Generating Lexical Options by Matching in a Knowledge Base

Mara Anita Miezitis

Technical Report CSRI-217
September 1988

Computer Systems Research Institute
University of Toronto
Toronto, Canada
M5S 1A1

Generating Lexical Options by Matching in a Knowledge Base

by

Mara Anita Miezitis

Department of Computer Science
University of Toronto
Toronto, Ontario, Canada

September 1988

A Thesis submitted in conformity with the requirements
for the degree of Master of Science

Abstract

The goal of this thesis is to explore the development of LOG, a Lexical Option Generator, to provide *different* lexical options that may be used to express the *same* information. In addition to syntactic information, LOG provides pragmatic and stylistic information with these options. The lexical units supplied by LOG may be words or phrases — including, possibly, *idioms*. LOG requires the *matching* of input information to a knowledge base of *situations* specifying the state of affairs that license the use of a particular lexical unit in output text. A representative survey of matching techniques is given, highlighting those aspects of the matchers desirable for LOG. The matching technique employed by LOG, using a *magnetization* process for directing information to situations likely to match, is presented in detail.

Acknowledgements

First and foremost, I thank Graeme Hirst, my supervisor. His patience, guidance, encouragement, and generosity with his much sought-after time were invaluable. He always believed in me even when I did not believe in myself.

I also thank John Mylopoulos, my second reader, who read this thesis on very short notice.

I wish to thank Chrysanne DiMarco who was always eager to read any portion of this thesis and provided helpful suggestions. I also thank Diane Horton, Mark Catt, Evan Steeg, and Dan Lyons who gave useful comments.

The pictures in this thesis were drawn by Gabe Nemeth, and I truly thank him.

I wish to thank my parents, Ligita and Juris, and brother and sister-in-law, Ivy and Line, for lifting my spirits and for providing encouragement and support.

My office mates have supplied the perfect mix of brilliant academic discussion, intensely non-academic discussion, encouragement, fun, aggravation, and laughter.

I thank my friends, both in and out of the department, who were precious in this endeavour. They were always there when I needed them.

Finally, I gratefully acknowledge financial support from Bell-Northern Research Limited and from the Natural Sciences and Engineering Research Council of Canada.

Contents

List of Figures v

List of Tables vi

1 Introduction 1
- 1.1 Lexical Options 1
- 1.2 Lexical Units and Situations 4
  - 1.2.1 Words and Lexical Units 4
  - 1.2.2 Idioms 5
  - 1.2.3 Situations 15
- 1.3 Summary 17

2 Matching in a Knowledge Base 18
- 2.1 Partial and Full Matching 18
- 2.2 The Best Match 19
- 2.3 The Roles of Matching 20
- 2.4 A Survey of Matching 20
  - 2.4.1 Multiply-Specified Matching 21
  - 2.4.2 KL-ONE Classification 24
  - 2.4.3 Numerical Determination of Matches 26
  - 2.4.4 Purely Structural Mapping 27
  - 2.4.5 Structured Associations 27
  - 2.4.6 A Marker-Passing Approach 28
- 2.5 Summary of Desirable Matcher Characteristics 29

3 LOG: The Lexical Option Generator 31

4 Lexicon Organization 35
- 4.1 A Hierarchical Network Representation of a Lexicon 35
- 4.2 Structuring the Lexicon 37
  - 4.2.1 Linking the Nodes 37
  - 4.2.2 Inheritance 40
- 4.3 The Node Structure 40
  - 4.3.1 The Situation Template 41
  - 4.3.2 The Syntactic Template 42

4.3.3 Idioms in a Syntactic Template 45

5 Matching in LOG **47**
5.1 Marker Passing – A Precursor to LOG 47
5.2 Input to LOG 49
5.3 Magnetization 50
5.4 Passing Information in the Network 51
5.5 The Reference Number and Clearing the Network 53
5.6 Matching Situations in a Node 54
5.7 Examples 54
5.8 Summary 60

6 Conclusion **61**
6.1 Contributions 61
6.2 Related Research 62
6.2.1 LOQUI 62
6.2.2 DIOGENES 63
6.2.3 FIG 65
6.2.4 Penman 67
6.2.5 PAULINE 68
6.2.6 Summary 68
6.3 Areas for Future Research 68

References **70**

List of Figures

3.1 LOG: The Lexical Option Generator 32
3.2 Partial LOG output for `(love (agent mary1)(patient john1))` 33

4.1 `Narcissism` as a restricted `love` situation. 37
4.2 Adding an attribute to the `eat` situation to make a `gobble` situation. . . 38
4.3 Three syntactic templates for `narcissim` 43
4.4 Syntactic template for "bury the hatchet" 46

5.1 A sample portion of the lexicon 50
5.2 The `love` portion of a network. 55
5.3 LOG output for Example 1. 56
5.4 Additional LOG output for Example 2. 57
5.5 The `eat` portion of a network. 58
5.6 The `violent-action` portion of the network. 59

List of Tables

1.1 Verb plus particle or preposition idioms 11
1.2 Complex idioms 12

4.1 Formality levels in the syntactic template 45

Chapter 1

Introduction

This thesis explores the development of a Lexical Option Generator to be used by a natural language generator to allow the incorporation of greater variety and style into its output text. The Lexical Option Generator provides lexical units that are appropriate for conveying the desired information. Its output is the *different* options that may be used by the natural language generator to express the *same* information. A generator could then use this "mini-lexicon" to achieve more "natural" language.

The appropriate lexical options are found by *matching* the input information to situations in the knowledge base, each indicating the state of affairs necessary for expressing a portion of the input information by one or more of the lexical units associated with the situation. This thesis proposes a matching technique, that includes a "magnetization" process for expediting the matching of the input information to the situations in the knowledge base.

This chapter elaborates upon lexical options and differentiates them from syntactic variations. The relationship between lexical units and situations is then explained. The importance of collapsing the notions of word and idiom into the one term *lexical unit* is detailed by an in-depth characterization of idioms.

Chapter 2 presents a survey of matching techniques to determine those characteristics of a matcher desirable for the Lexical Option Generator. Chapter 3 provides an overview of the Lexical Option Generator, LOG. Chapters 4 and 5 detail further the lexicon organization and the matching operation.

1.1 Lexical Options

Undoubtedly, the same propositional information can be expressed in many different ways with variations on the lexical, syntactic, and (for spoken language) phonological levels of language.

For instance, all of the following sentences convey the same fact about John kissing Mary:

(1) John <u>kissed</u> Mary.

(2) Mary <u>was kissed</u> by John.

(3) What John did to Mary was kiss her.

(4) A kiss is what John gave Mary.

(5) It was a kiss that John gave Mary.

(6) It was Mary that John kissed.

(7) It was John that kissed Mary.

(8) John gave Mary a kiss.

(9) John gave Mary a big wet one.

(10) A big wet one is what John gave Mary.

(11) John pressed his lips to Mary's.

Sentence (2) is a passivization of (1), reflecting a purely syntactic variation. Sentences (3) to (7) seem to be answers to different questions about the fact. Through syntactic variation (clefting, topicalization, pseudo-clefting, etc.) different portions of the fact are brought into focus. Note that these syntactic variations are accomplished through standard transformations that do not involve a great deal of lexical choice other than using the noun or verb form of the concept *kiss*. However, sentences (8) to (11) involve variations in lexical choice.

Consider the concept *kiss*, which is expressed as:

- a noun "kiss" in sentences (4) and (5),
- a verb "kiss", appropriately inflected, in sentences (1), (2), (3), (6), and (7),
- a verb phrase idiom "give a kiss" in sentence (8),
- a verb phrase idiom "give a big wet one" in sentence (9),
- a noun phrase idiom "big wet one" in sentence (10), and
- a verb phrase idiom "pressed his lips to" in sentence (11).

Each of these lexical units conveys the same information[1] in the context of sentences (1) through (11). However the use of a particular option in a sentence imposes syntactic constraints depending on the part of speech of the lexical unit and its transitivity. The importance of lexical options in language generation is emphasized by Pustejovsky and Nirenburg:

> ... lexical selection is not just a side effect of grammatical decisions but rather acts to flexibly constrain concurrent and later generation decisions of either lexical or grammatical type. (Pustejovsky and Nirenburg, 1987, page 202)

[1]Although the same literal information is conveyed by the different lexical units for the concept *kiss*, each may convey different pragmatic information.

This thesis is about determining the lexical options possible to convey the same information. Lexical selection is the process of choosing a lexical unit from among the lexical options, and will not be dealt with in this thesis.

Extensive consideration of generating lexical options and of lexical selection has been set aside in most generation systems to date. Generation systems have been concerned with other fundamental issues in language generation, such as grammatical knowledge and control (Pustejovsky and Nirenburg, 1987), that are required in order to convey *any* information. However, with the recent interest in producing "more natural" natural language by incorporating pragmatic constraints (Hovy, 1987), focus and other discourse information (Pustejovsky and Nirenburg, 1987), and stylistics (DiMarco and Hirst, 1988), lexical options and selection must come into play.

Consider the following two sentences:

(12) John hoped for snow on the ground at Christmas.

(13) John hoped for a white Christmas.

The idiom "a white Christmas", which can be considered to be one lexical unit (see section 1.2.2), summarizes the same state of affairs as "snow on the ground at Christmas" even though it is superficially not a *single* lexical unit but a noun phrase *composed* of lexical units. Note that "December 25th" could be replaced for "Christmas" in (12) but not (13) due to the idiomaticity of "a white Christmas". This emphasizes the difference between *providing* a lexical option and correctly *organizing* these lexical options both syntactically and pragmatically.

Totally different sentences, unrecognizable as lexical substitutions, may result from different lexical choices. It is important to note that the choice of one lexical unit may constrain the choice of other lexical units to express the rest of the information. For instance:

(14) John was in a no-win situation.

(15) John had no viable options to pursue.

There is no obvious one-to-one correspondence between the lexical items in (14) and those in (15), where "no-win situation" is an idiom and thus a single lexical unit. The choice of *one* particular word in *one* sentence (except for "John") may preclude the use of a word in the other, either because the information conveyed may already have been covered by a chosen lexical unit, or because there is no way to fit the new lexical unit into the structure built so far.

This thesis proposes a method for determining the different options that may be used to express the same information, through the recognition that both words and idioms indicate a *situation*, or state of affairs, which must be present for a word or idiom to be used. As shown above, sometimes lexical units can be directly substituted for one another in a given state of affairs and, at other times, different lexical units cover different portions of the state of affairs and are thus not substitutable. The lexical option problem becomes one of *matching* the information to be conveyed to situations which can be encompassed by a word or idiom. Different lexical units may cover different portions of the information.

1.2 Lexical Units and Situations

Disagreement exists about the linguistic definitions of, and distinctions between, *word* and *idiom*, but for the purposes of this thesis, both will be called *lexical units*. No real distinction need be made between words and idioms, since the important factor to note is that they both represent *situations*. Section 1.2.1 clarifies the meaning of *lexical unit*, and section 1.2.2 illustrates why *idioms* need not be distinguished from *words* in the definition of lexical unit for the Lexical Choice Generator. Finally, the meaning of *situation* is clarified with some examples.

1.2.1 Words and Lexical Units

A naïve definition of word might read: "a sequence of letters delimited by punctuation marks and blanks". However, this is much too simplistic. For example, the information that is expressed by syntactic structuring in English is often incorporated into a single "word" in an agglutinative language (Winograd, 1983).

"Word" is usually considered to be a *notion* rather than something concrete and definable. Generally, two characteristics are used to distinguish this notion: the *external mobility* and *internal stability* of a word (Winograd, 1983). The external mobility characteristic means that a word has positional mobility, as a whole, within a sentence. Words tend to be the smallest mobile units. Internal stability means that words tend to not be interruptible by insertion of new material. A morpheme is considered to be the basic unit of meaning and words are composed of these morphemes. DiSciullo and Williams (1987) claim there are three separate concepts identified with "word":

1. *Morphological objects* are defined by a set of morphemes (basic units of meaning) and their rules of combination (e.g., through affixing and compounding). For example, "compute", "-er", and "-ize" are each morphemes composing the morphological object "computerize".

2. *Syntactic atoms* are those units unanalyzable with respect to syntax. Syntactic atoms are in most cases equivalent to morphological objects. However, included within the class of syntactic atoms are *syntactic words* such as verb-complement compounds in French (e.g., *essui-glace* for windshield wiper), opaque to syntactic rules though they are phrases reanalyzed as words:

   > In fact both syntactic words and listed syntactic phrases [e.g., verb phrase idioms] are idiomatic; that is, they are listed and have non-compositional meanings. Their difference is structural; the syntactic words are X^0s [map exactly into a word class such as noun] and exhibit syntactic atomicity, whereas the listed phrases are X^{max}s and exhibit syntactic transparency. (Di Sciullo and Williams, 1987, page 80)

3. *Listemes* are the "listed" units of language (such as in dictionaries or, according to some cognitive theories, in the mental lexicon). The form and properties of a listeme must be "memorized", since they are not a function of the other listemes. For example, the word "love" is a listeme, but the inflections of "love" would not

be listemes as they follow a regular pattern. Similarly, an idiom such as "bury the hatchet" would be a listeme, while the various transformations and inflections of the idiom would not.

In this thesis the focus will be on the second concept, syntactic atoms, encompassing the notions of internal stability and external mobility. These syntactic atoms, together with idioms, form what will be called *lexical units*.

This thesis will concentrate on *content* lexical units (e.g., those lexical units in the open-class categories of English) which have well-defined semantic relations indicating the situation in which the lexical unit may be used. Usually, the closed-class items (grammatical morphemes) must be determined syntactically rather than through the semantic approach taken in this thesis:

> While open-class words express the meaning of the sentence, closed-class units contribute to sentence meaning only indirectly by signalling a small set of semantic relations between referents. (Summary of the work of Forster (1979) and Garrett (1975) in (Morrow, 1986, page 424))

However, Morrow proposes:

> Whereas content words express object and relation categories (e.g. *car, run*), grammatical morphemes express a relatively small set of conceptual distinctions that apply to most object and relation categories. (Morrow, 1986, page 424)

These grammatical morpheme semantic distinctions would be high up in the semantic hierarchy, requiring much abstraction to determine. This thesis will not focus on such grammatical morphemes.

A lexical unit must thus have one semantic unit and one syntactic unit. Furthermore, lexical units may, and in general do, have different *senses* in that they may have a single syntactic unit but more than one semantic unit. For example, the lexical unit "bank" has at least two unrelated senses (homonyms (Hirst, 1987, page 5)): one for an institution of financial services, and a second for the ground beside a river. The distinction between senses is oftentimes more subtle. The lexical unit "mouth" has the following closely related senses (McLeod and Hanks, 1982, first three senses of "mouth"):

1. The opening through which animals take food and issue vocal sounds.
2. The system of organs surrounding this opening, including the lips, tongue, teeth, etc.
3. The visible parts of the lips on the face.

Each of the senses of a syntactic unit will be considered to constitute a distinct lexical unit in the lexicon.

1.2.2 Idioms

Idioms are an important part of language if for no other reason than their abundance. They take on a variety of structures and properties, a diversity no other part of language possesses. Unfortunately, idioms have been an understudied phenomenon and are a stumbling block to current linguistic theories.

In this section, an outline is given of the various properties of idioms, including those indicated by the results of cognitive studies. It is demonstrated why the Lexical Option Generator should consider idioms as a lexical unit semantically, though syntactically a language generator must treat idioms also as a sequence of individual words to a certain extent. Idioms will be differentiated from other categories of language with which they are sometimes confused — metaphors and indirect speech acts — since each requires very different semantic processing.

A Definition of Idioms?

There does not exist agreement on a single operational definition of idioms. Scholars' views range from including every morpheme in a language as an idiom (Hockett, 1958) to disregarding their existence (Chomsky, 1982). The single common element in all definitions is that the meaning of the idiom is not a compositional function of the meanings of its parts. This characterization also includes other categories of language such as metaphors and indirect speech acts. I will not attempt to toss yet another definition of idiom into the already varied grab bag of existing definitions. Instead, a representative list of the various constructions that have been considered to be idiomatic, from a list compiled by Strassler (1982), is presented:

- **Proverbs and Sayings**, such as "A bird in the hand is worth two in the bush" and "let the cat out of the bag".
- **Phrasal verbs**, such as "to give in" or "to look up".
- **Prepositional verbs**, such as "to look after" or "to object to".
- **Binomials**, such as "spic and span" or "bag and baggage".
- **Frozen similes**, such as "as bold as brass".
- **Ungrammatical but generally accepted phrases**, such as "who did you see".
- **Logical connective prepositional phrases**, such as "for instance" or "on the other hand".
- **Phrasal compounds**, such as "White House" or "red herring".
- **Incorporating verb idioms**, such as "to baby-sit" or "to sight-see".
- **Formula expressions**, such as "at first sight" or "how do you do".
- **Tournure idioms**, the receptacle for idioms which do not fall under the above classifications, such as "to kick the bucket". These idioms tend to be the most obscure in their meaning unless already memorized.

Scholars' definitions of idioms incorporate subsets of the above list, sometimes using different terminology.

As suggested by Boisset (1978), possibly the best way to define an idiom is by a *law cluster concept*, where an idiom is characterized by a cluster of properties. In the next subsection, an outline of some of these properties which contribute to the characterization of idioms is given.

One of the major research areas still open in the study of idioms *is* the definition itself. From the above list of various idiom constructions, it would appear that rather than having just one definition to encompass *all* idioms, we have subclasses of idioms, each with its corresponding definition. Nevertheless, it is not surprising that idioms have not been extensively dealt with, given that the concept itself has not yet been well defined.

Characteristics

One of the most distinctive characteristics of an idiom is that a person must have prior knowledge of the conventionalized meaning of the particular string of words in order to understand it. Generally, neither the entire meaning nor a partial one can be attributed to any *constituent* part of the idiom. Weinreich (1969) attempted to assign *subsenses* relating the idiomatic meaning of the string to constituents of the string. However, the assignments are not well motivated (Boisset, 1978). As well, paraphrases of idioms can be shorter, the same length, or longer than the idiom itself and so the manner of determining these assignments is arbitrary. A related problem: the parts of an idiom generally cannot be assigned referents (Boisset, 1978). For instance, "the bucket" in the idiom "kick the bucket" has no referent. This would seem to indicate that, to be able to use idioms, a generator must treat them as a whole, or as one lexical unit, rather than as a syntactic complex.

Idioms are *ambiguous*, since in many cases they can have either a *literal* or *non-literal* meaning. *Ill-formed* idioms have no literal counterpart, since they violate selectional restrictions or semantic features or have lexical constituents not present in a non-idiomatic environment (Boisset, 1978). People are not usually aware of this ill-formedness in the non-literal use of the idiom, for example in the use of "trip the light fantastic".

Idioms tend to lose their idiomatic reading under some syntactic transformations. For example, the sentence "Joe kicked the bucket" loses its idiomatic reading to most people when it is made passive: "The bucket was kicked by Joe". The qualifier *to most people* is important, since the idiomaticity of sentences is based primarily on intuition, and thus judgements can vary from person to person, including from linguist to linguist. *Frozenness* is a term used to refer to the unwillingness of an idiom to undergo transformations, and must be taken into consideration when proposing a lexical option, since a language generator must be informed of these transformational deficiencies.

Alan Healey[2] classified idioms into *twenty-one* different sentence functions, including nouns, noun phrases, adjectives, intensifiers, prepositions, adverbs of manner, time, frequency or place, and intransitive verbs. Such variation is staggering!

[2]As pointed out by Makkai (1972), whose information was extracted from (Healey, 1968).

Idioms, especially tournure idioms, cannot acceptably be used in all social situations, since they have a tendency to be informal. A natural language generator that takes pragmatics or stylistics into account must be aware of any such nuances in its use of these idioms.

Finally, idioms seem to appear more often in their idiomatic rather than their literal sense, though as pointed out by Fraser (1970), this has never been proven.

Reviewing these features, one can see that the characterization of idioms spans the syntactic, semantic, and pragmatic levels of language.

The Processing of Idioms

A model of idiom processing, whether it be cognitive, linguistic, or computational, generally is in accord with either the Idiom List Hypothesis or the Lexical Representation Hypothesis (using Swinney and Cutler's (1979) terminology).

