide an explanation of the relationship, in an abductive manner. However, the ontology we use consists of a connected network of concepts (not words or predicate calculus terms and expressions) which are used in a compositional manner to represent the lexical semantics of a word. Since we have a complex arc weighting scheme, and perform a best-path search, we obtain significantly better results. In general, since we don't spread positive and negative activation energy (only pursue, in a best-first manner, paths that reflect a monotonically-increasing cost), we avoid the combinatorial explosion expense or a necessity of a post-filter, inhibition, damping, etc. Spreading activation reflects semantic relatedness, in that if two nodes have multiple links connecting them, they will be very "close", while for our application we are looking (abductively) for the single cheapest path that reflects the best relation between a constraint and a filler (in time linear to the number of links, without any combinatorial explosion caused by lack of equilibrium or over-activation). Additionally, most efforts that use spreading activation (including Charniak) use the mechanism itself for the control structure, while we use the ontology graph search to consider constraints pairwise, and rely on the SDS-building process to provide overall control.

One severely constraining factor on the use of network-based approaches (whether semantic distance or spreading activation) is the availability of the network resource, and the mapping of lexical items to nodes in the network. One potentially appealing source is the Machine Readable Dictionary (MRD), since the natural language **dictionary definition** can be used as a set of links to other words in the dictionary. However, the words in the definitions are themselves words (not word senses), and therefore need to be disambiguated themselves. Lesk (1986) approaches this difficulty by considering the overlaps of dictionary definitions of all senses of words in a sentence.

In other words, he attempts to find the subset of word senses (one per word) for which the dictionary definitions share the most words in common. The search space for this approach, however, is very large. To address this as an optimization of parameters problem, Cowie *et al.* (1992), Wilks *et al.* (1992), and Wilks and Stevenson (1997) use a **simulated annealing** approach (see Section 5.3.2 for an explanation of simulated annealing) with strong results. Veronis and Ide (1990), on the other hand, attempted a similar experiment using a **neural net** approach, however, on much smaller scale.

In addition to the definition, some MRDs provide another source of potential (paradigmatic) disambiguation knowledge, in the form of domain or **subject codes**. For example, the electronic form of the LDOCE dictionary (Procter *et al.* (1978)) provides about 500 subject tags. A number of approaches, such as much of the work described in Boguraev and Briscoe (1989) or, more recently, Wilks and Stevenson (1997), make use of these codes as a weak source of information for WSD, generally in conjunction with other methods. If the subject domain of a text is known *a priori*, then word senses in that domain can be preferred over other senses. In the absence of knowledge about the domain, it is sometimes possible to determine it automatically by tallying "votes" for each domain by taking the code from each sense for each word in the text; then WSD is performed by taking the sense that appears in the most frequent domain (or is closest to it). Harley and Glennon (1997) use the subject codes in the CIDE resource (Procter (1995)) in much the same way. Of course, this approach requires that an appropriate resource be available, which isn't the case for most of the world languages. And even for English, the coverage of these resources is such that many word senses (or entire words) have no domain information, resulting in no evidence for WSD.

As an alternative to domain codes, McRoy (1992), Morris and Hirst (1991), and others apply similar WSD methods using **domain clusters** of semantically associated words. The association or relationship between words can be (using McRoy's terms) either *categorial* (for synonyms or near-synonyms, like WordNet syn-sets), *functional* (reflecting relations such as part/whole), or *situational* for clusters of terms in the same domain, but with thematic role or more distant relationships, like *lawyer/court/testify/...* Although these clusters can be acquired relatively economically, neither this method nor the subject code or dictionary definition approaches provide any information about how the terms are related in a way specific enough to facilitate SDS building; despite the drawbacks of the semantic network methods, they are able to offer information which provides structure for the SDS process. In fact, the ontological graph search mechanism (described in Section 5) can be used for this cluster approach, in that synonymous words are mapped to the same (or sibling) ontological concepts, and semantic relations are uncovered by the search process, so long as they are represented in the ontology; the ontology is weakest, however, on the situational relational information. The cluster and other non-structure-providing methods could be used to provide secondary weak WSD information in our approach.

8.1.2 Related Syntagmatic Approaches

The central premise of the syntagmatic approaches is summarized in a slogan, attributed to Firth: "a word is known by the company it keeps". So for the immediate context of WSD, this suggests that the knowledge needed for disambiguation of a word sense should reflect the context in which it is expected to appear. We differentiate two approaches to acquiring and structuring such information. One general class of approaches, which has regained popularity in the past 5 years, involves the **statistical** acquisition of information identifying words that appear together in a corpus, generally in the form of uninterpreted frequencies, information measures, or language models. The other general approach involves (typically manual) acquisition of specific syntactic, semantic, or pragmatic expectations of what other classes of words are expected to appear in a specific linguistically motivated relationship to the word in question.

All the statistical approaches, for example Church and Hanks (1989), Gale *et al.* (1992), Alshawi and Carter (1994), Yarowsky (1995), Dagan *et al.* (1991), Dagan *et al.* (1993), Leacock *et al.* (1993a,b), or Brown *et al.* (1991), are **corpus-based** in that they require corpora of substantial size for each particular language, domain, and text type. In fact, much of this type of work (except some of the work by Yarowsky (1995) and relatively few others) requires that the texts be tagged or annotated in some way, whether with word senses relative to a fixed resource, such as WordNet, or with translations of running text, as in Melamed (1997). Building such annotated resources is very difficult, labor-intensive, and not very reliable (manual annotation isn't very accurate or consistent); even the premise of Melamed (1997), that bilingual corpora will soon be prevalent, won't necessarily prove as fruitful as expected (he has the further complication that his word senses are defined only relative to another language, which won't necessarily support the sort of reasoning necessary for resolution of metonymy, reference, or lexical divergences).

We agree with Wilks and Stevenson (1997) and Lehman (1994) that such statistical methods, based on corpus co-occurrence, are unlikely to achieve WSD (not to mention SDS-building) on running text with high accuracy, if used alone. None of these methods has been demonstrated on WSD of running text with any accuracy, if at all. It should be expected that such methods could typically be augmented and improved by the addition of well-understood linguistic knowledge; however, statistical models are typically virtually impossible to interpret and modify in a linguistically informed manner. In fact, Lehman (1994) finds that, for the statistical **co-occurrence** approaches, the frequency model doesn't add much value, and that most of the discriminating power comes from the inventory of word pairs (which was, albeit, discovered by the statistical techniques).

An additional problem with the range of collocational approaches (whether statistical or symbolic), as identified by McRoy (1992), is that many of these approaches, such as Smadja and McKeown (1990) (but not the McRoy (1992) or Dyer and Zernik (1986) symbolic approaches), aren't able to identify collocations between word senses, but between word strings, which still leaves much work to be done for using the results for WSD in support of Machine Translation in our framework (and other models, except for Brown *et al.* (1991)).

Despite the acquisition burden and inaccuracy, the symbolic **expectation**-based approaches have the advantage of allowing (or requiring) interpretability of the knowledge source by humans. Information with specific linguistic information content can be evaluated and generalized by human acquirers. Straightforward linguistic information, such as **syntactic category** and **subcategorization** has been used extensively, for example, recently by Wilks and Stevenson (1997), McRoy (1992), Harley and Glennon (1997), and the MikroKosmos system described in this document; although not achieving full WSD for fine-grained senses, this sort of knowledge is sufficient for disambiguating at the homograph level, and for pruning out a substantial percentage of fine-grained senses (at least a third, in our work). In fact, Wilks and Stevenson (1996) show surprising results on pruning homographs just using syntactic category information.

Perhaps the most widely-used type of expectation for WSD relies on **selectional restrictions** or semantic constraints, ever since Katz and Fodor (1963). This approach involves an expectation that a word of a specified semantic class will fill an argument position of a verb. The set of semantic classes may be small, involving only half a dozen binary distinctions, or could use a large ontology (as in our case). WSD then relies on selecting the combination of verb and argument senses for which the selectional restrictions are satisfied, for example in Charniak and Goldman (1988), Oflazer and Yilmaz (1996), Harley and Glennon (1997), and many others.

Selectional restriction approaches since Wilks (1975a,b) and his notion of **preferences**, including Fass (1986) or Fass (1988), have not required that selectional restrictions be completely satisfied (i.e., allowing some kind of relaxation), nor have they required that 100% of all constraints be satisfied in order for a sense to be picked. The complication then is how to control the relaxation of the constraints. If relaxation is too extensive (or if the constraints are too coarse-grained to begin with), then the restrictions lose all discrimination ability; if the mechanism is too restrictive in that it doesn't relax sufficiently or in the correct way, then too many correct senses are filtered out, particularly in the case of metaphor or metonymy. McRoy (1992), for example, finds that constraint-based discrimination isn't very effective; this probably results from the very limited relaxation of constraints (one level in her hierarchy) supported by her approach. One of the goals of the ontological graph search mechanism (described in Section 5) is overcoming exactly this problem: exerting tight, informed control over relaxation, sufficient to allow metonymic arguments, but restrictive enough for correct WSD.

Another problem with the selectional restriction approach is the knowledge acquisition expense, since not only do verbs need to be marked with constraints on each argument, but each noun also needs to be marked with a semantic class (or features). Wilks (1975a) identified that "it is a hypothesis of this work that we can build up a finite but useful inventory of bare templates adequate for the analysis of ordinary language: a list that can be interpreted as the messages that people want to convey at some fairly high level of granularity". But this acquisition is expensive and subject to human error, especially when exacerbated by lack of methodology. Zernik and Jacobs (1990) attempted, in a small-scale experiment, to use a statistical training technique to acquire information that amounted to selectional preferences, but with inventories of words (found in a corpus) in the place of semantic classes and constraints. Resnik (1997) extended that approach, but in an unsupervised training context (unsupervised in the sense of not requiring a tagged corpus, not in the sense of not using a pre-defined set of senses). He looked for words which appear after a verb with higher frequency than their general distribution over the corpus, then abstracted up the WordNet tree to identify a syn-set as a selectional category (since the nouns were undisambiguated, he simply gave credit to each syn-set having a sense of the noun). His results weren't great, however, possibly because metonymic language causes abstraction too far up the hierarchy, or maybe because he needs to disambiguate better during the training.

The use of coherent semantic classes is a common element of virtually all approaches to selectional restrictions. Spreading activation over semantic nets, however, (including Waltz and Pollack (1985)), mixes semantic and lexical knowledge in the same network. Although some IS-A relations may exist in these semantic networks, they are intermixed with other syntactic information (parse structures, syntactic categories) and other semantic relations. The activation then spreads over all types of information for all competing word senses for all words in a sentence, while selectional restriction tests are typically pair-wise. Although some selectional constraint information may exist in such semantic networks, they tend to be overshadowed by other semantic relations, more like the situational domain relations mentioned above. Other approaches to spreading activation, such as those using WordNet, don't even have semantic argument information available.

Although based on the traditional Katz and Fodor (1963) notion of selectional restrictions, our framework uses a significant generalization of the approach, taking advantage of the knowledge-based paradigm to address practical concerns of the application, namely supporting SDS-building.

- We use a finer granularity of constraints, in that any of the 5000 concepts in the ontology are available for use as a semantic constraint.
- Our framework provides multiple levels of constraint: a basic constraint (the SEM facet), a DEFAULT (always a subtype or specialization of the basic constraint, which, if matched, is stronger than satisfaction of the basic constraint), and a relaxation limit RELAXABLE-TO, which is a generalization of the basic constraint, which bounds how far vertical (taxonomic link-only) relaxation can proceed in the ontological graph search process.
- We rely on a relaxation method which not only relaxes along *IS-A* links, but also along any other links, especially metonymic ones, as described in Section 5 and Section 9. Previous WSD efforts that relied on selectional constraints were often weakened because of an inability to handle metonymy in an integrated way; they either had to weaken their constraints to the point of uselessness or they would fail to resolve upon encountering a metonymy.
- We generalize selectional constraints beyond case roles to various other semantic relations as well. Each EVENT and ENTITY in the ontology, on average, has 14 relational links to other concepts, each with a “selectional restriction” on the destination concepts of the links.
- We employ multiple sources of semantic constraints on each candidate filler of a relational slot: a) the definition of the RELATION itself has constraints on *DOMAIN* and *RANGE*; b) each concept in the ontology can have constraints on any relation for which it appears in the *DOMAIN*; c) (multiple) inheritance of constraints from parent concepts; d) the lexical entry for a given word, via the syntax-semantics interface, can further constrain relations, roles, or slots beyond the constraint that is in the ontology (see Section 4).
- We allow a much broader notion of what can be “selected for”: prepositions “select” both up and down (i.e,. object and attachment position), nouns can select (e.g., what it can be an *AGENT* of or what its *LOCATION* is likely to be), adjectives select for their attachment, as well as *v.v.*, verbs select not only for “inner arguments”, “thematic roles”, or “case roles”, but can also impose constraints on “outer arguments” or incidental relations that they may have to other concepts via prepositional phrases, relative clauses, etc.

As an example of the large number of constraints available in this model, on the training data discussed in Section 8.3 below, in the first text an average of 231 constraint were checked per sentence, which amounts to 14 constraints per word!

Critical to our approach is an explicit separation of information between the semantic (ontological) level and the strictly linguistic (lexical) level. Fass (1988), however, doesn’t appear to make this distinction: “In our view, semantic relations between terms are complex systems of mappings or structural relationships between linguistic descriptions of these terms”. Our approach to WSD and SDS-building involves identifying the ontological relationships between the concepts that are used to define the meaning of each word sense, then actually producing a full meaning representation for running text.

The **scripts** developed by Schank and Abelson (1977) and Cullingford (1981) could be seen as a generalization of selectional constraints. Instead of providing information just about argument roles of a verb in any context, the scripts provide expectations of what the argument roles might be in a specific context for a semantic class of verbs, and the context may include a set of related events. Norvig (1989), Bouaud *et al.* (1996), Dyer and Zernik (1986), Dahlgren *et al.* (1989), Charniak and Goldman (1988), (and even Wilks (1975a) who uses “semantic templates” for general behavior) all make use of (at least shallow) script-like mechanisms for WSD. Acquiring scripts at a significant depth, however, proves to be too expensive for practical applications.

The approach in Wilks (1975a) and Charniak and Goldman (1988) uses a hybrid of shallow scripts, which provide selectional restrictions for roles, and logical inference rules. Wilks’ inference rules included both role constraints and causality information (e.g., what the results of getting shot are). More traditional logic-based approaches, such as the abductive **logic** model of Hobbs *et al.* (1988), Grosz *et al.* (1986), or Hobbs (1991), encode some amount of selectional and script-like relational information by means of axioms, but also suffer from severe acquisition problems.

The **Word Expert Parser** approach of Small and Rieger (1982) involves an extreme case of encoding contextual knowledge (in their case, procedurally) to support inference for WSD, which was doomed to failure in scaling up because of the extensive amount of knowledge that had to be hand-crafted for each word.

8.1.3 Hybrid Approaches

Most work on practical applications that require WSD assume that some combination of the above techniques will be necessary to achieve the desired WSD accuracy. In some cases, such as McRoy (1992), these multiple techniques are independent, and the results are combined in some way to get an overall WSD preference (see Section 7.5.4). In a group of approaches described below, including our own, elements of both syntagmatic and paradigmatic approaches are combined into a single technique to allow for better-informed WSD.

Miller and Teibel (1991) use their WordNet hierarchy in combination with a corpus, by looking for examples where synonyms of the word in question appear in the same context in a corpus. So for WSD, they replace, in an abstraction of the sentential context, the word in question with other words in the syn-set for each sense, and see if any local contexts appear in the pre-tagged corpus. This method suffers from an extremely heavy reliance on a huge sense-tagged corpus, however. Instead of using an MRD or WordNet, Yarowsky (1992) uses Roget’s thesaurus. Using an on-line corpus (an encyclopedia), he collected statistics of the categories of words (of the 1000+ categories or groups in Roget’s) that appeared in a 50-word left and right context. This demonstration, however, was very narrow in scope, and remains to be proven as scalable to full running text.

Resnik (1995a,b) also uses WordNet, but as a semantic net for computing semantic distance, but he augments this with corpus-based frequencies for each node (syn-set) in WordNet. By computing the *information content* of each syn-set (how much unique information it conveys, inverse to the frequency), he gets a more informed distance metric that is sensitive to frequency of use at each node in the path. He finds that this outperforms regular semantic distance metrics, but expects that a richer network would be more informative, as do Richardson *et al*. (1994), Agirre and

Rigau (1996), and as do we. Richardson *et al.* (1994) also use the Resnik information content measure, but rely on the more elaborate Rada *et al.* (1989) semantic distance metric to mitigate a weakness of the information content measure, where the size of syn-sets affects the relative importance.

Sowa (1993), on the other hand, makes use of the simple semantic distance over an IS-A tree to help pick right conceptual graph for WSD, and for finding missing elements of meaning. Since he does not use an *a priori* full-sized ontological or taxonomic resource, his distance calculation is contextually restricted and only reflects the conceptual graphs invoked by sentential context.

Kozima and Furugori (1993) and Kozima and Ito (1995) also use word frequency to indicate informativeness, but in conjunction with spreading activation over LDOCE definitions. For them, initial activation is proportional to the significance of the word, reflected by the inverse of the frequency of occurrence in LDOCE definitions. But they have a big problem in that the defining vocabulary words in the definitions aren't disambiguated themselves.

Basili *et al.* (1997) share a concern with us regarding unsupervised methods for inducing WSD information statistically from corpora (monolingual or bilingual), since they too believe that explicit semantic classes are needed to avoid "mysterious scores with no linguistic flavor", because the semantic classes and data are needed for some applications/domains for their explanatory power, as is the case for supporting KBMT. They do use corpus statistics, however, to "tune" the WordNet taxonomy to reflect only the domain of immediate interest.

The overall approach for the work described here also falls into this hybrid syntagmatic/paradigmatic category. Crucial to our framework is the paradigmatic ontological knowledge source, which has some flavor of a semantic net in that it includes substantial relational and situational links between the concepts, but differs from many of the resources described above in that there is no linguistic knowledge in the resource (it is a network of concepts, not words). However, our decision procedure does not follow the spreading activation model, but (in addition to syntactic category and subcategory matching) is very much like selectional restriction satisfaction, although substantially generalized (as discussed in detail above) to include more sources and types of constraints. Additionally, the ontological graph search provides an elaborate relaxation procedure, which has some of the "semantic distance" flavor (but differs from distance in that the search checks for compliance of a filler with a constraint instead of assessing "semantic similarity"); also, it uses an elaborate mechanism for assigning weights to each link on the basis of type and context (vs. depth or breadth of the tree at that point), and returns a single chain of meaningful links between the filler and the constraint (resulting in much simpler computational complexity). Unlike many of the WSD approaches described above, particularly the statistical ones, our approach has been tested on running text, and produces not only WSD, but also builds a TMR representation of the meaning of the text.

8.2 The SDS-Based Model for WSD

After discarding word senses with inappropriate syntactic categories and syntactic (e.g., subcategorization) frames, the central mechanism for selecting word senses is a heuristic which relies on the satisfaction and relaxation of constraints on combinations of concepts. This heuristic attempts to find the best match between i) semantic constraints on roles or relational slots of a concept and ii) the candidate fillers for those roles or relational slots. The process of word-sense disambiguation prefers the set of word senses with the overall best-satisfied set of constraints, as described above.

Hobbs and Martin (1987) observed that "every morpheme in a sentence corresponds to a predication, and every predicate imposes selectional constraints on its arguments. Since entities in the text are generally the arguments of more than one predicate, there could well be inconsistent constraints imposed on them." We find that we this observation holds in our work, in that we have a proliferation of constraints, as described above. For this reason, the SDS-building process combines evidence from all available constraints, with some amount of constraint relaxation typically found in every sentence.

As mentioned above, all concepts have relational slots (in addition to arguments or case-roles in the case of EVENTs, or attributes in the case of ENTITYs). On average, each concept has 14 local and inherited relational slots; we use about 350 discrete types of relational slots. Each relational slot has constraints on what concepts (or complexes of concepts) may fill that slot, as well as an expected (DEFAULT) filler for some of the slots. The constraint can be any concept from the ontology (or Boolean combinations of concepts). As described in more detail above, one advantage that we have over previous constraint-based approaches to WSD is that we have many more sources of constraints and a much richer set of possible constraints (the ontology), which allows fairly fine-grained constraints on some slots, along with a knowledge-intensive constraint checking heuristic.

In the easiest case, the selectional constraints on the correct set of senses are all satisfied, and are violated for incorrect combinations of senses. Satisfied selectional constraints appear in the heuristic as a simple path over the *IS-A* hierarchy between the candidate concept and the constraint. But because natural language use is not literal or precise (because of metonymy, metaphor, etc.), we often need to relax constraints; however, relaxing or discarding semantic constraints unrestrictedly would result in egregious proliferation of readings in semantic analysis.

In our heuristic, controlled constraint satisfaction is managed by considering all arcs, not just *IS-A* arcs, and by levying a cost for traversing any of those other arcs. We treat the ontology as a directed (possibly cyclic) graph, with concepts as nodes and relational slots as arcs. Thus constraint satisfaction is treated as a cheapest path problem, between the candidate concept node and the constraint nodes; the best path thus reflects the most likely underlying semantic relation, whether it be metonymic or literal.

In the simple, base case, the process of matching a potential filler against a semantic constraint is simply checking whether the filler *IS-A* whatever the constraining concept is. This check identifies whether the constraining ontological concept is a parent of the potential filler in the ontological tree (viewing only the "vertical" hierarchical arcs in the tree); a *DOUGHNUT *IS-A* *PASTRY which *IS-A* *PREPARED-FOOD, which (skipping some generations) *IS-A* *PHYSICAL-OBJECT, so therefore a *DOUGHNUT can also be said to *IS-A* *PHYSICAL-OBJECT (the relation is transitive). This process of constraint checking is viewed as an application of a best path algorithm from the (potential) filler to the constraint concept, as described in Section 5.1. Since the cheapest arc is *IS-A*, then the ontological graph search algorithm finds whether the potential filler has an *IS-A* relationship to the constraining concept. Typically, the weight on the *IS-A* arc is *1.0*, or very close to *1.0*.

In many cases the constraint isn't a single simple concept but a Boolean combination of constraining concepts. Currently the AND and OR operators are supported in an obvious manner. In the current version of the ontological search algorithm, the OR case is treated by placing all of the disjuncts on the target search frontier (i.e., identify as acceptable path termination points, and in bidirectional search, place on list of target-side nodes available for expansion), and searching as in the base case; the difference is that the shortest path to reach *any* of the target nodes succeeds. The AND case is treated by iterating through all the conjuncts, and checking that the constraints are met — the resulting preference is the multiplication of all of the weights of all the conjunct preferences. With both of the operators, the juncts are actually treated in a recursive manner to handle possible imbedded Boolean combinations. The effect of a negation operator NOT can be simulated, for example NOT fu, by an OR of the remaining siblings of the concept fu, and the remaining relevant siblings of the parents of fu. For example, (AND *ANIMAL (NOT *HUMAN)) would be simulated by disjoining all of the remaining children of the parent of *HUMAN (for illustration, assume it to be *PRIMATE), and also adding as disjuncts the roots of all of the remaining maximal subtrees under *ANIMAL which do not include *PRIMATE.[1]

In some cases, although there is only one concept constraining the semantic filler, that concept may be a complex constraint, not just a node from the ontology (identified as a *univocal mapping* in Meyer *et al*. (1990)). For example, the lexeme **+jet-v1**, meaning "to travel by jet", has a constraint that the *INSTRUMENT*, if mentioned, typically needs to be a jet-propelled aircraft (see below for discussion of relaxation of constraints). The lexical semantics of this lexeme may include the following structure:

```
SEM-STRUC:
    (%MOVE
                 . . .
                 (INSTRUMENT
                          (SEM *AIRCRAFT
                                   (PROPELLED-BY
                                    (VALUE *JET-ENGINE)))
```

The semantic constraint-checking procedure needs to not only check that the possible filler is of the appropriate ontological type, but also that it has appropriate internal structure, for example, as in the instance:

```
TMR:
    (%AIRCRAFT234
                 (OWNER        (VALUE %PERSON223))
                 (MAKE         (VALUE "LEAR")) ;;this sloppiness for illustration
                 (PROPELLED-BY  (VALUE JET-ENGINE)))
```

Ignoring the very imprecise semantic representation used here (for perspicuous illustration purposes only), this example demonstrates that once the conceptual type of the potential filler (i.e., **AIRCRAFT234**), *IS-A* of the appropriate semantic class (i.e., *AIRCRAFT), any slots that the constraint and filler may need to be verified as well. Thus this process can be seen to be very similar to a recursive feature structure unification process, augmented with relaxation. Thus, if the prospective filler has internal structure, the semantic constraint checking process remains the same, since the algorithm is recursive.

