From a Children's First Dictionary to a Lexical Knowledge Base of Conceptual Graphs

by

Caroline Barrière

B.Eng. Ecole Polytechnique de Montréal, 1989

M.A.Sc. Ecole Polytechnique de Montréal, 1991

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
June 1997

APPROVAL

Name:	Caroline Barrière
Degree:	Doctor of Philosophy
Title of thesis:	From a Children's First Dictionary to a Lexical Knowledge Base of Conceptual Graphs
Examining Committee:	Lou Hafer Chair
	Dr. Frederick Popowich, Senior Supervisor
	Dr. Robert Hadley, Supervisor
	Dr. Veronica Dahl, Superviso
	Dr. Jim Delgrande, SFU Examiner
	Dr. Randy Goebel, University of Alberta External Examiner
Date Approved:	June 25th 1997

Abstract

This thesis aims at building a Lexical Knowledge Base (LKB) that will be useful to a Natural Language Processing (NLP) system by extracting information from a Machine Readable Dictionary (MRD). Our source of knowledge is the American Heritage First Dictionary[1](AHFD) which contains 1800 entries and is designed for children of age six to eight learning the structure and the basic vocabulary of their language. Using a children's dictionary allows us to restrict our vocabulary, but still work on general knowledge about day to day concepts and actions.

Our Lexical Knowledge Base contains information extracted from the AHFD and represented using the Conceptual Graph (CG) formalism. The **graph definitions** explicitly give the information contained in all the noun and verb definitions from the AHFD. Each sentence of each definition is tagged, parsed and automatically transformed into a conceptual graph. The **type hierarchy**, extracted automatically from the definitions, groups all the nouns and verbs in the dictionary into a taxonomy. **Covert categories** will be discovered among the definitions and will complement the type hierarchy in its role for establishing concept similarity. Covert categories can be thought of as concepts not associated to a dictionary entry, such as "writing instrument" or "device giving time". They allow grouping of words based on different criteria than a common hypernym, and therefore augment the space to explore for finding similarity among concepts. The **relation hierarchy** is built manually which groups into subclasses/superclasses the relations used in our CG representation of

definitions. The relations can be prepositions such as **in, on** or **with** or deeper semantic relations such as **part-of, material** or **instrument**. **Concept clusters** are constructed automatically around a trigger word to put it into a larger context. Its graph representation is joined to the graph representations of other words in the dictionary that are related to it. The set of related words forms a concept cluster and their graph representation, showing all the relations between them and other related words, is a **Concept Clustering Knowledge Graph**.

One important aspect of the thesis is the underlying thread of finding similarity through concept and graph comparison as a general way of processing information.

The ideas presented in this thesis are implemented in a system **ARC-Concept**. We present and discuss the results obtained.

Keywords: Lexical Knowledge Base, Machine Readable Dictionary, Concept Clustering, Type hierarchy, Conceptual Graphs

A mes parents,
pour tout leur amour et encouragement au fil des ans.

Acknowledgments

I thank my advisor Fred Popowich for believing in my good will to pursue my Ph.D. from a distance. I thank him for his moral support, his patience, his helpful comments and ideas on the present work but mostly for his positive approach to life.

I thank Mark Trumbo, my husband, for his encouragement, his love, and his support. I thank him also for the useful discussions we have had relating to this work and for his good review of the whole thesis.

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

Lexical knowledge is knowledge expressed by words. Words can be used in many different ways. They can be nice or mean, whispered or shouted, direct or ambiguous. They can also be said or implied. Words help us discover, interpret, and remember the world around us. They can trigger many images, feelings, situations. Words can be put together to form an infinite number of sentences, each one expressing a different meaning, a meaning that can also differ depending on the context of utterance.

The study of words in the goal of understanding their meaning and how they relate to each other is a very large and complex field in itself. Aiming to render this information usable by a computer presents an even larger problem. Researchers have tried to constrain this problem in a few ways.

Words can be taken as entities. A lot of them are looked at without investigating their meanings. The interaction between words is given through statistical measurements [42, 40]. Probabilities are involved at different steps of sentence analysis; the tagging process, the syntactic analysis and word sense disambiguation.