The Idiom List Hypothesis maintains that specialized processing is required for idioms by way of an *idiom mode* as opposed to the "normal" *literal mode* of processing. The idiom mode facilitates storage in and access from an *idiom lexicon*, distinct from the "normal" non-idiom lexicon. A sentence or utterance will undergo a literal analysis first, and then if necessary undergo an idiomatic analysis. In essence, this hypothesis is a relative of Grice and Searle's model for the processing of figurative speech (Levinson, 1983).

The Lexical Representation Hypothesis does not distinguish an idiom lexicon or process. Instead, both the literal and idiomatic meanings are computed in concert when the initial word of an idiom is encountered. Within the lexicon are the individual words of the idiom as well as a single lexical item representing the entire idiom. Structural analysis of the individual words operates in tandem with the processing of the idiom as a whole.

The major difference between the two hypotheses is that the former expects the processing of idioms to be a special function separate from literal processing, while the latter encourages a uniform handling of both idioms and literal statements. This thesis takes the latter view, claiming that both words and idioms can be used in certain situations, possibly interchangeably, and that the *process* for determining if such a situation holds is the same.

Cognitive experiments indicate some interesting characteristics of idiom processing. In general, they tend to demonstrate that the treatment of idioms and the rest of language should be uniform.

Bobrow and Bell (1973) attempted to give support for the Idiom List Hypothesis by giving empirical evidence of the literal/idiom mode division. They concluded that an idiom bias predisposed the subjects into the idiom mode of processing. The methodology of the experiment, has, however, come under question (Swinney and Cutler, 1979).

A seemingly indisputable result is that idioms are processed *faster* than unambiguous literal strings and that the literal interpretations of idioms take at least as long, if not longer (Ortony et al., 1978), than unambiguous literal strings of equal length and frequency (Gibbs and Gonzales, 1985; Rakowsky, 1984; Ortony et al., 1978; Swinney and Cutler, 1979): strong evidence for the Lexical Representation Hypothesis. If the single idiomatic lexical item is accessed and determined to be appropriate, the idiomatic meaning can then be determined more quickly than the literal reading since the additional lexical accesses and syntactic analysis for each individual lexical item are not required. The *literal* reading of an idiom may take longer since the idiomatic reading must first be processed, then rejected, and the literal analysis resumed (Rakowsky, 1984).

Gibbs and Gonzales (1985) attempted to empirically demonstrate that the syntactic frozenness of idioms affected their processing. The subjects of the experiment were faster at responding to frozen idioms than to the more flexible, less frozen, idioms. Their claim is that this observation can be explained by the fact that each form of a more frozen idiom is more familiar than the varied individual forms of the less frozen idiom. Alternatively, this phenomenon could be explained by there being a separate entry in the lexicon for each syntactic form of an idiom and thus more processing required for the resolution of ambiguity.

In conclusion, the cognitive studies on idiom processing indicate that the Lexical Representation Hypothesis is likely the most cognitively accurate model.

Idioms as a Lexical Unit

Should an idiom be treated as a unit, a single lexical item? There is convincing evidence for both a unit and non-unit treatment of idioms.

In favour of a unit treatment of idioms, the following can be said:

1. Clearly, the meaning of an idiom is not a compositional function of the meanings of its elements. In fact, the relation of the idiomatic meaning to the literal meaning, if a literal meaning does exist, is in most cases arbitrary.

2. Sometimes the elements of an idiom cannot appear outside an idiomatic environment. For example, in the idioms "kith and kin" and "spic and span", the words *kith* and *spic* do not appear outside of these idioms in text.

3. As an extension to (2), *entire* idioms sometimes do not have a corresponding literal interpretation, as with "trip the light fantastic".

4. Idioms have transformational deficiencies in that they lose their idiomatic reading when undergoing some transformations, as with the passive form of "to kick the bucket".

5. Cognitive studies have indicated that idioms are processed faster than unambiguous literal phrases of equal length and lexical frequency as well as faster than their own literal interpretations.

However, equally convincing arguments can be presented *against* a unit treatment of idioms:

1. Idioms comply with the same phonological rules as their literal counterparts (Boisset, 1978).

2. The individual words of an idiom observe the same morphological rules as do their literal counterparts. For instance, if the idiom "kick the bucket" was to be treated

as a unit, one would expect the inflection for the past tense of the idiom to be "kick the bucket*ed*" (or some similar inflectional morpheme) rather then "kick*ed* the bucket" (Newmeyer, 1974).

3. The very fact that, for many idioms, transformations *are* possible is argument for a non-unit treatment (Boisset, 1978).

4. Variable slots appear in many idioms, such as in "to break x's heart", where x could take on an infinite number of values. The possible bindings for these slots, however, are not arbitrary as evidenced by the examples: "Joe broke Jane's heart" and "Joe lost his temper" (Schenk, 1986). Joe could not break his own heart, nor could Joe lose Jane's temper.

5. The inability of an idiom to undergo certain transformations can sometimes be overridden by using a different stress pattern in speech (Boisset (1978) gives some French examples of this phenomenon). Also, for two-word idioms such as "White House", the stress pattern can distinguish the idiomatic and literal interpretations (Strässler, 1982). Thus, idioms are sensitive to the stress patterns of their individual words.

As indicated by Boisset (1978), the arguments for each side of the controversy do not all concern the same level of language. Evidence in favour of the unit treatment tends to be semantic, while that against the unit treatment tends to be syntactic. Idiom usage arguments fall on both sides of the controversy. In this thesis, which deals with words and idioms at a semantic level (i.e., the situations in which they may be used), treating idioms as a single lexical entity is advisable, though additional syntactic information must be provided with each lexical option so that a language generator can properly deal with the idioms syntactically.

Linguistic Theories of Idioms

Idioms come in many syntactic shapes, as is emphasized in Tables 1.1 and 1.2[3]. However, the syntactic forms usually have transformational deficiencies which are unique to each idiom. Unfortunately, no linguistic theory to date has been able to account for the multifarious characteristics of idioms.

Fraser (1970) represented the meaning of an idiom in a deep-structure representation, and attempted to account for the transformational deficiencies of idioms within the transformational grammar framework. As with previous transformational grammar models attempting to incorporate idioms (Katz and Postal, 1963; Weinreich, 1969), idioms were divided into two types:

1. *Lexical idioms* dominated by one of the lowest syntactic categories such as noun or verb, and

2. *Phrase idioms* which include all idioms not covered by item 1.

[3]Tables 1.1 and 1.2 were compiled from information in the introductory sections of (Cowie and Mackin, 1975) and (Cowie and Mackin, 1983), respectively.

Pattern	**Intransitive**	**Transitive**
v + particle	The vase *fell over* with a crash.	John packed the bag and *zipped* it *up*.
v + prep	John *fell into* many bad habits.	John *foisted* his problems *on* me.
v + particle + prep	John *fell in with* a bad crowd.	I *put* John's bad mood *down to* the weather.

Table 1.1: Verb plus particle or preposition idioms

A significant contribution of Fraser's was the proposal of a *frozenness hierarchy* of seven levels of allowed transformations:

L6: *Unrestricted* kinds of transformations allowed.

L5: *Reconstitution* of the idiom into another constituent structure is allowed, such as action nominalization transformations (e.g., "he laid down the law to his daughter" → "his laying down of the law to his daughter").

L4: *Extraction* of a constituent to outside the idiom is allowed, such as in particle movement formation (e.g., "look up the information" → "look the information up").

L3: *Permutation* of two consecutive constituents is allowed, such as in yes-no question formation (e.g., "lay down the law" → "lay the law down").

L2: *Insertion* of some constituent into the idioms is allowed, such as in indirect object movement (e.g., "John read the riot act to the class" → "John read the class the riot act").

L1: *Adjunction* of a non-idiomatic constituent is allowed, such as in gerundive nominalization (e.g., "John hit the ball" → "John's hitting the ball").

L0: *Completely Frozen*: no transformations allowed.

Fraser proposed that when an idiom belongs to one level, it also belongs to all the other levels below it. His approach had the advantage that the only information that must be associated with an idiom to specify its transformational deficiencies is its level. Unfortunately, the hierarchy does not hold for everyone's dialect within a particular language[4], and as Strässler (1982) pointed out, Burger (1973) proved that the transformation hierarchy is not applicable to German.

Thus, unfortunately, the marking of transformational deficiencies for idioms is not as straightforward as one would hope: idioms have not been able to be simply categorized to indicate their possible transformations.

It is clear that *internal structure*, such as is shown in Table 1.2, should be demonstrated in the lexical entry for an idiom. Newmeyer (1974), a generative semanticist, presented a non-unit treatment of idioms by means of an *idiom inventory* made up of

[4]Fraser did indicate that he was dealing only with his own idiolect.

Pattern	**Example**
Sentence	*A bird in the hand is worth two in the bush.*
Clauses	
v + comp	John has *gone berserk.*
v + obj	John *spilled the beans* about the party.
v + obj + comp	John and Mary *painted the town red.*
v + (iobj) + obj	John *blew* Mary *a kiss.*
v + (obj) + comp	John *drove* Mary *mad.*
v + obj + adjunct	John never *plays it straight.*
v + (obj) + adjunct	John *took* the news *hard.*
Possessive Clauses using "get", "give", or "have"	
v + obj	The groom was *getting cold feet.*
v + (iobj) + obj	The wait was *giving* the groom *cold feet.*
Noun Phrases functioning as:	
obj	John and Mary found *common ground* to talk about.
comp	The facts are *common knowledge.*
obj of prep	John lectured Mary on *the errors of her ways.*
Adjective Phrase functioning as:	
comp	The sales make shopping *easy on the pocket.*
adjunct	John travels *as the spirit takes him.*
Prepositional Phrases functioning as:	
comp	John is a man *of good standing.*
adjunct	John acted *at his own discretion.*
disjunct	*On the other hand*, John is right.
conjunct	*By the same token*, John should have apologized.
Adverbial Phrases	John ran *as fast as his legs could carry him.*
noun + noun	John worked *night and day.*
adj + adj	John got up *bright and early.*
adv + adv	John got up *slowly but surely.*
v + v	John was willing to *bow and scrape* for a promotion.
det + det	*Each and every* girl loves John.

Table 1.2: Complex idioms

idiom sources. An idiom source consisted of an ordered pair of semantic representations, (M1, M2), where M1 was the meaning of an idiom, and M2 was the semantic representation of the idiom's literal counterpart. Within the lexicon, if a word appeared in any idiom, it was marked with the designation(s) of the idiom source(s) in which it could appear.

Newmeyer claimed that idiom transformations behaved in such a way that the rules of both M1 *and* M2 restricted the possible transformations of the idiom. Thus, as with Chafe (1968), he considered the allowed transformations to be dependent on the meaning of the idiom, a notion which appears to be intuitively correct as a partial solution.

However, Boisset (1978) gives some examples in French where the above approach fails, such as where some idioms can be clefted, given a certain stress pattern on the words. Also, as Boisset points out, these types of rules depend on the chosen paraphrase of an idiom, which can vary. Newmeyer also did not account for ill-formed idioms and phrasal compounds (Strässler, 1982).

Thus, unfortunately, it does not seem that the transformational deficiencies of idioms can be systematically determined, either by simple categorization as with Fraser's frozenness hierarchy, or as a function of the meaning of the idiom (and its literal counterpart), as with Newmeyer's theory.

Idioms versus Metaphor

Metaphors and idioms share some principal characteristics:

- Their meaning is not a compositional function of the meanings of their parts.
- They are not distinguishable by syntactic form (Loewenberg, 1975).
- It is the task of the hearer or reader to determine if the literal or non-literal interpretation of the string of words should be taken.

Thus, the two linguistic categories are closely related, causing them to often be confused with each other. However, there are important distinctions between the two categories, affecting their semantic processing.

Metaphor is "a figure of speech in which a word or phrase is applied to an object or action that it does not literally denote in order to imply a resemblance" (McLeod and Hanks, 1982, page 709). The weakening or elimination of the semantic features of the words of a metaphor, while still taking into account the context and analysis with respect to the semantic features, allows *novel* metaphors to be understood (Boisset, 1978).

Idioms, on the other hand, are characterized by the arbitrariness of their idiomatic meaning. There is not merely a suppression of the semantic features of words but a total replacement (if semantic features can be assigned at all to the individual words of an idiom). A hearer or reader must in general have prior knowledge of the conventional meaning of an idiom to understand it. However, as pointed out by Weinreich (1969) there are some tendencies to have groups of idioms with common subparts such as "bury the hatchet" and "bury the tomahawk", as well as antonymous relations such as "bury the hatchet" and "dig up the hatchet"[5].

[5]The phrases "bury the tomahawk" and "dig up the hatchet" are not within all people's idiolects.

Intuitively, many idioms are *dead* metaphors, having lost their original metaphoric tie, and thus only the conventional meaning of the string of words is recognized. It is an amusing pastime to guess the origins of such idioms. This brings up an important point: one person's metaphor may be another person's idiom, depending on their knowledge. The fuzzy line between metaphor and idiom becomes even more indistinct when we consider that this line varies from person to person, and even for a single person over time. The distinction rests on the underlying psychological processes involved in each utterance or hearing of an idiom or metaphor. An idiom's meaning is *conventionalized*, whereas the meaning of a metaphor is *realized* through its use (Sadock, 1972).

Idioms versus Indirect Speech Acts

Indirect speech acts are another category of speech that could be confused with idioms. An indirect speech act is a sentence which "does not have the 'literal force' ... associated by rule with their sentence types, but rather some other force ..." (Levinson, 1983, page 357). For example, the question "Can you reach the book?" is effectively a *request* rather than a *question*.

Sadock (1972) held that an indirect speech act (ISA) is an idiom, since both have meanings that are not a functional composition of the meanings of their parts, and both usually have literal counterparts. Another similarity is that their additional meanings come about through conventionalization. Nevertheless, a distinction should be made between the two.

ISA's generally carry *both* their literal and non-literal meanings. For example, "Can you shut the door?" conveys both the literal question meaning, and the indirect request meaning since responses to both the question or the request would be acceptable: "No, I can't get up now" or "OK, right away!" (Levinson, 1983; Strässler, 1982). Idiomatic usage, in contrast, normally excludes the literal meaning, except in the case of humour where the dual meaning of the idiom is the conscious source of the humour.

ISA's convey different non-literal meanings, depending on the circumstances under which they are uttered. The ISA "It is cold in here" can have the non-literal meaning of a request to shut the door, or "Start the cold-air experiment since it is now cold enough" and so on (Strässler, 1982). Each time an ISA is used, a *new* non-literal meaning for that ISA may be realized. This means that a licensing situation could be difficult to define for an ISA. Idioms usually have only one non-literal meaning, which does not change with respect to the situation within which they are uttered.

Finally, ISAs are generally considered to serve as polite speech, whereas the findings of Strässler (1982) indicate that the usage of idioms subtly establishes a social structuring in which the user of an idiom is either of equal or greater status than the other participants in the dialogue.

Thus idioms and indirect speech acts are different categories of language.

Summary

As outlined in this section, the Lexical Option Generator treats idioms as a single lexical unit but provides additional syntactic information (e.g. possible transformations and inflections) to a language generator. Unfortunately, no linguistic theory to date has accounted clearly for this additional syntactic information. This problem is further aggravated by the fact that each person may have their own views on what is idiomatic and what is not. Thus, this thesis will treat each idiom on an individual basis, specifying the syntactic information for the idioms. However, it must be noted that idiomaticity varies from person to person.

1.2.3 Situations

The basis of this thesis is that each state of affairs, or *situation*, licenses the use of a certain lexical unit (or lexical units). There is no direct correspondence between the structural complexity of this lexical unit (e.g., the number of words in the entity) and the complexity of the compatible situation:

- An individual word can correspond to:
  - a simple situation composed of one concept (e.g., the word "red" represents the relatively simple concept *red*).
  - a complex situation (e.g., the word "octogenarian" represents a *person who is between 80 and 90 years old*).
- Analogously, an idiom can correspond to:
  - a simple situation (e.g., the idiom "for good" represents the concept *forever*).
  - a complex situation (e.g. the idiom "a gentleman's agreement" represents *an agreement between two people that is not legally binding but based on trust*).

Note, however, that calling something a *simple* concept is purely relative to the primitives of the knowledge representation used to describe a situation. I make no claims as to what the primitives *should* be, but simply that there should be primitives. Having a knowledge representation scheme whose primitives mirrored exactly each and every situation (which would essentially correspond to the magnitude of the number of lexical units times the number of senses) would require an unnecessary and unmanageable proliferation of primitives. Also, as new lexical units are coined, which is very common, especially in the case of idioms, new primitives would always have to be added to the knowledge representation. A dynamic knowledge representation would result. Different languages would require different knowledge representations, since not all languages have a word or idiom to represent the same situation. Nevertheless, there should be enough primitives to be able to represent each of the concepts and situations, but what constitutes *enough* is an unresolved research issue.

Thus, situations should be used as a means to *organize* knowledge of concepts in a lexicon, but not as the knowledge representation itself. Unfortunately, within the current state of research there is no standard or complete set of primitive concepts to work with, but for the purposes of this thesis it need only be noted that the situations are *composed* of the less complex entities in the knowledge representation.

A *hierarchical lexicon* results whose nodes define the *situation*, or state of affairs, that must be present for a lexical unit to be used. Lexical units used for exactly the same situation will be on the same node, while near synonyms reside close-by. Having a lexicon for both words and idioms whose entries are situations (groupings of concepts) arranged into a hierarchical net has some obvious advantages:

- An intuitively pleasing semantic organization is placed on the lexicon. Synonyms and near-synonyms for lexical units are placed closely in the representation and are easy to determine.
- Hyponyms and superordinates are equally easy to determine.
- New lexical entities can be added easily.

The following are examples of the types of options the Lexical Option Generator should provide.

Complex superordinates should be provided by the Lexical Option Generator. For example, a situation of ice cream topped with chocolate sauce, nuts, and whipped cream would have the lexical option "chocolate sundae" covering the entire situation, in addition to the lexical options covering the individual ingredients (e.g., "ice cream", "chocolate sauce", etc.).

The Lexical Option Generator should supply synonyms for a single situation. For example "to kiss", "to press one's lips to", and "to give a big wet one", are all synonyms. In general, language *blocks* absolute synonyms (Cruse, 1986; Di Sciullo and Williams, 1987). Even though a set of lexical options may be interchangeable in one situation, in a different situation only a subset or overlapping set of lexical options may be able to be used. For instance, in a situation where there is a gathering of geese, the lexical options would include "group", "gaggle", and "flock". However, for a similar situation where there is a gathering of sheep, the lexical options would not include "gaggle".

Related to synonyms are different views of the same situation. For example, it is assumed that the knowledge representation for *buying* and *selling*[6] would be the same except the mapping onto the case frames for "to buy" and "to sell" would be different. For example, if there is a commercial transaction of a book from John to Mary with a tender transfer of $50, the options include "John *sold* Mary a book for $50" and "Mary *bought* a book from John for $50". In such examples, it would depend on the knowledge representation of the language generator whether Charniak's (1981) case-slot identity theory would be upheld. The case-slot identity theory states that there is a one-to-one mapping between the slots of a frame representation and the cases of a verb, where the verbs map onto the frame description itself. I do not adhere to the case-slot identity theory, as demonstrated by the following two examples where a verb plus its arguments may be determined by more than the direct mapping suggested above. Nevertheless, an attempt is made to prevent completely arbitrary mappings between slots and cases.

Consider the situation where John hits Mary. The system should be able to provide the lexical units "strike", "hit", etc. But suppose that in addition, the knowledge is input that Mary hit John back, and so forth. Then a new situation arises where each of the hits can be described (using the lexical units provided in the above situation), but one can also say that John and Mary "fought", an abstraction of the hitting actions. The Lexical Option Generator should be able to do this kind of abstraction.

[6]It is assumed that the Lexical Option Generator uses a frame representation in the knowledge base.

Suppose that John is eating a sandwich. The lexical unit "eat" should be provided. If John were eating this sandwich quickly, along with "eat" and "quickly", "gobble" should be included as a lexical option. Thus, a verb can be partially determined by the slots of a frame, in addition to the generic frame description.

1.3 Summary

In this chapter, the need for lexical options for the production of "natural" language has been emphasized. An in-depth characterization of idioms was given to demonstrate that the notions of word and idiom should be collapsed into one notion, the *lexical unit*, for the Lexical Option Generator. The relationship between *situations* and lexical units was then outlined, with examples. Situations are the state of affairs that must be present for a lexical unit to be used. A hierarchical lexicon organization is proposed wherein each entry in the lexicon is a situation.