For example, given an input sentence such as *The independent candidate jetted around the*

1. A NOT macro will be implemented to avoid this unwieldy notation.

country in his two-seater Cessna, given the lexical semantics of the word *jet* as identified above, the strict semantic constraint would fail. Given that the search is for a path from the potential filler to the constraint, the arc weights for that particular search would include:

```
    (INSTANCE-OF              1.0)
    (IS-A                     0.997)
    (SUBCLASSES               0.9)
    . . .                     ;all other arcs have costs < 0.997
```

The other arc weights would be defined with smaller multiplicative weights. So the search path would begin at AIRCRAFT337, traverse the *INSTANCE-OF* arc (cost still *1.0*), succeed in meeting the constraint *IS-A* `*AIRCRAFT`, but fail in the secondary constraint `(`*PROPELLED-BY* `(VALUE JET-ENGINE))` since world knowledge would identify the Cessna two-seater as a prop-driven plane. Since *PROP-ENGINE does not satisfy the constraint *IS-A* *JET-ENGINE, the search would continue. The next cheapest arc to traverse from *PROP-ENGINE would be *IS-A* to the parent concept *PROPULSION-ENGINE, the net cost would still be *1.0*. But then to reach the concept *JET-ENGINE, which also *IS-A* *PROPULSION-ENGINE, the path would have to traverse the *SUBCLASSES* arc from the path head (i.e., *PROPULSION-ENGINE) to the target, *JET-ENGINE. The cost of that last arc would be *0.9*, giving a net cost of *0.9* for the path. No cheaper path between the two concepts (*PROP-ENGINE and *JET-ENGINE) would be found. The net effect of this semantic constraint test would be two-fold: an *IS-A* relation on the head concept, and a sibling relation on a secondary constraint. Thus directed constraint relaxation served here to allow the most plausible semantic dependency structure to be built without allowing a proliferation of alternative readings.

There are cases where there are constraints on the filler of a slot from the head, as well as constraints from the element that is providing the filler on what the head can be. For example, one sense of the English word *wooden* is that the head is *MADE-OF* *WOOD. This sense of the adjective (say, **+wooden-adj2**) has a constraint on what it can modify:

```
SEM-STRUC:
    (^$VAR1
            (INSTANCE-OF
                    (SEM *ARTIFACT))
            (MADE-OF
                    (VALUE *WOOD))))
```

Here, the constraint is that this sense of *wooden* can only modify things that are made by people, i.e., artifacts, and that the meaning of the word is carried by filling the *MADE-OF* slot. Meanwhile, the head that is being modified will also have constraints (local or inherited) on what it can be made of. For example, the furniture sense of the English word *table* will specify what tables can be made of. Thus, in order to produce the representation for the phrase *wooden table*, two constraint checks need to be carried out: a constraint on the head that *wooden* requires, and the constraint on the filler of *MADE-OF* that *table* requires.

8.3 Results

The entire Mikrokosmos SDS process was evaluated on the WSD problem on Spanish. This evaluation used a Spanish lexicon acquired as part of the overall effort. The lexicon consisted of more than 7000 lexical entries, along with 37,000 "virtual" dynamically-producible entries resulting from lexical rules, as described in Viegas *et al.* (1996) and Onyshkevych and Nirenburg

(1995), where lexical rules apply to well-defined base lexemes to transform the orthography, syntax, and semantics in a regular way, much the same way as a derivational process.

Table 1 shows statistics and results of the application of the Mikrokosmos implementation of the SDS-building process and ontological graph search as described in previous chapters. The table shows system results on 4 out of 400 Spanish texts used in knowledge acquisition (training). It

Table 1: Word Sense Disambiguation Results on Spanish Training Data

	Text 1	Text 2	Text 3	Text 4	Average
# of words in text	347	385	370	353	364
# of words/ sentence	16.5	24.0	26.4	20.8	21.4
# of open-class words	183	167	177	177	176
# of ambiguous open-class words	57	42	57	35	48
# of senses per ambiguous word	2.65	2.24	2.47	2.06	2.36
# of words resolved by syntax	21	19	20	12	18
# of words resolved by semantics	30	22	25	22	25
# correctly resolved: syntax + semantics	51	41	45	34	43
% correct of ambiguous: random guess	62	70	64	67	65
% correct of ambiguous: pick first sense	67	45	54	46	54
% correct of all: pick first sense	90	87	86	90	88
% correct of the ambiguous words	**89%**	**98%**	**79%**	**97%**	**91%**
% correct of all open-class words	**97%**	**99%**	**93%**	**99%**	**97%**

is important to note that the context of this experiment involves full-scale building of the Text Meaning Representation as described in Section 3.3, not just WSD experiments in isolation. So these results also reflect errors and successes in syntactic parsing, syntax-semantics interface, and construction of TMR through SDS-building. The four texts represented in this table are real-world texts from the EFE newswire, from the financial domain. The texts had, on average, 17 sentences each, with over 21 words per sentence. The correct senses for all the open class words in these four texts were selected (by a native speaker) from the Mikrokosmos lexicon, as described above; the native speaker was a member of the lexicography team that built the lexicon, so was able to pick correct senses with high reliability (we did not have multiple independent sense annotation). In cases of apparent ambiguity in the input text, the lexicographer still selected the most plausible sense. As the table shows, the system is able to select the correct sense of ambiguous open-class words 91%, or of all open-class words about 97% of the time, despite the fact that several micro-theories that impinge on WSD have not been implemented yet, including reference, anaphora, context, stylistics, or attitude/modality resolution. The results are significantly better than two common baselines reported in the literature: picking word senses at random (actually modeled probabilistically, not computed) and always picking the first sense.

The sources of disambiguation knowledge used in this experiment were syntactic category, subcategorization, semantic constraint checking, and a weak domain model. The table identifies what percent of disambiguation was achieved by the syntactic methods, as compared to semantic constraint checking via the ontological graph search. As an example, the total number of constraints checked in Text 1 was 4857, with an average of 231 constraints checked per sentence. This works out to about 14 constraints per word! The Hunter-Gatherer control structure (see Beale (1997)) for SDS building caches each constraint satisfaction call to the ontological graph search for reuse of results when the call is identical (resulting in about a 2/3 reduction in the number of explicit calls).

Notice that the syntactic analysis (including both syntactic category and syntactic subcategorization, as described in Section 3.4.2), contributed to about 38% of WSD results. This is consistent with results on verb pruning by subcategorization by Jing *et al.* (1997). Perhaps stronger WSD results could be achieved by syntax alone with a more informed parser, in conjunction with more extensive lexical information, such as syntactic classes of the type described in Levin (1991), Levin (1993), or Mitamura (1990); however, because of the strength of the semantic contribution in this approach, as well as for robustness against syntactic anomaly, ill-formedness, or mis-parsing, we do not intend to extend the syntactic disambiguation beyond the current level of disambiguating power.

As Table 1 shows, the first and third texts had significantly lower accuracy in WSD that the other two texts, because they had longer sentences, many more ambiguous words, and difficult constructs (e.g., ambiguous words embedded in appositions). In those two texts, in a number of cases a single word appeared multiple times, and was incorrectly disambiguated multiple times. For example, the Spanish word *operacion* occurred several times and the system was not always able to disambiguate correctly between its WORK-ACTIVITY, MILITARY-OPERATION, SURGERY, and FINANCIAL-TRANSACTION senses (although the SURGERY sense was easily eliminated). An additional theory of context or a stronger microtheory of domain would contribute to resolving this ambiguity in a fully-implemented end-to-end system.

The Mikrokosmos analyzer was then tested on a text that was not used for any stage of lexical acquisition, ontology acquisition, ontological graph search weight training, or any other stage of development. As Table 2 shows, results on this unseen text are not significantly different from re-

Table 2: Word Sense Disambiguation Results on Unseen Spanish Test Data

	Text 5
# of words in text	215
# of words/ sentence	26
# of open-class words	104
# of ambiguous open-class words	26
# of senses per ambiguous word	3.0
# of words resolved by syntax	9

Table 2: Word Sense Disambiguation Results on Unseen Spanish Test Data

	Text 5
# of words resolved by semantics	14
# correctly resolved: syntax + semantics	23
# correctly resolved: random guess	64
% correct of the ambiguous words	**88%**
% correct of all open-class words	**97**

sults on the training data, despite the fact that the syntactic parser used for this made numerous parse errors, and the analyzer itself had some variable binding problems due to previously unseen constructions. This unseen text contained 19 words that hadn't been entered into the Mikrokosmos lexicon. The Mikrokosmos analyzer produces dummy entries for all unknown words, marks them as nouns, and dynamically builds a **SEM-STRUC** mapping the meaning to ALL, the root concept in the ontology; this effectively means that semantic constraints will have little if any effect. These 19 words were treated as unambiguous in the WSD results in Table 2. Of these 19 unknown words, 12 were actually proper nouns, which could easily have been identified with high accuracy by Name Tagging technology, as described in Sundheim (1995). The name taggers would have also identified the proper nouns as referring to HUMAN, ORGANIZATION (or even FOR-PROFIT-CORPORATION), GEOGRAPHICAL-ENTITY (a place), or named ARTIFACT (such as a product), instead of just ALL; this additional level of information would certainly be useful for disambiguation of surrounding words. Of the other 7 unknown words, about half would have had multiple senses had they been entered into the lexicon.

The average number of senses per ambiguous word is 2.36 for the training data. This number is relatively low, in comparison with MRDs and resources such as WordNet. However, the sense distinction is finer than homograph level, so the results given below aren't comparable to Wilks and Stevenson (1996), or, for that matter, Yarowsky (1995), who only used binary distinctions. Although the lexicon was not acquired as a domain-specific lexicon, the fact that a mergers and acquisitions financial corpus was used to guide acquisition suggests that a higher percentage of senses relate to the training and testing domain than would be the case for a general resource; given the approach to semantic analysis described here, it is probably the case that domain-specific terms are more difficult to disambiguate than word senses from other domains.

In order to explore the effect of a more fine-grained lexicon (more senses per word) on disambiguation results, an additional experiment was run. For this experiment, an additional 40 senses were added to the lexicon entries of a total of 30 words used in the texts (these senses were generally beyond the domains encountered in the 400 training texts). The evaluation was re-run with this expanded lexicon (but without an expanded domain microtheory), giving overall results only 3.6% worse on total WSD.

The results, as a whole, are difficult to compare to other work for a number of reasons. Firstly, the results above are on Spanish, while the vast majority of other work is for English text. Second, we actually performed WSD on running text, while efforts such as Yarowsky (1995), Leacock *et al.* (1993a), Luk (1995), or Bruce and Wiebe (1994) gave results on a very small number of words (between one and twelve), but many occurrences. Third, our effort is not merely a word-sense tagging or WSD effort, as most of the efforts described in Light (1997) are, but a full semantic analysis effort, involving the building of SDS, which introduces a variety of complications that a tagging effort wouldn't encounter. Fourth, our lexical and ontological resources are of a substantially different granularity than most other efforts, which tend to use WordNet or LDOCE. That said, the results given on the experiments above are very competitive, especially since the scale of the effort is not toy; testing was not conducted on more texts not because the system wouldn't handle more texts, but because of the high cost of building answer keys for evaluation.

9. Application of the SDS-Building Mechanism: Metonymy Processing

Lakoff and Johnson (1980) identify metonymy as “using one entity to refer to another that is related to it.” Following Gibbs (1993), metonymy crucially differs from metaphor in that metonymy uses an entity to refer to another, related, entity from the same domain, whereas metaphor necessarily relies on the replacement of an entity from one domain by an entity from another conceptual domain.

As has been well-established in the literature, metonymic language use is pervasive in written and spoken language. NLP efforts addressing specific corpora, such as Hobbs and Martin (1987), Stallard (1993), and MADCOW (1992), all had to address metonymic phenomena because of its high frequency. The training and test data collected for this effort (as described below in Section 9.8 an Section 9.9) also found high volumes of metonymy in newswires.

9.1 Why Resolve Metonymy?

We find that we need to identify and resolve metonymy during the semantic analysis phase of Machine Translation for a number of reasons, given below. (Of course, some of these arguments assume that the generation component of the MT system is able to take advantage of the additional inferences and information that is provided as a result of the resolution.)

- The most compelling argument for resolving metonymy as part of the analysis process in MT is that metonymies do not necessarily translate literally into other languages. Although often they do translate felicitously, an informal investigation into the translatability of 15 examples of metonymy easily found a number of cases where a literal translation would be bizarre, misunderstood, or just ill formed. For example, in *The newspaper fired the editor in chief*, the word for *newspaper* (*shinbun*) must be rendered as *newspaper company* (*shinbunsha*) to make the example understandable in Japanese. In Kannada, the example *Alicia de Larrocha plays Mozart exquisitely* would be misunderstood if translated literally (would, at best, appear to refer to mind games that she played on him), as would an example such as *The sax has the flu tonight* (which would be understood as referring to the sax’s sound). These results are consistent with the more thorough field work in Kamei and Wakao (1992) and Wakao and Helmreich (1993) on English, Chinese, and Japanese; they cite additional examples, such as *He read Mao* being unacceptable in Chinese. The series of examples concerning the *Prix Goncourt*, developed by Kayser (1988) and extended by Horacek (1996) and others, also illustrate cases where well-formed metonymies in English are unacceptable in French or German; Horacek also presents other examples, such as *Mary finished her beer* and *we need a couple of strong bodies over here* as ill formed when translated into German.
- In addition to the cases where literal translation of metonymy is unacceptable, there are numerous other examples where the literal translation is understandable but not fluent. Examples of this include *Clinton invaded Haiti* when translated into Hebrew, *The newspaper fired the editor in chief* when translated into Ukrainian, and *We usually go out to lunch after church* when rendered in Japanese. Gibbs (1993) suggests that in metonymy “a thing may stand for what it is conventionally associated with”, so, since not all conventions are shared (or not shared with equal strength) across cultures and languages, there should be no expectation that a literally-translated metonymy will be felicitous.
- The replaced entity may need to be available for anaphoric and other referential mecha-

nisms. In the utterance *The sax has the flu tonight, so the boss docked his pay*, the pronoun refers to human (the musician) that the metonym replaced. Similarly, for identifying the definite reference in *I bought a used Volvo, even though the engine was shot,* one needs to identify that the company name is used metonymically for its product. Anaphora and definite reference function in various unique ways in different languages, so resolution is necessary for fluent translation.

- Agreement mechanisms may reference not the metonymic expression, but the replaced entity, in some examples. In the saxophone example above, the pronoun agrees with the replaced musician's gender, not the metonym's. In Japanese and other languages with counters or classifiers, expressions such as *six Volvos* require the classifier for cars, not for companies.
- Since word sense disambiguation (WSD) mechanisms typically rely on sentential context in some form, unresolved metonymies can cause inaccurate resolution. This is certainly the case for constraint-based WSD methods (such as the one described in Section 8, or methods using LDOCE *box codes*), domain-tag methods (such as those using *LDOCE subject codes*), *semantic distance* and other semantic net methods (such as Agirre and Rigau (1996), Collins and Loftus (1975), or Resnik (1995a)), or dictionary-definition-intersection methods (such as Lesk (1986) or Kozima and Furugori (1993)); all of these methods would be more accurate if the metonymies were resolved and the replaced entity added to the local context. Of course, these sorts of methods are often used to identify metonymies, so there is a circularity regarding whether WSD is needed to resolve metonymies, metonymies resolved to aid in WSD, or both needing to be done in parallel.

9.2 Related Computational Approaches to Metonymy

Two general classes of mechanisms for handling metonymy have been implemented (either implicitly or explicitly) in earlier work. In one approach, the lexical entries for argument-selecting heads can be extended to allow arguments to be of the type of typical metonyms (e.g., specifying that the agent of *announce* needs to be either a person, an organization, or a place of residence to handle expressions such as *The White House announced...* or *Tokyo announced...*); similarly, the lexical entries for typical metonyms can be expanded, typically by duplicating them with different type specification (e.g., having a second lexical entry for *White House* or *Tokyo* as a government organization). In addition to excessive proliferation or generalization of lexical entries and the inevitability of lexical acquisition gaps (resulting in missed metonymies and WSD errors), this approach misses the important insight or generalization that such processes are productive. Most efforts on sublanguages (e.g., Grishman and Kittredge (1986)) or in narrow domains, such as Stallard (1993), incorporate "domain-specific" lexical relaxations in their lexicons in this way, sometimes effectively avoiding the issue of metonymy.

The explicit approach involves the addition or incorporation of metonymy-handling processes to selectional constraint satisfaction, thus allowing full generalization; one danger here is overacceptance/overgeneration (which can substantially weaken word sense disambiguation), and the burden of acquisition of knowledge shifts from lexical to ontological or processing knowledge resources. The other systems described below are of this type.

A significant distinction in how metonymies are treated in computational systems that actually produce a representation of some sort is found in the reflex of the metonymy in the representation.

Sowa (1993), Hobbs and Martin (1987), Grosz *et al.* (1986), and Charniak and Goldman (1988) all rely on instantiating the entity replaced by the metonym and making it part of the representation (in addition to the surface metonym). In fact, the latter three assume metonymy as part of every relation between entities, instantiate an underspecified entity, then only delete it if constraints suggest that the intermediate entity is the same as one of the two actual entities.

On the other hand, Nunberg (1978), Pustejovsky (1991), Pustejovsky and Bouillon (1995), and others adopt a mechanism for coercing the argument role or the predicate to make the relation appropriate. In fact, Nunberg (1993) denies that some examples that are generally-regarded as metonymy actually are metonymies: he argues that in *We're parked around the corner* there is no metonymy, only coercion of the predicate, as indicated by agreement and anaphoric availability of the supposedly replaced entity. Stallard (1993), however, argues that while sometimes it is necessary to instantiate the replaced referent (what he calls *referential metonymy*), other times it is desirable to merely coerce the argument place (*predicative metonymy*), but considers both types as metonymy proper.

Given the MT and knowledge-based context of our work as described here, however, we find it always necessary to instantiate the replaced referent. One of the reasons, as identified in Fauconnier (1985), is that sometimes the same expression can produce (in different contexts) as antecedents either the metonym or the replaced expression (*Plato is on the top shelf, and it is bound in leather*, and *...and he is a very interesting author*); any system would need to have a highly accurate mechanism for handling context to be able to select one or the other with sufficient confidence. Stallard (1993) is only able to do so because of the extremely limited scope of the ATIS domain. Additionally, in our context of Machine Translation, we find it necessary to always insert the missing referent because the target language may require it to be lexicalized explicitly, as would be the case for many of the examples in Section 9.1 above.

While some of the approaches above instantiate the replaced entity, based on specific relational information and constraints on that relation, Sowa (1993), instantiates a concept (based on selectional constraints) and then attempts to determine the metonymic relation that holds between the inserted concept and the surface metonym, based on some inventory of lambda expressions encoding some kinds of background knowledge. This reliance on explicit background knowledge is not unlike Bouaud *et al.* (1996), who don't use an inventory of explicit metonymies, but use relational mini-Conceptual Graph models of background information. Their metonymy resolution search space expansion actually involves splicing together (joining) model graphs. They have no good way of preferring one CG model over another if the constraints are the same, and they use very weak constraints. As stated in their paper, they don't have sufficient resolution capability because the "semantic analyzer goes directly from grammatical relations to conceptual relations without any intermediate semantic representation", and thus they lack the case-role or thematic information which further constrains the arguments of the models. Thus they have to rely on their models or mini-scripts (in CG formalism), without which they aren't able to offer resolution for a given metonymy.

One element that many earlier approaches have in common is reliance on matching a fixed inventory of metonymic relations or patterns, as in Kamei and Wakao (1992), Fass (1986b), or Stallard (1993). However, an approach such as Hobbs and Martin (1987) assumes the existence of metonymy everywhere, but doesn't appear to have to identify the nature of the metonymic relation itself; likewise, Stallard doesn't identify the metonymic relation. Fass does attempt to identify the metonymic relation, using four inference rules that apply to specific topological relations over an

IS-A hierarchy; after identifying the relation, he substitutes in the replaced entity and then reprocesses the utterance from the beginning. However, he is unable to identify any metonymies that do not conform to these discrete rules.

All of the inventory-based approaches have a fundamental gap. As is evident from Nunberg (1978) or Fauconnier (1985), the number and types of relations or referring functions that may be used metonymically is effectively unrestricted. For that reason, semantic and pragmatic frameworks such as Grosz *et al*. (1986), Hobbs and Martin (1987), or Fauconnier (1985) assume that metonymic referring functions are effectively unrestricted. Unfortunately, for practical computational semantics system-building efforts, this means an explosion of the semantic search space beyond the already virtually-intractable. For this reason, our approach restricts the possible referring functions to individual relations or compositions of relations from the ontology. The discussions below will make evident that our approach uses both an inventory and a general (yet not unconstrained) mechanism to handle metonymies outside the inventory.

A further issue involves the interaction between metonymy processing in computational systems and WSD. (In some discussions, either one or the other problem is being tackled in a laboratory setting, so there is no requirement to integrate the solution into a complete system, thus the issue of integration never arises.) The approach taken in Kamei and Wakao (1992) and Wakao and Helmreich (1993) is to always accept any metonymy in their inventory, regardless of language; this hurts the WSD search space, by not restricting the search space to only consider metonymies licensed for a particular language. Additionally, if integrated into a full MT system, their approach would generate metonymies in a target language if and only if there was a metonymy in source the source language; Horacek (1996) shows why this results in infelicitous translations, in addition to limiting expressibility in generation.

9.3 Framework for Metonymy Processing

The metonymy identification and resolution mechanism is an integral part of the overall SDS-building process in our paradigm, as it is in Hobbs and Martin (1987) or Charniak and Goldman (1988), as opposed to relegating metonymy processing to an error-recovery process, as in Fass (1986b). Because it is an integral part of the word-sense disambiguation process, we gain efficiency and unified control, which has a high payoff because of the high prevalence of metonymy in text from real corpora, as discussed in Section 9.8 and Section 9.9.

A further benefit of this integration is that it correlates with psycholiguistic results concerning human metonymy processing, as discussed in Gibbs (1993):

> ... there are good reasons and experimental evidence to suggest that these tropes [metonymy, irony, hyperbole, understatements, oxymora, idioms, metaphor] do not require special cognitive processes to be understood, contrary to the widely held assumption in linguistics and philosophy that tropes violate, or ‘flout,’ norms of cooperative conversation.... a major reason why people use different tropes so frequently in everyday speech and writing is that human cognition is fundamentally shaped by various processes of figuration.