On the other hand, it is possible to work with a sublanguage where the number of words is limited and the sentence structures are more restricted [68]. Investigating a smaller set of words allows researchers to go deeper in their analysis and understanding of the meaning of words.

In this research, we take an approach of the second type as our interest is in representing the meaning of words. We are addressing the following problem: How

can one, from existing information, (semi-)automatically create a substantial lexical knowledge base that is useful for processing, disambiguating and understanding sentences. This is an important problem because manual construction of an LKB can be labour intensive, error-prone and less systematic than an automatic process.

Our first hypothesis is that a children's first dictionary is a good source of information to build an LKB if we focus toward NLP applications that need to understand sentences used in a non-technical, daily usage type of context.

Our second hypothesis is that the processes developed to automatically build the LKB, can also be used to augment and restructure the information contained in that LKB.

Our third hypothesis is that the LKB can be structured so that words are "defined" in terms of their relationship to other words in the dictionary.

The **American Heritage First Dictionary** (AHFD) will be our guide. It is a dictionary for young children made to give them simple explanations about a limited number of words that are commonly used in daily life. The AHFD will tell us what it knows about these words: what they mean and how they are used in different sentences. Through the definitions showing typical situations, we will learn about people and things, how they behave and interact. From that source, our goal is to build a **Lexical Knowledge Base** (LKB) that can be consulted by a natural language processing system for the basic task of sentence disambiguation and understanding. Some software tools will be developed to extract knowledge about the words in the AHFD and to test different ideas on the manipulation, disambiguation and reorganization of this knowledge.

So, our LKB will contain knowledge about the words found in the AHFD, but what does that mean exactly? What knowledge do we want our LKB to contain? Figure 1.1 shows a summary of types of knowledge relating to words as defined by the different authors who participated in the SIGLEX workshop [111]. We emphasize hereafter to what extent this research has an interest in these different types of knowledge.

We approach different types of word-related knowledge with different personal interests, and therefore investigate them to different degrees. Our main interest is in lexical knowledge. We are interested in analyzing the definitions given in the AHFD

1. Lexical Knowledge
   - What you put in the lexicon
   - Words are symbols (pointers) into the conceptual world
   - Contains the essential features of objects
   - A rich semantic structure
   - Lexical rules (changes on syntactic features based on semantic class)
   - Associative patterns between words' feature structure
   - Knowledge that can be expressed in words
2. World Knowledge
   - General inference mechanisms (abductive, deductive), commonsense reasoning
   - World Model, ontology, organization of concepts
3. Basic General Knowledge
   - What you need to talk about a topic (or understand)
4. Encyclopedic knowledge
   - a large amount of information on everything (purpose, composition, applications, usually used for, etc)
   - contains the superfluous features of objects
5. Domain-Specific Knowledge
   - knowledge on all the properties of specific objects
   - knowledge restricted to a particular domain
6. Semantic knowledge
   - Word meanings
   - Predicates
   - Sentence meanings
   - Text meaning
7. Pragmatic knowledge
   - information coming from the context
   - speakers and hearers roles and attitudes toward the discourse
8. Linguistic knowledge
   - morphology, syntax, subcategorization patterns
9. Non-lexical knowledge
   - tense and aspect

Figure 1.1: Different types of knowledge mentioned in SIGLEX

to find the meaning of words. Via a word's definition, we will find the relationships of that word to the other words in the dictionary. To do so, we need to analyze the meaning of the defining sentences and in that respect we are interested in semantic knowledge.

We will expand from lexical to world knowledge as we address the problems of concept organization and world modeling. We want our LKB to be a world model. It will be the world viewed through the eyes of our AHFD guide, with the ontology of concepts it defines. On the other hand, we do not explore general inference mechanisms (abductive, deductive), and commonsense reasoning that should be part of world knowledge [80, 102, 94]. Further research to include these reasoning mechanisms could use the LKB developed here as a starting point.

We expand in a similar way from semantic to pragmatic knowledge as we believe that words should be seen in context, and that their meaning is influenced by context. For our research and application, the context is given by the AHFD and we can look at words within that context. It can be considered a general context, a context which teaches children the way things in daily life "normally" work, but that is still in a probabilistic world, it is just one context with a high probability. We are only partially covering pragmatic knowledge as we do not work with models of the hearer and speaker [67] and belief revision systems [124].