Chapter 2

Matching in a Knowledge Base

The input to LOG is a frame representation of the information a natural language generator wishes to convey. The key process used by the Lexical Option Generator is *matching* this input knowledge to the situations represented in the lexicon. A situation, expressed in the natural language generator's frame language, defines one sense of a particular lexical unit. The lexicon thus provides a rich set of lexical options by differentiating the limited frame vocabulary into situations associated with lexical units. A combination of full and partial matching is used in this problem in that a *full* match of a situation is performed on *parts* (and possibly all) of the input knowledge.

This chapter outlines full and partial matching and the roles they can play in Artificial Intelligence systems. A survey of a representative set of matchers is given to highlight the variety of approaches to the problem, as well as the variety of applications, and to reveal desirable characteristics of the matcher for the Lexical Option Generator.

2.1 Partial and Full Matching

Partial matching is the process by which the similarities of at least two descriptions are identified. The knowledge representation used to express these descriptions can take many forms. However, irrespective of the representation, a partial match will consist of an *abstraction*, which contains the common properties or elements of the two descriptions, and two *residual* terms each of which are the unique elements associated with one of the descriptions (Hayes-Roth, 1978; Hayes-Roth, 1979). For example, consider a very simple knowledge representation where a description is a set of letters. If description $A = \{a, b, c, d\}$ and description $B = \{b, d, f\}$, then the abstraction of A and B, $A * B$, is $\{b, d\}$, the residual of A, $A - A * B$, is $\{a, c\}$, and the residual of B, $B - A * B$, is $\{f\}$. Humans do partial matching in daily life without apparent effort, such as when recognizing people from different angles or after many years, or when recognizing parallels in pieces of literature. However, as with many problems of Artificial Intelligence, the ease with which humans do this is not reflected in a known general partial matching algorithm: this results in a need to restrict the size and complexity, and thus increase the specialization, of the descriptions analyzed.

When one description fully matches another, it means that they are in total correspondence. A description may be a *pattern* description which consists of constants and possibly some variables. These variables can become bound, or be given a value, by placing them in correspondence with the elements of the *target* description. The criteria for a full match include (Kline, 1981):

1. Each description must have the *same* structure. That is to say, each of the two descriptions must have isomorphic structures with a one-to-one correspondence between the properties and relationships between the properties.

2. *All* of the pattern variables must become bound during the pattern match. A variable may bind to a substructure.

3. *All* constants must correspond between the two descriptions.

The keywords *all* and *same* in these criteria indicate the inflexibility of a full pattern matcher. The pattern and the target are compared to determine if there exists a set of substitutions, the variable bindings, that could make the pattern be identical to the target. The pattern and the target are then said to *unify*. For example, consider a pattern `A = (love (agent ?x) (patient ?y))`, where `?x` and `?y` are variables, with targets `B = (love (agent mary)(patient john))` and `C = (love (agent mary) (time 3:00p.m.))`. Pattern A and the target B unify with `?x = mary` and `?y = john`. However, no bindings can be found for which A and C unify.

A partial match will attempt to fulfill the above criteria, but they are *not* absolutely required for a partial match. The residuals of the partial match will contain those elements of the descriptions that do not comply with the full match criteria, while the abstraction consists of those elements of the description which do comply.

This thesis will investigate matching *parts*, but possibly all, of the input knowledge with *full* templates for situations. Thus, a combination of full and partial matching will be performed. The residual input knowledge of any part of the situation matched is assumed to be matched by other situations, such that all of the input knowledge can be covered by matched situations. It will be the job of the natural language generator to choose an appropriate cover of the information to be conveyed.

2.2 The Best Match

At this point in the discussion of what partial matching is, one may get the impression that comparing one description to another is very straightforward. On the contrary, there still remains ambiguity in that there are many ways of perceiving two descriptions as similar, depending on one's priorities or points of view. What makes one description *more* similar to a given description than another? This becomes the problem of the *best match*.

A common criterion for the best match is that the greater the number of elements common to two descriptions, and thus the larger the abstraction, the better the match (Kline, 1981; Joshi, 1978). In the Lexical Option Generator, this would correspond to situations that cover the largest portion of the input knowledge and thus are associated with the most specific lexical unit. When this does not provide a unique solution, other criteria can be established to overcome the impasse of selecting the best match.

In general, these criteria are dependent upon the kind of best match desired, which in turn is dependent upon the application. For example, depending on the goals of a natural language generator provided with the lexical options, a less abstract option may be desired, or other specific stylistic constraints may need to be fulfilled. Other criteria may include minimizing the size of the residuals, or considering the *importance* or *desirability* of parts of descriptions depending upon the application. In many cases an arbitrary choice can be made.

This thesis will not be concerned with finding a best match, since all of the lexical units provided will be correct and usable by the process to which they are given. However, it will be up to that process to select the optimal lexical units to produce language according to its current goals and constraints.

2.3 The Roles of Matching

One role of matching is to abstract commonalities and identify differences between descriptions (Hayes-Roth, 1979). Commonalities can be viewed as generalizations of two or more descriptions, whereas the differences indicate the special cases of a generalization. This role is useful in such areas as concept learning, analogical reasoning, inductive inference, and predicate and pattern discovery (Hayes-Roth, 1979; Falkenhainer et al., 1986; Bobrow and Winograd, 1977).

A second role of matching is in the interpretation of data (Hayes-Roth, 1978; Hayes-Roth, 1979). If each element of a description can be interpreted in more than one way, the "best" interpretation must be found. The data can be partially matched to a set of frames or templates to find the best match. For example, the resolution of lexical and syntactic ambiguities for semantic interpretation requires matching senses and categories of words to syntactic and semantic information gathered from a sentence (Hirst, 1987).

Another role of matching is in pattern recognition and classification by constraint satisfaction (Hayes-Roth, 1978; Hayes-Roth, 1979; Rau, 1987; Bobrow and Winograd, 1977). The majority of complex problems which need to be solved in such areas as Artificial Intelligence and knowledge acquisition must be solved without a *complete* set of data. Rather than total failure in these incomplete cases, the data can be compared to the *ideal* or *prototypical* description to arrive at a best or most plausible course of action for the time. The commonalities of the two descriptions can be viewed as the constraints, while the residuals are unsatisfied constraints. The current data can then be recognized as an instance of some prototypical instance (Hayes-Roth, 1978; Bobrow and Winograd, 1977).

2.4 A Survey of Matching

Concept matching is usually seen as a by-product of a system rather than as a focus of the system itself. However, as evidenced by the variety of applications for matching outlined in the following section, matching is a key theoretical issue of Computational Linguistics and other Artificial Intelligence programming.

This section will briefly outline the knowledge representation and matching strategies of a representative set of matchers. The final section of this chapter will summarize those aspects of the matchers reviewed which are of interest to this thesis.

2.4.1 Multiply-Specified Matching

Multiply-specified matching is where more than one matching strategy is used by the matcher and may be specified in more than one data structure in the system. This section outlines two such systems.

KRL

KRL (Bobrow and Winograd (1977) and (1979), and (Lehnert and Wilks, 1979)) was meant to be a general knowledge representation tool to build systems and theories of language understanding. Knowledge (conceptual objects) was organized hierarchically and associated with descriptions and procedures. These conceptual objects could be described partially or completely, as well as from different viewpoints. The data structures of KRL are best summarized by Bobrow and Winograd:

> The data structures of KRL are built of *descriptions*, clustered together into structures called *units*, that serve as unique mental referents for entities and categories. Each unit has a unique *name*, is assigned to a *category type* ...and has one or more named *slots* containing descriptions of entities associated with the conceptual entity referred to by the unit as a whole. Slots are used among other things to describe those substructures of a unit that are significant for comparison. (Bobrow and Winograd, 1977, page 267)

Units were used to collect a set of descriptions into one structure that could then be related to a set of procedures. The category types of units were as follows:

- **Basic categories** created non-overlapping partitions of the knowledge. An individual could not belong to more than one basic category, and therefore a simple match strategy was facilitated for finding "easy" conflicts, when two individuals were from different basic categories. For example, *collie* may have belonged to the basic category *dog*, and *E. Coli* to the basic category *bacterium.* Thus, a collie and an E. Coli could immediately have been determined not to match, since they had different basic categories.

- **Specialization categories** represented descendents of a basic category, and thus were further distinguished in their procedural attachment and described properties. For example, *collie* could be a specialization category of *dog.*

- **Abstract categories** provided a means of collecting descriptions (with their procedures) to be inherited by other entities for which the abstraction was a prototype. An example of an abstract category was *action* or *living-thing.*

- **Individual categories** represent unique entities, thus providing another simple match strategy, since two different individuals could not match. For example, a particular person, *John Doe*, would be an individual category.

- **Manifestation categories** provided a means of attaching additional descriptions to an entity. For instance, information specific to one context, time, or role could be attached to the entity.
- **Relation and Proposition categories** were used for representing a predicate and the instantiation of a predicate, respectively. For example, there could be a relation *greater-than* with propositions of greater-than for each instantiation of the relation.

The KRL matcher took a description as a pattern and another description as a target. For each pattern, a set of strategies was specified for finding potential targets. These strategies could be defined by the user. For example, the algorithm for aligning the pattern to correspond to the sequence of descriptions in the target made decisions about which descriptor-matching subtasks would be attempted, and in what order. Thus, procedures could be attached to an entity and invoked depending on the actions of those entities. Matching could be facilitated by partitioning (scoping) of the knowledge base: a family of procedures could be associated with each scope, as well as a scope limiting the space of search. User-specified parts of an entity could act as index *keys*, further facilitating matching strategies. Primitives were provided for retrieving units under a specific key pattern. KRL itself provided only basic matching algorithms and the rest had to be defined by the user, possibly using previously specified matchers as building blocks, specifically for the application in mind. This accorded with their goal of providing *process frameworks* for basic operations:

> Rather than having a semantically complete definition of what happens in a match, the system provides a matching framework which contains processes for setting up the structures to compare two descriptions, doing the alignment of comparable descriptors, looking for procedures attached to specific patterns, and handling all of those cases in which a simple syntactic match will work. (Bobrow and Winograd, 1977, page 280)

The user could also indicate, in a call to the matcher, a *directory* that specified additional matching strategies. The ability to compile the system aided in efficiency.

Thus, the user had control over the matching algorithms to be used; KRL itself did not provide many mechanisms for matching. The expertise of the builder of the knowledge base and of the programmer was assumed to yield the best strategies, rather than having much of the strategy involved implicit in the system itself. With this much control given to the builder of a KRL application, and in addition, having these various procedures (attached to various entities in the system) being invoked willy-nilly, seems conducive to a resultant ad hoc and complicated system with potentially unpredictable results. Directions to guide the matcher should be within the knowledge representation itself, as *part* of the representation, rather than *attached* to it.

Finin's Semantic Interpretation for Nominal Compounds

An approach very similar to KRL was taken by Finin (1980) in his system for the semantic interpretation of nominal compounds, built around a frame-based representation system of concepts and relationships between them. An abstraction hierarchy existed, supporting the inheritance of attributes. The central portion of this system was its concept matcher.

As with KRL, a selection of matching strategies could be suggested at varying times during a matching task. These suggestions could be specified in the arguments to the matching function itself, in a list of default strategies encoded in the system, and within the pattern or target concept itself.

Finin specified a set of possible strategies to choose from for the matching of two concepts:

- **Recurse**, the most powerful matching method, was a *general recursive* matcher. There were two steps to the method:

    1. *Slot alignment*, found a corresponding slot in the target for each slot in the pattern, thus aligning the slots. If there was no corresponding slot, the match failed.
    2. *Match slots* then matched the corresponding values of the aligned slots using methods from a set of matching strategies. Furthermore, these slot matching methods could be attached to particular slots or to the frame which described a slot, or specified in a default list of methods.

- **Same-frame** was a very simple test of whether the pattern and target were identical. It was assumed that this was a relatively common phenomenon.

- **Recall-result** attempted to recall a previous result of matching the two concepts. Previous results were kept in the short-term memory of each of the two concepts. Thus all that was required to see if two concepts matched was a check of the special fields in the two concepts indicating whether they had already been matched. A background daemon cleaned these fields up as a global buffer became full.

- **Assume-goal** simply assumed two concepts matched, to prevent infinite loops in recursive structures. When an attempt was made to match two concepts that were already in the process of being matched, the new match attempt was assumed to match.

- **A.k.o.** (a kind of) checked if the target was a subconcept or instantiation of the pattern.

- **Basic** was a basic quick check. As with KRL, the knowledge was partitioned, and with each entity the basic categories to which the entity belonged were stored. Thus, obvious mismatches between two entities could be detected by checking if categories were not the same or were not sub-categories of each other.

Finin's matching strategies were geared to quickly find non-matches and to choose the most likely procedures to lead to results in a timely fashion. This led to a knowledge base built by trial and error. Finin's system had the same pitfalls as KRL: it was dependent upon the builder of the knowledge base *and* the application.

2.4.2 KL-ONE Classification

KL-ONE (Lipkis, 1981; Brachman, 1985; Schmolze and Lipkis, 1983) is a general knowledge representation system where the fundamental units of information, *concepts* denoting a set of objects, are arranged hierarchically. The KL-ONE classifier provides a means of inserting a new concept, A, into a KL-ONE network by searching the network to find the most specific concepts that subsume A, and the most general concepts that A subsumes. The new concept A is then inserted into the network by establishing explicitly the subsumption relationships in the network.

Concepts are defined solely by their structure specified in the network, and thus have no attached, non-syntactic, meaning. Primitive concepts are those that cannot be fully defined by the structure. The epistemological primitives used to structure the concepts into a network are the following:

- **Specialization** links indicate a subsumption relationship between two concepts (i.e., whether an instance of the first concept is always an instance of the latter). For example, *person* specializes *mammal.*
- **Rolesets** describe potential (attribute) relationships between instances of a concept and other concepts. For example, roles exist from a concept to other concepts for its properties, parts, etc. Rolesets have *value restriction* (v/r) arcs which specify the concept that describes the potential filler of a role (i.e., of the property, part, etc.). If more than one v/r is present, the restrictions are taken conjunctively. Rolesets also have *number restrictions*, a lower and upper bound, on the cardinality of the fillers of a role. The roles of a concept are *inherited* by all of the subsumed subconcepts, unless modified by a roleset restriction or differentiation (see the two items below).
- **Roleset restrictions** between rolesets denote that a subset of the subsuming v/r concept is the v/r of the subsumed v/r concept. For example, if the v/r of the *sender* role of the concept *message* is *person*, the concept *important-message* may be roleset restricted to have the v/r of the role *sender* be *manager*, a subconcept of *person.*
- **Roleset differentiation** differentiates the role relation of a subsuming concept into a subrelation. For example, the *executive-officers* role of a *company* concept may be differentiated into *president*, *vice-president*, etc.
- **Structural descriptions** are used to specify interdependencies between the roles of a concept. For example, a *transaction* could be described as two *transfers*, where the *giver* of the first transaction is the *receiver* of the second, and vice versa. The use of structural descriptions is very complex, and "messy" (by the designer's own admission (Brachman, 1985)), and will not be expounded on here.

Both roleset restrictions and differentiations can themselves be restricted and/or differentiated (and are inherited until modified). Note that there is no cancellation of any of the above links, only further specification.

KL-ONE asserts the existence of an individual by connecting it to a *nexus* (via a *description wire*), with a particular context. A *nexus* is an entity serving as the point of coreference for the various descriptions specifying the *same* object in the context.

A new concept, A, is classified by first classifying the concepts that constitute its roles, and then the head of concept A is classified (i.e., in general, concepts are classified bottom-up). If a cycle exists in the description, each of the concepts in the cycle are classified in parallel, by assuming that the necessary subsumption relationships exist for each concept in the cycle, and then checking that this is the case (as for the Assume-goal method of Finin, section 2.4.1). Thus, only one concept, whose subparts have already been classified by the bottom-up nature of the classifier, need be taken into consideration at any given time. The most specific subsuming concepts and the most general subsumed concepts are found in separate independent searches.

The most specific subsuming nodes are found as follows:

> The search for subsumers starts at the top of the lattice and performs a depth-first traversal, testing each node to see whether it subsumes the concept. If it does not, none of its children could either, so the subtree below it need not be searched. If it does, it is remembered as a subsumer and its children are searched. If none of its children subsume the concept, then it is the the most specific subsumer (in this subtree) and a superconcept cable [specialization] is established between the concept and its newly found subsumer. (Lipkis, 1981, pages 134–135)

Additional constraints can be used to limit the search for the most general nodes subsumed by A. The transitivity of the subsumption relation means that only the sub-concepts of the superconcept found in each stage of the above search need to be considered. Also, since the subsumed nodes inherit the roles of the subsumer, only the subtrees in the network consisting of all of the roles need to be searched. The concept A is established in the network as a superconcept of the highest concept in each subtree that it subsumes, if one exists in that subtree (Lipkis, 1981).

The KL-ONE classifier is central to the KL-ONE knowledge representation system. In addition to automatically inserting new concepts into the network (guaranteeing the consistency of the network semantics), the classifier can be used for generalized search. A search pattern can be encoded into a concept, P, and classified, thus discovering the concepts that P subsumes. A target concept can be described as a concept, T, and classified as well. If P subsumes T, then there is an indication of a match between the two concepts, and thus of the original pattern and target (Brachman, 1985). However, the information packaged into T must be predetermined.

The use of the transitivity of the subsumption in the classification algorithm to constrain search is an appealing feature. It is unfortunate that the specification of structural descriptions in KL-ONE is complicated, as interrelationships between the participants in a situation are an important factor in specifying situations in the Lexical Option Generator.

2.4.3 Numerical Determination of Matches

Dan Fass (1986a; 1986b; 1988) computes *semantic vectors* as scores for the matches between pairs of word-senses to determine metonymic relations and word sense ambiguities in sentences.

The knowledge is organized into a lattice of *sense frames*, in which each sense frame describes one semantic sense of a word. These sense frames can themselves point to other sense frames, thus creating a lattice. The lattice is hierarchical, with various arc labels describing the connection between the word senses, usually some kind of membership relation. Some examples of arc types are: supertype (also known as IS-A), superscale (membership of a scale, e.g., *heat* belongs to the scale of *temperature*), and superpart (also known as IS-PART). The nodes themselves usually contain *preferences* (restrictions on the context in which a word sense can occur), and *assertions* (what is to be asserted on the word sense to which it is applied). Noun senses contain only structural and property information about the noun itself. Preferences distinguish among literal, metaphorical and anomalous relations. Assertions can distinguish among novel, redundant, and inconsistent relations in a sentence.

The matching is done by two techniques:

Graph Searching: The source and target sense frames are compared by the type of path between them. There are five such types of paths:

1. The source and target are the *same*.
2. The target is an *ancestor* of the source.
3. The target is a sister, cousin, or second cousin of the source (i.e., has a common mother, grandmother or great-grandmother, respectively, with the same types of arcs connecting them to the common ancestor). This type of relation Fass terms *congeneric*.
4. The target is a *descendent* of the source.
5. The source and target do not fall into any of the above (*estranged*).

Frame Matching: The cells (slots) of two sense frames are compared to find the relations between the values of the attributes of the two sense frames. A tally is kept of the relationships between corresponding cells, where the possible relationships include the five types of paths, as for graph searching, plus tallies of those cells which are in the source but not the target, and vice versa (the *residuals*). The sense frames may be expanded to include inherited attributes and/or include only *salient* cells. A cell is deemed salient when the cell has a reference to the main sentence verb. A seven-slot vector is created to represent a tally of each kind of relation.

Through the graph searching and frame matching (including matching frames which are referenced in the node to determine metonymic relations), semantic vectors are determined. The types of relations in the semantic vectors are used to determine the type of match, if any. For instance, meaningless relationships would have a high proportion of residuals in the frame matching, and the graph searching would result in an estranged relation. Another example is metaphorical relations: they are detected by having an estranged graph relation but parallel semantic structures indicated by congeneric relationships in the preference field of the frame.

Not only is it appealing to have a numeric calculation of a match, but the vectors themselves itemize possibly important relations between the sense frames and also between their cells. Relationships using congeneric relations can only detect parallel structures to a depth of three, which makes the detection of such relationships dependent on how the network is built and in what detail. Fass had an exhaustive search and did not emphasize efficiency in his system.

2.4.4 Purely Structural Mapping

Gentner's Structure Mapping Engine, SME (Falkenhainer et al., 1987; Falkenhainer et al., 1986), based on her theory of analogy, takes a purely structural approach to matching.