The approach described in this chapter relies on a fundamental observation about metonymy, namely that it reflects (conventional) semantic contiguity, as described in Gibbs (1993) or Jakobsen and Halle (1956). The premise of our approach is that relations in the ontology coincide with the relations of semantic contiguity at some level, thus the task of the ontological graph search is to identify the nature of contiguity in each case by identifying the best path.

Gibbs (1993) discusses how metonymic relations, identified by Nunberg (1978) or Fauconnier (1985) referring functions, reflect cultural conventions, thus language-specific conventions. For these types of metonymy, an inventory can be very useful; we make use of an inventory, for example, of about 40 frequent metonymic relations for English (probably the most comprehensive inventory built).

Gibbs (1983) also identifies that prior context can set up a mutually-understood local referring function: "any given instance of a referring function needs to be sanctioned by a body of beliefs encapsulated in an appropriate frame". But there are infinite such local contexts that can generate locally-sanctioned referring functions (all the "ham sandwich" types of metonymies, for example), thus an unrestricted range of notions of contiguity. While we aren't able to fully make use of context at this stage of development, the ontological graph search can make use of any ontological relation or predicate (event) in establishing a metonymic link. So any of the 300+ (non-inventory) relations in the ontology can all be identified as the contiguity relation and establish the metonymic link, if they provide the most plausible explanation for an apparently necessary constraint relaxation (if describing the problem from an abductive inference perspective).

The inclusion of both conventionally-established and locally-sanctioned referring functions thus corresponds with the approach taken within the ontological graph search program.

In general, the attitude taken here is consistent with Apresjan (1973), who identified that "Selectional restrictions can be in principle violated, e.g., for stylistic purposes. The semantic theory should not only make provision for such cases but it should also specify for each of them what he resulting stylistic effect or trope is." The mechanism described below crucially depends on violated selectional restrictions, and resolves the trope explicitly. Apresjan, however, believes that it is practically impossible to mark each noun in the dictionary with certain features (such as "displaceability") which therefore cannot be used for restrictions. By relying on the ontology to capture such features (instead of the lexicon), and by making extensive use of inheritance in the ontology, we find that we can use a very wide range of features for constraining relations.

Thus the metonymy-processing approach described below essentially consists of two steps: a) the application of the general constraint-satisfaction process (the ontological graph search process outlined in Section 5), and b) identification of the concept that was replaced by the metonym in the path returned by the graph search process.

This approach allows full use of the relations defined in the ontology. If only the strict *IS-A* relations from the ontology were used, with either vertical relaxation of constraints (see Section 8.2) or a relaxation utilizing a small set of topological relations over a hierarchy (such as Fass 1986, 1988), then the wealth of metonymic expressions would be unprocessable without either allowing excessive ambiguity or not recognizing numerous uninventoried examples of metonymy. The framework outlined here allows metonymic expressions to be processed by utilizing the general word-sense disambiguation mechanisms, namely semantic constraint checking and relaxation over the full range of metonymic relations, combined with taxonomic generalization; note, however, that not all combinations of relations or arcs in the ontology identify paths of acceptable weights, that is, the arc weight mechanism allows for identifying varying degrees of acceptability of relations that comprise potential paths between filler and constraint.

As inventoried in detail below in Section 9.6, the *metonymic arcs* reflect the types of metonymic relations which have been identified, such as *PART-OF* for the Part-for-Whole metonymy, *LOCATION-OF* for the Place-for-Event metonymy, *PRODUCTS* for the Producer-for-Product metonymy, etc. Thus for each identified metonymy, the arc(s) is found in the ontology that reflects the metonymy in defining the path from the metonym to the constraint. For example, in *he drove his V8...* the constraint on what can be driven is ENGINE-PROPELLED-VEHICLE, but the candidate filler is ENGINE (of a certain type). The part is the engine, the whole is the vehicle, and the arc from ENGINE to ENGINE-PROPELLED-VEHICLE is *PART-OF*; the potential filler is the metonym, and the constraint identifies what is being replaced. Thus in Producer-for-Product, a candidate filler (such as *Chevrolet*) has a certain relation, identified by the metonymic arc (such as *PRODUCER-OF*), to the constraint, which is what is being replaced (such as an automobile).

Heuristic IX. Metonymies in text are identified by the ontological graph search, when the best path traverses an arc which reflects a metonymic relation.

The general problem of metonymy handling is thus reduced to identifying the list of metonymic relations, establishing relations in the ontology to reflect these metonymic relations, and assigning weights to these arcs. The metonymic arcs would be less expensive than the rest of the unmentioned arcs, but more expensive than the weights for straightforward constraint satisfaction (i.e., *IS-A* and *INSTANCE-OF*). Since these arc weights are available to every constraint satisfaction check, metonymy processing is an integral part of SDS-building, not a special disjoint mechanism to be called upon failure. Yet if a straightforward constraint satisfaction path is found, the metonymic paths need not be pursued, thus not adding to the computational cost.

Once a metonymic relation is found by the constraint satisfaction process, the metonym needs to be represented. The metonymic relation is represented by a slot on the metonym, which is filled by an instantiation of the concept that the metonym replaces.

Heuristic X. In representing metonymies in the text meaning representation, it is necessary to make an inference (by instantiation) about the existence of the entity replaced by the metonym.

In other words, if X-for-Y is the metonymy, X is the metonym actually used, and Y is what it replaces, then in addition to instantiating X (from the lexical trigger), we also instantiate Y, and we connect X and Y with the metonymic arc reflecting the relation. Since every relation in the ontology has an inverse, X will have a slot *FU* filled by Y, and Y will have a slot FU^{-1} which is filled by X. A specific example of this appears below.

9.4 Determination of Arc Weights

As identified above, metonymy processing is essentially reduced to selection of a path including a metonymic arc by the constraint satisfaction mechanism (the ontological graph search). Thus success depends on having the appropriate arcs in the ontology, with the appropriate cost assessing mechanism. However, this cost assessment is specific to metonymy type as well as the individual context. The *PRODUCER-OF* arc, as the *COMPOSED-WORKS* arc (for musical composers), might appear only once in the ontology, since all concepts that are descendants will inherit the slot. Individual subconcepts for various kinds of products would not need to *define* this slot, although they may be able to *fill* the slot where the filler is known and fixed.

The heuristic that underlies this approach to metonymy processing is that the inventoried metonymic arcs will be traversed by the ontological graph search in constraint satisfaction if taxonomic (*IS-A*) relations do not satisfy the constraint, but would be preferred to arcs identifying other (non-inventoried) real-world relations between the candidate filler and the constraint.

Heuristic XI. Inventoried metonymic relations are less preferable than taxonomic (*IS-A*) relations, but still preferable over all other relations over the ontology.

For some of the metonymic relations (such as Part-for-Whole), the chaining of more than one traversals of a metonymic arc (such as the *PART-OF* arc) is acceptable; for others (such as Place-for-Event), the dynamic arc weight mechanism (described in Section 5.2) increases the arc cost of that particular arc to near-infinite cost, once one occurrence of that arc appears in the path currently being explored.

9.5 Metonymy Processing: An Example

Once an acceptable path is found, the preference of the reading is adjusted to reflect the cost incurred in allowing the metonymy (vs. a straightforward taxonomic satisfaction over *IS-A* arcs) and the path is analyzed to produce the appropriate combination in the instantiation-combination process. Thus, for example, for the sentence *Lynn drives a Saab*, the semantic constraint for the appropriate slot of the appropriate sense of the verb *drive* would be *ENGINE-PROPELLED-VEHICLE. Yet the potential filler *Saab* is of type (or a subtype of) *FOR-PROFIT-MANUFACTURING-CORPORATION which is a violation of the constraint. The ontological concept *FOR-PROFIT-MANUFACTURING-CORPORATION has a slot *PRODUCER-OF*, which has an “inverse” relation called *PRODUCED-BY*. The path which is found by the ontological search process is (expressed in the `[SOURCE-NODE` *OUTGOING-ARC* `--> DESTINATION NODE]` notation):

ONTOLOGY PATH:

[**FOR-PROFIT-MANUFACTURING-CORP417** *PRODUCER-OF* --> *AUTOMOBILE]
[*AUTOMOBILE *IS-A* --> *WHEELED-ENGINE-VEHICLE]
[*WHEELED-ENGINE-VEHICLE *IS-A* --> *ENGINE-PROPELLED-VEHICLE]

If **FOR-PROFIT-MANUFACTURING-CORPORATION417** were a concept in the onomasticon (with knowledge about Saab Scania AB), i.e., with slot/fillers such as (*NAME* $SAAB), (*PRODUCER-OF* *AUTOMOBILE *JET-AIRCRAFT), the above path would be found. But even if that world knowledge tidbit (about Saab’s products) were not available, the path that the ontological search process finds is:

ONTOLOGY PATH:

[**FOR-PROFIT-MANUFACTURING-CORP417** *PRODUCER-OF* --> *ARTIFACT]
[*ARTIFACT *SUBCLASSES* --> *VEHICLE]
[*VEHICLE *SUBCLASSES* --> *ENGINE-PROPELLED-VEHICLE]

The latter path has a lesser preference (i.e., a greater cost) than the former, because of the more expensive traversed arcs (*SUBCLASSES* is always more expensive than *IS-A)*, but illustrates that the mechanism is still able to identify the metonymy in the absence of the specific product information.

Once a path is found (let’s assume the latter no-onomasticon-information case), it is inspected for the appearance of a metonymic relation arc. If such an arc is found, the inverse of that arc is available in constructing the final SDS of the sentence.[1] For the above example, the most specific information that is available from the path (identifiable by following *SUBCLASSES* arcs after the metonymic arc) is utilized in making an inference about the replaced metonym and instantiating

1. The inference notation for use in the TMR was developed by Sergei Nirenburg and Kavi Mahesh and implemented by Stephen Beale.

an appropriate concept **%ENGINE-PROPELLED-VEHICLE460**:

```
TMR:
    (DRIVE435
        (AGENT  (VALUE PERSON440))  ; abbreviated of course
        (THEME
            (SEM *ENGINE-PROPELLED-VEHICLE)
            (VALUE
                (source FOR-PROFIT-MANUFACTURING-CORP417)
                (inference inference306 ENGINE-PROPELLED-VEHICLE460))))
    (PERSON440
        (NAME $LYNN))
    (inference480
        (TYPE metonymy)
        (ENGINE-PROPELLED-VEHICLE460
            (MANUFACTURED-BY
                (VALUE FOR-PROFIT-MANUFACTURING-CORP417))))
    (FOR-PROFIT-MANUFACTURING-CORP417
        (NAME (VALUE $SAAB))
        (PRODUCER-OF inference480
            (SEM *ARTIFACT))
            (VALUE ENGINE-PROPELLED-VEHICLE460)))
```

(Here the SEM facets are left in from processing for illustration, but are not properly part of the TMR). We could augment the episodic memory about Saab (if this reading were found to be the most plausible one for the sentence) to include information about the type of product that this manufacturer produces.

The inference notation used in this example is more generally available to represent inferences made by a variety of specialized mechanisms or microtheories during the course of semantic analysis. This notation preserves the original literal interpretation, while making available the replaced entity that was inferred to exist by the metonymy processing mechanism; this inferred information (in this case, the existence of a produced vehicle) satisfies the goals of metonymy resolution mentioned above in Section 9.1.

The real challenge to this approach is when the episodic memory has no information about the word *Saab* at all. As a system heuristic, one of the most likely possibilities for an unrecognized word in noun position (particularly if we utilize the English capitalization information) is that it is a name for some named entity (i.e., (**NAMED-ENTITY239** (*NAME* "Saab"))). In fact, we can do better by relying on Name Tagging (i.e., shallow extraction) capabilities that are available for integration into MT and other NLP applications.[1] Name Tagging technology can suggest, with high reliability (93-94%) that the string represents an organization, say **ORGANIZATION 240**, in which case we can expect the path found by the ontological search process to be:

1. Numerous such Name Tagging systems, with accuracy very near human, have been evaluated in the scope of the Message Understanding Conferences (MUCs) and are described in Sundheim (1995).

```
ONTOLOGY PATH:
        [ORGANIZATION239  INSTANCE-OF  -->  *ORGANIZATION]
        [*ORGANIZATION  PRODUCER-OF  -->  *ARTIFACT]
        [*ARTIFACT  SUBCLASSES  --> VEHICLE]
        [*VEHICLE  SUBCLASSES  -->  *ENGINE-PROPELLED-VEHICLE]
```

This path, albeit expensive, is found by the search algorithm; the challenge of this approach is to adjust all of the arc weights to return these weights with fairly low cost relative to other returned paths, as a subset of the problem identified in Section 5.2.

9.6 Inventory of Metonymic Relations

Although not receiving nearly as much attention in the literature as metaphor, there have been a few attempts in the various literatures to categorize metonymy into types. None of the inventories are comprehensive enough to support the population of a working ontology for use in the analysis of real-world texts. Thus the strategy used here to build such an inventory consisted of combining multiple sources in the literature, experiments and analysis of corpora, and carefully filtering inventories of other kinds of semantic relations (e.g., syntagmatic and paradigmatic lexical relations, meaning change, cognitive meronymic classification) for relations that do reflect metonymic use of language in English.

As mentioned above, it is not possible to build an exhaustive inventory of metonymy (especially in the pragmatic relation group, below). So although this inventory was compiled for the purpose of seeding the metonymy processing mechanism, it is augmented with the mechanism for handling novel or unexpected (i.e., uninventoried) metonymic relations and combinations (chains) of metonymic relations.

Below is a combined inventory of metonymy types derived from various sources; since these works span theoretical linguistics, lexicography, cognitive science, philosophy of language, and computational linguistics, and did not necessarily deal explicitly with metonymy, the relations identified below are those for which metonymic examples could be found in the cited work or constructed. Each relation below is marked by one or more initials for the various sources from which the relation was derived:

- (A) for regular polysemy relations as identified in Apresjan (1974)
- (F) for the computational semantics treatment of metonymy in Fass (1986)
- (K) for the inventory of metonymy treated by Kamei and Wakao (1992)
- (L) for the seminal treatment of metaphor and metonymy in Lakoff and Johnson (1980)
- (M) for lexical functions which can serve as metonymy in Mel'chuk and Zholkovsky (1984)
- (N) for referring functions described in Nunberg (1978)
- (S) for a discussion of diachronic and synchronic change of meaning in Stern (1965)
- (W) for the categorization of part-whole relations (meronymy) in Winston *et al*. (1987)
- (Y) for the examples of metonymy in Yamanashi (1987)

Metonymies found in the course of the work described here are also marked:

- (O) for relations that were identified in experiments during the course of this work.

Notice the subcategories identified below for some of the categories (the subcategories do not necessarily define the entire partition for the category). The top level distinction (Semantic vs. Conventional vs. Pragmatic) is based on Yamanashi (1987), and is in accord with distinctions among types of reference described in Nunberg (1978).

9.6.1 Semantic Metonymic Relations

Metonymic relations that fall into this group of relations reflect fundamental semantic relationships between the metonym and the replaced entity, and are understood independent of context. In many cases the metonym could be lexically defined to mean the replaced entity, as described in Yamanashi (1987). These metonymies tend to be very productive, and in many cases are cross-linguistic, because they reflect fundamental semantic relationships.

- Part-for-Whole:
  - Material-for-Object-made-from-it: as in *He drinks a lot of alcohol*, or *He pumped lead into the assailant* (for *bullets*), or *glass* (W, S, A, Y, O)
  - Component-for-integral-object: as in *he drove his V8*; or *He always carries his blade* (for *sword* or *knife*) (W, S, L, F, Y)
  - Member-for-collection: as in *We need to pay attention to Ivan* (meaning Russia or the Russians) (W, O)
  - Body-part-for-Animal: as in *there are many new faces around here, She is wearing mink,* or *We need another pair of hands* (L had Face-for-Person, S, A, K)
  - Container-for-Contents: as in *he drank two whole bottles.* (F, S, A, K, Y)
  - Container-for-Measure: as in *he drank a pitcher of soda* (A)
- Producer-for-Product: as in *he drove his Ford...* (L, K, O).
  - Artist-for-Artform: as in *She plays Mozart exquisitely* (F) Except for some very well-known inventions (such as engines referred to as *diesels* or *wankels*), this type seems to be restricted to literature, art, or music.
- Role-for-Person: as in *the President* (A, O)
  - Voice-for-Singer: as in *the baritone will stand on the right* (A).
- Information-conveyer-for-Information: as in *Today's paper will cause the President to issue a retraction.* (K, O)

9.6.2 Conventional Metonymic Relations

The metonymies in this category are also understandable independent of any special context or pragmatic environment. But these metonymies are more the reflective of conventional associations than the semantic relations, and understanding these metonymies may require encyclopedic knowledge. These metonym generally cannot be lexically defined to refer to the replaced entities. Because the relationship is conventionalized, a given language or culture may not share a particular convention that may be prevalent in another language, resulting in non-translatability.

- Object-for-Material: as in *This jacket is made of recycled soda bottles* (O)
- Institution-for-PeopleResponsible: as in *IBM announced an agreement*...(L, O)
- Controller-for-Controlled: as in *Clinton invaded Haiti,* (L, O)
- Place-for-Entity

- ••Place-for-Occupants: as in *the pit* (for *orchestra*) (S, A, Y, O)
- ••Place-for-Institution: as in *Tokyo announced...* (L, O)
- • Symbol-for-Thing-symbolized: as in *He fought for the Crown*, or *He hung up The Last Supper in the corner* (meaning a picture of that event), or *I have only three names so far for tomorrow's luncheon.* (S, A, O)
- ••Concrete-for-Abstract: as in *Max is down to his last pennies* (meaning that he has no means or resources) (Y, O)
- • Product-for-Producer: as in *PowerPC announced a new version of the chip...* (O)
- • Entity-for-Action (O)
- ••Co-Agent for Activity: as in *The Orioles played the Brewers* (F, A)
- ••Instrument-for-Action: as in *Under the baton of Ozawa, the concerto took on a new life* (S, A, O)
- ••Place-for-Event: as in *Antietam was a critical point in the war, After my shower,...* or *We go out to lunch after church.* (L, S, A)
- ••Organ-for-Capability: as in *Paul's nose can identify the year of any burgundy, The point guard had hot hands tonight,* or *Max is the brains of the company* (S, O)
- ••Body-Part-for Disease: as in *She suffers her joints in humid weather*, or, in Russian, *she has kidneys* (translation) can mean that she is afflicted with a disease of the kidneys (M, A)
- • Action-for-Entity: as in *Accounting won the annual softball game.* (O)
- ••Production-process-for-Product: as in *a painting, an etching* (O, A)
- ••Event-for-Location: as in *He arrived at the dinner at 8:15*. (O)

9.6.3 Pragmatic Metonymic Relations

These metonymies necessarily require a specific pragmatic context to establish the metonymic entailment. Such metonymic relationships are not conventionalized, lexicalized, or reflective of a fundamental semantic defining condition. Many of these cases will not be translatable into other languages verbatim.

- • ObjectUsed-for-User: as in *the sax has the flu tonight* (L, A, O).
- ••Article-of-Dress-for-Person: as in *The boardroom is overrun by suits*, or *The redcoats are coming*. (S, A)
- ••Equipment-for-Operator: as in *The F16 requested clearance to land* (O)
- • User-for-ObjectUsed: as in *Jack is parked around the corner.* (O)
- • Feature-for-Activity: as in *I'm going to spend money this afternoon* (for *going out shopping*) (W, O)
- • Generic-for-Specific: as in *That animal needs to be fed* (for *dog*) (M, Y, O)
- • Group-for-Individuals: as in *The Canadians won the pennant* (referring to a group of individuals forming a sports team), or *I want you to meet this couple* (referring to two individuals married to each other). (M)
- • Abstract-for-Concrete: as in *Power and privilege will no longer shout down the voice of the people*. (O)

- Other: Other physio-spatial or arbitrary context-specific relations, such as *The ham sandwich needs another fork* (N, O).

In general, most non-inventoried metonymies that were found in the English corpora were of the pragmatic group; for this reason, we include the Other category in this group.

9.6.4 Utility of this classification

There are some examples which seem to fall into multiple categories: *The White House announced that...* could be either Symbol-for-Thing-symbolized or Place-for-Occupants. It is also the case that it isn't clear which high-level group of relations a given metonymic relation falls into, particularly between the Semantic and Conventional groups. This classification is primarily for conceptual organizational purposes, and is not reflected by the processing mechanism described in this chapter.

There is a group of alternations that reflect a semantic relation that could be arguably treated as either metonymy, regular polysemy (i.e., handled by Lexical Rules in our format or by generative processes in Pustejovsky (1995)), or derivational processes.

- Product-for-Plant: (or is it *vice-versa*?) as in *raspberry, cotton, rose, mustard,* or *cedar* (A)
- Animal-for-Meat: as in *I'll have the goose* (A)
- Music-for-Dance: as in *waltz, jig*, etc. (A)
- Phenomenon-for-Discipline: as in *history, mathematics* (A)
- Government-form-for-Country: as in *a democracy* (A)

9.7 Metonymic Relations in the Ontology

This inventory is abstracted from the previous section, with certain metonymies weeded out for lack of productivity (often because there is only a limited possibility of examples of the metonymy, and those are diachronically lexicalized). For each metonymic relation below, we list a relation that is used in the ontology to represent the relation between the referent and the metonym (i.e., from the thing being replaced to the thing that replaces it), along with the inverse relation (which is what actually appears in the path in a filler-to-constraint search).

- Part-for-Whole:
    - Material-for-Object-made-from-it: *MADE-OF/MATERIAL-OF*
    - Component-for-integral-object: *HAS-PARTS/PART-OF*
    - Member-for-collection: *HAS-MEMBER/MEMBER-OF*, can also be reflected by a *MEMBERS* slot in a TMR set notation.
    - Body-part-for-Animal: *HAS-PARTS/PART-OF* with a `SEM` constraint of EXTERNAL-ANIMAL-PART on the *RANGE*.
    - Container-for-Contents: *CONTAINED-IN/CONTAINS*
    - Container-for-Measure: uses quantity and measure representation
- Producer-for-Product: *PRODUCED-BY/PRODUCER-OF*
    - Artist-for-Artform: *CREATION-RELATION/INVERSE-CREATION-RELATION*, with subtypes such as *COMPOSED-BY/COMPOSER-OF*, *AUTHORED-BY/AUTHOR-OF*, *INVENTED-BY/INVENTOR-OF*, etc.