As we are using a children's dictionary, we are looking more at basic general knowledge rather than encyclopedic knowledge, although these two types again do not form two distinct sets. The boundary between what is necessary and what is superfluous in a definition is quite fuzzy.

Some linguistic knowledge will be assumed known a priori to allow us to transform the sentences in the dictionary into a different knowledge formalism, and to allow us to extract information from these sentences and construct the knowledge base.

The ultimate goal would be a knowledge base which encompasses all kinds of knowledge, as there is no clear separation between any of them. They all influence each other and should blend in a certain way. In the present research, we emphasize lexical knowledge. Based on the definitions presented in Figure 1.1, what we call a lexical knowledge base will be a cross-over between different types of knowledge even

if we concentrate more on the lexical aspect. This LKB should tell us as much as it can about the meaning of words, so that when we encounter those words in a text, we can access the LKB to help us understand the meaning behind that text. The content of our LKB is of course influenced by the choice of our source of knowledge. As mentioned before, we chose a children's dictionary as our guide. This choice is quite unique to this work. Other researchers have chosen other places to find lexical knowledge.

One possibility is to hand-code, using a specific formalism, all we know about day to day life, about arts or communications, about social experiences, about the nature around us, about the things we find in a house, about how to behave in a restaurant, about everything! This is what the project Cyc intends to do, which is the largest enterprise in building a Knowledge Base. The group, under the direction of R.V. Guha and D.B. Lenat, started the Cyc project 10 years ago with an ambitious goal of building a knowledge base of "foundational knowledge", by which they mean:

> *...the knowledge that, for example, a high school teacher assumes students already have, before they walk into class for the first time. ... common sense notions of time, space, causality, and events; human capabilities, limitations, goals, decision-making strategies, and emotions; enough familiarity with art, literature, history, and current affairs that the teacher can freely employ common metaphors and allusions.* [70]

Another dictionary, the EDR Electronic Dictionary [100], is the result of nine years of efforts to manually develop a bilingual Japanese-English dictionary. Many other groups designing Natural Language Processing systems, probably not having 10 years to spend on their knowledge base, preferred to manage with limited knowledge, testing their systems on a small lexicon. With the increase in the amount of information that is available electronically, there has been a corresponding increased interest in building large Lexical Knowledge Bases. Machine Readable Dictionaries (MRDs) and many text corpora are widely available and NLP systems can use such information to evolve from toy prototypes to real-size systems.

We think that the dictionary is a good place for finding lexical units and their

semantic relations since this is part of the purpose of a dictionary. The definitions from a Machine Readable Dictionary can contain significant information about lexical and world knowledge. That information is often presented in an implicit way, and we aim at rendering it explicit so that an NLP system can make good use of it.

After deciding on the source of information, we have to decide on the means of representation. We opt for Conceptual Graphs for their closeness to natural language, their intuitive graphical representation, and their comparison algorithms that will allow us to compare definitions, to find common information and to join information.

We also have to decide on the structure of our LKB. This thesis is built on the hypothesis that words are defined with words. If the knowledge extracted from the dictionary is represented as a list of separate words, with no relations between them, these words are then still just isolated words with almost no meaning by themselves. With links, they form a network of interconnected words that define objects and situations.

An important part of this research will be to introduce the idea of word clusters. More precisely we should talk of concept clusters, using a simple definition of concept as being a word sense. A cluster will extend the meaning of a concept to include the meaning of other concepts that are related to it in various ways. A concept can be related to another concept, not only by being a synonym or antonym, but also by interacting with that concept in particular situations.

One important relation between concepts is the hypernym/hyponym relation. It has been given a lot of importance in recent work on knowledge extraction from dictionaries, especially looking at noun definitions [35, 41, 4, 34]. Finding the supertype (hypernym) of each noun allows the building of a type hierarchy. We will investigate the type hierarchy with the goal of expanding it, so that we can create a richer ontology showing more classes or groups of words sharing something else than the same supertype.