The matcher is specified by a collection of rules that indicate what things might match and that estimate how strongly these matches should be believed. Thus this set of rules determines the type of matching to take place, and can be easily changed.

No guidelines are given for representation other than systematicity: the knowledge should be a system of related knowledge rather than an assortment of unconnected facts.

An algorithm is used in which first the facts of the source and target are compared pairwise, then these *local hypotheses* are merged into *global maps*. Global maps together with inference constructions (predicates existing in the base but not the target) become the final mapping. All consistent ways to interpret the analogy are constructed.

The SME is quite general in that the the matching rules can be respecified to determine the type of matching done. Empirically, it runs in linear time on very systematic, small knowledge bases ($O(n!)$ on the number of facts, theoretically). But, due to the fact that the SME takes a purely structural approach to matching, the engine is extremely dependent upon the representation chosen. Having more information encoded directly into the knowledge base structure would be more appealing, but of course runs counter to the original goal of flexibility. Finally, as Gentner points out herself, the SME does not deal with the problem of retrieving the appropriate information from memory.

2.4.5 Structured Associations

The KING Natural Language Generator (Jacobs (1985a; 1986)), a system meant to alleviate some of the problems of PHRED (Jacobs, 1985b), organizes conceptual and linguistic knowledge (categories) into a hierarchy of objects using *structured associations* in the Ace representation language. These structured associations (SAs) include:

- **Dominate:** an IS-A link connecting a subcategory with its parent category,
- **Manifest:** a role link connecting a category with a role of the category,
- **Instantiate:** used to indicate an instance of a concept,
- **View:** connecting concepts to other concepts that have a referential or metaphoric relation (the definition of these kinds of relations is not given),

- **Ref:** connecting a linguistic structure to its meaning (a concept),
- **Role-Play:** to specify correspondences between different concepts, and
- **Macro-Association:** a somewhat ad hoc structured association that allows biasing towards certain views by the generator.

These SAs are used to structure the knowledge hierarchically, with both linguistic and conceptual knowledge represented uniformly. However, node structure and the incorporation of constraints are not clearly explained.

Using heuristics to specify an order in which SAs should be applied, the SAs that are retrieved and applied relate the concept to be expressed by KING to the other knowledge in the hierarchy until, finally linguistic structures are reached. Templates for linguistic patterns that satisfy the input constraints, and the structures derived from the mapping, are then accessed. Finally, the constraints are applied to the patterns selected, in sequential order, and the first one satisfying the constraints is chosen to be inserted into the surface structure. It is unclear what input constraints can be specified. To illustrate the process, consider when the input is a particular *commercial transaction* event where Mary sells John a book. Structured associations are applied from this input commercial transaction event (a category in the hierarchy) until the *buying* structure is applied. The structure has ref arcs identifying linguistic relations needed to express *buying* in a surface structure. The templates for linguistic patterns are then chosen sequentially from among these linguistic relations to create a (possibly partial) sentence structure. Restrictions on these patterns map the roles of the event into the appropriate positions in the sentence structure to form the final sentence "Mary sold John a book". The role mapping may require recursive calls to the mapping–pattern selection–restriction sequence to obtain the surface form of the roles.

This approach predisposes the generator to *always* produce the same, *single*, mapping for the same input, especially through the use of macro-associations, and thus the same output generation is received. There is no way to incorporate synonyms or different ways of combining the same knowledge into a different form.

The representation of idioms is unduly complex, with many ad hoc concepts created to accommodate the transformational deficiencies of idioms. Though Jacobs claims that this method of representing idioms works in his system, he shows no working examples for the real system.

2.4.6 A Marker-Passing Approach

Rau's natural language System for Conceptual Information Summarization, Organization and Retrieval (SCISOR) (Rau, 1987) analyzes, answers questions, and summarizes newspaper stories about corporate takeovers and finance. Retrieval (matching) is performed in a two-stage process, the first stage using marker passing to retrieve a set of candidate answers, and the second using graph matching to find the final answer(s).

SCISOR's knowledge is organized as is KING's, using the Ace representation scheme, but the knowledge is divided into three partitions: event, abstract (generalized episodic), and semantic (for making inferences about input) memory. In addition, *tag* nodes with a numerical threshold value are used to connect related episodic and abstract concepts.

The marker-passing stage instantiates new inputs and performs constrained spreading activation. Each marking increases the activation value of a *tag* by one. If a *tag* reaches its threshold, usually one-third of the number of concepts in the *tag* group, it becomes a candidate retrieval. The candidates are then sent on for syntactic graph matching, which notes nodes that are more general than the input question, or that do not correspond, that could be thought of as presuppositions. These differences are reported to the user.

Rau's method has the advantage of being able to tolerate partial or contradictory input. This approach also allows content-addressable memory, since every input acts as an index to a retrieval.

2.5 Summary of Desirable Matcher Characteristics

In this section those characteristics of the matchers profiled in this chapter, and their knowledge representations, that are appealing for the Lexical Option Generator, and those that should be left out, are outlined.

Conceptual matching is the major component of the Lexical Option Generator. Thus, giving the system builder total control over which matching strategies are to be used and when (as in KRL and Finin's system for the interpretation of nominal compounds) is undesirable. For this same reason, although having a modular replaceable matcher system such as that of Gentner's Structure Mapping Engine is appealing for its versatility, it is not required for the type of system envisaged in this thesis.

The specification of situations for lexical units in the Lexical Option Generator requires representing interdependencies among the participants in the situation. For example, the situation for *narcissism* requires that the *agent* and *patient* of a *love* situation be equal (i.e., "John loves John" → "John is narcissistic"). Thus, a straightforward means of specifying the interdependencies is desired (as opposed to the complicated method of KL-ONE structural descriptions).

The numerical determination of matches, such as in Fass's system, is interesting from a computational point of view — numbers are easy to work with. A natural use of such an approach in the Lexical Option Generator might be assigning saliency in the parts of a situation. However, to assign numbers to parts of a situation, semantic entities, would inevitably require intuitive guessing to determine absolute values. Furthermore, a premise of this thesis is that all of the parts of the input knowledge are equally important, since small variations in any part can result in different lexical options being viable. Thus, a numerial approach is not advisable for the Lexical Option Generator.

In keeping with the idea that situations are not the knowledge representation itself but rather use a knowledge representation, syntactic information and situation information do not need to be uniformly represented as in Jacobs's KING system.

The following are the characteristics of the matcher and the knowledge representation desired for the Lexical Option Generator:

- The conceptual information should be arranged hierarchically (as in KL-ONE and KING), or network-like, to allow maximum use of abstraction and inheritance in the representation for searching of situations.
- A richer set of arc labels than just IS-A links should be used to inter-connect

conceptual information (as in Fass's system, KING, and KL-ONE).

- Partitioning or some other scoping of the knowledge should be used to aid in the tractability of the match.
- The matcher should use some variation of marker passing, where constraints must be imposed to avoid searching the entire network. This approach is appealing since every input acts as an index to the solution(s), and thus every part of the input information is considered to be equally important in the search for matches.

The Lexical Option Generator, LOG, incorporates the above desirable characteristics and avoids the undesirable characteristics outlined earlier. The overall operation of LOG is given in Chapter 3. Chapter 4 delves more deeply into the knowledge representation used for LOG, and Chapter 5 details the matching method used to search for situations in the lexicon.

Chapter 3

LOG: The Lexical Option Generator

This chapter gives a brief overview of the operation of LOG, the Lexical Option Generator. LOG provides appropriate lexical units that can be used by a natural language generator, a NLG, to produce language with variety and style. In other words, the different options that may be used by the NLG to express the same information are the output of LOG.

As shown in Chapter 1, sometimes lexical units can be directly substituted for one another and, at other times, different lexical items cover different portions of the information to be conveyed by the NLG. For example, there is no obvious one-to-one correspondence between the lexical items in sentence (1) and those in sentence (2) below, where "no-win situation" is an idiom and thus a single lexical unit:

(1) John was in a no-win situation.

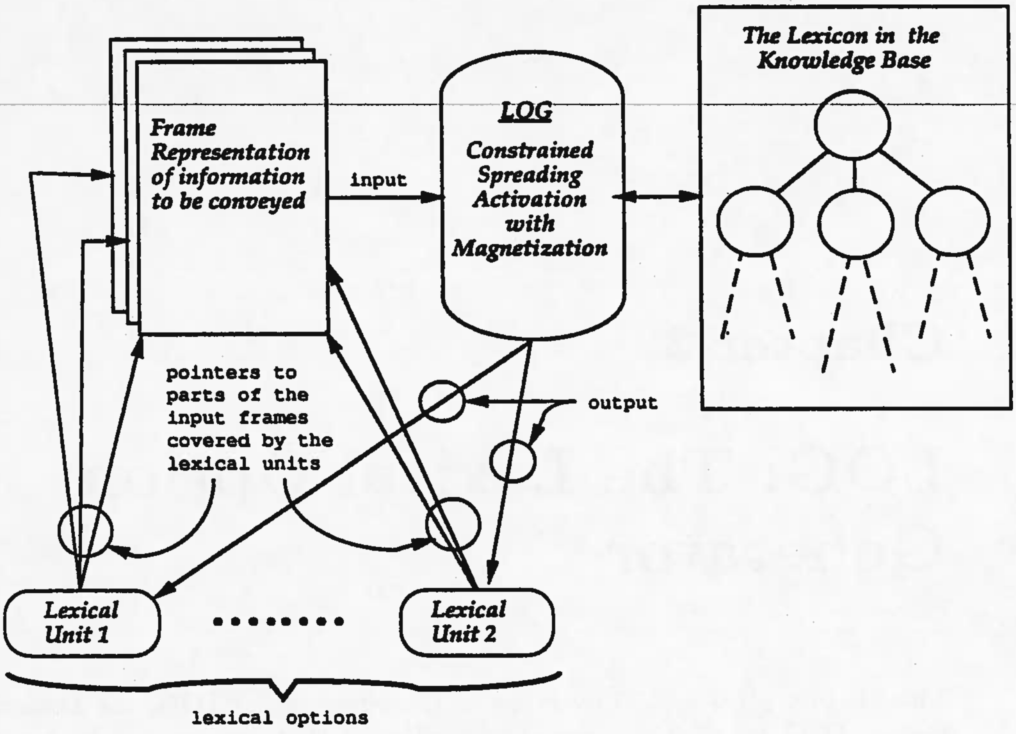
(2) John had no viable options to pursue.

(3) John had no tenable options to pursue.

However, sentence (3) conveys the same information as sentence (2) with the substitution of the lexical unit "tenable" for "viable". Thus, with each lexical option offered by LOG, the portion of the input information that the option covers is also output.

The input to LOG is a frame representation of the information a natural language generator wishes to convey. It is assumed that frames are the semantic representation scheme of the NLG, and that this frame-language has at its disposal an ample vocabulary of generic frametypes, slots, and individuals. This same vocabulary is used to specify, in the lexicon, the situations, or state of affairs, under which a lexical unit can be used.

The lexicon is organized into a hierarchical network of nodes. Each node defines a situation indicating the state of affairs necessary to use the lexical units associated with the node. The situation, expressed in the NLG's frame language, defines one sense of the lexical units associated with the node. The hierarchical structure of the lexicon mirrors the hierarchical nature of the situations defining the lexical units associated with the nodes. The nodes in the LOG lexicon are connected to other nodes by taxonomic



The Lexicon in the Knowledge Base
Frame Representation of information to be conveyed
input
LOG
Constrained Spreading Activation with Magnetization
pointers to parts of the input frames covered by the lexical units
output
Lexical Unit 1
Lexical Unit 2
lexical options

Figure 3.1: LOG: The Lexical Option Generator

and role links indicating the semantic relationships between the situations, and thus the lexical units, for each node.

The lexicon thus provides a rich set of lexical options by differentiating the more limited frame vocabulary into situations associated with lexical units. For example, the NLG may have the generic frametype `love` with slots for `agent` and `patient`. However, in the LOG lexicon, in addition to there being a node for the situation *love* with an arbitrary agent and patient, there is a differentiated *love* node, *narcissism*, where the *love* situation is further defined as having the agent the same as the patient. The NLG need not know specifically about the concept of *narcissism* and yet LOG could provide it with the correct lexical units for expressing such a concept.

The appropriate lexical units in the lexicon are found by *matching* the input frame to the situations in the lexicon. Matching is performed through a constrained spreading activation process. This thesis proposes a matching algorithm using a limited number of types of messages sent between the nodes of the lexicon. *Magnetization* is proposed as a method of dynamically orienting the nodes of the lexicon so that message information is channelled to likely matches efficiently (i.e., likely solutions act as "magnets" for messages, thus the magnetization metaphor). Furthermore, magnetization constrains the search for matches.

Each match in the network is reported. The values output from LOG are partial functional descriptions of the lexical units. Each functional description also contains pointers to the parts of the input frame covered by the lexical unit.

The LOG process is summarized pictorially in Figure 3.1. A simple example of part of the output from LOG for the input `(love (agent mary1) (patient john1))`

```
COVERING: 1.2, 1.1

[category = s                        ]
|subject = '1.1'                     |
|predicator = [category = v]         |
|             [lex = <love>]         |
[object = '1.2'                      ]


[category = s                                                      ]
|formality = informal                                              |
|transforms = passive                                              |
|subject = '1.2'                                                   |
|predicator = [category = v]                                       |
|             [lex = <be>  ]                                       |
|object = [category = np                                          ]|
|         |determiner = [category = article       ]                ||
|         |             |number = singular        |                ||
|         |             |definiteness = definite  |                ||
|         |             [lex = "the"              ]                ||
|         |head = [category = n     ]                              ||
|         |       |number = singular|                              ||
|         |       [lex = "apple"    ]                              ||
|         |modifier = [category = pp                            ] ||
|         |           |prep = [category = prep]                 | ||
|         |           |       [lex = "of"     ]                 | ||
|         |           |modifier = [category = poss]             | ||
|         |           |           [head = '1.1'   ]             | ||
|         |           |head = [category = n]                    | ||
[         [           [       [lex = "eye" ]                    ]]]]
```

Figure 3.2: Partial LOG output for (love (agent mary1)(patient john1))

is presented in Figure 3.2, showing the lexical unit information for "love" and "be the apple of x's eye". Note that the pointer 1.1 points to the value `mary1`, and the pointer 1.2 points to the value `john1`.

The process outlined in this chapter is dealt with in greater detail in Chapters 4 and 5. Chapter 4 presents the primitives for structuring the lexicon and the contents of the nodes of the lexicon, namely the situation and syntactic templates used for matching and providing the output, respectively. Chapter 5 details the matching algorithm used by LOG. The process of magnetization is explicated, and the passing of information in the lexicon is specified.

Chapter 4

Lexicon Organization

The representation of the lexicon is central to the operation of the Lexical Option Generator (LOG). The lexicon is organized into a hierarchical network of nodes, where each node defines the situation that licenses the use of the lexical unit(s) associated with the node. That is to say, the node delineates an equivalence class of lexical units with respect to a situation. Nodes are connected to other nodes by taxonomic and role arcs indicating the semantic relations between the lexical units for each node.

This chapter describes the lexicon representation. The primitives for structuring the nodes in the hierarchical network will first be described. The contents of the nodes themselves will then be outlined.

4.1 A Hierarchical Network Representation of a Lexicon

The semantic representation of a lexicon as an association network is not a new concept. Quillian (1966) organized words into a network representation, a semantic net:

> His [Quillian's] intent was to capture in a formal representation the "objective" part of the meanings of words so that "humanlike use of those meanings" would be possible. The representation was composed of *nodes*, interconnected by various kinds of *associative links*, and closely reflected the organization of an ordinary dictionary. The nodes were to be considered "word concepts" and links from a concept node pointed to other word concepts, which together made up a definition, just as dictionary definitions are constructed from sequences of words defined elsewhere in the same volume. (Brachman, 1979, pages 5–6)

Each of Quillian's nodes, or *planes*, represented one sense of a word. His lexicon was organized around the words themselves, an objective approach, rather than the meanings of the words.

Amsler (1981) has investigated the classificational organization of the lexicon, using the implicit classifications employed by the writers of a dictionary[1], for use in natural

[1]The dictionaries used for the analysis were from machine-readable copies of the *Merriam-Webster New Pocket Dictionary* and the *Merriam-Webster Seventh Collegiate Dictionary.*

language understanding applications. Each definition in the dictionary contains a noun or verb phrase with at least one kernel term. Sets of word pairs, the defined and the disambiguated defining kernel terms, were connected into hierarchical lattices (one for nouns with 24,000 noun senses, and one for verbs with 11,000 verb senses). The kernel words were hand-disambiguated. The two lattices were representative of the structure of the entire English lexicon as it used data from a complete English dictionary. Intuitive conclusions were confirmed about the structure in the resulting lattices:

> [The] bottom is a set of terminal disambiguated words that are not used as kernel defining terms; these are the most specific elements in the structure. The tops of the structure are senses of words such as "cause", "thing", "class", "being", etc. ...If all of the top terms are considered to be members of the metaclass "⟨word-sense⟩", the tangled forest becomes a tangled tree.

Sequences of interrelated definitions whose kernels formed a loop indicated primitives of the language. For example, circularity in definitions was found with the set of words "class", "group", "type", "kind", "set", "division", "category", "species", "individual", "grouping", "part", and "section". The definitions of this set of words embodied a primitive concept related to the *set* concept in mathematics.

Amsler has found that two taxonomic relationships emerge naturally from the assembling of the lattice: IS-A and IS-PART. For example, a fern IS-A plant and a frond IS-A leaf, but a leaf IS-PART of a plant as a frond IS-PART of a fern. IS-PART relationships, although expressed in the dictionary, do not contribute to the *meaning* of fern, but rather provide information that can be used to *infer* that something is a fern.

In this thesis, the lexicon is a hierarchical network of *meanings*, or situations, that *define* one or more lexical units. Each node represents one of these situations, and a node is connected to other nodes by taxonomic and role links. PART-OF relationships are not considered, as they do not define a word, per se, but rather are used to infer that an item A might be an item B (e.g., if you know a plant has fronds as its outgrowths, it might be inferred that the plant is a fern). The organization of the lexicon is based on the property that the values of the slots and the frametypes can themselves be expressed in natural language. In other words, adapting the desirable properties of partial matching in (Hayes-Roth, 1979), the situations necessary for lexical entities, as well as the input frames, have the following properties:

1. *Part recognizability:* A part of the situation being described can be recognized as a whole in itself. That is to say, it is itself a situation and expressible in at least one lexical unit.

2. *Attribute combination effect:* A part of the situation being described can be recognized because some combination of its attributes is represented in the description.

3. *Part-whole continuity:* This property implies that the preceding two properties can be applied recursively. Thus each situation is made up of parts which are themselves situations, and recognizable as such, and thus expressible by lexical units.

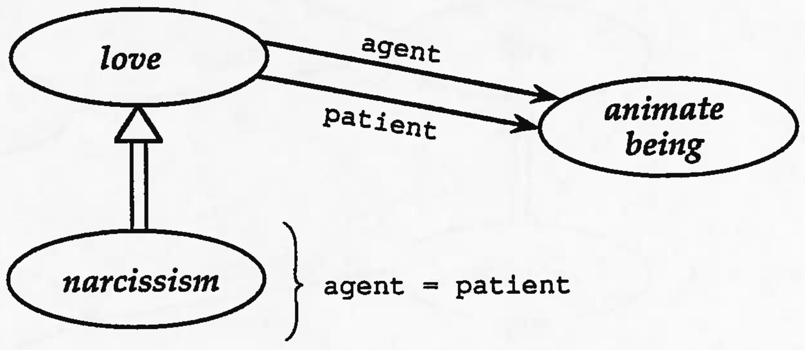


Figure 4.1: `Narcissism` as a restricted `love` situation.

Clearly, these properties hold for the descriptions of situations in the lexicon, for otherwise, we would not be able to define words and idioms. In other words, all of the nodes in the lexicon are *expressible*, either directly by the lexical unit(s) specified in the node, or, for those nodes without an associated lexical unit, by some grammatical composition of the expression of the situations which make up the new situation.

4.2 Structuring the Lexicon

This section will outline the primitives used for organizing the nodes in the lexicon. The nodes themselves will then be described in detail in section 4.3.

4.2.1 Linking the Nodes

Links are used in LOG to create the hierarchical network of nodes. The types of arcs used in the representation of the lexicon are based on structuring primitives found in other knowledge representations such as KL-ONE (Brachman, 1985) and Ace (Jacobs, 1985a; Jacobs, 1986). The types of arcs include taxonomic, role, restriction, differentiation, and instantiation links.