- Role-for-Person: typically no relation necessary, because social roles are subtypes of humans in the ontology
- Information-conveyer-for-Information: *CONVEYED-BY/CONVEYS*
- Object-for-Material: *MATERIAL-OF/MADE-OF*
- Institution-for-PeopleResponsible: *HEAD-OF/HEADED-BY, OWNER-OF/OWNED-BY, CONTROLS/ CONTROLLED-BY* or even *MEMBER-OF/HAS-MEMBER*; often in series with Role-for-Person, because constraints on *HEADED-BY* etc. are SOCIAL-ROLEs, not HUMANs.
- Controller-for-Controlled: *HEADED-BY/HEAD-OF, OWNED-BY/OWNER-OF,* and *CONTROLLED-BY/CONTROLS*
- Place-for-Entity: *LOCATION/LOCATION-OF*
  - Institution-for-PeopleResponsible: *LOCATION/LOCATION-OF* or *OCCUPANTS-OF/OCCUPANTS* where the occupants would be constrained to be HUMAN or ORGANIZATION.
  - Place-for-Institution: *LOCATION/LOCATION-OF* where the institution is constrained to be an ORGANIZATION
- Symbol-for-Thing-symbolized: *SYMBOL/SYMBOL-OF, INVERSE-EVERYDAY-NAME-RELATION/ NAME-OF (for pictures etc.)*
  - Concrete-for-Abstract: varies
- Product-for-Producer: *PRODUCER-OF/PRODUCED-BY*
- Entity-for-Action
  - Co-Agent for Activity: *ACCOMPANIER/ACCOMPANIER-OF*
  - Instrument-for-Action: *INSTRUMENT/INSTRUMENT-OF*
  - Place-for-Event: *LOCATION/LOCATION-OF*
  - Organ-for-Capability: *INSTRUMENT/INSTRUMENT-OF* on PERCEPTUAL-EVENTs, with fillers constrained to SENSORY-ORGANs
- Action-for-Entity: *ROLE-FOR-ACTIVITY/ACTIVITY-FOR-ROLE*, *AGENT-OF/AGENT*
  - ProductionProcess-for-Product: generally reflected by the *THEME-OF/THEME* slot of an event
  - Event-for-Location: *LOCATION-OF/LOCATION*
- ObjectUsed-for-User: *UTILIZES/USED-IN*, and other various slots
  - Article-of-Dress-for-Person: *WEARING/WORN-BY*
  - Equipment-for-Operator: a two-step relation, first *INSTRUMENT/INSTRUMENT-OF* then *AGENT-OF/AGENT*, or a one-step *OPERATOR-OF/OPERATED-BY* relation
- User-for-ObjectUsed: *INSTRUMENT-OF/INSTRUMENT* of an event after an *AGENT/AGENT-OF*, or a one-step *OPERATED-BY/OPERATOR-OF*
- Feature-for-Activity: this relation would pick out a single (most salient?) subtask of a complex event; these subtasks or subevents would need to be represented by script-like mechanisms as in Cullingford (1981), in contrast with all the core relations which are represented by simple relations from the ontology. In the absence of a well-defined theory of complex events in our paradigm, these types of examples are not currently treated in the work described.

- Generic-for-Specific: just the *IS-A/SUBCLASSES* relation.
- Group-for-Individuals: the relation appears to identify an aggregate unit of some sort; potentially this could be represented by *MEMBER-OF/HAS-MEMBER* arcs from the ontology. However, the aggregate entity needs to be treated at its first mention in the text, before any such membership links are established. This would need to be remedied by default reasoning, as represented by relations such as *TYPICAL-MEMBER-OF/TYPICALLY-HAS-MEMBER*. Actual set membership is represented in the TMR by using the set notation, which is treated entirely non-ontologically, thus would not be available to the metonymic treatment mechanism as described in this section.
- Abstract-for-Concrete: any of a variety of relations relating EVENTs or OBJECTs to specific OBJECTs
- Other: any of the other (350 or more) relations in the ontology may be needed for specific metonymic constructions in input texts

Notice that there are cases of both relations and their inverses as allowed metonymic links, and also note that some metonymies that are distinct in the literature (as reflected by the inventory) end up with the same relation.

The biggest challenge with some of the metonymies where there wasn't a specific slot specified is that the triggering conditions may differ from the canonical metonymy, where a selectional restriction violation is a clear indicator that some kind of relaxation is necessary. In particular, there might not be any selectional restriction violation for the examples given for some of the pragmatic group metonymies. In other words, for the example *I'm going to spend money this afternoon* there would be no indication that a trope of some sort is being used, because the literal interpretation is perfectly adequate. Chances are that there is no selectional restriction violation in the examples such as *That animal needs to be fed* either. If we adopt the strategy of variable depths of description discussed in Hirst and Ryan (1992) or conditional interpretation of Pollack and Pereira (1988), however, we can identify that all of these secondary phenomena can be handled adequately at the literal level, under the premise that for any realization in another language in a MT application, the same literal level of description might be adequate. In applications requiring more elaborate event or domain inferencing, however, these examples would need to be treated (but fall outside the scope of the current work).

9.8 Knowledge Base for Metonymy Processing

The knowledge required for processing metonymy by the ontological search mechanism is not specifically differentiated from the constraint satisfaction data requirements of the overall processing mechanism, which includes the relaxation mechanism. In other words, the same list or table of arcs and weights that were described in Section 5 and Section 7 are used for metonymy processing. Those static resources do, however, reflect arcs and weights that are used for identifying and resolving metonymy. The knowledge acquisition consisted first of identifying the arcs that needed special treatment because they are used in resolving frequently-occurring metonymies, then second by setting weights for those arcs by the automated training mechanism (using simulated annealing) described in Section 5.3.2. The latter part of the task, however, required manually building a training data set, as described in the same section above.

The example below illustrates the training data. The example from the corpus is quoted, followed by an enumeration of the metonymy categories in effect in the example. The matrix verb is the source of constraints on the metonym in this case, so the concept is listed, along with the constraint that it places on the *AGENT* role. The path given in this example needs to be matched by the ontological graph search exactly.

```
;;; "The White House said it does not know" (USA Today)
;;; Metonymy Type: PLACENAME-FOR-OCCUPANTS
;;; Metonymy Type: ROLE-FOR-PERSON
;;; "said" = ASSERTIVE-ACT
;;; ASSERTIVE-ACT.AGENT= HUMAN (SEM constraint)
WHITE-HOUSE (HUMAN)
     (((WHITE-HOUSE -)
       (PRESIDENT OCCUPANT)
       (ELECTED-GOVERNMENTAL-ROLE IS-A)
       (GOVERNMENTAL-ROLE IS-A)
       (SOCIAL-ROLE IS-A)
       (HUMAN IS-A)))
```

9.8.1 Training the Static Weight Assignment Mechanism

The training process for the static weight assignment mechanism simply produces a weight for each of the arcs represented in a manually-produced inventory of arcs, mostly reflecting the arcs (actually, the second of each pair) identified above in Section 9.7. In our experiment, the arc types that receive special weights are manually specified, and the training mechanism assigns weights. It would have been possible for the training mechanism to assemble the list of arcs, as well, by examining the arcs reflected in the training data; one drawback of the latter approach would have been the inability to call out specific arcs that aren't used in the training data, in expectation of their occurrence in other corpora.

This section describes the training process and the success of metonymy identification on the training data. Section 9.9 describes the success of metonymy identification on a test data set that was not used in training.

The data acquisition effort involved three separate data sets. Table 9A below addresses Data

Table 9A. Metonymy Training Results on Data Set 1 (Static Weight Assignment)

	Data Set 1 (Collected examples)		
	% of Total	# Correct: Manual Weights	# Correct: Trained Weights
Symbol for Symbolized	1.7%	0/1	1/1
Role for Person	1.7%	1/1	1/1
ProductionProcess for Product	1.7%	0/1	1/1
Product for Producer	5.1%	3/3	3/3

Table 9A. Metonymy Training Results on Data Set 1 (Static Weight Assignment)

	Data Set 1 (Collected examples)		
	% of Total	# Correct: Manual Weights	# Correct: Trained Weights
Place for Institution	1.7%	1/1	1/1
Other	24.1%	11/14	14/14
ObjectUsed for User	3.4%	2/2	2/2
Object for Material	3.4%	0/2	2/2
Instrument for Action	1.7%	1/1	1/1
Institution for PeopleResponsible	18.9%	7/11	11/11
InfoConveyer for Information	1.7%	0/1	1/1
Generic for Specific	5.1%	0/3	3/3
Event for Location	3.4%	2/2	2/2
Equipment for Operator	3.4%	2/2	2/2
Entity for Action	1.4%	1/1	1/1
Concrete for Abstract	3.4%	1/2	2/2
Action for Entity	10.3%	1/7	7/7
Abstract for Concrete	6.9%	4/4	4/4
TOTAL	100%	37/59 = 62.7%	59/59 = 100%

Set 1, used as a training data set. The data set consisted of 56 examples in English (three involved a chain of two metonymies) that included artificial examples concocted to illustrate particular metonymies, in addition to actual examples that were assembled from the literature and from four Spanish newswire articles. This data set essentially reflects an opportunistic collection of metonymies, and is in no way exhaustive or reflective of the distribution of metonymies over a corpus. The table shows the results of the ontological graph search process on the data set, first using static weights that were assigned based on intuition. The table also reflects the results of the process using weights produced by the simulated annealing training process; the training was able to produce a set of weights that accounted for 100% of the training data. A typical set of such weights is given below:

IS-A	0.979727
SUBCLASSES	0.762831
HAS-MEMBER	0.787453
PRODUCER-OF	0.779002
PRODUCED-BY	0.80813
INSTRUMENT-OF	0.741613
REPRESENTS	0.793012
HEADED-BY	0.889488
LOCATION	0.767951
OWNED-BY	0.714717
MADE-OF	0.941093
ACTIVITY-FOR-ROLE	0.702713
NAME-OF	0.9999
AGENT	0.832912
THEME	0.9666362
THEME-OF	0.601492
SOURCE	0.906768
CONVEYS	0.69802
RANGE	0.825514
DOMAIN	0.973727
HAS-ELEMENT	0.836513
HAS-AS-PART	0.602793
PART-OF	0.86144
MEMBER-OF	0.982538
INSTRUMENT	0.863454
HEAD-OF	0.82394
OPERATED-BY	0.802672
CONTROLLED-BY	0.669301
LOCATION-OF	0.74336
MATERIAL-OF	0.663279
SYMBOL-OF	0.656622
INVERSE-CREATION-RELATION	0.796057
CONTROLS	0.9999
OWNER-OF	0.62836
USED-IN	0.962146
EQUIPMENT-FOR	0.875004
ELEMENT	0.632047
WORN-BY	0.536994
ACCOMPANIER-OF	0.859984
OPERATOR-OF	0.889231
OCCUPANT	0.956381
EMPLOYER-OF	0.869145
ACCOMPANIER	0.504643
INSTANCE-OF	1.0
NAMED-INSTANCE-OF	1.0
*	0.58028

The last line reflects the weight used for all arcs not explicitly inventoried.

This set of weights (produced by training on Data Set 1) was tested against Data Set 2. This second training set was produced more systematically from English-language newswire, specifically the February 9-11 1997 edition of USA Today (hardcopy) and the February 11 1997 edition of the on-line edition of the San Jose Mercury News. Table 9B below shows the distribution of

Table 9B. Metonymy Training Results On Data Set 2 (Static Weight Assignment)

	Data Set 2 (SJMN and USA Today)					
	# Correct: Trained on DS 1	#Wrong: Due to missing concept	#Wrong: Due to bad path	#Wrong due to missing link	#Wrong after concepts and links added (before retraining)	# Correct: Trained on DS1+DS2
Role for Person	2/7		2/5			7/7
ProductionProcess for Product	1/1					1/1
Product for Producer	8/9	1/1			2	9/9
PlaceName for Occupants	0/5		2/5	3/5 1 link		5/5
Place for Institution	3/4		1/1			4/4
Other	0/3	2/3 2 concepts		1/3	1	2/3
ObjectUsed for User	0/1			1/1	1	0/1
None	7/7					7/7
Instrument for Action	0/1		1/1			0/1
Institution for PeopleRespons.	30/36	1/6	4/6	1/6		36/36
InfoConveyer for Information	4/4					4/4
Feature for Activity	1/4			3/3 1 link		4/4
Event for Location	1/2					2/2
Equipment for Operator	0/5			5/5 2 links	5 1 link	5/5
Entity for Action	0/3	1/3		2/3 1 link	2 1 link	2/3
Concrete for Abstract	3/3				1	2/3
Action for Entity	2/3				1	3/3

Table 9B. Metonymy Training Results On Data Set 2 (Static Weight Assignment)

	Data Set 2 (SJMN and USA Today)					
	# Correct: Trained on DS 1	#Wrong: Due to missing concept	#Wrong: Due to bad path	#Wrong due to missing link	#Wrong after concepts and links added (before retraining)	# Correct: Trained on DS1+DS2
Abstract for Concrete	1/1					1/1
TOTAL	62/99, but 67%[a]	5/32 5 concepts	10/32	16/32 4 links	86/99 but 85%	94/99 but 94.25

a. The TOTAL reflects percentage of correct examples; because some examples involve chained metonymies, the ratios in the breakdown (reflecting individual metonymies) do not match the total.

metonymy types in this data set, which consists of 88 examples for a total of 99 metonymies (because of chained metonymies in some examples). Just using the ontology as-is and arc weights produced by training on Data Set 1, 67% of the examples were covered (compared to chance, where less than 5% of the examples would be covered). The table also shows the error analysis at this level, also broken down per metonymy type. Only four additional links needed to be added to the ontology to account for the 99 metonymies, as well as five additional concepts. After those concepts and links (slots) are added, using the weights trained on DS1, 85% accuracy is achieved. Note that both numbers (67% and 85%) include a 3 point penalty due to apparent metonymies that the developer wasn't sure how to represent, resulting in automatic deductions for those examples.

After the ontology was augmented as required, new weights were produced by simulated annealing. The annealing run used the same annealing schedule and Cauchy cooling rate described in Section 5.3.2, and began by initially "heating" the temperature (by 10 complete randomizing annealing iterations) to an energy of *0.97* (in the interval *[0.0, 1.0]*). The simulated annealing run resulted in final energy of *0.0575*, or 94.25% example accuracy (percentage of example sentences correctly analyzed, as compared to metonymic link accuracy, where examples with a chain of multiple metonymies count multiple times). Of the remaining errors (i.e., metonymic relations not found by the ontological search program), one is unsolvable by the current approach. The example, *Eddie Jones had a hot hand in today's game* has no selectional constraint violations (and is, in fact, understandable and incorrectly acceptable literally).[1] Handling this type of non-literal expression is beyond the scope of the work described here, and would require a substantially differ-

1. This example is from CNN, dated February 9 1997, not from San Jose Mercury News or USA Today, so doesn't really belong in this data set.

ent approach.

Of the other four examples that were not solvable after training, one is actually ambiguous, and the ontological search mechanism suggested a reading not supported by context: *Fujimori told Peruvian radio that*[1]... appeared in a context which suggested that he talked to the nation via radio, vs. talking to the people in charge of the Peruvian radio service, as the ontological search program suggested. Two of the other examples, *Other dinners brought in more money* and *The dinner is adding to the questions being asked about fund-raising activities*, were incorrectly analyzed as using "dinner" to refer to the people who prepared the dinner, not the people who attended the dinner (in the former case); in is unclear how to analyze the latter of these, which is complicated by ellipsis, so there is no correct answer given in the training data, resulting in an automatic failure.[2] The last of the incorrect examples, *... will move people from welfare rolls into jobs,* also involves some metaphorical or elliptical mechanism.[3]

9.8.2 Training the Transition Table

The training mechanism for the dynamic transition-table-based weight assignment mechanism also just assigns weights. In preparation for the automated training, however, the developer needs to manually suggest the various states that may be encountered. In addition, for the sake of efficiency, it is useful to group arcs that could be expected to have similar behavior into arc clusters (for example, grouping *CONTAINS*, *HAS-MEMBER*, *HAS-AS-PART,* and *HAS-ELEMENT*). This is merely to reduce the search space for the simulated annealing, as well as for adding robustness to handle metonymic expressions that were not encountered in the training data; thus instead of having 45+ inputs requiring different transition table columns, we deal with about 25. Still, however, the transition table has so many entries that it would require a very large training corpus to get even single examples of each state for the training process. For that reason (and, again, to improve robustness), cells are grouped to reflect similar expected arc weights; for example, for the grouping of arcs mentioned above we have six groups of cells, with the grouping reflecting whether the previous path is an *IS-A* vs. a *SUBCLASSES* path, etc. So instead of a transition table requiring 46 (unique slots) * 40 (states) = 1840 cell values, we only require a total of 80. The simulated annealing mechanism can still handle that number of parameters (although there still are some parameters which can vary freely because the training data never causes those states to be reached).

Other than this simplification, the process of training of the transition table paralleled the static mechanism training process described above. Table 9F, on page 155, illustrates the transition table built manually for handling metonymy, which also served as the starting point for training. Table 9G, on page 158, shows the grouping of transition table cells as described above; the first element of each cell reflects the cell grouping, while the second element of each cell reflects the next state. For each cell grouping, the simulated annealing training mechanism sets a value in *[0.0, 1.0]* (actually in *[Threshold, 1.0]*).

Results of training on Data Set 1 and Data Set 2 are given in Table 9C, which shows the accuracy of the search process when trained on the data set indicated and applied to the data set indicated in the top row. As expected, applying the training results to the data trained on typically approaches (but doesn't always reach) 100%; when using weights generated by training on a data

1. This example due to the on-line San Jose Mercury News service, article dated 11 February 1997.
2. Both these examples due to USA Today, 9-11 February 1997 edition (hardcopy).
3. This example due to the on-line San Jose Mercury News service, article dated 11 February 1997

Table 9C. Metonymy Training Results On Data Sets 1 and 2 (Transition Table Weight Assignment)

	Tested on DS1	Tested on DS2			Tested on DS1+DS2		
	Trained on DS1	Trained on DS1	Trained on DS2	Trained on DS1+DS2	Trained on DS1	Trained on DS2	Trained on DS1+DS2
TOTAL	58/59 = 98%	63/87 = 72%	83/87 = 95%	83/87 = 95%	121/146 = 82%	124/146 = 85%	139/146 = 95%

set other than the data set to which the results are applied, the results are worse. As the total training data set increases, however, the loss decreases.

Four examples in DS2 were not solvable after training (compared to five by the static mechanism). The two unsolvable examples were obviously not solved, and the two other examples that ended up unsolved were the ambiguous ones, where both static and dynamic mechanisms picked the option that was not supported by the context. To address these sorts of genuinely ambiguous examples (for example, *peruvian radio* as referring to the audience not the management, and *diners* bringing in money as referring to attendees not preparers), a context mechanism is needed. This mechanism could make use of both actual context (entities previously referred to, for example) and virtual context, expectations, or world knowledge, as would be encoded in script mechanism such as in Cullingford (1981).

9.9 Results

A test set was produced in exactly the same way as training Data Set 2, from USA Today and San Jose Mercury News articles (7 March 1997 editions). The test data reflects the first fifty me-

Table 9D. Metonymy Test Results On English Test Data

	# Correct	Errors due to missing arc	Errors due to representation gap	Errors due to bad path
Static Arc Weight Assignment	47/50 = 94%	0/3	1/3	2/3

Table 9D. Metonymy Test Results On English Test Data

	# Correct	Errors due to missing arc	Errors due to representation gap	Errors due to bad path
Dynamic Arc Weight Assignment	45/50 = 90%	2/5	1/5	2/5

tonymies found in the two sources; actually, many repeat metonymies of the form *X announced... X also announced...* were omitted; the inclusion of all these (easy) metonymies would have resulted in a ratio of about 95/100 for the test set. Table 9D shows results on this test set for both the dynamic and the static arc weight assignment mechanisms, using weights trained on DS1 and DS2 together.

A certain class of errors was disregarded in these test results: concepts missing from the ontology, which suggests that the word (sense) for the metonym or the selecting word (usually the verb) would not have been in the lexicon, so, in computational processing, the metonymy resolution mechanisms would not have been invoked in the first place. In this test, similar concepts were substituted for these missing concepts or instances (for example, *Liberia* was missing from the onomasticon, so *Algeria* was used instead).

The test corpus had one "guaranteed" error, because the key had no correct answer for it: *the pro-Beijing newspaper.* The meaning of *pro* is captured by the ADVOCACY relation; a metonymy arises because our ontology requires that the *DOMAIN* of ADVOCACY be restricted to ABSTRACT-OBJECT, perhaps capturing the notion that only ideas etc. can be advocated. Thus this example contains at least a Place-for-Institution metonymy. In addition, the ontological constraint suggests that it isn't the government that is being advocated, but perhaps their policies or actions. It is uncertainty about this second chained metonymy that caused the metonymy to be unresolved in the key.

Both examples (for both assignment mechanisms) where the wrong path was returned (*the Tamil rebels attacked two army bases* and *Tamil rebels shelled a town*) required that the *THEME* of ASSAULT be HUMAN; the search returns an Institution-for-PeopleResponsible metonymy instead of the Institution-for-PeopleResponsible metonymy. However, the *OCCUPANT*s relation is not defined for SMALL-GEOPOLITICAL-ENTITY and MILITARY-INSTALLATION; the *LOCATION-OF* relation is defined, but too generally to have been preferred by the search.

The texts used for training and testing for the Spanish WSD experiments (see Section 8.3) were also examined for metonymies produced as part of the SDS-building/TMR-construction process. The results there showed, realistically, how metonymy resolution and WSD are interrelated, in that many of the WSD failures correlated with (wrong) metonymies being produced and vice versa. Table 9E shows the cumulative counts for different categories of metonymies produced during the course of producing TMRs for the four training and one test text. The table shows that seven kinds of metonymic expressions were found in the four texts, of which Institution-for-Per-

Table 9E. Metonymic Inferences in 5 Spanish Texts

CORRECT INFERENCES DUE TO METONYMIES	Institution for PersonResponsible	23
	ObjectUsed for User	1
	Action for Entity	15
	Generic for Specific	10
	Symbol for Symbolized	1
	Product for Producer	2
	Instrument for Action	1
INCORRECT INFERENCES	wrong preposition selected	6
	conjunction problems	2
	all other TMR errors (including bad metonymy resolution and missing microtheories)	37

sonResponsible was the most common, as expected, due to the nature of the texts. The Action-for-Entity type was also well-represented, often from the use of "imports" or "exports" to refer to products (since they are represented as the *THEME* of an event in the lexical semantic specifications). The table also shows a count for various classes of errors in TMR production. A number of these errors are just reflections of missing microtheories; for example, "millions of dollars" and other numerical expressions cause odd TMR constructions that cause trouble when they are linked to other elements of the SDS, resulting in type mismatches and. therefore, metonymic inferences. Another class of anomaly is due to temporal expressions, for which no microtheory has been developed, and whose absence causes funny metonymic expressions to appear in the TMR. These various missing microtheories account for about half of the errors. It is difficult to pinpoint the cause of errors, so no breakdown of the error types can be produced; for example, it is difficult to determine whether bad metonymic resolution is the cause or the effect of bad WSD on open class words or prepositions, informed by a range of other knowledge sources other than the ontological graph search. Some indeterminate fraction of the errors is due to the ontological graph search returning a bad path, or returning an appropriate path but at a cost that resulted in bad overall disambiguation. Thus many of these errors, real and apparent, would be eliminated by further development of the MikroKosmos system that formed the test environment in this case, namely by developing the following microtheories: numeric expressions, temporal expressions, reification of case roles, and prepositional semantics.

The goal of these experiments was not to attempt to solve all cases of metonymy, but to identify how far this general mechanism can lead us in addressing metonymies. In fact, the results are rather promising, in terms of coverage. A handful of examples are mentioned in Section 9.8 above and in this subsection as difficult or impossible within the framework of the approach described here; however, they seem to account for less that five percent of metonymies occurring in real corpora. Not unexpectedly, overall results improve with improvements to the ontology and with increased training data for inducing the arc weights.