Here are the four interesting aspects of the Lexical Knowledge Base we are creating. We justify each one in more length hereafter.

1. The choice of a children's dictionary
2. The use of conceptual graphs

3. The expansion of the type hierarchy
4. The expansion of the meaning of a word into a cluster

1.1 Choosing a dictionary

Deciding to work with Machine Readable Dictionaries (MRDs) reflects our belief that the dictionary is a good source of lexical knowledge.

> *Still, dictionaries are the largest available repositories of organized knowledge about words, and it is only natural of computational linguists to turn to them in the hope that this knowledge can be extracted, formalized and made available to NLP systems.* [25]

Another popular source of lexical knowledge are text corpora. Both corpora and dictionaries are rich in linguistic and domain information [22, 21]. Corpora can be viewed as the raw information from which dictionaries are made. Lexicographers digest this raw information, looking for generalizations, contexts of usage, dependencies, and organize it into a dictionary. In doing so, they are making available to others information that is not easily obtained from the raw text. However, the drawback is that no two lexicographers will extract or represent the information in exactly the same way[1].

There is also information that can be extracted from corpora that is not present in dictionaries, such as frequency data, collocations [125], word cooccurrences [43, 142, 132, 31, 108] and subcategorizations [38, 30, 117]. As mentioned in [42] lexicographers tend to pay little attention to the corpora as a whole but look more for selected citations and complement their findings with introspection. But the trend is starting to change, as dictionary making starts to depend heavily (mostly in Britain as the authors note) on machine-readable corpora, for example, to find concordances using tools.

[1] For a critical analysis of dictionary making, definitions reliability and usefulness for LKB building, see [13] as well as [82].

Ideally, a natural language system should have access to as much information as possible about words to help its task of sentence understanding. The use of corpora can nicely complement the use of dictionaries [13] in building an LKB. Given the vast amount of knowledge that is contained in dictionaries (which also reflects the vast amount of human preprocessing), it seems natural to use the dictionary as a starting point.

Assuming we are looking into a general-purpose dictionary (by which we mean a non-technical dictionary), the scope of the vocabulary used in a dictionary is the same as unrestricted text. Moreover, the language used in dictionaries cannot appropriately be called a specialized language given that it does not operate in a specialized domain. At the syntactic level, the variety of constructions (if not in the definitions, certainly in the given examples) is comparable to that of textual corpora. The regularity of the language used within dictionary definitions lies in the frequent occurrences of lexical and syntactic patterns to express particular conceptual categories or semantic relations. Much research has been done on trying to extract semantic relations from dictionaries [35, 41, 92, 3, 139, 85, 58]. All has been done on adult dictionaries, which often give complex definitions, and assume a lot of implicit knowledge from the user.

In this research, we perform knowledge extraction from a children's first dictionary: **The American Heritage First Dictionary** (AHFD). For our needs, we produced an electronic version of the AHFD[2].

The AHFD is addressed to children of ages six to eight. It contains 1800 entries, and about 650 pictures. The definitions can always be interpreted independently from the pictures. The AHFD is the second dictionary of a series of four. The first is the American Heritage Picture Dictionary (AHPD) addressed to children of ages four to six. It contains about 900 words with a pictorial representation of each word. The third dictionary in the series, is the American Heritage Children's Dictionary (AHCD), for children of ages eight to eleven, which contains 37000 entries. From the nearly 2000 words learn in the AHFD, the AHCD expands into 37000 words. This does not imply that all the words in the AHCD are uniquely defined using words from

[2]Copyright ©1994 by Houghton Mifflin Company. Reproduced by permission from THE AMERICAN HERITAGE FIRST DICTIONARY.

the AHFD[3]. The child has acquired enough basic concepts and relationships between concepts from this first dictionary, to be able to expand to a much larger world, a world that specializes into multiple different domains, and for each domain there is more vocabulary to learn.

We favored the AHFD for the following reasons:

Limited size: Its limited number of entries allows us to constrain our experiments to a corpus of a manageable size.

Day to day knowledge: Although it is quite limited in size, the AHFD contains a lot of knowledge about basic daily tasks and simple world generalizations. This kind of information is useful for an LKB designed to be the source of knowledge for a Natural Language Processing (NLP) system trying to understand a non-technical conversation. This information is often not stated in an adult's dictionary because it is assumed to be known by the user.