Taxonomic Links

A taxonomic link is commonly called an IS-A link. The true meaning of such a link has come under much question in the past (see (Woods, 1975) and (Brachman, 1979)). For the purposes of this thesis, if there is an IS-A link from a node B to a node A, that is, "B IS-A A" (as there is from `narcissism` to `love` in Figure 4.1, and from `gobble` to `eat` in Figure 4.2), the following are true:

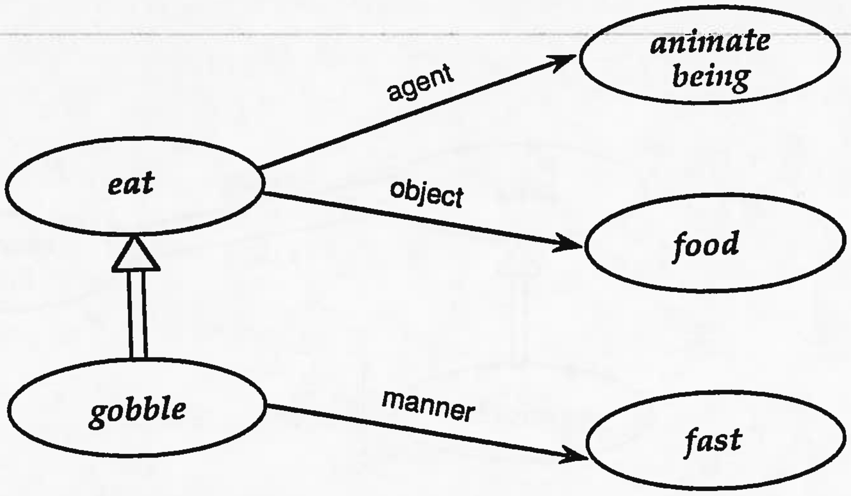


Figure 4.2: Adding an attribute to the eat situation to make a gobble situation.

1. The link can be characterized by a change in the situations which correspond to B and A. This change is achieved by one or both of the following:
   - A is made more specific in B. For example, in Figure 4.1, `narcissism` is specified as a `love` situation where the `agent` is the same as the `patient`.
   - A new attribute is added to the situation in A to result in the situation in B. For example, in Figure 4.2, a new attribute `manner` is added to the `eat` situation to create the `gobble` situation.
2. If a situation can be described by the lexical units in the B node, then it can also be described by the lexical units in the A node. For example, in Figure 4.1, a `narcissism` situation can be described as a `love` situation. Likewise, in Figure 4.2, a `gobble` situation can also be described as an `eat` situation.
3. If a situation can be described by the A node, then it cannot necessarily be described by the B node. For example, an `eat` situation cannot necessarily be described as a `gobble` situation, nor can a `love` situation necessarily be described as a `narcissism` situation.

As a matter of terminology, the notion of travelling from a node along an IS-A link to its subsuming node will be called "going up" the IS-A link. Conversely, going the other way along the IS-A link will be "going down" the IS-A link.

Role Links

Role arcs link nodes to associated nodes that describe the components of a situation in the node, like the slots in a frame representation. These associated nodes may have the following relationships with the node originating the role link:

- *Collection.* For example, the node for weekend would have role links to the nodes for Saturday and Sunday, with a constraint in the weekend node that the Saturday and Sunday in question must be consecutive. As another example, the node for ice cream sundae would have role links to the nodes specifying the parts making up a sundae: ice cream, nuts, whipping cream, etc.

  Note that these role links should not be confused with partonomic links. Partonomic, or PART-OF, links in knowledge representations generally specify the structural pieces of a whole. For example, a hand is a PART-OF an arm, and a screw is a PART-OF a desk. The role links must associate nodes aiding in the *definition* of the meaning of the node. The parts of a sundae define the sundae, but a screw does not define a desk, even though a desk may have screws among its parts.

- *Subcategorization.* For example, the situation for the node love, corresponding to the verb "love", subcategorizes for an agent and patient, and thus there would be role arcs corresponding to the agent and patient of the love situation (see Figure 4.1).

- *Additional Attributes.* For example, the eat situation does not necessarily have to have a manner slot, but to define gobble, an additional attribute of the manner being fast differentiates gobble from eat (see Figure 4.2).

The notion of travelling from a node along a role link to the role node will be called "going along" a role link. Conversely, travelling the opposite way on a role link will be called "going backwards" on a role link.

Restriction Links

Restriction links specify that a role link of a subsuming node is being further specialized to a subsumed node. For example, the idiom "pass the buck" may have transfer as its subsuming node where the object role must be restricted to blame, rather than an arbitrary transfer object. Thus, this type of link causes the replacement of a subsumer's role link with a more specific role.

Differentiation Links

Differentiation links specify that a role link of a subsuming node is further distinguished into two or more role links in the subsumed node. For example, suppose that the node for transitive-action has a role for the instrument of the action. The subsumed node, paint, could differentiate the instrument slot into the roles for applier (as in the brush used for the painting action) and applied (as in the paint used in the painting action) (Charniak, 1981).

Instantiation Links

Instantiation links connect an *individual* to its subsuming nodes. For instance, if John1 is an individual who is a person and a male, then there should be instantiation links from John1 to person and male. It should be noted that it is these instantiation links that connect the definitional situation hierarchy to the rest of the knowledge base expressing assertional knowledge. Individual nodes do not have a situation associated with them, but may have associated lexical units.

Summary

Taxonomic, role, restriction, differentiation, and instantiation links are used to structure the LOG lexicon in the knowledge base. This representation is similar in spirit to other knowledge representation languages such as KL-ONE (Brachman, 1985) and Ace (Jacobs, 1985a; Jacobs, 1986). For perspicuity, in general only the roles resulting from differentiation and restriction will be referred to and shown in diagrams. For this thesis, the role number restrictions of KL-ONE are performed by the specification of the situations within the node itself. One can think of the role and IS-A links in LOG's representation as *channels* along which information can be passed. This is in fact the way the lexicon structure will be viewed in the matching process outlined in Chapter 5.

4.2.2 Inheritance

It is advantageous to maximize the use of inheritance in the network. For the purposes of the matching algorithm in LOG, only the topmost node(s) in the hierarchy that uses a specific role arc, including those specified by restriction and differentiation arcs, is required. It is implicit that these roles are inherited down the network along the taxonomic links (as in KL-ONE, see section 2.4.2). Also, since the taxonomic links are transitive up the IS-A hierarchy, an IS-A link only connects to the node(s) immediately subsuming it. This use of inheritance minimizes the number of arcs in the network, thus constraining the number of possible channels along which information can be passed in the network during the matching process.

4.3 The Node Structure

Each node in the LOG network corresponds to a single situation. In most cases, this situation defines one or more lexical units. However, for completeness of the network, not all of the situations need to be expressible by a single lexical unit.

There are three major components of a node:

1. A *reference number* is used to obviate the need to clear the network of old information after each new input.

2. A *situation template* defines the situation which must be satisfied for the node to be matched.

3. A *syntactic template* is used, if the situation template is matched, to report the necessary syntactic and stylistic information associated with each of the lexical units pertinent to the matched node. This component is not present for those nodes without associated lexical units.

The use of the reference number is detailed in section 5.5. The situation template and the syntactic template are described in the following two subsections.

4.3.1 The Situation Template

The *situation template* has two components, the first being a representation of the pieces of information participating in the situation, and the second being the constraints on or between those participants. These two components will be called the *template* and the *constraints*, respectively.

The templates are in the following form:

$$
\begin{array}{llll}
((origin_1 & slotname_1 & ?filler_1 & inframe_1) \\
 & \vdots & \vdots & \vdots \\
(origin_n & slotname_n & ?filler_n & inframe_n))
\end{array}
$$

where:

- *origin$_i$* specifies the node which is expected to supply the filler for that field in the template,
- *slotname$_i$* specifies the slotname for the incoming information from the input frame,
- *?filler$_i$* is a variable name to be used in the constraints section of the situation template, and
- *inframe$_i$* specifies the generic frametype that the incoming information should originate from. The entries in the template that must be in the exact same frame are also specified.

For example, referring to Figure 4.1, the template for the node `narcissism` is as follows:

```
((animate-being agent ?agent love)
 (animate-being patient ?patient love))
```

Thus, the node `animate-being` is to supply a filler for both the `agent` and the `patient` slot of this situation. The template would also specify that both the `agent` and the `patient` must originate from the same input `love` frame.

The constraints are specified by a predicate followed by the arguments to the predicate. These arguments may include the filler variables specified in the template, which indicates that the filler value should be used in the constraint test. For example, consider the portion of the network shown in Figure 4.1. A differentiation must be made between the concept of `love` and the concept of `narcissism` by the use of a constraint on the fillers for the participants of the love situation in the `narcissism` node:

```
(equal ?agent ?patient)
```

specifying that the agent and patient of the `love` situation must be the same animate being, thus defining narcissism. If both the `agent` and `patient` of the `love` frame were `John1`, then it would be possible to say that "John is narcissistic" using the information from the `narcissism` node, and also, using the `love` concept, that "John loves John" (a sarcastic remark such as in "Who does John love?", "John loves John!").

The predicates can specify arbitrary interrelationships between the fillers of a template, as they are defined separately in a procedure detailing the operation of the predicate. It is these constraints that play the part of the *structural descriptions* in KL-ONE, which are not as straightforward:

> The need to handle the various possible relations among Roles makes the technical details of Structural Descriptions (SDs) a bit messy. However, the intent is straightforward — an SD allows the formation of a description whose essential difference with its proximate genus is a relationship among more than one of its Roles. (Brachman, 1985, page 192)

The argument to a constraint in the situation template may also contain field specifications in the filler value. For example, consider the node for `fight` where a fight consists of two, or more, `violent-actions`. The template would be as follows:

```
((violent-action nil ?violent1 nil)
 (violent-action nil ?violent2 nil))
```

where each of the `violent-actions` would originate from a different frame in the input. Note that in this template there are no slotname or inframe specifications. Each of the the `violent-actions` is a frame in itself, and thus does not have a slot value in a frame. A `violent-action` may be a `hit` or a punch. The constraints on the `fight` node would be:

```
((equal ?violent1.agent ?violent2.patient)
 (equal ?violent2.agent ?violent1.patient))
```

Thus, if John$_{(agent)}$ kicks Mary$_{(patient)}$, and Mary$_{(agent)}$ punches John$_{(patient)}$, each matching a `violent-action` situation, a `fight` situation is satisfied.

4.3.2 The Syntactic Template

The syntactic template is used to report the necessary syntactic and stylistic information associated with a lexical unit when a node is matched. Although the content and format of this information would depend on the specific syntactic and stylistic theories chosen by a natural language generator (e.g., see (Watt, 1988)), an attempt has been made to provide the generic information which could be tailored to a natural language generator's individual needs. A partial functional description was chosen for the format of the syntactic output, since each item in such a description can be considered independently, conforming to LOG's function of providing possibly partial lexical coverings of the input information.

For each lexical unit associated with a node there is a template for the functional structures covered by the situation of the node. For example, consider the situation for `narcissism` from Figure 4.1 once again. The syntactic templates for the node are as shown in Figure 4.3.

The notation used for the various fillers is as follows:

- Angle brackets, ⟨*lexical-form*⟩, delimit a lexical form which can differ only in respect of inflections to create a realized word form. For example, ⟨be⟩ could be inflected

```
┌category = s                                           ┐
│subject = '?agent'                                     │
│predicator =  ┌category = verb       ┐                 │
│              └lex = <be>            ┘                 │
│complement =  ┌category = adjective                   ┐│
└              └lex = <narcissistic> | <conceited>     ┘┘

┌category = s                            ┐
│formality = informal                    │
│subject = '?agent'                      │
│predicator =  ┌category = verb┐         │
│              └lex = <be>     ┘         │
│complement =  ┌category = adjective┐    │
└              └lex = <stuck-up>    ┘    ┘

┌category = np                 ┐
│head =   ┌category = noun     ┐│
│         │number = singular   ││
│         └lex = <narcissism>  ┘│
└modifier = '?agent'           ┘
```

Figure 4.3: Three syntactic templates for `narcissim`

to "is" or "are". These possible *inflectional* variations should not be confused with *derivational* variations which produce lexical units with new meanings. For example, the derivational variations from "true" → "*un*true" or "child" → "child*ish*" would not hold in the same situation. Inflectional changes, however, merely add feature markings to a lexical form. Inflections may include:

- the *conjugation* of verbs for number, tense, and person, or
- the *declension* of nouns, adjectives, and adverbs according to number, person, case, and gender.

These inflections are assumed to be performed by the natural language generator during the realization of a sentence. A lexical form is oftentimes not able to be inflected within an idiom. For example, some idioms cannot be pluralized (e.g., "John is the cat's meow" but not "John and Mary are the cat's meows"). In these cases, double quotes surround the word(s).

- Single quotes, '*variable*', are used to indicate that in the output, pointers to the parts of the input frame covered by the filler variable are to be specified. The natural language generator (NLG) would then fill in this part of the functional definition(s) with the functional definition covering the parts of the input frame specified by the pointers. For example, `'?agent'` in the `subject` field in Figure 4.3 indicates that the NLG should find another functional description covering the parts of the input frame pointed to by the `subject` field in the output.

- Braces, { ...}, indicate an optional portion of the functional definition having no impact on the meaning of the form. For example, for many idioms, optional insertions may be made that have no impact on the meaning of the idiom (e.g., "kick the bucket" or "kick the proverbial bucket").

- The OR bar, |, is used to indicate equivalent alternate choices.

In addition to the syntactic information, stylistic information is associated with each lexical unit. As with the form of the syntactic output, the form and detail of style markings would ultimately depend on the natural language generator's needs. However, as an indication of the type of information which would be included with each lexical unit, the level of formality and some attitudinal markings are given with each lexical unit.

Formality level reflects three dimensions of information simultaneously (Cowie and Mackin, 1975; Cowie and Mackin, 1983; Strässler, 1982):

1. The *social relationship* between the speakers. For instance, slang would not be used by a worker talking to a superior.

2. The *medium*, that is, whether the language is spoken or written.

3. The seriousness, detachment, or importance of the *occasion* in which a lexical unit is being used. For instance, a going-away speech for a friend at a personal party would require less formal speech than one for the Vice-President of a company at an official banquet.

Formality	Social Relation	Medium	Occasion
neutral	all	written or spoken	all
formal	distant, elevated	spoken	serious
		writing	all
informal	intimate	spoken	modest
slang	intimate	spoken	particular sub-group
taboo	generally unacceptable except among intimate sub-groups	spoken	very informal, often expressing tension, irritation, or anger

Table 4.1: Formality levels in the syntactic template

Lexical units are marked using five levels of formality: neutral, formal, informal, slang, and taboo. This marking is adapted from (Cowie and Mackin, 1975). The meaning of these markings is summarized in Table 4.1. A neutral formality level is the default for lexical units.

Attitudinal markings indicate whether the use of a lexical unit is derogatory, facetious, or humorous. For example, the lexical unit "a flea pit" is a derogatory term for a cheap theatre or cinema (Cowie and Mackin, 1983).

The formality and attitudinal markings, if any, are added as another field in the functional description of the lexical unit.

4.3.3 Idioms in a Syntactic Template

The main considerations for incorporating idioms into syntactic templates are the syntactic complexity of the idioms and their transformational deficiencies.

In Chapter 1 of this thesis, the multifarious syntactic structures of idioms, as well as their transformational deficiencies, were outlined. It was also pointed out that, to date, no linguistic theory has been able to completely account for this variability. The approach in this thesis is to mark each idiom with the transformations it *can* undergo, while giving internal syntactic structure to the idiom. The possible transformations are indicated by a `transforms` field in the functional description. For example, the lexical entry for "bury the hatchet" would be as shown in Figure 4.4. If there is no `transforms` field in the functional description, then the lexical unit has no transformational deficiencies, as for non-idioms.

```
[category = s                                                ]
|transforms = passive                                        |
|formality = informal                                        |
|subject = '?agent'                                          |
|predicator =  [category = verb]                             |
|              [lex = <bury>   ]                             |
|object =   [category = np                                  ]|
|           |determiner =   [category = article      ]      ||
|           |               |number = singular       |      ||
|           |               |definiteness = definite |      ||
|           |               [lex = "the"             ]      ||
|           |head =   [category = noun   ]                  ||
|           |         |number = singular |                  ||
[           [         [lex = "hatchet"   ]                  ]]
```

Figure 4.4: Syntactic template for "bury the hatchet"

Chapter 5

Matching in LOG

In this section, the matching operation is outlined. The matching algorithm can be described as a constrained spreading activation process where each node has limited operations, and there are a limited number of types of messages being sent between nodes. The key to constraining the search is the *magnetization* of nodes.

5.1 Marker Passing – A Precursor to LOG

One inspiration for the matching algorithm came from Fahlman's "marker passing" in NETL (Fahlman, 1979), which incorporated type hierarchies and property inheritance. In NETL, for each component of the object being looked for, a marker would be given to all the nodes in the network representing that component. Each mark would then be passed up the IS-A hierarchy from the marked component until no new nodes were marked. Marks were sent in succession until intersections of marks from all origins were found by a query to the nodes in the network. It was assumed that the network operations were totally parallel and that marks could be sent to all of the nodes in the network at the same time. The classic example: if the goal was to find the colour of Clyde the elephant, a marker would be sent first to the node for Clyde in the network, and then marks would be passed up the IS-A hierarchy until all of the subsumers for Clyde were marked. Then, a new marker would be sent to any node which was connected by a colour link to an already-marked node. The resulting intersection would be the colour of Clyde.

Charniak (1983) then proposed "dumb marker passing", which was not concerned about the types of links that markers were passed along, unlike Fahlman's. Given a string of disconnected words (i.e., without functional structure), his system would disambiguate word senses, establish case relations, and propose explanatory actions using this dumb marker passing scheme. With such a scheme, false positives were a very real problem, as intersections could occur from unrelated components of the search:

> One of the key differences between these two formulations of marker-passing [smart and dumb] is the treatment of variables. In a smart marker-passing scheme we must take into account the bindings, in the dumb marker-passing scheme we needn't. Thus, the smart marker-passer would not find a path

between "Bill going to the restaurant" and "John losing a wallet", while the dumb marker-passer would. (Hendler, 1986, page 6)

Thus, a separate deduction program was needed to interpret the paths found by the marker passer. An inevitable false positive would occur at the top of the hierarchy where all of the marks would intersect. Also, sending marks both up and down IS-A's would result in the *entire* network being marked. These problems with marker passing thus demonstrated the need for *constraining* the passing of markers, which may include any number of the following rules outlined in (Hendler, 1988), (Hendler, 1986), and (Charniak, 1983):

1. Determine an absolute (yet arbitrary) limit on the number of markers passed and/or on the breadth of activation. For example, if an absolute limit of 10 is set on the breadth of activation, and a correct path has a length of 11, the longer path will not be found. In consequence, the results are dependent on the form of the knowledge, not the meaning. In one variation of this constant, this limit is often called *zorch*, and is partially consumed as each marker is passed. Zorch is determined empirically, often by a factor of the average length of a valid path.

2. Check for *promiscuous* nodes. If a node has more links to other nodes than some predetermined (and again, arbitrary) limit, that node cannot send out markers. Hendler (1986) also proposes not even allowing promiscuous nodes to be marked as they will tend to have intersections and report false paths. For example, there are many things which are physical objects, and thus intersections not leading to a desired result, in most cases, would take place at the physical object node. The limit used to determine promiscuity is dependent on the knowledge structure chosen. For example, a node for *animal* may have all of the animals (e.g. dog, cat, elephant) as direct descendents, or may be broken down into the taxonomic classification of the animal kingdom, with the animals being the eventual leaves in the taxonomy. If the limit is the same in both cases, *animal* may be classified as a promiscuous node in the former, but not in the latter.

3. Pass markers only one way along links. For example, pass markers only up IS-A links, but not down. Charniak (1983) proposed not allowing marks to change the direction they take on IS-A links, once one direction has been chosen.

4. To avoid looping in the marker passing, leave traces of the origins of each mark at the nodes. If a cycle occurs in the knowledge base, which is not uncommon, a node in the cycle can propagate a mark, but when the same mark is received a second time, the marker propagation is terminated.