Table 9F. Manually Acquired Arc-Cost Determination Transition Table (for basic dependency and metonymy)

STATE / SLOT	IS-A	INSTANCE-OF NAMED-INST.-OF	SUBCLASSES	CONTAINS HAS-MEMBER HAS-AS-PART HAS-ELEMENT	PART_OF MEMBER-OF	MATERIAL-OF	HEADED-BY OPERATED-BY CONTROLLED-BY	HEAD-OF CONTROLS OWNER-OF	LOCATION-OF	SYMBOL-OF NAME-OF	PRODUCER-OF INV-CREAT-REL etc.[a]	PRODUCED-BY	LOCATION	INSTR.-OF USED-IN EQUIPMENT-FOR	ACCOMPANIER-OF	CONVEYS REPRESENTS	ACTIVITY-FOR-ROLE AGENT	MADE-OF	THEME	THEME-OF	AREA-OF-ACTIVITY	EMPLOYER-OF	OCCUPANT	DOMAIN	RANGE	*Comments State name*
A	0.999/B[b]	1.0/A	0.85/C	0.85/D	0.9/E	0.9/H	0.9/I	0.9/J	0.9/K	0.85/L	0.9/M	0.9/N	0.9/O	0.9/P	0.9/Q	0.9/BB	0.9/DD	0.9/FF	0.7/HH	0.7/JJ	0.85/LL	0.9/NN	0.9/PP	0.9/A	0.7/A	*Start state*
B	0.999/B	[c]	0.65/C	0.85/D	0.9/E	0.9/H	0.9/I	0.9/J	0.9/K	0.85/L	0.9/M	0.9/N	0.9/O	0.9/P	0.9/Q	0.9/BB	0.9/DD	0.9/FF	0.7/HH	0.7/JJ	0.85/LL	0.9/NN	0.9/PP	0.9/B	0.7/B	*Basic/ISA*
C	0.65/B		0.85/C	0.85G	0.9/F	0.9/R	0.9/S	0.9/T	0.9/U	0.85/V	0.9/W	0.9/X	0.9/Y	0.9/Z	0.9/AA	0.9/CC	0.9/EE	0.9/GG	0.7/II	0.7/KK	0.85/MM	0.9/OO	0.9/QQ			*Basic/Subcl*
D	0.999/D		0.85/G	0.9/D	0.3/D	0.8/H		0.85/J			0.8/M	0.8/N	0.8/O	0.8/P	0.8/Q			0.9/FF		0.7/JJ		0.9/NN				*Contains/ISA*
E	0.999/E		0.85/F	0.3/E	0.99/E		0.8/I			0.8/L		0.8/N	0.8/O	0.8/P	0.9/Q	0.9/BB	0.8/DD	0.9/FF	0.7/HH	0.7/JJ		0.9/NN	0.9/PP			*Part-of/ISA*
F	0.3/E		0.85/F	0.3/F	0.99/F		0.8/S			0.8/V		0.8/X	0.8/Y	0.8/Z	0.9/AA	0.9/CC		0.9/GG	0.7/II	0.7/KK		0.9/OO	0.9/QQ			*Part-of/Subcl*
G	0.3/D		0.85/G	0.9/D	0.3/E	0.8/H		0.85/T			0.85/W	0.8/X		0.8/Z	0.8/AA	0.9/CC		0.9/GG		0.7/KK		0.9/OO				*Contains/Subcl*
H	0.999/H		0.85/R	0.85/D	0.85/E	0.9/H			0.8/K	0.8/L		0.9/N	0.9/O	0.9/P		0.85/BB		0.3/FF		0.7/JJ			0.9/PP			*Material-of/IS-A*
I	0.999/I		0.85/S		0.8/E			0.3/J		0.8/L	0.8/M				0.8/Q					0.7/JJ		0.9/NN				*Headed-by/IS-A*
J	0.999/J		0.85/T	0.8/D	0.85/E		0.3/I			0.8/L	0.9/M		0.8/O		0.8/Q					0.7/JJ		0.9/NN				*Head-of/IS-A*
K	0.999/K		0.85/U	0.85/D	0.85/E	0.85/H	0.85/I	0.85/J	0.9/K	0.8/L	0.85/M	0.85/N	0.3/O	0.85/P	0.85/Q		0.8/DD	0.8/FF	0.7/HH	0.7/JJ		0.9/NN	0.9/PP			*LocationOf/ISA*
L	0.999/L		0.85/V	0.85/D	0.85/E		0.85/I	0.85/J	0.85/K	0.9/L	0.85/M	0.85/N	0.85/O	0.85/P	0.85/Q	0.8/BB	0.8/DD	0.8/FF	0.7/HH	0.7/JJ	0.85/LL	0.9/NN	0.9/PP			*SymbolOf/ISA*
M	0.999/M		0.85/W	0.85/D	0.85/E	0.8/H				0.85/L		0.3/N	0.8/O	0.85/P		0.9/BB		0.8/FF		0.7/JJ						*Creator-of/ISA*
N	0.999/N		0.85/X	0.85/D	0.8/E		0.85/I	0.8/J					0.8/0		0.8/Q					0.7/JJ		0.9/NN	0.9/PP			*ProducedBy/ISA*
O	0.999/O		0.85/Y	0.8/D	0.85/E		0.85/I			0.8/L	0.8/M		0.3/0							0.7/JJ			0.9/PP			*Location/ISA*

Table 9F. Manually Acquired Arc-Cost Determination Transition Table (for basic dependency and metonymy)

STATE / SLOT	IS-A	INSTANCE-OF NAMED-INST-OF	SUBCLASSES	CONTAINS HAS-MEMBER HAS-AS-PART HAS-ELEMENT	PART_OF MEMBER-OF	MATERIAL-OF	HEADED-BY OPERATED-BY CONTROLLED-BY	HEAD-OF CONTROLS OWNER-OF	LOCATION-OF	SYMBOL-OF NAME-OF	PRODUCER-OF INV-CREAT-REL etc.[a]	PRODUCED-BY	LOCATION	INSTR.-OF USED-IN EQUIPMENT-FOR	ACCOMPANIER-OF	CONVEYS REPRESENTS	ACTIVITY-FOR-ROLE AGENT	MADE-OF	THEME	THEME-OF	AREA-OF-ACTIVITY	EMPLOYER-OF	OCCUPANT	DOMAIN	RANGE	*Comments State name*
P	0.999/P		0.85/Z						0.8/K	0.8/L		0.85/ N	0.8/O	0.85/P			0.75/ DD	0.8/ FF	0.7/ HH	0.7/JJ						*InstrOf/ISA*
Q	0.999/Q		0.85/ AA		0.8/E		0.8/I						0.8/O				0.8/DD		0.7/ HH	0.7/JJ						*AccompOf/ISA*
R	0.3/H		0.85/R	0.85/G	0.85/ F	0.9/R			0.8/U	0.8/V		0.9/X	0.8/Y	0.9/Z		0.85/ CC		0.3/ GG		0.7/ KK			0.9/ QQ			*MaterialOf/ Subc*
S	0.3/I		0.85/S		0.8/F			0.3/T		0.8/V	0.8/W				0.8/ AA					0.7/ KK		0.9/ OO				*HeadedBy/ Subcl*
T	0.3/J		0.85/T	0.8/G	0.85/ F		0.3/S			0.8/V	0.9/W		0.8/Y		0.8/ AA					0.7/ KK		0.9/ OO				*HeadOf/Subcl*
U	0.3/K		0.85/U	0.85/G	0.85/ F	0.85/ R	0.85/S	0.85/T	0.9/U	0.8/V	0.85/W	0.85/ X	0.3/Y	0.85/Z	0.8/ AA		0.8/EE	0.8/ GG	0.7/II	0.7/ KK		0.9/ OO	0.9/ QQ			*LocationOf/ Subcl*
V	0.3/L		0.85/V	0.85/G	0.85/ F		0.85/S	0.85/T	0.85/ U	0.8/V	0.85/W	0.85/ X	0.85/ Y	0.85/Z	0.8/ AA	0.8/CC	0.8/EE	0.8/ GG	0.7/II	0.7/ KK	0.85/ MM	0.9/ OO	0.9/ QQ			*SymbolOf/ Subcl*
W	0.3/M		0.85/W	0.85/G	0.85/ F	0.8/R				0.85/ V		0.3/X	0.8/Y	0.85/Z		0.9/CC		0.8/ GG		0.7/ KK						*CreatorOf/ Subcl*
X	0.3/N		0.85/X	0.85/G	0.8/F		0.85/S	0.8/T					0.8/Y		0.8/ AA					0.7/ KK		0.9/ OO				*ProducedBy/ Subcl*
Y	0.3/O		0.85/Y	0.8/G	0.85/ F		0.85/S			0.8/V	0.8/W		0.3/Y							0.7/ KK			0.9/ QQ			*Location/Subcl*
Z	0.3/P		0.85/Z						0.8/U	0.8/V		0.85/ X	0.8/Y	0.85/Z			0.75/ EE	0.8/ GG	0.7/II	0.7/ KK						*InstrOf/Subcl*
AA	0.3/Q		0.85/ AA		0.8/F		0.8/S						0.8/Y				0.8/EE		0.7/II	0.7/ KK						*AccompOf/ Subcl*
BB	0.999/ BB		0.85/ CC							0.8/L		0.9/N				0.85/ BB	0.8/DD			0.7/JJ	0.85/ LL					*Conveys/IS-A*
CC	0.3/BB		0.85/ CC							0.8/V		0.9/X				0.85/ CC	0.8/EE			0.7/ KK	0.85/ MM					*Conveys/Subcl*
DD	0.999/ DD		0.85/ EE					0.65/J			0.65/M		0.8/O				0.3/DD			0.7/JJ		0.9/ NN				*Act4Role/IS-A*
EE	0.3/DD		0.85/ EE					0.65/T			0.65/W		0.8/Y				0.3/EE			0.7/ KK		0.9/ OO				*Act4Role/Subcl*

Table 9F. Manually Acquired Arc-Cost Determination Transition Table (for basic dependency and metonymy)

STATE / SLOT	IS-A	INSTANCE-OF NAMED-INST.-OF	SUBCLASSES	CONTAINS HAS-MEMBER HAS-AS-PART HAS-ELEMENT	PART_OF MEMBER-OF	MATERIAL-OF	HEADED-BY OPERATED-BY CONTROLLED-BY	HEAD-OF CONTROLS OWNER-OF	LOCATION-OF	SYMBOL-OF NAME-OF	PRODUCER-OF INV-CREAT-REL etc.[a]	PRODUCED-BY	LOCATION	INSTR.-OF USED-IN EQUIPMENT-FOR	ACCOMPANIER-OF	CONVEYS REPRESENTS	ACTIVITY-FOR-ROLE AGENT	MADE-OF	THEME	THEME-OF	AREA-OF-ACTIVITY	EMPLOYER-OF	OCCUPANT	DOMAIN	RANGE	*Comments* *State name*
FF	0.999/ FF		0.85/ GG			0.3/H				0.8/L								0.9/ FF		0.7/JJ						*Made-of/IS-A*
GG	0.3/FF		0.85/ GG			0.3/R				0.8/V								0.9/ GG		0.7/ KK						*Made-of/Subcl*
HH	0.999/ HH		0.85/II	0.85/D	0.8/E	0.8/H	0.8/I	0.8/J	0.7/K	0.8/L	0.8/M	0.8/N	0.8/O	0.8/P	0.8/Q	0.9/BB	0.8/DD	0.8/ FF	0.7/ HH			0.9/ NN	0.9/ PP			*Theme/IS-A*
II	0.3/HH		0.85/II	0.85/G	0.8/F	0.8/R	0.8/S	0.8/T	0.7/U	0.8/V	0.8/W	0.8/X	0.8/Y	0.8/Z	0.8/ AA	0.8/CC	0.8/EE	0.8/ GG	0.7/II			0.9/ OO	0.9/ QQ			*Theme/Subcl*
JJ	0.999/JJ		0.85/ KK		0.8/E								0.7/O				0.8/DD		0.3/ HH	0.7/JJ	0.7/ LL					*Theme-of/IS-A*
KK	0.3/JJ		0.85/ KK		0.8/F								0.7/Y				0.8/EE		0.3/II	0.7/ KK	0.7/ MM					*Theme-of/Subcl*
LL	0.999/ LL		0.85/ MM													0.9/BB				0.7/JJ						*AreaOfAct/IS-A*
MM	0.3/LL		0.85/ MM													0.9/CC				0.7/ KK						*AreaOfAct/Sub*
NN	0.999/ NN		0.85/ OO	0.85/D	0.8/E		0.8/I	0.8/J			0.8/M	0.8/N	0.8/O		0.8/Q		0.8/DD			0.7/JJ			0.9/ PP			*EmployOf/IS-A*
OO	0.3/NN		0.85/ OO	0.85/G	0.8/F		0.8/S	0.8/T			0.8/W	0.8/X	0.8/Y		0.8/ AA		0.8/EE			0.7/ KK			0.9/ QQ			*EmployOf/Subc*
PP	0.999/ PP		0.85/ QQ	0.85/D	0.8/E		0.8/I	0.8/J			0.8/M	0.8/N	0.8/O		0.8/Q		0.8/DD			0.7/JJ		0.9/ NN				*Occupant/IS-A*
QQ	0.3/PP		0.85/ QQ	0.85/G	0.8/F		0.8/S	0.8/T			0.8/W	0.8/X	0.8/Y		0.8/ AA		0.8/EE			0.7/ KK		0.9/ OO				*Occupant/Subc*

a. Other slots of this cluster are: *COMPOSER-OF, AUTHOR-OF, INVENTOR-OF, PAINTER-OF, SCULPTOR-OF, FILMER-OF*

b. The first element of each cell reflects the arc weight assigned to the input arc (top), if in the given state (left); the second element of each cell determines the next state.

c. **Note**: Unmarked cells = Default cost (0.4) / same slot
See Section 9.8.2 for an explanation of this table.

Table 9G. Transition Table Equivalence Sets for Training

STATE / SLOT	IS-A	INSTANCE-OF NAMED-INST.-OF	SUBCLASSES	CONTAINS HAS-MEMBER HAS-AS-PART HAS-ELEMENT	PART_OF MEMBER-OF	MATERIAL-OF	HEADED-BY OPERATED-BY CONTROLLED-BY	HEAD-OF CONTROLS OWNER-OF	LOCATION-OF	SYMBOL-OF NAME-OF	PRODUCER-OF INV-CREAT-REL etc.[a]	PRODUCED-BY	LOCATION	INSTR.-OF USED-IN EQUIPMENT-FOR	ACCOMPANIER-OF	CONVEYS REPRESENTS	ACTIVITY-FOR-ROLE AGENT	MADE-OF	THEME	THEME-OF	AREA-OF-ACTIVITY	EMPLOYER-OF	OCCUPANT	DOMAIN	RANGE	*Comments State name*
A	1/B[b]	2/A	5/C	9/D	15/E	21/H	26/I	30/J	36/K	40/L	44/M	49/N	55/O	59/P	63/Q	68/BB	74/DD	79/FF	13/ HH	22/JJ	35/LL	45/ NN	59/PP	75/A	79/A	*Start state*
B	1/B	[c]	6/C	9/D	15/E	21/H	26/I	30/J	36/K	40/L	44/M	49/N	55/O	59/P	63/Q	68/BB	74/DD	79/FF	13/ HH	22/JJ	35/LL	45/ NN	59/PP	75/B	79/B	*Basic/ISA*
C	3/B		5/C	9G	15/F	21/R	26/S	30/T	36/U	40/V	44/W	49/X	55/Y	59/Z	63/ AA	68/CC	74/EE	79/ GG	13/II	22/ KK	35/ MM	45/ OO	59/ QQ			*Basic/Subcl*
D	1/D		7/G	11/D	23/D	25/H		32/J			46/M	51/N	57/O	61/P	65/Q			81/FF		27/JJ		50/ NN				*Contains/ISA*
E	1/E		7/F	4/E	18/E		28/I			42/L		51/N	57/O	61/P	67/Q	70/BB	76/DD	81/FF	16/ HH	27/JJ		50/ NN	64/PP			*Part-of/ISA*
F	4/E		8/F	4/F	18/F		29/S			43/V		52/X	58/Y	62/Z	67/ AA	70/CC		82/ GG	17/II	31/ KK		50/ OO	69/ QQ			*Part-of/Subcl*
G	4/D		8/G	11/D	23/E	24/H		33/T			47/W	52/X		62/Z	66/ AA	71/CC		82/ GG		31/ KK		50/ OO				*Contains/Subcl*
H	1/H		7/R	12/D	19/E	21/H			38/K	42/L		53/N	53/O	53/P		72/BB		23/FF		27/JJ			64/PP			*Material-of/IS-A*
I	1/I		7/S		19/E			23/J		42/L	46/M				65/Q					27/JJ		50/ NN				*Headed-by/IS-A*
J	1/J		7/T	10/D	19/E		23/I			42/L	48/M		57/O		65/Q					27/JJ		50/ NN				*Head-of/IS-A*
K	1/K		7/U	12/D	19/E	25/H	28/I	32/J	36/K	42/L	46/M	51/N	23/O	61/P	65/Q		76/DD	83/FF	16/ HH	27/JJ		50/ NN	64/PP			*LocationOf/ISA*
L	1/L		7/V	12/D	19/E		28/I	32/J	38/K	40/L	46/M	51/N	57/O	61/P	65/Q	72/BB	76/DD	83/FF	16/ HH	27/JJ	41/LL	50/ NN	64/PP			*SymbolOf/ISA*
M	1/M		7/W	12/D	19/E	24/H				42/L		23/N	57/O	61/P		70/BB		83/FF		27/JJ						*Creator-of/ISA*
N	1/N		7/X	12/D	19/E		28/I	32/J					57/0		65/Q					27/JJ		50/ NN	64/PP			*ProducedBy/ ISA*
O	1/O		7/Y	10/D	19/E		28/I			42/L	46/M		23/0							27/JJ			64/PP			*Location/ISA*
P	1/P		7/Z						38/K	42/L		51/N	57/O	61/P			77/DD	83/FF	16/ HH	27/JJ						*InstrOf/ISA*

Table 9G. Transition Table Equivalence Sets for Training

STATE / SLOT	IS-A	INSTANCE-OF NAMED-INST.-OF	SUBCLASSES	CONTAINS HAS-MEMBER HAS-AS-PART HAS-ELEMENT	PART_OF MEMBER-OF	MATERIAL-OF	HEADED-BY OPERATED-BY CONTROLLED-BY	HEAD-OF CONTROLS OWNER-OF	LOCATION-OF	SYMBOL-OF NAME-OF	PRODUCER-OF INV-CREAT-REL etc.[a]	PRODUCED-BY	LOCATION	INSTR.-OF USED-IN EQUIPMENT-FOR	ACCOMPANIER-OF	CONVEYS REPRESENTS	ACTIVITY-FOR-ROLE AGENT	MADE-OF	THEME	THEME-OF	AREA-OF-ACTIVITY	EMPLOYER-OF	OCCUPANT	DOMAIN	RANGE	Comments State name
Q	1/Q		7/AA		19/E		28/I						57/O				76/DD		16/ HH	27/JJ						*AccompOf/ISA*
R	4/H		8/R	11/G	20/F	21/R			39/U	43/V		53/X	53/Y	53/Z		73/CC		23/ GG		31/ KK			69/ QQ			*MaterialOf/ Subc*
S	4/I		8/S		20/F			23/T		43/V	47/W				66/ AA					31/ KK		50/ OO				*HeadedBy/ Subcl*
T	4/J		8/T	14/G	20/F		23/S			43/V	48/W		58/Y		66/ AA					31/ KK		50/ OO				*HeadOf/Subcl*
U	4/K		8/U	11/G	20/F	24/R	29/S	33/T	36/U	43/V	47/W	52/X	23/Y	62/Z	66/ AA		78/EE	84/ GG	17/II	31/ KK		50/ OO	69/ QQ			*LocationOf/ Subcl*
V	4/L		8/V	11/G	20/F		29/S	33/T	39/U	40/V	47/W	52/X	58/Y	62/Z	66/ AA	73/CC	78/EE	84/ GG	17/II	31/ KK	41/ MM	50/ OO	69/ QQ			*SymbolOf/ Subcl*
W	4/M		8/W	11/G	20/F	24/R				43/V		23/X	58/Y	62/Z		71/CC		84/ GG		31/ KK						*CreatorOf/ Subcl*
X	4/N		8/X	11/G	20/F		29/S	33/T					58/Y		66/ AA					31/ KK		50/ OO				*ProducedBy/ Subcl*
Y	4/O		8/Y	14/G	20/F		29/S			43/V	47/W		23/Y							31/ KK			69/ QQ			*Location/Subcl*
Z	4/P		8/Z						39/U	43/V		52/X	58/Y	62/Z			77/EE	84/ GG	17/II	31/ KK						*InstrOf/Subcl*
AA	4/Q		8/AA		20/F		29/S						58/Y				78/EE		17/II	31/ KK						*AccompOf/ Subcl*
BB	1/BB		7/CC							42/L		54/N				72/BB	76/DD			27/JJ	41/LL					*Conveys/IS-A*
CC	4/BB		8/CC							43/V		54/X				73/CC	78/EE			31/ KK	41/ MM					*Conveys/Subcl*
DD	1/DD		7/EE					34/J			34/M		57/O				23/DD			27/JJ		50/ NN				*Act4Role/IS-A*
EE	4/DD		8/EE					34/T			34/W		58/Y				23/EE			31/ KK		50/ OO				*Act4Role/Subcl*
FF	1/FF		7/GG			23/H				42/L								81/FF		27/JJ						*Made-of/IS-A*

Table 9G. Transition Table Equivalence Sets for Training

STATE / SLOT	IS-A	INSTANCE-OF NAMED-INST.-OF	SUBCLASSES	CONTAINS HAS-MEMBER HAS-AS-PART HAS-ELEMENT	PART_OF MEMBER-OF	MATERIAL-OF	HEADED-BY OPERATED-BY CONTROLLED-BY	HEAD-OF CONTROLS OWNER-OF	LOCATION-OF	SYMBOL-OF NAME-OF	PRODUCER-OF INV-CREAT-REL etc.[a]	PRODUCED-BY	LOCATION	INSTR.-OF USED-IN EQUIPMENT-FOR	ACCOMPANIER-OF	CONVEYS REPRESENTS	ACTIVITY-FOR-ROLE AGENT	MADE-OF	THEME	THEME-OF	AREA-OF-ACTIVITY	EMPLOYER-OF	OCCUPANT	DOMAIN	RANGE	*Comments* *State name*
GG	4/FF		8/GG			23/R				43/V								82/ GG		31/ KK						*Made-of/Subcl*
HH	1/HH		7/II	12/D	19/E	25/H	28/I	32/J	37/K	42/L	46/M	51/N	57/O	61/P	65/Q	70/BB	76/DD	83/FF	16/ HH			50/ NN	64/PP			*Theme/IS-A*
II	4/HH		8/II	11/G	20/F	24/R	29/S	33/T	37/U	43/V	47/W	52/X	58/Y	62/Z	66/ AA	73/CC	78/EE	84GG	17/II			50/ OO	69/ QQ			*Theme/Subcl*
JJ	1/JJ		7/KK		19/E								56/O				76/DD		23/ HH	27/JJ	41/LL					*Theme-of/IS-A*
KK	4/JJ		8/KK		20/F								56/Y				78/EE		23/II	31/ KK	41/ MM					*Theme-of/Subcl*
LL	1/LL		7/MM													70/BB				27/JJ						*AreaOfAct/IS-A*
MM	4/LL		8/MM													71/CC				31/ KK						*AreaOfAct/Sub*
NN	1/NN		7/OO	12/D	19/E		28/I	32/J			46/M	51/N	57/O		65/Q		76/DD			27/JJ			64/PP			*EmployOf/IS-A*
OO	4/NN		8/OO	11/G	20/F		29/S	33/T			47/W	52/X	58/Y		66/ AA		78/EE			31/ KK			69/ QQ			*EmployOf/Subc*
PP	1/PP		7/QQ	12/D	19/E		28/I	32/J			46/M	51/N	57/O		65/Q		76/DD			27/JJ		50/ NN				*Occupant/IS-A*
QQ	4/PP		8/QQ	11/G	20/F		29/S	33/T			47/W	52/X	58/Y		66/ AA		78/EE			31/ KK		50/ OO				*Occupant/Subc*

a. Other slots of this cluster are: *COMPOSER-OF, AUTHOR-OF, INVENTOR-OF, PAINTER-OF, SCULPTOR-OF, FILMER-OF*

b. The first element of each cell reflects the arc weight assigned to the input arc (top), if in the given state (left); the second element of each cell determines the next state.

c. **Note**: Unmarked cells = Default cost / same slot

10. Ontology Acquisition

One of the chief impediments to the use of knowledge-based systems is the heavy burden of acquiring the knowledge sources. In this section we present some thoughts on the acquisition of the ontology, as well as the ramifications of gaps in the ontology. While the acquisition of knowledge for the ontology and lexicons is outside the scope of the work discussed here, this section will speculate on ways of acquiring knowledge in a semi-automated manner, on methodologies for augmenting that knowledge to support the sort of reasoning discussed in Section 7 and subsequent sections, and on effects of gaps in the ontology on those reasoning processes.