Sentence structure: In the AHFD, all words are defined using complete simple sentences. By doing so, the AHFD does not respect the convention of other dictionaries which always define a word of a certain part of speech by a word or phrase of the same part of speech. But for the purpose of knowledge extraction, this renders the parser's task no different than parsing plain text. The simplicity of the sentences (limited length, limited number of relative clauses) leads to a more limited set of possible parse trees, and this will propagate to the next steps to limit the disambiguation.

Closed world: Almost all defined words in the AHFD use in their definition other words that are themselves defined in the AHFD. Our view of meaning sees the meaning of a word consisting of a set of schematas or situations in which that word is seen in relation to other words. All words are seen as part of an

[3]It would have been an interesting experiment, had the AHCD been in electronic form, to see how many words are actually defined using the 2000 words from AHFD. The words found form a set of X words, and then we see how many more words are defined using the words in set X, and so on. This is the idea of bootstrapping presented later in this section.

interconnected network of relations to other words. Again, words are defined with words. This is a circular process and it will be possible to close the loop if we are in a closed world system. The AHFD gives us an almost closed world since only a few words used in definitions are not present in the dictionary.

Bootstrapping: We can think of some NLP applications directed toward children that could make good use of our LKB, such as a language teaching tool, or machine translation of a children's book. But we can also think of our LKB as the starting point of a bootstrapping process. For a particular task, if the information contained in the LKB is not sufficient, we can continue to update it by acquiring more information dependent on the domain of application.

By its nature, as being simple and forming a closed-world system, the AHFD makes it easy to build a coherent LKB which can become the seed of a bootstrapping process. The LKB built from the AHFD could be at the core of a more extended LKB, acquiring new information from other dictionaries or text corpora. The LKB expands, and at each step, the new words added are defined using mostly the words in the core, and then the core becomes a bigger core on top of which new words can be added and so on. That way the core grows gradually to include more and more words.

This is a different view than the one used by the lexicographers who built the Longman Dictionary of Contemporary English (LDOCE), which is a favorite among the research community working on knowledge extraction from MRDs [27]. LDOCE contains extra semantic information in the form of codes that can be read, but its appeal comes mostly from the fact that it uses a restricted core vocabulary of about 2000 words and then the rest of the 40000 words in the dictionary are defined using only words from this core. An interesting work by Wilks *et al.* [142] presents an attempt at disambiguating by hand the core lexicon and then performing automatic analysis on the rest of the dictionary.

We state hereafter a few problems that will be encountered by researchers working with the LDOCE. One first problem of expressing a set of 40000 words by using only a subset of 2000 words is that it requires some long explanations.

leading potentially to complex sentence structures. Alternatively, we prefer an approach that builds a larger and larger set of words that can define the other words.

If we start by extracting the information from the children's first dictionary, we will have a core of about 2000 words. Then later on, we could expand to a larger core, incorporating any word from a dictionary or corpus that is expressed using the first core of words. Maybe 5000 words could be defined. Then repeating this process, maybe 10000 more words could be defined using the first 5000 and so on.

A second problematic consequence of using a small core vocabulary for a large dictionary is that it will result in a fairly flat hierarchy as we try to extract the noun taxonomy. As no intermediate terms are allowed to be defined and therefore all 40000 words are defined uniquely using the subset of 2000 words, the only possible non-leaf nodes are those 2000 words.

> *Taxonomies derived from LDOCE would be expected to have a greater percentage of leaf nodes than those from other dictionaries because of the restricted core vocabulary.* [48]

Researchers working on dictionaries and/or lexical-based systems agree on the importance of building a taxonomy of words. For LKBs, inheritance is performed along this hierarchy. A type inherits the attributes and default values of its supertype. As well, in tasks of disambiguation of texts, if a word's context does not allow us to disambiguate it, we might have to search in the taxonomy to find contextual elements surrounding a supertype or a subtype of the word. For a taxonomy to be useful it must contain many intermediate nodes between the root and the leaves of the tree. For example, a flat hierarchy in which all the nouns are subtypes of a unique root "thing" would be totally useless.