5. Stop passing marks at a particular node once an intersection is found at that node. This approach can help to prevent redundant paths from being reported, but can also prevent a necessary solution from being found (Hendler, 1986). If a record of the paths originating a mark are kept by each node, redundant paths can be checked for at the nodes themselves when an intersection occurs. Markers are not propagated if an intersection has already been found on a path between the same

two nodes. For example, if a node, A, receives a mark whose originating path contains B, and then receives a second mark whose originating path also contains B, the second mark can be discarded, as it is redundant. The assumption that the first path found is the best one is made in this case.

Options 1, 2, 3, and 5 above have an unfortunate side effect of stopping short of finding all of the correct answers, and may find none at all.

The approach taken in LOG is to return to a "smart" method of passing information in the network: having the information paths depend on the type of links between nodes and sending out different types of information depending on the state of the node. The state of a node is maintained by binding pertinent information at each node. Local matching and binding at each node ensure that no false matches are reported. Furthermore, the search for matches will be aided by *magnetization*, to be discussed in section 5.3.

5.2 Input to LOG

The input to LOG is a representation of the information to be conveyed. Although there are many different, yet equivalent, forms of knowledge representation, I will assume that the input is in the form of frames. That is to say, the input is of the form:

$$\begin{array}{llll}
(frame_1 & (slotname_{11} & value_{11}) \ldots (slotname_{1i} & value_{1i})) \\
\vdots & \vdots & \vdots & \vdots \\
(frame_n & (slotname_{n1} & value_{n1}) \ldots (slotname_{nm} value_{nm}) &
\end{array}$$

where any of the value$_{ij}$s may themselves be frames. If more than one frame is input (as shown above) then there is an implicit AND relation between them. For example, if John kicks Mary and Mary punches John (i.e., John and Mary fight), the input would be:

```
(punch (agent mary1)(patient john1))
(kick (agent john1)(patient mary1))
```

It is assumed that these input frames are the working language of a natural language generator (NLG). The various frametypes and the individuals are the vocabulary of the NLG for semantic representation. This is also the vocabulary used for expressing the situations in the lexicon. The LOG lexicon has the means for determining more complex relationships among the elements of the NLG representation. Thus, there is a tradeoff between the complexity of the NLG's language and the complexity of the situations in the lexicon. In the past, most natural language generators have obviated the need for such a tradeoff by maintaining a one-to-one relationship between the items in a lexicon and the elements of the underlying representation (Cumming, 1986; Hovy, 1987). However, LOG's goal is to provide a richer system for expressing the semantic knowledge without enforcing further complexity on the semantic representation of the NLG. This is fundamental for a system to be used for translation or incorporating stylistics.

The input also includes a reference number, which is discussed in section 5.5.

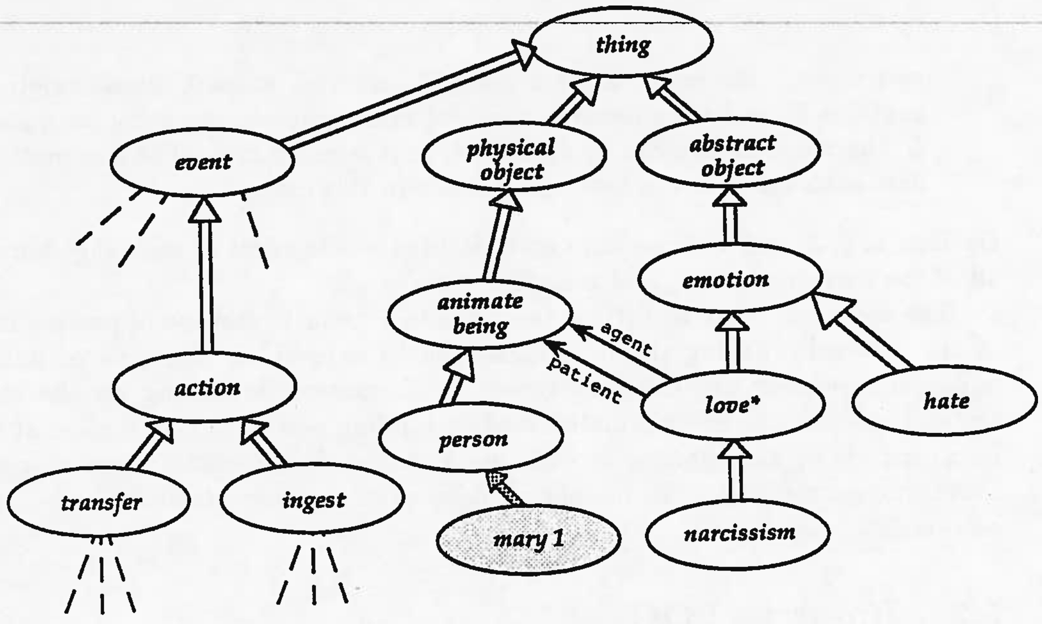


Figure 5.1: A sample portion of the lexicon

5.3 Magnetization

When searching for something in a network, it would be advantageous to have the correct node act as a *magnet*, attracting the bits of information being passed around the network. It was with this concept in mind that the *magnetization* metaphor was conceived for describing the method in LOG used to efficiently expedite the passing of information to nodes which are likely to need the information to match the situation associated with the node.

Magnetization can be thought of as orienting the nodes on an IS-A path to channel information to the origin of the magnetization. Whenever information arrives at a magnetized node, it is immediately passed to the origin of the magnetization, thus avoiding passing any information along extraneous IS-A and role links. It should be recalled that the only role links present in the network are those to the *topmost* nodes in a hierarchy requiring a specific role arc (see section 4.2.1).

It is assumed that throughout the lexicon there are distinguished nodes, the magnets, that can begin the magnetization process. These distinguished nodes correspond to the *highest* nodes in the network, along each IS-A path, containing a particular generic frametype in the situation for the node. For example, consider Figure 5.1, in which a portion of the lexicon is pictured. In this figure the distinguished node, with respect to the `love` frametype, is marked with a "*" (most of the role nodes and role arcs have been omitted for simplicity). The `love` node describes a situation involving the generic *love* frametype, as does the `narcissism` situation. If one travelled down the IS-A links from the root of the hierarchy, `thing`, the node first encountered on each path containing a situation using the `love` frametype would be a distinguished node. In this example, the `love` node would be a distinguished node (but note that `narcissism` would not be distinguished). This example also suggests a simple algorithm for determining these distinguished nodes. An inventory is kept for the distinguished nodes of each frametype.

The highest such node is deemed the distinguished node, as it is desirable that the magnetic origin be the most likely node to be matched. Since, by the requirements of a valid IS-A hierarchy, none of the subsumed nodes could possibly be matched unless the subsuming node is matched, the highest node is the most likely to be matched. For example, the `narcissism` node in Figure 5.1 could not be matched unless the `love` node was also matched. However, those nodes above `love` would not necessarily need to be considered, thus constraining the search.

Each of the generic frametypes in the input to LOG will have at least one distinguished node associated with it to begin the magnetization process. All of the frametypes in a situation must be present for a node to become magnetized. From a magnetized distinguished node, a message is sent up the node's IS-A path indicating the origin (the distinguished node) of the magnetization, and the information sought, thus creating the "magnetic field".

5.4 Passing Information in the Network

The matching in LOG is performed by the passing of information in the network. This information takes the form of messages (described below) and is incorporated into the situation definition of each node. When a complete situation has been accumulated, a match has occurred and it is reported.

There are four types of messages proposed that a node can send to another node: magnetize, i-am-a, i-have-a, and subsumer-matches.

Magnetize

As was outlined in the previous section, the distinguished nodes pertaining to the input representation are considered to be the origins of the magnetization. These magnetic origins will then send *magnetize* messages up their IS-A links to begin creating the magnetization path.

The magnetize message contains only the reference number, the magnetic origin, and the nodes (message origins) that require the relaying of their information to the magnetic origin. Upon receiving a magnetize message, the node records for future reference the magnetic origin and the message origins sought, and then proceeds to send magnetize messages up its own IS-A links.

For example, consider the `love` node of Figure 5.1. Suppose the template for `love` was:

```
((animate-being agent ?agent love)
 (animate-being patient ?agent love))
```

See section 4.3.1 for a review of the structure of the situation template. The relevant information in the template, for this example, is `animate-being`, the node from which the information for filling in the `agent` and `patient` roles of the situation template is expected. If `love` is made a magnetic origin, a magnetize message containing the magnetic origin, `love`, and the message origin, `animate-being`, from which to look for information, is sent up `love`'s IS-A link to `emotion` and propagated up through to `thing`.

I-Am-A

The *i-am-a* message is sent from a node under three conditions:

1. If the input representation contains an instantiation of an individual, an i-am-a message is sent up all of the instantiation links of the individual.
2. If the situation of a node is matched, an i-am-a message is sent up all of the IS-A links of the matched node, and to the node itself.
3. If a node receives an i-am-a message, then the message is propagated up all of the node's IS-A links. However, if the i-am-a message received covers the exact same portion of the input as a previous i-am-a message, then the message is discarded.

The following information is included in an i-am-a message:

1. The reference number.
2. The original matched situation (or in item 1 above, a representation of the individual).
3. The origin of the message (i.e., of the original matched node or individual).
4. The slot and frame information of the origin of the message. For example, consider if the slot value for the slot `agent` of an input frame, A, is a frame itself, B. If the frame B matches a situation, B_{sit}, the i-am-a message would include the information that the B_{sit} matched in the `agent` slot of the input frame.
5. Pointers to the parts of the input frame(s) covered by the matched situation.

For example, suppose the input frame is `(love (agent john1)(patient mary1))`, with reference number 42. The individual node `mary1` would receive an i-am-a message, and propagate the message to the `person` node. The `person` node would then propagate the following information to the `animate-being` node:

```
(42 mary1 person patient love ⟨1.2⟩)
```

where `42` is the reference number, `mary1` is the matched situation information, `person` is the message origin, `patient` is the slotname, `love` is the inframe value, and `⟨1.2⟩` is the pointer to the slot `(patient mary1)` portion of the input frame.

I-Have-A

An *i-have-a* message is sent to nodes that could possibly use a matched situation to match their own situation. Thus, these types of messages are sent backwards along role links. The i-have-a messages are sent under either of the following two conditions:

1. A node receives an i-am-a message. In this case, the information to be sent out is exactly the information received in the i-am-a message. For example, suppose the node for **animate-being** receives an i-am-a message from the **person** node (as in the example of an i-am-a message in the previous section). This **person** might be able to play the role of the **animate-being** in some other situation. Thus, the information about the **person** is sent out backwards along all of the role links to **love**.

2. If a magnetized node receives an i-have-a message, then the message is simply relayed to the magnetic origin(s) as an i-have-a message (thus acting as if there had been a role link directly from the magnetic origin to the origin of the i-have-a message). The information sent in an i-have-a message is the same as for the i-am-a message.

When a node receives an i-have-a message, the new information is incorporated into the partially matched situations in the node. If a situation thus becomes matched with the newly incorporated information, the syntactic information of the node is reported (see section 4.3.1). The node then sends itself an i-am-a message. If a duplicate i-have-a message is received (i.e., from the same origin covering the same portion of the input), the message is discarded.

Subsumer-Matches

Once a subsuming node has been matched, the opportunity arises for subsumed nodes to be matched. For example, in Figure 5.1, if the **love** node is matched, possibly then the **narcissism** node can also be matched. When a node is matched, a *subsumer-matches* message is sent to all of its subsumed nodes (e.g., **love** would send a subsumer-matches message to **narcissism**). The message contains:

1. The reference number.

2. The matched subsumer situation.

3. Pointers to the parts of the input frame(s) covered by the matched situation.

An attempt is made by the subsumed nodes to incorporate the matched situation. If the attempt is unsuccessful, no match in the subsumed node is possible using the subsumer matched situation. Otherwise, the matched situation together with any new information the node might have received so far, or receives later, may combine to match the node.

5.5 The Reference Number and Clearing the Network

The reference number is used to avoid the need for clearing the network on each new input. When a new reference number is used, usually for each new input, it indicates that any information using a different reference number is now invalid. When a node receives information with a new reference number (i.e., not corresponding to the one recorded in the node), all previous information received by the node is discarded before processing the new information.

5.6 Matching Situations in a Node

As outlined in section 4.3.1, a situation template is present at each node specifying the participants in, and the constraints for, a match.

Associated with the template is a list of all potential mappings into the template of the information received so far. Initially, of course, this list has one empty element. As each new piece of information arrives at a node (via i-have-a and subsumer-matches messages), all of the possible incorporations of the new information are found. At present, i-am-a messages are not incorporated as a means to constrain the number of lexical options reported. For example, if `mary1` is a `person`, and a `person` is an `animate-being`, only the lexical options for `mary1` will be reported. The lexical options for "a person named Mary" and "an animate being named Mary" are not considered for this thesis.

It should be noted that situations are separate from considerations such as formality level and attitude. The situations deal with the semantics of the lexical units, and only semantic matches are found. A further filter would be required at each node, or on the entire output, to accommodate screening on the pragmatic aspects of lexical units.

5.7 Examples

The output from LOG is a set of instantiated syntactic templates. This section will present some examples of the operation of LOG. For each example, a sample portion of the lexicon is presented. Not all of the role arcs, role nodes, and IS-A links are shown in the lexicon figures, to simplify the illustration. The nodes have been given names that imply the differentiation between the nodes, and thus in most cases the situation template and constraints for the node are obvious. The nodes marked with a "*" are the distinguished nodes for the particular example. Pointers to the input frame in the LOG output specify the nesting level and relative slot number of the slot covered. For example, consider the input:

```
(frame1
   (slot1 value1)
   (slot2 (frame2 (slot21 value21)
            (slot22 value22))
   (slot3 value3))
```

A pointer to `slot1` would be `1.1` (the first slot of the outmost frame), and a pointer to `slot21` would be `1.2.1` (the first slot of the frame in the second slot of the outer frame).

Example 1

Input: `(love (agent mary1)(patient john1)), reference number = 1`

Consider the portion of the lexicon pictured in Figure 5.2. The frametype `love` is in the input, and thus its distinguished node, `love`, is magnetized. As a result, `emotion`, `abstract2`, `abstract-object`, and `thing` also become magnetized, ready to relay any information received from an `animate-object` or `person` origin.

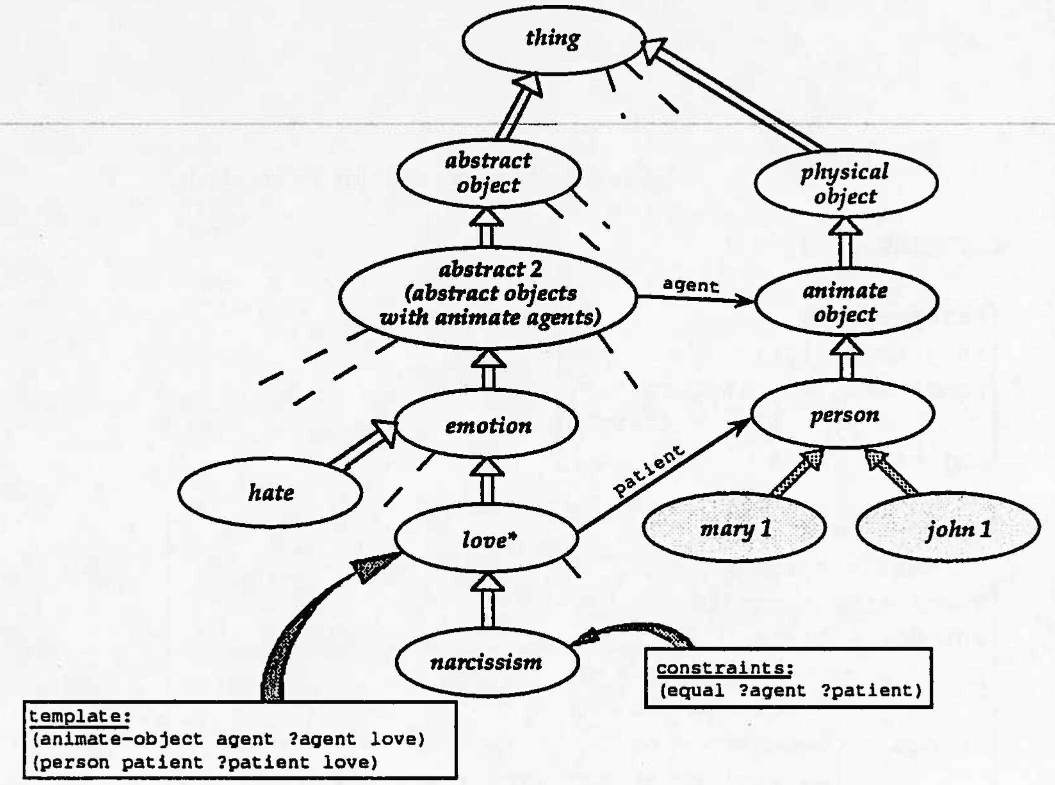


Figure 5.2: The **love** portion of a network.

An i-am-a message is sent to **mary1**, with indications that it is the **agent** slot of the **love** frame. The syntactic information for the **mary1** node is then reported.

The i-am-a information is propagated to **person**, **animate-object**, **physical-object**, and **thing**. When each of these nodes receives the i-am-a message, the information is also sent backwards along their role links (most of which are not shown in the figure) as i-have-a messages. In particular, the **animate-object** node will send the **mary1** information to the **abstract2** node, which has been magnetized to look for information from **animate-objects**. **Abstract2** relays the **mary1** information to **love** as an i-have-a message (as if there had been a direct message from **animate-object** to **love**). This information is incorporated into a **love** situation. Note that **person** would also send the **mary1** information backwards along its role links to **love**, but this message would be discarded since, according to the situation template, **love** expects the **agent** to come from **animate-object**[1].

The **john1** information is passed in the network in a similar fashion as **mary1**, except when the i-have-a message is received by **love** from **person**, the information received is incorporated (since it is from the correct origin according to the template). Once this information is received at the **love** node, a potential mapping is complete, satisfying the restrictions on the node (in this case, there are no constraints), and thus the **love** node matches the input. The syntactic information for the **love** situation is reported.

I-am-a messages would be propagated from **love** to its subsuming nodes (up the IS-A

[1]The somewhat less than intuitive roles for **love** (i.e., agent being an **animate-object** and patient being a **person**) were used here to demonstrate the matching process with an example where two roles are filled from different message origins.

Figure 5.3: LOG output for Example 1.

```
COVERING: 1.2, 1.1

[category = s                      ]
|subject = <1.1>                   |
|predicator = [category = v]       |
|             [lex = <love>]       |
[object = <1.2>                    ]

[category = s                                                      ]
|formality = informal                                              |
|transforms = passive                                              |
|subject = <1.2>                                                   |
|predicator = [category = v]                                       |
|             [lex = <be>  ]                                       |
|object = [category = np                                          ]|
|         |determiner = [category = article      ]                ||
|         |             |number = singular       |                ||
|         |             |definiteness = definite |                ||
|         |             [lex = "the"             ]                ||
|         |head = [category = n      ]                            ||
|         |       |number = singular |                            ||
|         |       [lex = "apple"     ]                            ||
|         |modifier = [category = pp                            ] ||
|         |           |prep = [category = prep]                 | ||
|         |           |       [lex = "of"     ]                 | ||
|         |           |modifier = [category = poss]             | ||
|         |           |           [head = <1.1>   ]             | ||
|         |           |head = [category = n]                    | ||
[         [           [       [lex = "eye" ]                    ] ]]

COVERING: 1.1

[category = proper]
[lex = "Mary"     ]

[category = proper              ]
|formality = formal             |
[lex = "Miss Mary McDougall"    ]

COVERING: 1.2

[category = proper]
[lex = "John"     ]

[category = proper              ]
|formality = formal             |
[lex = "Mr. John Juggernauts"   ]
```

```
COVERING: 1.2, 1.1

[category = s                                                  ]
[subject = <1.1>                                               ]
[predicator = [category = v]                                   ]
[             [lex = <be>  ]                                   ]
[complement = [category = adj                               ]  ]
[             [lex = <conceited> | <narcissistic>           ]  ]


[category = s                       ]
[formality = informal               ]
[attitude = derogatory              ]
[subject = <1.1>                    ]
[predicator = [category = v]        ]
[             [lex = <be>  ]        ]
[complement = [category = adj  ]    ]
[             [lex = <stuck-up>]    ]


[category = np                     ]
[head = [category = n          ]   ]
[       [number = singular     ]   ]
[       [lex = <narcissism>    ]   ]
[modifier = <1.1>                  ]
```

Figure 5.4: Additional LOG output for Example 2.

links), and i-have-a messages would be sent backwards along any role links to `love`, to see if the `love` situation can participate in any other situations (which in this case it cannot). The `love` node then sends a subsumer-matches message to `narcissism`, attempting to incorporate the information from the matched `love` node. However, it is found that the constraint, `(equal ?agent ?patient)`, is not met, and thus the `narcissism` node does not fully match. No more information is passed, and the matching process finishes. The output from LOG for this example would be as shown in Figure 5.3.