10.1 Acquisition of Ontological Knowledge

The ontology is organized as a hierarchical tree; the content of the uppermost levels of the ontology is crucial to the inheritance, organization, and utility (by various reasoning mechanisms) of the ontology itself. These upper levels of the ontology are necessarily hand-crafted. The balance of the ontology, which is expected to consist of thousands (probably under ten thousand) of nodes, could be constructed by a combination of semi-automated importation of existing resources that have ontological knowledge implicitly or explicitly encoded within, manual acquisition to support the processing of particular domains or corpora, and tool-enhanced manual acquisition in a more thorough manner. The sections below discuss some such possibilities to expedite or automate the ontology acquisition process. However, it should be noted that an ontology has already been built that is sufficient for the experiments and results presented in previous chapters; this ontology, with about 5000 nodes and over 15,000 (local) relations, is described in Mahesh and Nirenburg (1995) or Mahesh (1996).

10.1.1 Importing Existing Ontological Resources

There is a range of available resources which contain some amount of either taxonomic or ontological knowledge; the depth of these resources varies from shallower to deeper than is required for the work discussed here, but, uniformly, all of these resources are impoverished in terms of relations and attributes on concepts or ontology entries. In order to make use of any of these resources it becomes necessary to reformat the resource into a hierarchy of the form that the ontology-building tools can manipulate, then to allow a human ontology acquirer to peruse the resources and import appropriate fragments. Although the uppermost levels of the hierarchical resources tend to be organized differently from what our paradigm expects, the mid levels of these resources can be suitable for importation; the lower levels vary from being too detailed to being useful as well. An approach which involved merging a number of these (or other) resources could also be attempted, but would run into the same difficulties as mentioned above. All these resources only provide the taxonomic structure — the relational and attribute information (i.e., the *differentia*) need to be added to the taxonomic (the *genus*). Note that all of these resources, in addition to providing some ontological information, generally provide lexical information for English, perhaps including some lexical semantic mapping.

- WordNet (Miller *et al.* (1993)) — The WordNet resource contains clusters of related word senses (called syn-sets), along with indications of hyponymy, hypernymy, meronymy, and antonymy between clusters. Although the clusters are defined by their membership of English words, in effect they identify clusters akin to ontological concepts. Numerous efforts have used this resource in building English-language lexicons or ontologies. Several complications for using WordNet for our ontology exist, however, including a difference in top-level choices of categories. Additionally, the WordNet resource has a much finer granularity at the leaf nodes than we would want to have in our ontology; either manual pruning, or perhaps a corpus-specific method such as Cucciarelli and Velardi (1997), could reduce the taxonomy to a more appropriate depth and granularity.

- LDOCE (Procter *et al.* (1978)) — The Longman Dictionary of Contemporary English has also been extensively used, typically in building a lexicon. The presence of semantic constraint information and typology (via the *box codes*) and the domain information (the *subject codes*), in the electronic form, allows some degree of taxonomization; the limited vocabulary and grammar of the definitions has allowed researchers to induce semantic definition and/or classification information, including Guthrie *et al.* (1990), Bruce and Guthrie (1992), Vossen *et al.* (1989), Alshawi (1989).
- CYC (Lenat *et al.* (1990)) — The CYC knowledge base offers the opportunity for importing knowledge into our ontology, of the taxonomic, attributive, and relational nature. Substantial work will need to be done, however, to determine whether the rather disparate knowledge types and structures in CYC would be appropriate for MIKROKOSMOS-style inferencing, and to perform the conversion. An initial investigation into the utility of such a large-scale importation is described in Mahesh *et al* (1996b), with generally pessimistic conclusions. The axiomatic information in CYC isn't necessarily of the sort that the ontology encodes, and the number of concepts isn't as significant as expected. The way information is encoded doesn't sustain the frame-based approach that we need (although it could be reformatted, with some effort). In general, the information in CYC isn't designed to sustain the sort of reasoning that we perform, namely inheritance and relaxation of constraints, which necessarily relies on hierarchical organization of information (a type of knowledge that CYC is deficient in).
- SIC (Office of Management and Budget, US Government) — The Standard Industrial Classification manual presents a very large, comprehensive taxonomization (bushy, 6 levels deep) of all areas of business, commerce, industry, agriculture, etc. in the United States. Portions of this taxonomy can be imported to populate appropriate portions of our ontology. From the occurrence of the same type of product in numerous categories (for example, research on computers, manufacturing of computers, sales of computers, repair of computers) it should be possible to develop a characterization of some events and relations that certain types of objects can participate in.
- Cambridge Linguistic Survey — This repository of lexicalized knowledge may be able to provide taxonomic and attribute information that could be imported into the ontology, whether directly or thorough the Cambridge International Dictionary of English (CIDE, Procter (1995).
- Word Menu from Random House — This compilation of words is organized in hierarchical thesaurus-like manner that may allow some derivation of ontological information.
- FrameNet, as described in Lowe *et al.* (1997), may be the best potential source for direct importation of ontological knowledge, although the project was just initiated at the time of writing. This resource will certainly have more relational information that WordNet, especially of the case role type. Since relational information will be important in building this resource, the taxonomy may end up not too dissimilar with the taxonomy in our ontology,

to facilitate inheritance.

- The ISI SENSUS ontology, as described in Knight and Luk (1994) — This resource was, in fact, constructed by integrating the taxonomic information from a number of other resources mentioned above, specifically LDOCE (although this resource was eventually removed for intellectual property rights reasons), the predecessor of the MikroKosmos ontology described here, and the WordNet taxonomy. Knowledge from these resources was merged with the ISI Upper Model, and upper tier ontology. This taxonomy does not have the relational information needed to support the kinds of inferencing that we require, and is very English-language specific, but could be used to bootstrap an English lexicon by mapping portions of the taxonomy to sections of our ontology.
- The information encoded in an MRD could be mined in an attempt to build a taxonomy, as in White (1988), but any such effort would have to contend with word sense ambiguity and other text processing problems on the dictionary definitions themselves.

10.1.2 Facilitating Building an Ontology

There will always be a need for some manual addition and modification of the ontology, even in the unlikely case of overwhelming success in (semi-) automated acquisition. In addition to an editor, there are other tools that could be provided to the lexicographer/ontologist which can assist in the manual review process.

- Tools for automatic derivation of subcategorization frames for verbs (such as those being developed by Levin and Hogan at CMU, or Briscoe and Carroll (1997)), can help the ontologist determine the sort of relations that a concept (which represents the meaning of a certain verb) can participate in over a corpus; the case roles identify slots and relations that need to be defined for the event concept.
- Corpus analysis tools, even as primitive as KWIC, can help identify the sorts of arguments that a verb takes; the concept(s) which represents the meaning of that verb will need to allow the appropriate set of fillers (through semantic constraints) for the various relations/ case-roles that the verb participates in. Similarly, KWIC on a noun can identify the sort of events and relations that the entity can participate in.
- Noun-Noun compounds in English provide an unusually rich source of relations which have objects as the domain and range (i.e., are slots on object concepts). In examining the N-N compounds, say from a syntactically marked-up corpus like Treebank, each such compound identifies a particular relation that exists between the concepts which represent the meaning of the head and subordinate nouns. Appropriate generalization of such relations should substantially enrich the ontology to reflect empirical facts.
- Collocational relations in corpora could be mined to identify the underlying semantic information, as in the preliminary work in Anick and Pustejovsky (1990)

The use of these tools, as described above, to analyze a corpus for the purpose of determining the appropriate relations and constraints that various concepts (as reflected by English words) can participate in provide an avenue for empirical verification and augmentation of the ontology, particularly the relational information.

10.2 The Effects of Gaps in the Ontology

The process of lexical acquisition ensures, to a certain degree, an appropriate degree of richness in the ontology; in order to write the **LEX-SEM** specification of a particular lexeme, it is necessary to have a specific ontological concept, a TMR structure, or a complex of concepts and structures to map the meaning of the lexeme into. Thus the ontology is incrementally enriched on an as-needed basis in the process of lexical acquisition.

It is unclear what the effect of varying degrees of richness over sections of the ontology would have on the reasoning and constraint relaxation processes. It is possible that a corner of the ontology, if defined to a greater granularity and bushiness, would be traversed at a different rate than another corner which is sparser, resulting in different "best paths" than would have resulted from an equally-balanced ontology. This will be a matter of investigation.

In general, the effect of missing relations and attributes result in the lack of information by which to distinguish between possible lexical realizations in generation. In analysis, lack of relations can result in missing metonymies or metaphors in constraint relaxation, and certainly in finding inappropriate relations for resolving N-N compounds. Since such effects are possible, the gap issue may contribute a heuristic to the ontology building constraints: that the ontology be as balanced as possible in terms of granularity.

Gaps in the ontology might also hinder recovery mechanisms for dealing with unknown words in the input text. Although such mechanisms have not been explored seriously in the context of MIKROKOSMOS yet (other than just guessing a lexeme with meaning ALL as described in Section 8.3), there are some specific approaches that follow from the use of the ontology both for defining selectional restrictions and lexical semantics. For example, unknown nouns can be approximated by instantiating the constraint on the verb (or preposition) argument slot in which the noun appears. Verbs might be approximated by determining the arguments, and perusing the EVENT subhierarchy in search of events with semantic constraints matching the semantics of the arguments, assuming a default syntax/semantic interface frame. If an appropriate verb subtree is found, instantiate the fairly general event concept (at the head of the tree) as the approximation of the verb's meaning. Similar exercises have been described in the literature, but perhaps by having a rich ontology and specific constraints it may be possible to get reasonable results.

10.3 Issues in Defining Relations

The success of the lexical semantic representation (in discriminating between senses) lies in the ontology having the appropriate richness, and mapping lexical meaning from the ontology. The success of the semantic analysis or SDS-building process also relies on the ontology, in that constraint satisfaction and relaxation, as well as determination of the relation between syntactically marked items (such as N-N compounds) also depends on the richness of nodes and, crucially, links in the ontology. There are numerous issues that arise about the nature and richness of the inventory of relations (in the RELATION corner of the ontology), as well as about the process of acquiring these relations.

- One question that can only be answered after extensive work on the ontology is whether or not we need to represent any three-place relations. If so, is there any penalty in treating them as events, where there may be arbitrarily many arguments. It might be the case that there are three-place relations which are treated by "lexicalizing" one of the arguments into the definition of the relation type.

- One critical issue that needs to be addressed is whether, as the ontological graph search assumes, there is in fact some sort of ontological proximity (i.e., some relatively short path over any arcs) between semantically related concepts. Here, "semantically related" needs to be defined in pragmatic terms, i.e., so that the notion can be used by the ontological graph search to perform disambiguation and to select appropriate relations between concepts. Thus we would expect holster and revolver to be related, chopsticks and eating, astronomer and star, stocks and bonds, and so on. This basic assumption underlies the algorithm in Section 5; success or failure of that approach will validate or deny this premise.
- Various observations have been made about the relatedness of the sets of relations that exist in derivational word formation (as reflected in Lexical Rules), in N-N compounds, in regular polysemy, in collocations, and in metonymy (usually these observations are on specific pairs from the above list). There are inventories in the literature for each these phenomena (e.g., Mel'chuk and Zholkovsky (1984), Lakoff and Johnson (1980), Apresjan (1974), Apresjan *et al.* (1969), Stern (1965), Ostler and Atkins (1992), among others); these inventories can be exhaustively reflected by sets of relations in the ontology. Additionally, each of these phenomena could be isolated in a corpus, and a study of the examples would lead to verifying the existence of appropriate relations in the ontology. On the other hand, after extensive development of the ontology and the lexicons, as verified by extensive processing of texts from various sources, an experiment could be set up to explore the degree to which these various phenomena share similar semantic relations, and the extent of relations that are only differentia in meaning discrimination but don't participate in any of these phenomena.
- Just as there are obvious parallels between English verbs and nouns (*destroy* and *destruction*, for example), there are parallels between event verbs (which have EVENT representations) and expressions which give rise to relational information in the TMR. For example, there might be an ontological event STORE_EVENT, which reflects the act of storing something somewhere. The issue is whether or not there should be a relation (such as *STORED_IN*) which relates one argument of an event with another (e.g., REVOLVER *STORED_IN* HOLSTER), to mirror the event, for example STORE_EVENT with *THEME* REVOLVER and *LOCATION* HOLSTER. Furthermore, if there were such a relation, what the exact semantics would be; in fact, for the above example, the relation (and the event from which it was derived) should probably be something akin to TYPICALLY_STORED_IN. Although representing weak default knowledge, such relations appear to be necessary to capture the relation between HOLSTER and REVOLVER, which might be necessary for N-N compound resolution (such as *revolver holster*); whether this relation could be represented by generic specification of HOLSTER is left to experimentation.

  If, in fact, such events do lead to relations, it needs to be determined how specifically they should be used in the ontology; should AIRPLANE have slots *TYPICALLY_FUELLED_IN*, *TYPICALLY_MAINTAINED_IN*, *TYPICALLY_STORED_IN*, etc. locally specified? Or should more generic knowledge be relied on, with substantial acquisition and storage savings, perhaps at the risk of some inappropriate inference?

  As an experiment to investigate the feasibility and nature of such relations derived from events, a *Gedanken* experiment could be conducted on a number of events from various points in the ontology (from fairly general to fairly specific). From each event, a number of relations could be derived by considering each of the five or seven most frequent case-roles for the domain of the relation, and the same for the range (for example MANUFACTURE, *AGENT* as domain, and *LOCATION* as range, yielding a relation *LOCATION_OF_MANUFACTURING*). A representative set of objects (from various points in the ontology, particularly including typical instruments, agents, locations, etc.) could be tested against the domain and range of the relation (for the above example, for BLACKSMITH the relation makes sense, and would be fillable by SMITHY or BLACKSMITH-SHOP, since that is the typical location where a blacksmith manufactures). If the relation is valid, there should be a well-defined subtree (however large or small) of object concepts for both the domain and range. A pragmatic decision in using these sorts of relations is whether or not the specificity, thus added discriminatory and disambiguation power, is enough to justify the overhead. In the absence of such specific information, the BLACKSMITH concept would still be linkable to SMITHY as a *LOCATION*.

11. Conclusion

11.1 Future Work

11.1.1 Noun-Noun Compound Resolution

The processing of N-N compounds for English (where there is no preposition that links the two nouns) would differ from most of the other applications of the instantiation-combination process in English in that there are no explicit syntactic relations identified in the **SYN-STRUC** zones of the participating nouns, nor are there semantic dependency selectors (i.e., ^$VAR#) expressions in the lexical semantic specifications of the nouns. A discussion of the way that this type of structural processing is handled in the state-space search process appears in Section 6.3.2.

The basic premise of this approach to N-N compound processing is that slots capture the relation between the two concepts:

Heuristic XII. In N-N compound processing, the relation between the subordinate noun and the head noun is captured by filling a slot on the head instance with the subordinate instance.

Since any relation that is defined in the ontology may serve as a slot on a concept, this approach allows a wide variety of relations to be used to define N-N compounds.

In implementing the SDS-building process for N-N compounding, certain differences from the base case need to be accounted for. Firstly, the invocation of the combination process is driven not by the syntax/semantic interface through the $VARs, but by syntactic cues. Secondly, since there is no indication of the appropriate slot for combination, the combination operator is invoked slightly differently. The same ontological graph search process is still appropriate for this task, but with a different set of arc weights than the ones for directed semantic constraint satisfaction. The search is initiated by searching for the best paths from the filler (subordinate concept) to the head concept; since any path to the constraining concept (in this case the head) goes through one of the slots on that concept, this approach serves to explore any of the slots on the head concept.

In assigning the arc weights for the graph search process, certain heuristics apply. The slots that are locally defined or are inherited by the head concept have semantic constraints which define what are the desired and allowed fillers for those slots, which serve to define the paths to the head.

Heuristic XIII. In N-N compound processing, slots inherited by the head concept, although still available to serve as the locus of combination, are slightly less preferred than locally-defined (therefore more specific) slots

This causes a decreased preference for every level above the head concept from which a slot/constraint is inherited. Thus, the initial path costs are adjusted to reflect the decreased preference for the more distantly-inherited slot constraints.

Examples such as *bottle opener* or *airline ticket reservation system* illustrate the need for "generic instances". There is no one specific instance of *BOTTLE that the particular opener was built to open, and there isn't one specific airline, one specific ticket, or one particular reservation that the system is to handle. On the other hand, in examples such as *boat bottom* or *toothbrush handle* there is a specific boat that the bottom is part of, and a specific handle that is part of the specific toothbrush in the model of discourse.

Although no comprehensive generalization is immediately available as to when the subordinate noun(s) is a generic vs. an instance, it appears clear that the relation controls the instantiation:

Heuristic XIV. The nature of the relation between the head and subordinate nouns in N-N compounds dictates whether the subordinate noun produces a generic or an instance.

It appears, at first blush, that when the relation is a kind of *TYPE* relation (such as *SYSTEM-TYPE*), the generic is called for; when the relation is *PART-OF*, the instance appears to be more appropriate. Obviously, extensive experimentation and corpus analysis will be necessary to determine this division and to validate this heuristic generalization. This partitioning of relations may, in fact, have ramifications on the design of the *RELATION portion of the ontology, if it appears that all generic-producing relations are somehow taxonomically grouped.

Much of the literature on N-N compound typology (e.g., Finin (1980, 1986), Lees (1960, 1966), Warren (1978), Lehnert (1988), or Zhivov (1978)) classifies compounds along structural equivalents that the compound correlates with, such as `SUBJECT-OBJECT`. Although this sort of typology is useful, it does not provide the semantic relations that are needed for the approach described here. A search of other, possible less related, literature and empirical studies will be needed to describe a typology of semantic compounding relations. Semantic classification of the two terms which comprise N-N compounds may allow specific constraints to be placed on the domain and range of each relation, which could substantially aid the search process.

11.1.2 Additional Microtheories

The model of semantic analysis presented here makes use of the SDS-building process with constraint satisfaction via the ontological graph search for core meaning construction from lexical information and inter-lexeme relationships. However, the model also expects to have a range of specialist microtheory processors to handle particular phenomena or aspects of meaning. Some of these microtheories may be interleaved or integrated into core SDS-building, while others may apply separately, as characterized in Figure 7A. Additionally, some of these microtheories will make use of the ontological graph search, but, again, not all. What follows below is some speculation on possible microtheories that may need to be added to the overall framework to address the range of meaning that we desire to represent, and to address the four desiderata for practical computational semantic theories as spelled out in Section 1.3.

- Reference resolution: the TMR notation already has a mechanism for representing coreference, but we don't have a procedure specified for discovery of coreference relations. The ontological graph search may assist such a procedure or microtheory, by exploring the paths over the ontology between candidate coreferring instance structures in the TMR (a different set of arc weights would be needed, because the ontological graph search would be asked to identify semantic similarity, instead of constraint satisfaction). Additional mechanisms would obviously be necessary; the coreference exercises in MUC (see Sundheim (1995)) required a wide range of techniques, including partial matching of name strings, to address the coreference problem in news wire texts.

  The ontological graph search can also assist in some cases of definite reference resolution, particularly with evoked or inferable information (see Prince (1981)). The ontological graph search can identify relationships between parts and wholes and other semantically related entities.

- The name tagging technology also described in Sundheim (1995) could also provide a useful tool for our framework, if inserted before the syntactic parser. In addition to identifying the extent of proper names in text (a surprisingly non-trivial problem) and tokenizing them, this technology can provide semantic type information for what would otherwise be an unknown word (or an ambiguous word), in many cases (such as novel company names not in the onomasticon, or names that could serve multiple types, such as *London* as a place or a person name, among others). In cases where a name isn't found in the onomasticon and lexicons, a name transliteration mechanisms could be incorporated, as appropriate.
- Some of the WSD algorithms in Section 8.1 could be integrated into our framework as an additional microtheory of sense selection, adding its preferences to the mechanisms already in place, as described in this document. Some of these algorithms, such as the collocational statistics-based approaches, could be added as a separate stand-alone microtheory, so long as the word senses were mapped into our lexicon. Other mechanisms, such as the Resnik (1995a) information content approach, could be explored as an enhancement to the ontological graph search, although it isn't clear whether it would provide added value, since the knowledge that his approach encodes is somewhat orthogonal to the path weights that we use.
- A more sophisticated approach to stylistics could be incorporated as a microtheory in the analysis process, also providing weak WSD clues on the basis of expected (or computed) stylistic mismatches with certain senses.
- A microtheory of certain types of metaphor is already under preliminary investigation; the current model being explored involves comparison of semantic features across the ontology in cases of constraint violations, and suggestions of substitutions of the event concept in a metaphor with semantically/metaphorically related ones (and reliance on the core SDS process to select among them). In essence, this involves relaxing the event or constraining concept in the case of violations, as opposed to relaxation of the constraint itself, as is the case in metonymy resolution. This effort is very preliminary at this stage, and only attempts to address a small subset of metaphors that conform to this relaxation model.

11.2 Summary and Heuristics

The paradigm presented in this paper developed from the observation that the depth of analysis required for high-quality translation and other applications exceeds the capabilities of syntactic and shallow semantic frameworks. The information that is derived from the semantic analysis (which, in our terms, includes contextual semantics, pragmatics, stylistics, treatment of unexpected input, resolution of deictic phenomena, and other tasks, in addition to what has traditionally been called lexical semantics — that is, static, syntax-driven constraints on meaning) is represented in the language-neutral representation (the TMR). In practice, this depth of analysis requires substantial amounts of world knowledge for disambiguation and the other inferencing that is required in the process of building the TMR. The lexicon becomes the point in which much of this world knowledge is referenced and indexed.

Basic semantic analysis, or Semantic Dependency Structure building, becomes a process of disambiguating the input into lexemes and predicate argument structure, the instantiation of those lexemes, and the combination of those instantiated fragments into the basis of the TMR. The search space of possible combinations and instantiations (thus possible semantic representations) is large, therefore elaborate constraint checking is utilized to restrict possible combinations of representations for lexemes.

This constraint check relies on semantic constraints on fillers of slots of instances, encoded either in the ontology or in lexical entries. Constraint checking is performed by determining the semantic relation between the potential filler and the constraint, by means of a best-path search over the ontology, where various relations other than hierarchical ones also provide arcs between nodes. Arcs have different weights to reflect the degree of semantic affinity reflected by that node, although the weight itself may be context dependent. Semantic affinity refers to the strength of the relation encoded by an arc, as judged against the extent to which it can serve as the underlying motivator which defines a metonymic or other relation between words.

The list of heuristics below summarizes the main insights which guide the effort described here.

Heuristic I. Semantic constraint satisfaction and relaxation corresponds to finding the best path over weighted arcs in an ontological graph from the candidate filler to the constraint.

Heuristic II. Use Equation (2) as the search heuristic in the graph search over the ontology.

Heuristic III. The weight on an arc from node A to node B in the ontology is proportional to the semantic affinity that node A has for node B.