There is a third problem caused by using a core vocabulary that is more specific to our work and the work of Richardson [118]. It is actually noted in [118] who makes use of the LDOCE for his project. During the clustering process presented

Table 1.1: Noun definitions from adult's AHD and AHFD

WORD	AHD	AHFD
air	a colorless, odorless, tasteless gaseous mixture chiefly nitrogen (78%) and oxygen (21%)	Air is a gas that people breath. Air is all around us. We cannot see it, but we can feel it when the wind blows.
bear	any of various usually large mammals having a shaggy coat and a short tail.	A bear is a large animal. It has thick fur and strong claws. Many bears sleep all winter.
bird	a warm-blooded, egg-laying, feathered vertebrate with forelimbs modified to form wings.	A bird is a kind of animal. It has two wings and is covered with feathers. Robins, chickens, eagles, and ostriches are all birds. Most birds can fly.
boat	A relatively small, usually open water craft.	A boat carries people and things on the water. Boats are made of wood,metal, or plastic. Most boats have engines to make them move. Some boats have sail.
bottle	A container, usually made of glass, with a narrow neck and a mouth that can be capped.	A bottle is used to hold liquids. It is made of glass or plastic. Al got a bottle of juice at the store.
boot	A kind of shoe that covers the foot and part of the leg.	A boot is a large shoe. Boots are made of rubber or leather. Most people wear boots in the rain or snow.
brain	the portion of the central nervous system consisting of a large mass of gray nerve tissue enclosed in the skull of a vertebrate, responsible for the interpretation of sensory impulse, the coordination and control of bodily activities and the exercise of emotion and thought	The brain is a part of the body. It is inside your head. The brain makes your arms, legs, eyes and ears work. People think with their brains.

in section 3.4 we will take a particular word and look into the definitions of all the words in the AHFD that are defined using that word. The goal is to build a larger context around a word. In LDOCE, such a process is only possible for the words in the core vocabulary as they are the only words used in other words' definitions. As clustering is a very important aspect of the LKB we are building, the LDOCE is definitely not adequate for our purpose.

Naive view: The AHFD gives us a naive view on things. When we look at the AHFD and then at an adult's dictionary, we feel like the adult one is quite complicated and abstract. AHFD is made for young people learning the structure and the basic vocabulary of their language. In comparison, an adult's dictionary is more of a reference tool which assumes knowledge of a large basic vocabulary. A learners dictionary assumes a limited vocabulary but still some very sophisticated concepts. To give a feel for that statement, Table 1.1 shows a few

examples of nouns defined in the AHFD and in the adult's American Heritage Dictionary (AHD).

The complexity in the sentence structures can be seen, mostly in the entry for **brain**. One long sentence will correspond to three or four simpler sentences in the AHFD. One main difference between the AHFD and the adult's dictionary is on the emphasis given by the definition. The adult's dictionary always tries to give a noun's definition using another noun. It follows the genus/differentia model of definition and therefore tries to find a genus at the expense of getting into complicated sentence structures, as well as sometimes finding obscure nominalizations (e.g. feathered vertebrate, gaseous mixture). The AHFD tends to give simpler definitions that are more usage oriented. In the cases of **boot** and **boat**, the AHFD has information about usage that is not present in the adult's dictionary, respectively on when people wear those boots, and that boats carry people. The AHFD's definition of **bird** gives many important aspects of being a bird (except for the egg-laying), as well as examples of birds. This information seems like exactly what we would want to put in an LKB where we need to say what people usually know about birds. Notice also the sentence **Most birds can fly**. This is the kind of generalization that is really useful to have in an LKB and that would be hard to extract from multiple texts or the adult's dictionary. We will use this kind of information later on to assign certainty factors to the facts stored in the LKB.