Example 2

Input: `(love (agent john1)(patient john1)), reference number = 2`

The matching of this input would proceed as for Example 1 above. However, when `narcissism` receives the subsumer-matches message from `love`, the constraint `(equal ?agent ?patient)` would be satisfied, and thus the `narcissism` node would be matched

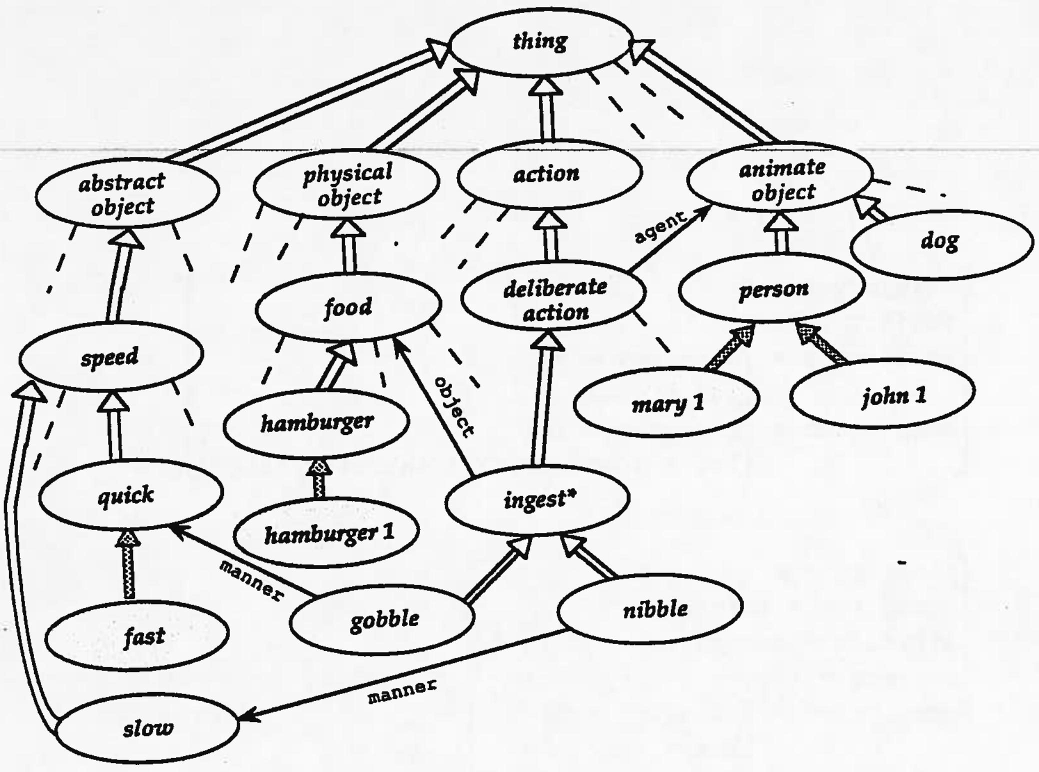


Figure 5.5: The eat portion of a network.

(and its lexical units reported). An i-am-a message would be sent from narcissism to love, but since no new parts of the input have been covered, the message would be discarded. Narcissism does not have any backward role links or down IS-As, and thus the matching is complete.

The output, in addition to that for Example 1 above (and omitting the lexical units for mary1), would be as shown in Figure 5.4. Note, in the figure, that the lexical unit information for more than one category is given (e.g., sentence information for "conceited", "narcissistic", and "stuck-up", and noun phrase information for "narcissism"). Thus, a natural language generator could choose, from the lexical options, the lexical unit with a category satisfying its current syntactic needs (cf. (Watt, 1988)).

Example 3

Input: `(ingest (agent john1)(object hamburger1)(manner fast)), reference number = 3`

Consider the portion of the network pictured in Figure 5.5. The matching process for this example would proceed in a very similar fashion as for Example 2 above, where ingest would magnetize up the IS-As to thing, looking for messages from animate-object and food. The ingest node would match, after receiving the i-have-a messages about john1 and his hamburger1, reporting such lexical options as "ingest" and "eat". A subsumer-matches message would then be sent to gobble and nibble.

The input (manner fast) would cause quick to get an i-am-a message, and thus an i-have-a message for fast would be sent to gobble, and would be incorporated into a potential mapping. When the gobble node receives the subsumer-matches message,

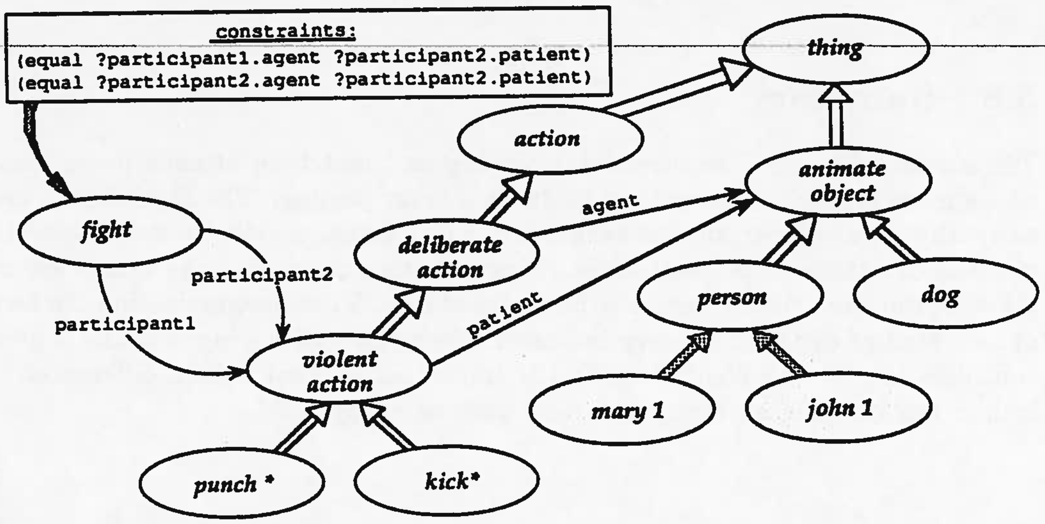


Figure 5.6: The `violent-action` portion of the network.

the `agent` and `object` information is also incorporated, and thus the node is matched. The `nibble` node would not be matched, as the `manner` slot would not be matched (the `manner` would not meet the node's situation constraints).

The output would include the lexical options "quickly" and "fast" (for `manner`), "eat" and "ingest" subcategorizing for an agent and object, and finally "gobble", which covers the entire input, subcategorizing for an agent and object as well. The lexical options "hamburger", "John", and "Mr. John Juggernauts" would be included as well. The exact output will not be shown here, as the functional descriptions are quite lengthy.

Example 4

```
Input: (punch (agent mary1)(patient john1))
       (kick (agent john1)(patient mary1))
       reference number = 4
```

Consider the part of the network shown in Figure 5.6.

The nodes punch and `kick` would match from the input in a similar fashion to the previous example, and would send an i-am-a message to `violent-action`, and thus i-have-a messages to `fight`. The input satisfies the constraints of the `fight` node (i.e., `(equal participant1.agent participant2.patient)` and `(equal participant2.agent participant1.patient)`), and thus in addition to the lexical options "punch" and "kick", the lexical option "fight" would be provided, covering both input frames (i.e., John and Mary would be the participants in the fighting).

5.8 Summary

The matching in LOG requires local binding and matching at each node, making the nodes more labour-intensive than in dumb marker passing. The algorithm is conducive to totally parallel operation at each node. LOG makes maximum use of inheritance in the network through *magnetization*. Magnetization constrains the search for matches, allowing the most "likely" nodes to be matched first. Thus, magnetization can be thought of as a kind of dynamic saliency indicator. Every part of the input frame is given *equal* consideration by the algorithm; this is important in that subtle differences in input frames can result in quite varied lexical units as output.

Chapter 6

Conclusion

6.1 Contributions

This thesis has explored the development of LOG, a Lexical Option Generator. LOG provides the different lexical options that may be used to express the same information. In line with a new interest in Computational Linguistics in generating more "natural" language, LOG provides a means by which a generator can produce a rich output without requiring an equally rich and complex internal semantic representation scheme. The finer and more complex variations of word options are stored in LOG's lexicon and found by its matcher. In addition to the syntactic information offered with a lexical option, pragmatic information such as formality level and attitudinal markings are output. Using this "mini-lexicon" output from LOG, a natural language generator has the means by which it can incorporate lexical style, and other pragmatic considerations, into its output text (cf. (DiMarco and Hirst, 1988; Watt, 1988)).

A hierarchical network structure has been proposed for the lexicon, where each node in the network represents the particular *situation* that licenses the use of the lexical units associated with the node. A template specifies the participants in the situation, and the constraints on and between those participants. Situations are arranged in the network with taxonomic, role, differentiation, restriction, and instantiation links expressing the relationships between the situations themselves. Furthermore, not only complex relationships between the slots of one frame, but also relationships between more than one frame and their slot values, can be specified in a situation.

Associated with a node may be one or more lexical units representing an equivalence class for conveying the same information. These lexical units may have internal syntactic structure for the incorporation of idiomatic constructions. This thesis has emphasized the importance of idioms to "natural" language, has surveyed their characteristics, differentiated them from other genres of figurative speech, and has indicated why they should be treated semantically as words, but syntactically as structures. Thus, *lexical unit* has been used as a term for both words and idioms.

The lexical options are found by *matching* the input frame representation of the information to be conveyed to the situations in the lexicon knowledge base. The matched situations may cover a portion, or all, of the input information to be conveyed. A survey of matching in knowledge bases indicated the characteristics of a matcher desirable for the

LOG matcher. A spreading-activation process, where each node has limited operations and where there are a limited number of types of messages sent between nodes, was selected as the matching technique. Spreading activation allows every aspect of the input to act as an index to a solution simultaneously. This attribute of spreading activation is important, since minor differences in frame and slot values in the input knowledge can result in the licensing of different sets of lexical options. *Magnetization* of nodes, a method for expediting the passing of information to nodes that are likely to need particular information, guided and constrained the search for matches.

6.2 Related Research

In this section, the lexical choice processes of some recent language generation systems are reviewed.

6.2.1 LOQUI

LOQUI (Horacek, 1987) is an English and German natural language interface to a database, concentrating on word choice in the generation process. The knowledge is arranged in a semantic network using a small set of epistemological primitives, like KL-ONE and LOG, though including quantifier links. In fact, one of the main concentrations of the system is the consideration of quantifiers in the generation process. If a node in the network is expressible, the node is linked to a discrimination network that is used to examine the particular characteristics of an instance to determine the appropriate word choice. A separate lexicon is used to provide the syntactic structures (e.g., the case frame of a verb) of the word choices. LOLA, the logic representation used to express either the meaning of an input sentence or the information that is to be conveyed to the user, has predicates corresponding to the links in the semantic network.

The input to the word choice task is a LOLA formula, where the nodes of the network are referred to by constants or variables bound by quantifiers. In general, constants and variables map to nouns, quantifier primitives to quantifiers, and property and value primitives to the verbs "have" and "is", respectively. The concepts representing an action generally have a verb associated with them. The input also includes a *dialogue move* giving semantic and discourse level information: the speech act required, the focus of the goal utterance, and an inventory of the exact variables and constants to be the content of the goal utterance. The output is the functional descriptions of the words chosen. A separate process in the generator integrates the functional descriptions and creates the surface structure.

Word selection begins with the variable in focus, specified in the dialogue move, from which either the generic counterpart, or nearest subsuming concept, is found. The discrimination net associated with this node is used to select a word to express the node on the basis of the roles or concepts associated with the variable in the input, possibly using information from the dialogue move or syntactic preferences. The discrimination net for word choice can be quite complex, specifying arbitrary combinations of which variables and constants in the content, or focus fields, of the dialogue move affect word choice. The choice may also depend on the omission of already known information. The syntactic preferences are specified by a switch indicating the current preference for a noun, verb, or adjective. The heuristic used for setting the switch in LOQUI is that the focus is preferred to be a noun, and then subsequent preferences alternate between verbs and nouns (or adjectives).

The result of the word choice task is similar to the output of LOG: functional description structures, possibly containing pointers to other structures already created or to be created later. The portions of the input covered by the functional descriptions are then marked in the input, and the process is repeated recursively for the other variables and constants from the content field of the dialogue move until all of the content field has been covered. The functional structures are then linked according to the pointers they contain.

Horacek proposes a *chunking* method for combining the information from several nodes in the network to result in a new word choice by recognizing node chains. For example, a part role, its role filler, plus the concept that has the part role, is a chain. If the concept and role filler are individuals of the same generic concept:

> ...the noun attached to this concept (if such a noun exists) is prefixed by "sub-" and the new word is looked up in the lexicon. If this lookup is successful, the linguistic verification is present (the prefixed word exists) and so is the technical verification (the system knows this word). In this case the prefixed word will be the mapping of the concept that is a part ..., the "part" role itself ..., and the linking value relation. (Horacek, 1987, page 129)

Thus, for example, "the project which is a part of a project" maps to "the subproject". It appears that each of these types of chains must be checked for explicitly. By contrast, this recognition of a particular situation is performed automatically in LOG, without a special separate process. Furthermore, the method of *deriving* words (e.g., "project" → "subproject") rather than having the exact words readily available from the network precludes finding derivative words not explicitly checked for or words for relationships that do not follow a regular derivation pattern (e.g., "*property* of a city" → "urban *property*").

As with LOG, there is a separation between the word choice stage and the integration of the choices into a sentence. LOQUI deals only superficially with style and other pragmatic issues (e.g., the focus in the dialogue move), and is not concerned with providing style and pragmatic information. Since functional descriptions are the output of LOQUI, idioms could probably be incorporated into their generator, but no such attempt is made. The importance of idioms to language is not emphasized, as it is in LOG.

6.2.2 DIOGENES

DIOGENES (Nirenburg and Nirenburg, 1988) is a distributed (blackboard control structure) natural language generator for a knowledge-based interlingual machine translation system. The system includes two knowledge sources relevent to lexical choice: a *concept lexicon* and a *generation lexicon*. The concept lexicon contains the object and event-type concepts for the particular domain of generation. The generation lexicon links the instances in the concept lexicon with the corresponding lexical units (of open-class items) of the target language. An *importance value* is associated with each of the meaning slots of the entry in the generation lexicon, indicating the saliency of the slot with respect to the entry frame. For example, the **gender** slot of the **boy** frame would have a much higher importance than its **age** slot. The entries also include syntactic information (e.g., verb-type, category, and morphological information), and the entry's synonyms, antonyms, hypernyms (e.g., boy → person), a pointer from the entry to the corresponding node in the concept lexicon (only one), and collocational information indicating the lexical units the entry usually appears with. For example, the entries for "boy" are as follows:

```
(make-frame boy
   (is-token-of (value person.CL)) ;pointer to generation lexicon
   (sex (value male)
        (importance 10))
   (age (value (2 15))
        (importance 4))
   (lexeme (value "boy"))
   (para-collocation (synonym lad kid child)
                     (antonym girl adult)
                     (hypernym person))
   (syn-collocations-in (value boy.syn))) ;pointer to syntactic collocations

(make-frame boy.syn
   (agent-of (value play throw run jump)
                     (strength 0))
   (place (value school playground ballfield)
                     (strength 0)))
```

The input to DIOGENES is a set of concept instances representing the propositional content to be conveyed, and a set of pragmatic parameter values. Each event and role instance in the input triggers the posting of the knowledge source to the blackboard. Lexical realization begins by checking if the input frame has already been mentioned, in which case a deictic or elliptical realization is attempted. Otherwise, lexical realization:

> ...consists, first of all, in scanning[1] the generation lexicon in search of a set of candidate realizations for the input frame. ...When such a set is produced, we attempt to filter it by removing those candidates that are not compatible with realizations already decided upon for other input frames in the same sentence. This processing is based on comparing the collocation information in the lexicon entries for the members of various candidate realization sets. (Nirenburg and Nirenburg, 1988, page 474)

The use of collocational information in the filtering process is in actuality a use of the *context* of the input (for the one sentence). It is unclear whether an overall context is also considered.

[1]It is unclear whether the generation lexicon is truly "scanned", or whether a more efficient method is used.

If, after the filter process is completed, only one candidate remains, then this is the result posted to the blackboard. Otherwise, a "well-defined inexact matching metric" is used to determine which of the candidates is closest to the input meaning (the "best" match). I assume that this is where the importance ranking of the slots comes into play. The best match is then posted as the result, and the process continues. Modifers are selected after the heads of their phrases have been selected.

The idea of an importance value is appealing as it allows inexact matching to occur. However, choosing the actual values (which range from 1 to 10 in DIOGENES) can be quite arbitrary, based only on the intuition of what are the most important meaning fragments of a concept. The use of collocational information to choose words in the context of the sentence is also an important feature of the system, but idioms are not handled, in contrast to LOG. The lexical units in the generation are given no internal structure, and thus idioms with variables (e.g., "x lost x's temper") could not be accommodated in a straightforward manner into the lexical selection process. Furthermore, the integration of a complex set of concepts into a situation, with constraints specified on the participants in the situation, is not attempted (e.g., "John hit Mary" and "Mary hit John" $\rightarrow$ "John and Mary fought"), thus precluding the lexical options found in this manner by LOG.

In LOG, the collocational issues dealt with in DIOGENES are handled in one of two ways. Restrictions on the adjectives or adverbs modifying a noun or verb are detailed in different situations in the lexicon. Frozen idiomatic collocation (e.g., "ladies and gentlemen", "kith and kin") is handled as an idiom, and appear together as a lexical unit, thus incorporating the generation of collocations into a single step, rather than as complex collocational considerations as in DIOGENES.

6.2.3 FIG

The Flexible Incremental Generator, FIG (Ward, 1988), is a spreading activation system designed for machine translation (a Japanese to English prototype) and cognitive modelling, based on the underlying rationale that speaking is a process of choosing one word after another. Ward takes into consideration many important issues of word choice. The knowledge, both world and linguistic, is represented in a single semantic net (as in KING (Jacobs, 1986) and PHRED (Jacobs, 1985b)). Nodes can represent concepts, words, syntactic features, syntactic constructions, and constituents of constructions. Each node has an energy level indicating the current relevance of the node at a particular point in time:

> A "relevant" word is one which could form part of the output, a "relevant" construction is one which could provide an appropriate structure to the output, and a "relevant" concept is one which is associated with the meaning to express. (Ward, 1988, page 726)

The nodes are connected by links indicating that there is an association between the nodes, possibly weighted, but the meaning of the association is not indicated.

The input to the system is a conceptualization of the information to be conveyed. Each of the input nodes supplies a source of energy to the network, to the corresponding node in the semantic network, in particular. I assume that an initial syntactic expectation is also input. The energy is propagated through the network along the links, and the energy level of a node is the sum of the energies reaching it from the other nodes. A part of the energy is consumed each time a link is crossed, thus biasing word choice to words near the input nodes, and thus possibly precluding finding some abstractions (e.g., "John hit Mary" and "Mary kicked John" $\rightarrow$ "John and Mary fought", as done in LOG) and favouring words that parallel the input structure, which of course depends on the implemented structure of the network. The most highly activated word node in the network is chosen, and emitted. The energy of the nodes representing the word emitted, the portion of the input that has been conveyed (and those that can be inferred to have been conveyed), and the constituent structures that have been completed, are all zeroed. For example, if the input nodes include *woman*, *old*, *live*, and *day*, and the syntactic expectations are activating verbs, then *live* will have the highest activation, since it receives input from both the syntactic and semantic considerations. The verb "live" will be emitted, and the node for the word "live" and the node indicating that a verb is being looked for are zeroed. The process then repeats itself until all of the input has been covered.

Ward stresses the fact that the relation between a word and the input can be complex, but his network cannot recognize relationships between participants in the meaning of a word. For example, situations in which John loves John (i.e., the agent and patient of the *love* event are equal) cannot be recognized. Therefore, the system could not emit "narcissism" as a word choice, in contrast to LOG.