Heuristic IV. The syntax/semantics interface identifies the correlation between elements in the syntactic argument structure and semantic dependency structure by means of parallel variable structures.

Heuristic V. Semantic Dependency Structure Building is accomplished by iterative application of the Instantiation and Combination operators.

Heuristic VI. The semantic analysis process proceeds by recursive descent down the syntactic parse tree.

Heuristic VII. The instantiation operator produces a meaning representation for a specific or generic instance of an entity, event, etc., as specified in the `SEM-STRUC` zone of a particular lexeme; these fragmentary meaning representations constitute fundamental building blocks of the TMR.

Heuristic VIII. The Combination operator attempts to combine TMR fragments according to either expectations from the syntax/semantics interface or as indicated by syntactic clues; the success of the operator is contingent on the successful application of the constraint satisfaction check, as embodied by the ontological graph search mechanism.

Heuristic IX. Metonymies in text are identified by the ontological graph search, when the best path traverses an arc which reflects a metonymic relation.

Heuristic X. In representing metonymies in the text meaning representation, it is nec-

essary to make an inference (by instantiation) about the existence of the entity replaced by the metonym.

Heuristic XI. Inventoried metonymic relations are less preferable than taxonomic (*IS-A*) relations, but still preferable over all other relations over the ontology.

Heuristic XII. In N-N compound processing, the relation between the subordinate noun and the head noun is captured by filling a slot on the head instance with the subordinate instance.

Heuristic XIII. In N-N compound processing, slots inherited by the head concept, although still available to serve as the locus of combination, are slightly less preferred than locally-defined (therefore more specific) slots

Heuristic XIV. The nature of the relation between the head and subordinate nouns in N-N compounds dictates whether the subordinate noun produces a generic or an instance.

These heuristics can be categorized according to their primary purpose. The core heuristics that deal with the SDS-building process are Heuristic IV and Heuristic X, while others deal with the search space for the process: Heuristic V, Heuristic VI, Heuristic VII, and Heuristic VIII. The largest set of heuristics, however, are concerned with defining the use of the graph search in the constraint satisfaction process: Heuristic I, Heuristic II, Heuristic III, Heuristic IX, and Heuristic XI. While these heuristics alone do not define the entire SDS-building process, search space, control flow, or knowledge sources, they are the chief guiding insights that led to the design of the approach as described above.

11.3 Significance

The body of work presented in this document and a constellation of related work in the DIANA/MIKROKOSMOS projects marks the first significant effort to use an ontological semantics framework for KBMT on unrestricted text. Although some of the theory and method discussed above may have been used in other contexts, we found that the ontological semantics framework provides opportunities to extend traditional concepts in significant ways. The results obtained on both the Word Sense Disambiguation task and on metonymy resolution suggest that not only is the overall framework viable, but also that the richness of knowledge and the inference mechanisms are adequate for addressing these very difficult NLP tasks.

The contributions of this work to the *theory* of ontological semantics include the following points:

• We define a framework for the unified treatment of semantic dependency structure building that encompasses, in an integrated manner, satisfaction and relaxation of constraints, word-sense disambiguation, and resolution and representation of metonymy (and, potentially, could also be extended to handle some cases of nominal compounding and reference resolution).

• We define a model for achieving very significant expressiveness of meaning representation.

• We extend the notion of traditional selectional restrictions or semantic constraints by using a wide range of sources of constraints, a rich inventory of potential constraints (the entire ontology), constraints on non-selected-for structures, and semantically nullifying constraints.

- We show how an ontology can be used as a search space for constraint satisfaction and identification of semantic relations.
- We built an extensive inventory and typology (via the ontology) of metonymic relations
- We present an approach to metonymy which is an integral component of semantic analysis, not a separate afterthought; this approach uses an inventory of metonymic relations but also identifies novel uninventoried uses, and it isn't specific to a pair of languages and can recognize and represent chains of metonymies in one semantic relation.

In addition, this work contributes to the *methodology* of practical computational semantics and KBMT:

- We define a framework for lexical semantics and a syntax-semantics interface which can support rich expressiveness, lexical decomposition and non-compositional combination of lexical meaning.
- This framework demonstrates a novel integration of syntagmatic (via extensive selectional constraints) and paradigmatic (via ontological graph search from constraint to filler) models to word-sense disambiguation, with experimental results that are very promising.
- We demonstrated the practical utility of rich ontological relations for a range of computational tasks for NLP, including semantic representation, word-sense disambiguation, and metonymic processing.
- We introduce an efficient mechanism for finding paths over graphs with context-sensitive weighting of arcs.

The practical results of the framework are such that they suggest continuing the exploration of these methods for practical NLP applications, first among them being Machine Translation.

We now revisit the four desiderata for practical computational semantic theories that we specified in Section 1.3. The expressiveness of meaning representation is addressed by the extensive propositional and non-propositional meaning structures in both lexical semantic and text meaning representations, including stylistics, inferences, speaker attitudes, and text relations. All the semantic primitives of the meaning representation are defined by means of an ontology that uses a multiple inheritance hierarchy, relational links between concepts, and features or properties on those concepts; thus we satisfy the second desideratum. The third one is addressed by the SDS-building and constraint satisfaction procedure outlined above, while the fourth desideratum is only partially addressed by the idiom processing, conventional language, and metonymy resolution mechanisms integrated into the overall framework.

The overall significance of the work outlined here is that it demonstrates that the ontological semantic and knowledge-based frameworks for practical computational semantics can provide leading-edge results on NLP tasks such as machine translation. Although the framework could still fail to scale well to full production-level applications, the results achieved on the non-trivial demonstrations are very promising.

12. Bibliography

Agirre, Eneko and German Rigau (1995). “A Proposal for Word Sense Disambiguation using Conceptual Distance”, in *Proceedings of the International Conference on Recent Advances in Natural Language Processing.* Tzigov Chark, Bulgaria. pp. 258-264.

Agirre, Eneko and German Rigau (1996). “Word Sense Disambiguation using Conceptual Density”, in *Proceedings of COLING 1996.*

Aho, Alfred, John Hopcroft, Jeffrey Ullman (1974). *The Design and Analysis of Computer Algorithms*. Reading Mass: Addison Wesley.

Alshawi, Hiyan (1989). “Analyzing the dictionary definitions”, in Bran Boguraev and Ted Briscoe, eds. *Computational Lexicography for Natural Language Processing.* New York: Longman

Alshawi, Hiyan and David Carter (1997). “Training and Scaling Preference Functions for Disambiguation”, in *Computational Linguistics*. vol 20:4.

Anick, Peter and James Pustejovsky (1990). “An Application of Lexical Semantics to Knowledge Acquisition from Corpora”, in *Proceedings of COLING-90*, pp. 7-11.

Apresjan, Yuriy (1973). “Synonymy and Synonyms”, in *Trends in Soviet Theoretical Linguistics*, F. Kiefer, ed. Dordrecht Holland: D. Reidel Publishing

Apresjan, Yuriy (1974). “Regular Polysemy” in *Linguistics* vol. 142, pp. 5-32.

Apresjan, Yuriy, Igor Mel’chuk, and Alexander Zholkovsky (1969). “Semantics and Lexicography: Towards a new type of Unilingual Dictionary”, in *Studies in Syntax and Semantics,* F. Kiefer, ed. Dordrecht: Reidel Publishing Company.

Arnold, Douglas (1996). “Parametrizing Lexical Conceptual Structures for Interlingual Machine Translation”, in *Machine Translation,* vol. 11:4.

Barwise, Jon (1989). *The Situation in Logic.* Stanford: CSLI.

Barwise, Jon and John Perry (1983). *Situations and Attitudes*. Cambridge MA: MIT Press.

Basili, Roberto, Michelangelo Della Rocca, and Maria Theresa Pazienza (1997). “Towards a Bootstrapping Framework for Corpus Semantic Tagging”, in *Proceedings of ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?* Washington DC.

Beale, Stephen (1997). *Hunter-Gatherer: Applying Constraint Satisfaction, Branch-and-Bound and Solution Synthesis to Computational Semantics.* (upcoming) Ph.D. diss., Program in Language and Information Technologies, School of Computer Science, Carnegie Mellon University.

Boguraev, Bran and Ted Briscoe, eds. (1989). *Computational Lexicography for Natural Language Processing.* New York: Longman.

Bouaud, Jacques, Bruno Bachimont, and Pierre Zweigenbaum (1996). “Processing Metonymy: a Domain-Model Heuristic Graph Traversal Approach”, in *Proceedings of COLING-96.*

Bresnan, Joan, ed. (1982). The Mental Representation of Grammatical Relations. Cambridge MA: MIT Press.

Briscoe, Ted and John Carroll (1997). “Automatic Extraction of Subcategorization from Corpora”, in *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP97).* Washington DC.

Brown, Peter, Stephen Della Pietra, Vincent Della Pietra, and Robert Mercer (1991). “Word-Sense Disambiguation Using Statistical Methods”, in *Proceedings of ACL91*. Berkeley CA.

Brown, Ralf (1996). “FramepaC User’s Reference”. Center for Machine Translation, Carnegie Mellon University Technical Memo CMU-CMT-96-MEMO.

Bruce, Rebecca and Louise Guthrie (1992). “Genus Disambiguation: A Study in Weighted Preference”, in *Proceedings of COLING-92.* Nantes.

Bruce, Rebecca and Janyce Wiebe (1994). “Word-Sense Disambiguation Using Decomposable Models”, in *Proceedings of ACL94.*

Budd, Timothy (1994). *Classic Data Structures in C++*. Reading Mass: Addison Wesley.

Buitelaar (1997). “A Lexicon for Underspecified Semantic Tagging”, *Proceedings of ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?* Washington DC.

Carbonell, Jaime (1981). “Metaphor: An inescapable phenomenon in natural language comprehension.” Department of Computer Science, Carnegie Mellon University, Tech Report CMU-CS-81-115.

Carbonell, Jaime, Teruko Mitamura and Eric Nyberg (1992). “The KANT Perspective: A Critique of Pure Transfer (and Pure Interlingua, Pure Statistics,...)” in *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation*. Montreal.

Carbonell, Jaime and Masaru Tomita (1987). “Knowledge-based Machine Translation, the CMU Approach”, in *Machine Translation: Theoretical and Methodological Issues,* Sergei Nirenburg, ed. New York: Cambridge University Press.

Carlson, Lynn and Sergei Nirenburg (1990). “World Modeling for NLP.” Center for Machine Translation, Carnegie Mellon University, Tech Report CMU-CMT-90-121.

Carlson, Lynn, Elizabeth Cooper, Ronald Dolan, Steven Maiorano (1994). “Representing Text Meaning for Multilingual Knowledge-Based Machine Translation” in *Proceedings of the First AMTA Conference*. Columbia MD.

Carter, Everett (1995). “Simulated Annealing Package and Documentation”. `ftp://taygeta.com/pub/c++/simanneal.tar.Z`

de Champeaux, Dennis and Lenie Sint (1977). “An Improved Bidirectional Heuristic Search Algorithm” in *Journal of the Association for Computing Machinery* 24:2, pp. 177-191.

Charniak, Eugene (1985). “A Single-Semantic-Process Theory of Parsing.” Brown University, manuscript.

Charniak, Eugene (1986). “A Neat Theory of Marker Passing”, in *Proceedings of AAAI-86,* Philadelphia.

Charniak, Eugene and Robert Goldman (1988). “A Logic for Semantic Interpretation”, in *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics* (ACL88).

Charniak, Eugene and Saadia Husain (1991). “A New Admissible Heuristic for Minimal-Cost Proofs”. Brown University Department of Computer Science Technical Report CS-91-11.

Charniak, Eugene and Solomon Shimony (1990). “Probabilistic Semantics for Cost-Based Abduction”, in *Proceedings of AAAI-90.*

Charniak, Eugene and Solomon Shimony (1994). “Cost-Based Abduction and MAP Explanation”, in *Artificial Intelligence.* vol. 66, pp. 345-374.

Church, Kenneth and Patrick Hanks (1989). “Word Association Norms, Mutual Information, and Lexicography”, in *Proceedings of ACL89*. Vancouver.

Collins, A.M. and E. Loftus (1975). “A spreading activation theory of semantic processing”, in *Psychological Review.* vol. 82, pp. 407-428

Cowie, Jim, Joe Guthrie, and Louise Guthrie (1992). “Lexical Disambiguation using Simulated Annealing”, in *Proceedings of COLING-92*. Nantes.

Cucciarelli, Alessandro and Paola Velardi (1997). “Automatic Selection of Class Labels from a Thesaurus for an Effective Semantic Tagging of Corpora”, in *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP97).* Washington DC.

Cullingford, Richard (1981). “SAM”, in *Inside Computer Understanding: Five Programs Plus Miniatures*, Roger Schank and Christopher Riesbeck, eds. Hillsdale NJ: Lawrence Erlbaum and Assoc.

Cullingford, Richard and Boyan Onyshkevych (1985). “Lexicon-Driven Machine Translation”, in *Proceedings of the International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-85)*, Colgate NY.

Cullingford, Richard and Boyan Onyshkevych (1987). “An Experiment in Lexicon-Driven Machine Translation”, in *Machine Translation: Theoretical and Methodological Issues*, Sergei Nirenburg, ed. New York: Cambridge University Press.

Dagan, Ido, Alon Itai, and Ulrike Schwall (1991). “Two Languages are More Informative than One”, in *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL91)*. Berkeley.

Dagan, Ido, Shaul Marcus, and Shaul Markovitch (1993). “Contextual Word Similarity and Estimation from Sparse Data”, in *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL93)*. Columbus, Ohio.

Dahlgren, Kathleen, Joyce McDowell, and Edward Stabler (1989). “Knowledge Representation for Commonsense Reasoning with Text”, in *Computational Linguistics* vol 15:3.

Davis, Lawrence (1987). *Genetic Algorithms and Simulated Annealing.* Los Altos CA: Morgan Kaufmann.

Dijkstra, Edsger (1959). “A Note on Two Problems in Connexion with Graphs”, in *Numerische Mathematik*. vol. 1, pp. 269-271.

Dorr, Bonnie (1993). *Machine Translation: A View from the Lexicon.* Cambridge MA: MIT Press.

Dorr, Bonnie (1994). “Machine Translation Divergences: A Formal Description and Proposed Solution”, in *Computational Linguistics,* vol. 20:4.

Dorr, Bonnie, Joseph Garman, and Amy Weinberg (1994). “From Subcategorization Frames to Thematic Roles: Building Lexical Entries for Interlingual MT” in *Proceedings of the First AMTA Conference*. Columbia MD.

Dorr, Bonnie and Clare Voss (1994). “ILustrate: a MT Developers’ Tool with a Two-Component View of the Interlingua” in *Proceedings of the First AMTA Conference*. Columbia MD.

Dyer, Michael and Uri Zernik (1986). “Encoding and Acquiring Meanings for Figurative Phrases”, in *Proceedings of the Conference of the Association for Computational Linguistics*.

Evans, David and Dana Scott (1986). “Concepts as Procedures”, *ESCOL ‘86, Proceedings of the*

Third Eastern States Conference on Linguistics. Pittsburgh PA, pp. 533-543.

Evens, Martha (1988). *Relational Models of the Lexicon: Representing knowledge in Semantic Networks.* New York: Cambridge University Press.

Fahlman, Scott (1982). *NETL: A System for Representing and Using Real-World Knowledge*. Cambridge Mass: MIT Press.

Farwell, David, Louise Guthrie, and Yorick Wilks (1993). “Automatically Creating Lexical Entries for ULTRA, a Multilingual MT System” in *Machine Translation* vol. 8:3, pp. 127-146.

Fass, Dan (1986a). “Collative Semantics: A Description of the Meta5 Program.” Computing Research Laboratory, New Mexico State University. MCCS-86-23.

Fass, Dan (1986b). “Collative Semantics: Lexical Ambiguity Resolution and Semantic Relations (with Particular Reference to Metonymy). New Mexico State University’s Computing Research Lab Technical Report MCCS-86-59.

Fass, Dan (1988a). “Collative Semantics: A Study in the Discrimination of Meaning.” Centre for Systems Science, Simon Fraser University. CSS/LCCR TR88-24.

Fass, Dan (1988b). “An Account of Coherence, Semantic Relations, Metonymy, and Lexical Ambiguity Resolution,” in *Lexical Ambiguity Resolution*, Small, Cottrell, and Tanenhaus, eds. Palo Alto: Morgan Kaufman.

Fass, Dan (1988c). “Collative Semantics: A Semantics for Natural Language Processing”. New Mexico State University’s Computing Research Lab Technical Report MCCS-88-118.

Fass, Dan (1989). “Lexical Semantic Constraints.” Centre for Systems Science, Simon Fraser University. CSS/LCCR TR89-11.

Fass, Dan (1991a). “met*: A Method for Discriminating Metonymy and Metaphor by Computer” in *Computational Linguistics* 17:1, pp. 49-90.

Fass, Dan (1991b). “Metonymy, Case Role Substitution and Sense Ambiguity,” in *Proceedings of the IJCAI Workshop on Computational Approaches to Non-Literal Language*.

Fauconnier, Gilles (1985). *Mental Spaces: Aspects of Meaning Construction in Natural Language*. Cambridge, Mass: MIT Press.

Fillmore, Charles, Paul Kay, and Mary O’Connor (1988). “Regularity and Idiomaticity in Grammatical Constructions: the Case of *Let Alone*”, in *Language* vol. 64, pp. 501-538.

Finin, Timothy (1980). “The Semantic Interpretation of Nominal Compounds,” in *Proceedings of AAAI*.

Finin, Timothy (1986). “Constraining the Interpretation of Nominal Compounds in a Limited Context”, in *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*, Ralph Grishman and Richard Kittredge, eds. Hillsdale NJ: Lawrence Erlbaum Assoc.

Gale, William, Ken Church, David Yarowsky (1992). “One Sense per Discourse”, in *Proceedings of the DARPA Human Language Technologies Workshop*, Harriman NY.

Garey, Michael and David Johnson (1979). *Computers and Intractability: A guide to the Theory of NP-Completeness*. New York: W. H. Freeman and Co.

Gibbons, Alan (1985). *Algorithmic Graph Theory.* New York: Cambridge University Press.

Gibbs, Raymond (1993). “Process and products in making sense of tropes,” in *Metaphor and*

Thought, Andrew Ortony, ed. Cambridge: Cambridge University Press.

Gibson, Edward (1990). “DIMORPH: A Morphological Analyzer.” Center for Machine Translation, Carnegie Mellon University. CMU-CMT-MEMO.

Gibson, Edward (1991a). “Bidirectional Active Chart Parsing.” Center for Machine Translation, Carnegie Mellon University. Draft.

Gibson, Edward (1991b). “BICHART: A Bidirectional Chart Parser.” Center for Machine Translation, Carnegie Mellon University. CMU-CMT-MEMO.

Givón, T. (1986). “Prototypes: Between Plato and Wittgenstein,” in *Noun Classes and Categorization*. Collette Craig, ed. Philadelphia: John Benjamins Publishing Co.

Granger, Richard, Kurt Eiselt, and Jennifer Holbrook (1984). “Parsing with Parallelism: A Spreading-Activation Model of Inference Processing During Text Understanding”. University of California at Irvine, Artificial Intelligence Project Technical Report #228.

Grishman, Ralph and Richard Kittredge (1986). *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*. Hillsdale NJ: Lawrence Erlbaum Associates.

Grosz, Barbara, Douglas Appelt, Paul Martin, and Fernando Pereira (1986). “TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces”. SRI Technical Note 356R.

Guthrie, Louise, Brian Slator, Yorick Wilks, and Rebecca Bruce (1990). “Is there content in empty heads?”, in *Proceedings of COLING-90*. Helsinki.

Guthrie, Louise, James Pustejovsky, Yorick Wilks, and Brian Slator (1996). “The Role of Lexicons in Natural Language Processing”, in *Communications of the ACM*, vol. 39:1.

Harley, Andrew and Dominic Glennon (1997). “Sense Tagging in Action: Combining Different Tests with Additive Weights”, in *Proceedings of ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?* Washington DC.

Hirst, Graeme (1987). *Semantic interpretation and the resolution of ambiguity.* New York: Cambridge University Press.

Hirst, Graeme and Mark Ryan (1992). “Mixed Depth Representations for Natural Language Text,” in *Text-Based Intelligent Systems: Current Research and Practice in Information Extraction and Retrieval*, Paul Jacobs, ed. Hillsdale NJ: Lawrence Erlbaum.

Hobbs, Jerry (1985). “Ontological Promiscuity,” in *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*. Chicago.

Hobbs, Jerry (1991). “Interpretation as Abduction,” in *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*.

Hobbs, Jerry and Paul Martin (1987). “Local Pragmatics”. SRI International Technical Note 429.

Hobbs, Jerry, Mark Stickel, Paul Martin, and Douglas Edwards (1988). “Interpretation as Abduction”, in *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics (ACL88).* Buffalo NY.

Horacek, Helmut (1996). “On Expressing Metonymic Relations in Multiple Languages”, in *Machine Translation*, vol. 11:1-3, pp. 109-158.

Hovy, Eduard (1988). *Generating Natural Language under Pragmatic Constraints*. Yale University, Ph.D. dissertation.

Hovy, Eduard, Julia Lavid, Elisabeth Maier, Vibhu Mittal, Cecile Paris (1992). “Employing

Knowledge Resources in a New Text Planner" in *Aspects of Automated NL Generation*, Dale, Hovy, Rosner, and Stock, eds. Lecture Notes in AI no. 587. Heidelberg: Springer Verlag.

Hovy, Eduard and Elisabeth Maier (1994). "Parsimonious or Profligate: How Many and Which Discourse Structure Relations?" Unpublished ms.

Hutchins, William John (1986). *Machine Translation: Past, Present, Future*. New York: John Wiley and Sons.

Jackendoff, Ray (1983). *Semantics and Cognition*. Cambridge MA: MIT Press.

Jackendoff, Ray (1988). "Conceptual Semantics" in *Meaning and Mental Representations*, Umberto Eco, Marco Santambrogio, and Patrizia Violi, eds. Bloomington IN: Indiana University Press.

Jackendoff, Ray (1990). *Semantic Structures*. Cambridge MA: MIT Press.

Jakobsen, Roman and Morris Halle (1956). *Fundamentals of Language.* 'S-Gravenhage: Mouton & Co.

Jing, Hongyan, Vasileios Hatzivassiloglou, Rebecca Passonneau, and Kathleen McKeown (1997). "Investigating Complementary Methods for Verb Sense Pruning", in *Proceedings of ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?* Washington DC.

Johnson-Laird, Philip (1988). "How is Meaning Mentally Represented?", in *Meaning and Mental Representations*, Umberto Eco, Marco Santambrogio, and Patrizia Violi, eds. Bloomington IN: Indiana University Press.

Jones, Douglas and Boyan Onyshkevych (1997). "Combining Knowledge Sources for Automatic Sense Tagging", in *Proceedings of ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, Washington DC.

Kamei, Shin-ichiro and Takahiro Wakao (1992). "Metonymy: how to treat it properly in a multilingual machine translation system", in *Proceedings of the First Singapore International Conference on Intelligent Systems*. pp. 493-497.

Kamp, Hans (1981). "A Theory of Truth and Semantic Representation" in *Formal Methods in the Study of Language*, J. Groenedijk, J. Janssen, M. Stokhof, eds. Amsterdam: Mathematical Center Tracts.

Katz, Jerrold and Jerry Fodor (1963). "The Structure of a Semantic Theory". *Language*, vol. 39, pp. 170-210.

Kayser, Daniel (1988). "What Kind of Thing is a Concept?", *Computational Intelligence*, vol. 4:2, pp. 158-165.