FIG prefers the most specific word (i.e., the word activated from the most inputs) and this precludes stylistic and other pragmatic goals that may require less concise (more verbose) word choice. However, LOG provides the specific as well as the less specialized options.

The assumption is made in FIG that in order to make a lexical *choice*, you must know at that time what syntax is expected. However, a generator may in fact require the lexical options available to make decisions on the syntactic structure of the output, especially with respect to style (DiMarco and Hirst, 1988; Watt, 1988). Certainly syntax is required for the final choice, but the options provided by LOG may be a desirable intermediate source of knowledge.

In general, FIG's goals were much the same as LOG's. All factors of the input were to contribute simultaneously. However, incorporating syntax into the same network as world knowledge and the lexicon results in possibly an unduly complex, and difficult to maintain, system as the network is scaled up to include stylistic considerations, or even just a larger vocabulary. Furthermore, no provision for idioms and their internal structure is made. The links that would be needed for idioms in FIG, especially those with variables, would be difficult to incorporate. The premise in LOG is that syntactic and stylistic considerations are dealt with by a separate process. Equivalent choices (e.g., synonyms) are not dealt with by FIG, and thus the same choice will always be made.

6.2.4 Penman

Penman (Sondheimer et al., 1988) is a natural language generator for English text. The knowledge base is a conceptual network[2] that is shared with other tasks as well as lexical selection, and thus not all concepts have associated lexical information (as in LOG). The conceptual network is connected to the lexicon through attached data, a *word id*. A concept can point to more than one word id (possibly the different syntactic categories of the word), but a word id can be pointed to by only one concept. This restriction means that either all of the senses of one word are taken care of in one concept, or the same word can appear under different word ids. The lexical item associated with a concept is a word or string of words with no internal structure, and thus idioms with variables cannot be accommodated.

The grammatical component of the generation process, Nigel, is based on the functional systemic framework at the level of sentence generation. It is the grammar that drives the lexical selection.

The input language is based on first-order logic with restricted quantification (i.e., the set being quantified over can be restricted), equality, and such quantifiers as "there exists exactly one". The formula input is then translated into a graph notation, the *verbalization graph*, stored as PENNI assertions, and a quantifier tree. *Presentation specifications* are then created specifying what conceptual information is to be included in each clause or phrase.

The concept in the knowledge base that most exactly describes the entity in the presentation specification is found directly by the KL-TWO knowledge representation system. If this concept has an associated lexical item, and the lexical item is of the category expected by the grammar, it is chosen. Otherwise, the lexical item associated with the nearest subsuming concept that has these properties is chosen. If more than one word id is returned, a *random* choice is made, assuming that they are synonyms. In either case, the information covered by the chosen item is removed from the presentation specification, and the process continues until all of the presentation specification has been covered. Of course, each time, different grammatical expectations are implemented.

Although Penman deals with some important issues in lexical choice, it also makes some major assumptions that preclude the offering of many lexical options required for stylistic and pragmatic freedom in generation. This is in contrast to LOG. For example, the most specific lexical item covering the information in the presentation specification is chosen, due to the use of KL-TWO to retrieve concepts. Also, a predetermined packaging of the information into presentation specifications may prevent certain lexical options from being found (e.g., "John hit Mary" and "Mary hit John" → "John and Mary fought"). The grammatical decisions are made before each word is searched for, and takes into account no stylistic information (although there are plans to incorporate this in the future). It is difficult to say how Penman would allow less specific choices, unless different presentation specifications were made each time. To add other stylistic considerations on top of this, such as formality, attitude, etc., may make the choice process unwieldly.

[2]The knowledge representation tool used by Penman is NIKL, a descendent of KL-ONE (see section 2.4.2), together with KL-TWO, a hybrid knowledge representation system that uses NIKL's formal semantics to link the reasoner PENNI to NIKL.

In LOG, all these options are found at once, with stylistic and pragmatic information included. The approach taken in LOG is to provide lexical options based on semantics and to allow the natural language generator to combine the syntactic and pragmatic information provided by LOG with its own goals, however complex, to choose a final word.

The lack of internal structure of lexical items in Penman prevents the use of idioms, in contrast to LOG.

6.2.5 PAULINE

PAULINE (Hovy, 1987; Hovy, 1988), Planning And Uttering Language In Natural Environments, is a generation system concerned with pragmatic constraints. The knowledge base is a property-inheritance network using Schank's (Schank, 1975) Conceptual Dependency as its basis. The input to PAULINE is a standard case-frame language. A discrimination net for determining lexical units is attached to pragmatically neutral concepts in the network. This attached data discriminates with respect to pragmatic parameters. For example, the concept *hit* would have as its generic lexical unit "hit", and discriminate amongst "smash" and "tap" depending on the affective biases (opinions) of the speaker. Thus, the discriminations are made on the basis of pragmatic issues, changing the meaning of the input concepts when outputting lexical choices. In contrast, LOG finds the different lexical options conveying the *same* information.

PAULINE, therefore, is quite different from LOG in that it looks for finer pragmatic nuances, while LOG looks at the *meaning* of a variety of lexical units.

6.2.6 Summary

In this section, several natural language generators with a concentration on the lexical selection process have been reviewed. However, none of them provide all of the following (as does LOG):

- a variety of lexical options that convey various parts of the input, including (but not only) the most specific option,
- offering of idioms, including idioms with variables (e.g., "the apple of x's eye"),
- recognition of inter-relationships between parts of the input to provide a richer output (e.g., "John hit Mary" and "Mary punched John" $\rightarrow$ "John and Mary fought", or "John loves John" $\rightarrow$ "John is narcissistic"), and
- an emphasis on providing stylistic and pragmatic information.

6.3 Areas for Future Research

The Lexical Option Generator proposed in this thesis has dealt with a portion of the research area of the structure and processes involved in the use of lexicons for Computational Linguistics. Of course, many areas for future research remain, some of which are outlined in this section.

LOG itself should have more extensive cardinality restrictions in the situations, and further investigation is required into exactly what kinds of inter-relationships are required between the participants of a situation.

With a suitable user interface (and a larger lexicon), LOG could be used as an on-line thesaurus taking into consideration fine nuances of meaning.

To get a true picture of the lexicon, a comprehensive study of the *entire* language is required. Of course, this is a formidable task if done by hand. This has been recognized by the Computational Linguistic community, and thus an effort has been made to automate lexical information acquisition (Amsler, 1981; Alshawi, 1988; Boguraev and Briscoe, 1988; Byrd et al., 1988, for example). In particular, using this information to build, or at least aid in the building of, a LOG lexicon could give an indication of the weaknesses in LOG with respect to situation representation, and possibly in the structuring of the lexicon itself. As stated in Chapter 1, the use of grammatical morphemes (i.e., in specifying the situations for their use) has not been dealt with in this thesis — an area in definite need of future study.

Incorporating the use of context and inference is an important next step in the search for lexical options. For example, if John is married to Mary, and Mary's mother is Martha then when dealing with Martha, the lexical option "John's mother-in-law" should also be provided in contexts pertaining to John. Context would also be required for determining when metonymic lexical options are possible. An immediate intuition on the inclusion of context into LOG is that a *priming* of the network with background information would be a first step. Also, maintaining previous input in the background could maintain the ongoing context of the realized text. Of course, this presents only a naïve first attempt at the problem. At present, LOG assumes that the exact information to be conveyed is input, precluding the use of inference. For example, if John is shot, it could be assumed that the word "assassinated" is appropriate if he is found to be a political figure.

Currently, LOG outputs simple stylistic considerations, such as formality and attitude. Research needs to be done on the parameters of style and pragmatics. Also, a filter for lexical options so that the desired stylistic, pragmatic, and possibly syntactic, lexical options are provided would prove useful for generators considering these aspects (e.g., ELOQUENCE (Watt, 1988)).

LOG does not provide generalized lexical options (e.g., if Mary is a person who is a mammal, etc., only the lexical options for Mary are provided) as a means of restricting the number of lexical options provided. Further research needs to be done on when the more general terms can be provided, and on their output representation.

Finally, a logical next step would be to use the LOG lexicon in parsing and semantic interpretation. Directed spreading activation processes have been developed for parsing (Tomabechi and Tomita, 1988), and such approaches should be investigated using the LOG lexicon.

References

Alshawi, Hiyan (1988). Processing dictionary definitions with phrasal pattern hierarchies. *Computational Linguistics*, 195–202.

Amsler, Robert A. (1981). A taxonomy for English nouns and verbs. In *19th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, Stanford, California, pages 133–138.

Bobrow, A.S. and Bell, S.M. (1973). On catching on to idiomatic expressions. *Memory & Cognititon*, 1(3):343–346.

Bobrow, D.G. and Winograd, T. (1979). KRL: another perspective. *Cognitive Science*, 8:29–42.

Bobrow, David G. and Winograd, Terry (1977). An overview of KRL, a knowledge representation language. *Cognitive Science*, 1(1):3–46.

Boguraev, Bran and Briscoe, Ted (1988). Large lexicons for natural language processing: utilising the grammar coding system of LDOCE. *Computational Linguistics*, 203–218.

Boisset, Jean-Hugues Alai (1978). *Idioms as Linguistic Convention (with illustrations from French and English)*. PhD thesis, The University of Florida.

Brachman, Ronald J. (1979). On the epistemological status of semantic networks. In Findler, N.V., editor, *Associative Networks: Representation and Use of Knowledge by Computers*, Academic Press, New York, pages 3–50. [Also published in (Brachman and Levesque, 1985)].

Brachman, Ronald J. (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9:171–216.

Brachman, Ronald J. and Levesque, Hector J., editors (1985). *Readings in Knowledge Representation*. Morgan Kaufman Publishers, Inc., Los Altos, California.

Burger, Harald (1973). Idiomatik des deutschen. *Germanistische Arbeitshefte*, 16. [Unter Mitarbeit von Harald Jaksche, cited in (Strässler, 1982)].

Byrd, Roy J., Calzolari, Nicoletta, Chodorow, Martin S., Klavans, Judith L., and Neff, Mary S. (1988). Tools and methods for computational linguistics. *Computational Linguistics*, 219–240.

Chafe, Wallace L. (1968). Idiomaticity as an anomaly in the Chomskyan paradigm. *Foundations of Language*, 4:109–125.

Charniak, Eugene (1981). The case-slot identity theory. *Cognitive Science*, 5(3):285–292.

Charniak, Eugene (1983). Passing markers: a theory of contextual influence in language comprehension. *Cognitive Science*, 7(3):171–190.

Chomsky, Noam (1982). *Some Concepts and Consequences of the Theory of Government and Binding.* MIT Press, Cambridge, Massachussetts.

Cowie, A.P. and Mackin, R. (1975). *Verbs with Prepositions and Particles.* Volume 1 of *Oxford Dictionary of Current Idiomatic English*, Oxford University Press, Oxford.

Cowie, A.P. and Mackin, R. (1983). *Clause & Sentence Idioms.* Volume 2 of *Oxford Dictionary of Current Idiomatic English*, Oxford University Press, Oxford.

Cruse, D.A. (1986). *Lexical Semantics.* Cambridge University Press, Cambridge.

Cumming, Susanna (1986). *The Lexicon in Text Generation.* ISI Research Report ISI/RR-86-168, University of Southern California, Information Sciences Institute.

Di Sciullo, Anna Maria and Williams, Edwin (1987). *On the Definition of Word.* The MIT Press, Cambridge, Massachusetts.

DiMarco, Chrysanne and Hirst, Graeme (1988). Stylistic grammars in language translation. In *COLING 88, Budapest, International Conference on Computational Linguistics*, Budapest, pages 148–153.

Fahlman, Scott E. (1979). *NETL: A System for Representing and Using Real-World Knowledge.* The MIT Press, Cambridge, Massachusetts.

Falkenhainer, B., Forbus, K.D., and Gentner, D. (1986). The Structure Mapping Engine. In *Proceedings AAAI-86, Fifth National Conference in Artificial Intelligence*, pages 273–277.

Falkenhainer, B., Forbus, K.D., and Gentner, D. (1987). *The Structure-Mapping Engine: Algorithm and Examples.* Technical Report UIUCDCS-R-87-1361, Department of Computer Science, University of Illinois at Urbana-Champaign.

Fass, Dan (1986a). *Collative Semantics: A Description of the Meta5 Program.* Memoranda in Computer and Cognitive Science MCCS-86-23, Computing Research Laboratory, New Mexico State University.

Fass, Dan (1986b). *Collative Semantics: An Approach to Coherence.* Memoranda in Computer and Cognitive Science MCCS-86-56, Computing Research Laboratory, New Mexico State University.

Fass, Dan (1988). *Collative Semantics: A Semantics for Natural Language Processing.* Technical Report MCCS-88-118, Computing Research Laboratory, New Mexico State University. [The technical report version of his Doctoral Dissertation from the Department of Computer Science, University of Essex, England.].

Finin, Timothy (1980). *The Semantic Interpretation of Compound Nominals.* PhD thesis, University of Illinois.

Forster, Kenneth (1979). Levels of processing and the structure of the language processor. In Cooper, W.E. and Walker, E.C.T., editors, *Sentence processing: Psycholinguistic studies presented to Merrill Garrett*, Erlbaum, Hillsdale, New Jersey.

Fraser, Bruce (1970). Idioms within a transformational grammar. *Foundations of Language*, 6:22–42.

Garrett, Merrill F. (1975). The analysis of sentence production. In Bower, G., editor, *Psychology of learning and motivation*, Academic Press, New York.

Gibbs, Raymond and Gonzales, Gayle (1985). Syntactic frozenness in processing and remembering idioms. *Cognition*, 20:243–259.

Hayes-Roth, Frederick (1978). The role of partial and best matches in knowledge systems. In Waterman, Donald A. and Hayes-Roth, Frederick, editors, *Pattern-Directed Inference Systems*, Academic Press, New York, pages 557–574.

Hayes-Roth, F. (1979). Matching and abstraction in knowledge systems. In *Proceedings of the Symposium on Artificial Intelligence in Information Science – 1979*, American Society for Information Science, Minneapolis.

Healey, Alan (1968). English idiom. *Kivung*, 1(2):71–108.

Hendler, James A. (1986). *Issues in the Design of Marker-Passing Systems.* Technical Report TR-1636, University of Maryland, Department of Computer Science.

Hendler, James A. (1988). *Integrating Marker-Passing and Problem Solving: A Spreading-Activation Approach to Improved Choice in Planning.* Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey. [Also his Doctoral Dissertation, Brown University, Providence, Rhode Island].

Hirst, Graeme (1987). *Semantic interpretation and the resolution of ambiguity.* Cambridge University Press, Cambridge.

Hockett, Charles F. (1958). *A Course in Modern Linguistics.* Macmillan Press, New York.

Horacek, Helmut (1987). Choice of words in the generation process of a natural language interface. *Applied Artificial Intelligence*, 1:117–132.

Hovy, Eduard (1987). *Generating Natural Language Under Pragmatic Constraints.* PhD thesis, Yale University.

Hovy, Eduard (1988). Two types of planning in language generation. In *26th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (ACL '88)*, Buffalo, New York.

Jacobs, Paul Schafran (1985a). *A Knowledge Based Approach to Language Production.* Technical Report UCB/CSD 86/254, Computer Science Division, University of California.

Jacobs, Paul Schafran (1985b). PHRED: a generator for natural language interfaces. *Computational Linguistics*, 11(4):219–242.

Jacobs, Paul Schafran (1986). The KING natural language generator. In *7th European Conference on Artificial Intelligence*, Brighton Centre, pages 193–202.

Joshi, Aravind K. (1978). Some extensions of a system for inference on partial information. In Waterman, Donald A. and Hayes-Roth, Frederick, editors, *Pattern-Directed Inference Systems*, Academic Press, New York, pages 241–257.

Katz, J. and Postal, P. (1963). *Semantic interpretation of idioms and sentences containing them.* MIT Quarterly Progress Report 70, MIT.

Kline, P.J. (1981). The superiority of relative criteria in partial matching and generalization. In *Proceedings of the 7th International Joint Converence on Artificial Intelligence*, pages 296–303.

Lehnert, Wendy and Wilks, Yorick (1979). A critical perspective on KRL. *Cognitive Science*, 3:1–28.

Levinson, Stephen C. (1983). *Pragmatics.* Cambridge University Press, Cambridge.

Lipkis, Tom (1981). A KL-ONE classifier. In Schmolze, James G. and Brachman, Ronald J., editors, *Proceedings of the 1981 KL-ONE Workshop*, Fairchild Laboratory for Artificial Intelligence Research, pages 126–143. [Fairchild Technical Report No. 618, FLAIR Technical Report No. 4].

Loewenberg, Ina (1975). Identifying metaphors. *Foundations of Language*, 12:315–338.

Makkai, A. (1972). *Idiom Structure in English.* Mouton & Co., The Netherlands.

McLeod, William T. and Hanks, Patrick (1982). *The New Collins Concise Dictionary of the English Language.* William Collins Sons & Co. Ltd., London & Glasgow.

Morrow, D.G. (1986). Grammatical morphemes and conceptual structure in discourse processing. *Cognitive Science*, 10:423–455.

Newmeyer, F.J. (1974). The regularity of idiom behavior. *Lingua*, 34:327–342.

Nirenburg, Sergei and Nirenburg, Irene (1988). A framework for lexical selection in natural language generation. In *COLING 88, Budapest, International Conference on Computational Linguistics*, pages 471–475.

Ortony, A., Schallert, D., Reynolds, R., and Antos, S. (1978). Interpreting metaphors and idioms: some effects of context on comprehension. *Journal of Verbal Learning and Verbal Behavior*, 17:465–477.

Pustejovsky, James and Nirenburg, Sergei (1987). Lexical selection in the process of language generation. In *25th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, Stanford, California, pages 201–205.

Quillian, M. Ross (1966). *Semantic Memory*. PhD thesis, Carnegie Institute of Technology, Carnegie-Mellon University. [Published as Report 2, Project 8668, Bolt, Beranek, and Newman Inc., 1966.].

Rakowsky, Amy B. (1984). Processing of dual idiomatic meanings. (Unpublished Manuscript).

Rau, Lisa F. (1987). Spontaneous retrieval in a conceptual information system. In *The Ninth Annual Conference of the Cognitive Science Society*, Seattle, Washington, pages 873–886.

Sadock, J.M. (1972). Speech act idioms. In *Papers from the 8th Regional Meeting of the Chicago Linguistics Society*, CLS, pages 329–339.

Schank, R. C. (1975). *Conceptual Information Processing*. North-Holland Publishing Company, Amsterdam.

Schenk, André (1986). Idioms in the Rosetta machine system. In *11th International Conference on Computational Linguistics, Proceedings Coling 86*, pages 319–324.

Schmolze, J.G. and Lipkis, T.A. (1983). Classification in the KL-ONE knowledge representation system. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany, pages 330–332.

Sondheimer, Norman K., Cumming, Susanna, and Albano, Robert (1988). How to Realize a Concept: Lexical Selection and the Conceptual Network in Text Generation. University of Sourthern California, Information Sciences Institute (Draft).

Strässler, J. (1982). *Idioms in English: A Pragmatic Analysis*. Gunter Narr Verlag Tübingen, Germany.

Swinney, David and Cutler, Anne (1979). The access and processing of idiomatic expressions. *Journal of Verbal Learning and Verbal Behavior*, 18:523–534.

Tomabechi, Hideto and Tomita, Masaru (1988). Application of the direct memory access paradigm to natural language interfaces to knowledge-based systems. In *COLING 88, Budapest, International Conference on Computational Linguistics*.

Ward, Nigel (1988). Issues in word choice. In *COLING 88, Budapest, International Conference on Computational Linguistics*, pages 726–731.

Watt, Murray (1988). *The Realization of Natural Language with Pragmatic Effects*. Master's thesis, University of Toronto.

Weinreich, Uriel (1969). Problems in the analysis of idioms. In Puhvel, Jaan, editor, *Substance and Structure of Language*, University of California Press, Los Angeles and Berkeley.

Winograd, Terry (1983). *Syntax*. Volume 1 of *Language as a Cognitive Process*, Addison-Wesley Publishing Company, London.

Woods, W.A. (1975). What's in a link: Foundations for semantic networks. In Bobrow, D.G. and Collins, A.M., editors, *Representation and Understanding: Studies in Cognitive Science*, Academic Press, New York, pages 35–82. [Also published in (Brachman and Levesque, 1985)].