King, Margaret (1992). "Epilogue: on the relation between computational linguistics and formal semantics," in *Computational Linguistics and Formal Semantics.* Michael Rosner and Roderick Johnson, *eds*. New York: Cambridge University Press.

Kirkpatrick, S., C. Gelatt, and M. Vecchi (1983). "Optimization by Simulated Annealing". *Science*, vol. 220:671-680.

Knight, Kevin and Steven Luk (1994). "Building a Large-Scale Knowledge Base for Machine Translation", in *Proceedings of AAAI94.*

Korf, Richard (1988). "Optimal Path Finding Algorithms", in *Search in Artificial Intelligence*, Laveen Kanal and Vipin Kumer, eds. New York: Springer Verlag.

Kozima, Hideki and Teiji Furugori (1993). “Similarity between Words Computed by Spreading Activation on an English Dictionary”, in *Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics* (EACL93), Utrecht.

Kozima, Hideki and Akira Ito (1995). “Context-Sensitive Measurement of Word Distance by Adaptive Scaling of a Semantic Space”, in *Proceedings of the International Conference on Recent Advances in NLP* (RANLP95). Tzigov Chark, Bulgaria.

van Laarhoven, P. and P. Aarts. (1987). *Simulated Annealing: Theory and Applications*. Dordrecht: Kluwer Academic.

Lakoff, George (1987). *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. Chicago: University of Chicago Press.

Lakoff, George (1988). “Cognitive Semantics”, in *Meaning and Mental Representations*, Umberto Eco, Marco Santambrogio, and Patrizia Violi, eds. Bloomington IN: Indiana University Press.pp. 119-154.

Lakoff, George and Mark Johnson (1980). *Metaphors We Live By.* Chicago: University of Chicago Press.

Leacock, Claudia, Geoffrey Towell, and Ellen Voorhees (1993a). “Corpus-based statistical sense resolution”, in *Proceedings of the ARPA Human Language Technologies Workshop.*

Leacock, Claudia, Geoffrey Towell, and Ellen Voorhees (1993b). “Towards building contextual representations of word senses using statistical models”, in *Proceedings of the SIGLEX Workshop: Acquisition of Lexical Knowledge from Text.*

Lees, Robert (1960). “The Grammar of English Nominalizations”, in *International Journal of American Linguistics*, vol. 26:3.

Lees, Robert (1966). *The Grammar of English Nominalizations*. Bloomington: Indiana University.

van Leeuwen, Jan, *ed*. (1990). *Handbook of Theoretical Computer Science: Algorithms and Complexity.* New York: Elsevier and Cambridge MA: MIT Press.

Lehman, Jill (1994). “Toward the Essential Nature of Statistical Knowledge in Sense Resolution”, in *Proceedings of AAAI-94.*

Lehnert, Wendy (1988). “The Analysis of Nominal Compounds,” in *Meaning and Mental Representations,* Umberto Eco, Marco Santambrogio, and Patrizia Violi, eds. Bloomington IN: Indiana University Press.

Lenat, Douglas, Ramanathan Guha, Karen Pittman, Dexter Pratt, and Mary Shepherd (1990). “CYC: Toward Programs with Common Sense”, in *Communications of the ACM (CACM).* vol. 33:8.

Lesk, Michael (1986). “Automatic Sense Disambiguation using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone”, in *Proceedings of the Fifth International Conference on Systems Documentation (ACM SIGDOC)*. Toronto.

Levin, Beth (1989). “English Verbal Diathesis”. Lexicon Project Working Papers #32. Massachusetts Institute of Technology.

Levin, Beth (1991). “Building a Lexicon: The Contribution of Linguistics” in *International Journal of Lexicography.* vol. 4:3, pp. 205-226.

Levin, Beth (1993). *English Verb Classes and Alternations: A Preliminary Investigation.* Chicago: University of Chicago Press.

Levin, Lori and Sergei Nirenburg (1994a). “Construction-Based MT Lexicons” in *Current Issues in Computational Linguistics: In Honour of Don Walker*, Antonio Zampolli, Nicoletta Calzolari, Martha Palmer, eds. Kluwer Academic and Giardini Editori E Stampatori in Pisa.

Levin, Lori and Sergei Nirenburg (1994b). “The Correct Place of Lexical Semantics in Interlingual MT”, in *Proceedings of COLING 94*.

Lewis, David (1972). “General Semantics”, in Davidson and Gil Harman, *eds., Semantics of Natural Language*. Dordrecht: Reidel.

Light, Marc, *ed.* (1997). *Proceedings of ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?* Washington DC.

Lowe, John, Collin Baker, and Charles Fillmore (1997). “A Frame-Semantic Approach to Semantic Annotation”, in *Proceedings of ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?* Washington DC.

Luk, Alpha (1995). “Statistical sense disambiguation with relatively small corpora using dictionary definitions”, in *Proceedings of ACL (ACL95).*

MADCOW (1992). “Multi-Site Data Collection for a Spoken Language Corpus”, in *Proceedings of the DARPA Speech and Natural Language Workshop*, Harriman NY. (MADCOW = the Multi-site ATIS Data COllection Working group).

Mahesh, Kavi (1996). “Ontology Development for Machine Translation: Ideology and Methodology”. New Mexico State University, Computing Research Laboratory Technical Report MCCS 96-292

Mahesh, Kavi and Sergei Nirenburg (1995). “A Situated Ontology for Practical NLP,” in *Proceedings of the IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing*. August 19-21, Montreal.

Mahesh, Kavi, Stephen Beale, Sergei Nirenburg, and Boyan Onyshkevych (1996a). “Ontology-Based Ambiguity Resolution and Non-Literal Interpretation”, in *Proceedings of the International Conference on Knowledge Based Computer Systems (KBCS-96),* Bombay India.

Mahesh, Kavi, Sergei Nirenburg, Jim Bowie, David Farwell (1996b). “An Assessment of CYC for Natural Language Processing”. New Mexico State University, Computing Research Laboratory Technical Report MCCS 96-302.

McDermott (1978). “Tarskian Semantics, or No notation without denotation!” in *Cognitive Science*. vol. 2:3.

McRoy, Susan (1992). “Using Multiple Knowledge Sources for Word Sense Discrimination”, in *Computational Linguistics*, vol. 18:1.

Melamed, Dan (1997). “Measuring Semantic Entropy”, in *Proceedings of ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?* Washington DC.

Mel’chuk, Igor and Alexander Zholkovsky (1984). *Explanatory Combinatorial Dictionary of Modern Russian: Semantico-Syntactic Studies of Russian Vocabulary.* Vienna: Wiener Slavistischer Almanach.

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller (1953). “Equation of State Calculations by Fast Computing Machines”, in *Journal of Chem. Physics*. vol 21:6, pp. 1087-1092.

Meyer, Ingrid and James Steele (1990). “The Presentation of an Entry and of a Super-Entry in an Explanatory Combinatorial Dictionary,” in *The Meaning-Text Theory of Language: Linguistics, Lexicography, and Practical Implications*, James Steele, ed. Ottawa: University of Ottawa Press.

Meyer, Ingrid, Boyan Onyshkevych, and Lynn Carlson (1990). “Lexicographic Principles and Design for Knowledge-Based Machine Translation.” Center for Machine Translation, Carnegie Mellon University. CMU-CMT-90-118.

Miller, George, Claudia Leacock, Randee Tengi, Ross Bunker (1993). “A Semantic Concordance”, *Proceedings of DARPA Speech and Natural Language Workshop*.

Miller, George and Daniel Teibel (1991). “A proposal for Lexical Disambiguation”, *Proceedings of DARPA Speech and Natural Language Workshop*. pp. 395-399.

Mitamura, Teruko (1990). “The Hierarchical Organization of Predicate Frames for Interpretive Mapping in Natural Language Processing.” Ph.D. Dissertation. Center for Machine Translation, Carnegie Mellon University. CMU-CMT-90-117.

Monarch, Ira (1989). “ONTOS: Reference Manual.” Center for Machine Translation, Carnegie Mellon University. CMU-CMT-MEMO.

Morris, Jane and Graeme Hirst (1991). “Lexical Cohesion Computed by Thesaural Relations as an Indicator of the Structure of Text”, in *Computational Linguistics*. vol. 17:1.

Nilsson, Nils (1980). *Principles of Artificial Intelligence*. San Mateo CA: Morgan Kaufmann Publishers.

Nirenburg, Sergei, Victor Raskin, and Allen Tucker (1987). “The Structure of Interlingua in TRANSLATOR” in *Machine Translation: Theoretical and Methodological Issues,* Sergei Nirenburg, ed. NY: Cambridge University Press.

Nirenburg, Sergei and Kenneth Goodman (1990). “Treatment of Meaning in MT Systems,” *Proceedings of the Third International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language*. Linguistic Research Center, University of Texas at Austin.

Nirenburg, Sergei, Lynn Carlson, Ingrid Meyer, and Boyan Onyshkevych (1990). “Lexicons for KBMT”, in *Proceedings of International Workshop on Electronic Dictionaries*, Kanagawa Japan.

Nirenburg, Sergei and Christine Defrise (1991). “Practical Computational Linguistics,” in *Computational Linguistics and Formal Semantics,* R. Johnson and M. Rosner, eds. Cambridge: Cambridge University Press.

Nirenburg, Sergei, Jaime Carbonell, Masaru Tomita, and Kenneth Goodman (1992). *Machine Translation: A Knowledge-Based Approach.* San Mateo CA: Morgan Kaufmann Publishers.

Nirenburg, Sergei and Lori Levin (1992). “Syntax-Driven and Ontology-Driven Lexical Semantics” in *Lexical Semantics and Knowledge Representation,* James Pustejovsky, ed. Heidelberg: Springer Verlag.

Nirenburg, Sergei and Victor Raskin (1996). “Ten Choices for Lexical Semantics”. New Mexico State University, Computing Research Laboratory Technical Report MCCS 96-304.

Nirenburg, Sergei, Victor Raskin and Boyan Onyshkevych. (1995). “Apologiae Ontologiae”, in *Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation (TMI-95),* Leuven, Belgium.

Nirenburg, Sergei, Steven Beale, Kavi Mahesh, Boyan Onyshkevych, Victor Raskin, Evelyne Viegas, Yorick Wilks and Remi Zajac. (1996a). “Lexicons in the Mikrokosmos project”, in *Proceedings of the Society for Artificial Intelligence and Simulated Behavior, Workshop* *on Multilinguality in the Lexicon*, Brighton, UK.

Nirenburg, Sergei, Kavi Mahesh, Evelyne Viegas, Steve Beale, Victor Raskin and Boyan Onyshkevych (1996b). “Technological and Conceptual Tools for Lexical Knowledge Acquisition”, in *Proceedings of DIALOGUE-96*, Moscow, Russia.

Norvig, Peter (1989). “Marker Passing as a Weak Method for Text Inferencing” *Cognitive Science*: 13:4

Nunberg, Geoffrey (1978). *The Pragmatics of Reference*. CUNY Dissertation. Reprinted by Bloomington IN: Indiana University Linguistics Club.

Nunberg, Geoffrey (1993). “Transfers of Meaning”, in *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL93)*. Columbus, Ohio.

Nyberg, Eric (1988). “The FRAMEKIT User’s Guide, Version 2.0.” Center for Machine Translation, Carnegie Mellon University. CMU-CMT-MEMO.

Nyberg, Eric and Teruko Mitamura (1992). “The KANT System: Fast, Accurate, High-Quality Translation in Practical Domains” in *Proceedings of COLING-92*. Trento.

Oflazer, Kemal and Okan Yilmaz (1996). “A Constraint-Based Case Frame Lexicon”, in *Proceedings of COLING-96*.

Onyshkevych, Boyan and Sergei Nirenburg (1991). “Lexicons, Ontology, and Text Meaning”, in *Lexical Semantics and Knowledge Representation: Proceedings of a Workshop Sponsored by the SIGLEX of the Association for Computational Linguistics*, Berkeley CA. Reprinted in *Lexical Semantics and Knowledge Representation*, James Pustejovsky, ed. Heidelberg: Springer Verlag (1992).

Onyshkevych, Boyan and Sergei Nirenburg (1994). “The Lexicon in the Scheme of KBMT Things”. New Mexico State University, Computing Research Laboratory Technical Report MCCS-94-277.

Onyshkevych, Boyan and Sergei Nirenburg (1995). “A Lexicon for Knowledge-Based MT”, *in Machine Translation*, 10:1-2

Ostler, Nicholas and B.T.S. Atkins (1992). “Predictable Meaning Shift: Some Linguistic Properties of Lexical Implication Rules” in *Lexical Semantics and Knowledge Representation*, James Pustejovsky, ed. Heidelberg: Springer Verlag.

Pearl, Judea (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. New York: Addison Wesley.

Pollack, Martha and Fernando Pereira (1988). “An Integrated Framework for Semantic and Pragmatic Interpretation”, in *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics* (ACL88).

Prince, Ellen (1981). “Toward a Taxonomy of Given-New Information”, in P. Cole, *ed.*, *Radical Pragmatics*. New York: Academic Press.

Procter, Paul *et al*. (1978). *Longman Dictionary of Contemporary English*. Harlow UK: Longman Group Unlimited.

Procter, Paul, *ed.* (1995). *Cambridge International Dictionary of English*. Cambridge: Cambridge University Press.

Pustejovsky, James (1991). “The Generative Lexicon” in *Computational Linguistics*, 17:4.

Pustejovsky, James (1995). *The Generative Lexicon*. Cambridge MA: MIT Press.

Pustejovsky, James and Pierrette Bouillon (1995). "Aspectual Coercion and Logical Polysemy", in *Journal of Semantics,* vol. 12, pp. 133-162.

Quillian, M. Ross (1968). "Semantic Memory", in *Semantic Information Processing,* Marvin Minsky, ed. Cambridge MA: MIT Press.

Rada, R., H. Mili, E. Bicknell, and M. Blettner (1989). "Development of an Application of a Metric on Semantic Nets", *IEEE Transactions on Systems, Man, and Cybernetics* vol.19:1, pp. 17-30.

Raskin, Victor (1990) "Ontology, sublanguage, and semantic networks in NLP", in *Advances in AI: NL and KB systems*. M. Golumbic, ed. New York: Springer Verlag.

Raskin, Victor and Sergei Nirenburg (1995). "Lexical Semantics of Adjectives." New Mexico State University, Computing Research Laboratory Technical Report, MCCS-95-288.

Raskin, Victor, Salvatore Attardo, and Donalee Attardo (1994). "Augmenting Formal Semantic Representation for NLP: The Story of SMEARR", in *Machine Translation*, 9:2.

Resnik, Philip (1995a). "Using Information Content to Evaluate Semantic Similarity in a Taxonomy", in *Proceedings of IJCAI-95.*

Resnik, Philip (1995b). "Disambiguating Noun Groupings with Respect to WordNet Senses", in *Proceedings of the Third Workshop on Very Large Corpora*. Cambridge MA.

Resnik, Philip (1997). "Selectional Preferences and Sense Disambiguation", in *Proceedings of ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?* Washington DC.

Richardson, R., A. Smeaton, and J. Murphy (1994). "Using Wordnet as a Knowledge Base for Measuring Semantic Similarity between Words". Working Paper CA-1294, School of Computer Applications, Dublin City University.

Rosch, Eleanor, Carolyn Mervis, Wayne Gray, David Johnson, and Penny Boyes-Bream (1976). "Basic Objects in Natural Categories." *Cognitive Psychology.* vol. 8:382-439.

Sampson, G. (1986). "A Stochastic approach to Parsing", in *Proceedings of the 11th International Conference on Computational Linguistics*. Bonn.

Santos, Eugene (1994). "A linear constraint satisfaction approach to cost-based abduction", in *Artificial Intelligence.* vol. 65, pp. 1-27.

Schank, Roger (1973). "Identification of Conceptualizations Underlying Natural Language" in *Computer Models of Thought and Language,* Roger Schank and Kenneth Colby, eds. San Francisco: W.H. Freeman Co.

Schank, Roger, *ed.* (1975). *Conceptual Information Processing.* Amsterdam: North Holland.

Schank, Roger and Robert Abelson (1977). *Scripts, Plans, Goals, and Understanding.* Hillsdale NJ: Lawrence Erlbaum.

Sedgewick, Robert (1983). *Algorithms.* Reading Mass: Addison Wesley.

Selman, Bart and Graeme Hirst (1987). "Parsing as an Energy Minimization Problem", in Lawrence Davis, *ed. Genetic Algorithms and Simulated Annealing.* Los Altos CA: Morgan Kaufmann.

Skuce, Douglas and Ira Monarch (1990). "Ontological Issues in Knowledge Base Design: Some Problems and Suggestions." Center for Machine Translation, Carnegie Mellon University. Technical Report CMU-CMT-119.

Smadja, Frank and Kathleen McKeown (1990). “Automatically Extracting and Representing Collocations for Language Generation”, in *Proceedings of ACL90.*

Small, Steve and Chuck Rieger (1982). “Parsing and Comprehending with Word Experts (A Theory and Its Realization),” in *Strategies for Natural Language Processing*, Wendy Lehnert and Martin Ringle, ed. Hillsdale NJ: Lawrence Erlbaum Associates.

Sowa, John (1993) “Lexical Structures and Conceptual Structures”, in *Semantics and the Lexicon*, James Pustejovsky, ed. Boston: Kluwer.

Stallard, David (1993). “Two Kinds of Metonymy”, in *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL93)*. Columbus, Ohio.

Stern, Gustaf (1965). *Meaning and Change of Meaning*. Bloomington: Indiana University Press. (first edition: 1931)

Sundheim, Beth, *ed.* (1995). *The Proceedings of the Sixth Message Understanding Conference (MUC-6)*. San Francisco: Morgan Kaufmann.

Touretzky, David (1986). *The Mathematics of Inheritance Systems.* Palo Alto CA: Morgan Kaufmann.

Tulving, Endel (1985). “How many memories are there?” in *American Psychologist*. vol. 40, pp. 385-398.

Tversky, A. (1977). “Features of Similarity”, in *Psychological Review.* vol. 84:4. pp. 327-352.

Veronis, Jean and Nancy Ide (1990). “Word Sense Disambiguation with Very Large Neural Networks Extracted from Machine Readable Dictionaries”, in *Proceedings of COLING-90.* Helsinki.

Viegas, Evelyne, Kavi Mahesh, and Sergei Nirenburg (1997). “Semantics in Action”, in *Predicative Forms in Natural Language and in Lexical Knowledge Bases,* Patrick Saint-Dizier, ed. Boston: Kluwer Academic Publishers.

Viegas, Evelyne, Boyan Onyshkevych, Victor Raskin and Sergei Nirenburg. (1996). “From *Submit* to *Submitted* via *Submission*: on Lexical Rules in Large-scale Lexicon Acquisition”, in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL-96)*, Santa Cruz, CA.

Vossen, Piek, Willem Meijs, and Marianne den Broeder (1989). “Meaning and structure in dictionary definitions”, in Bran Boguraev and Ted Briscoe, eds. *Computational Lexicography for Natural Language Processing.* New York: Longman

Wagner, Robert (1976). “A Shortest Path Algorithm for Edge-Sparse Graphs”, in *Journal of the Association for Computing Machinery.* vol. 23:1, pp. 50-57.

Wakao, Takahiro and Stephen Helmreich (1993). “Translation of Metonymy in an Interlingual MT System”, in *Proceedings of the Pacific Association for Computational Linguistics (PACLING)*, Vancouver.

Waltz, David and Jordan Pollack (1985). “Massively Parallel Parsing: A Strongly Interactive Model of Natural Language Interpretation”, in *Cognitive Science*. vol. 9, pp. 51-74.

Warren, Beatrice (1978). *Semantic Patterns of Noun-Noun Compounds*. Göteborg: Acta Universitatis Gothoburgensis.

White, John (1988). “Determination of lexical-semantic relations for multi-lingual terminology structures”, in Martha Evens, ed. *Relational Models of the Lexicon: Representing knowledge in Semantic Networks.* New York: Cambridge University Press

Wierzbicka, Anna (1992). *Semantics, Culture, and Cognition: Universal Human Concepts in Culture-Specific Configurations*. New York: Oxford University Press.

Wilks, Yorick (1973). “An Artificial Intelligence Approach to Machine Translation” in *Computer Models of Thought and Language*, Roger Schank and Kenneth Colby, eds. San Francisco: W.H. Freeman Co.

Wilks, Yorick (1975a). “An intelligent analyzer and understander of English”, *Communications of ACM*. vol. 18:5, pp. 264-274.

Wilks, Yorick (1975b). “Preference Semantics” in *Formal Semantics of Natural Language*, E.L. Keenan, ed. Cambridge: Cambridge University Press.

Wilks, Yorick (1992a). “Review of Ray Jackendoff’s *Semantic Structures”*, in Computational Linguistics, vol. 18:1 pp. 95-97.

Wilks, Yorick (1992b). “Form and Content in Semantics” in *Computational Linguistics and Formal Semantics.* Michael Rosner and Roderick Johnson, eds. New York: Cambridge University Press.

Wilks, Yorick (1996). “Interlinguas: natural languages, logics, or arbitrary notations?” in the *Working Notes from the Pre-Workshop on ILs and IL approaches to MT* (unpublished), Montreal CA.

Wilks, Yorick, Louise Guthrie, Joe Guthrie, and Jim Cowie (1992). “Combining Weak Methods in Large-Scale Text Processing”, in *Text-Based Intelligent Systems: Current Research and Practice in Information Extraction and Retrieval*, Paul Jacobs, ed. Hillsdale NJ: Lawrence Erlbaum Associates.

Wilks, Yorick and Mark Stevenson (1996). “The Grammar of Sense: Is word-sense tagging much more than part-of-speech tagging?” Computation and Language e-print archive, cmp-lg/9607028.

Wilks, Yorick and Mark Stevenson (1997). “Sense Tagging: Semantic Tagging with a Lexicon”, in *Proceedings of ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?* Washington DC.

Winograd, Terry (1983). *Language as a Cognitive Process.* Reading MA: Addison-Wesley.

Winston, Morton, Roger Chaffin, and Douglas Herrmann (1987). “A Taxonomy of Part-Whole Relations,” in *Cognitive Science*, vol. 11, pp. 417-444.

Wittgenstein, Ludwig (1921). *Tractatus Logico-Philosophicus.* Translation by D. Pears and B. McGuiness, Atlantic Highlands NJ: Humanities Press International, 1964.

Wittgenstein, Ludwig (1953). *Philosophical Investigations.* Translation by G. Anscombe, Englewood Cliffs NJ: Prentice Hall, 1958.

Woods, William (1970). “Transition Network Grammars for Natural Language Analysis”, in *CACM*. vol. 13:10.

Woods, William (1975). “What’s in a Link: Foundations for Semantic Networks”, in *Representations and Understanding*, D. Bobrow and A. Collins, eds. San Diego: Academic Press.

Yamanashi, Masa-aki (1987). “Metonymic Interpretation and Associative Processes in Natural Language.” in *Language and AI (Proceedings of international symposium in Kyoto 1986)*, Makoto Nagao (ed). Elsevier/North Holland, pp. 77-86

Yarowsky, David (1992). “Word Sense Disambiguation using Statistical Models of Roget’s Categories Trained on Large Corpora”, in *Proceedings of COLING-92*. Nantes.

Yarowsky, David (1995). “Unsupervised word sense disambiguation rivaling supervised methods”, in *Proceedings of ACL95*. Cambridge MA.

Zernik, Uri and Paul Jacobs (1990). “Tagging for Learning: Collecting Thematic Relations from Corpus”, in *Proceedings of COLING-90.* Helsinki.

Zhivov, Victor (1978). “Some Typological Observations Concerning Noun Compounds”, in *Linguistics,* vol. 208 pp. 5-12.