The AHFD's naive view on things makes it a perfect source of "shallow lexical knowledge" which is argued by Dahlgren [53] to be sufficient to disambiguate and build a discourse model of a text/sentence in real time. She describes her approach of "naive semantics" as follows:

> *... all language understanding occurs in the context of some knowledge. Within a subculture there is a body of common knowledge that is shared among participants. There is a relatively shallow layer of that*

> *common knowledge ("lexical semantic knowledge") which the hearer/reader employs in discourse interpretation; this shallow knowledge is accessed as "default values" in the absence of relevant contextual information to the contrary.* [53]

Limited polysemy: The AHFD gives a limited number of senses to each word. Comparing definitions of verbs between an adult dictionary and the AHFD demonstrates an important problem often mentioned in work on knowledge extraction from dictionaries: polysemy. Some words in the AHD have more than ten senses. For example the verbs **carry** and **catch** both have thirteen senses. In comparison, these verbs have only one sense in the AHFD. In the AHFD, some words have multiple senses, but it rarely exceeds three, and never exceeds five (only a few verbs contain five senses).

This limited polysemy might not be considered an advantage in itself, but our research focuses on other aspects of lexical semantics, and therefore not having to deal with polysemous words all the time becomes an advantage. We will address the question of word sense disambiguation within our lexical acquisition and clustering processes, but it is not the main emphasis of this research.

Now, let us summarize our justification for choosing the AHFD as our source of knowledge by answering some questions presented in [141].

Sufficiency: Is the dictionary sufficient to give a strong enough knowledge base for English?

Answer: We do not believe it is sufficient, but it is a very good starting point. With all the availability of data on the Internet, there is currently a keen interest in corpora. At the Euralex'96 meeting[4], the consensus seemed to be that corpora are absolutely necessary to the dictionary making process. Therefore the use of a dictionary becomes, as mentioned before, like looking at the result of a corpora

[4]Euralex'96 is a European Conference on lexicography that brings together dictionary publishers, lexicographers and computational linguists

analysis made by lexicographers. We think that any LKB should be dynamic, meaning it should be possible to update it with more data coming from either dictionaries or texts.

Extricability: Can a set of computational procedures that operate on an MRD, without any human intervention, extract general and reliable semantic information on a large scale, and in a general format suitable for a range of subsequent NLP tasks?

Answer: The dictionary contains the same vocabulary as plain texts, and uses the same syntactic structures, therefore any method used on the dictionary should work on plain text. The advantage of the dictionary is that some of its structures are used very often and give hints for ways to extract information. This is particularly true in the AHFD where the sentences are short and often use the same structures.

Bootstrapping: Is it possible to collect an initial set of information that is required by a set of computational procedures for extracting semantic information from the sense definitions in an MRD? Or how do we extract the initial information that will enable us to analyze the rest of the dictionary?

Answer: This is an important point. We must assume some things are known before we start, but we would like to keep it to a minimum. In our approach, we do not hand-code any semantic knowledge for individual words as was done in LDOCE. We start with the following elements:

1. 1800 words, with their part-of-speech and their textual definitions;
2. Morphological rules for the tagging of words;
3. A chart parser [66] with multiple parse rules for generating the parse tree(s), as well as additional information to help the parser:
   (a) some word specific heuristics, for example, parse rules unlikely to make sense for some words (ex. **a trip can go well**, using rule [np → n n] to make a compound noun of **trip can** would be unlikely);

(b) type specific heuristics, for example, parse rules unlikely to make sense for words of a particular semantic class (ex. **he dreams every night**, using the rule [vp → vt np] to make **every night** an object of the verb dream would be unlikely, as the word **night** is a subtype of the semantic class **time** which is unlikely to be used as an object);

(c) information about verb categorizations for a few verbs (ex. verb **give** can take 2 NPs, **give John a message**);

4. Transformation rules to transform a parse tree into a Conceptual Graph (CG);

5. Knowledge of certain semantic relations (**is-a, part-of, goal, instrument**) and the defining formulas used in the definitions to express them. We need CG transformation/reduction rules to find the defining formulas at the CG level and transform them into semantic relations within the CGs;

6. Knowledge of CG combination algorithms (finding common subgraphs, performing a join of two graphs) with the help of the type hierarchy (automatically constructed from the **is-a** relations found in the dictionary) to operate on CGs, discover common knowledge and build larger structures of knowledge.

The first three items could be used by any knowledge base constructor, they allow us to analyze the sentences from which we want to extract knowledge. The last three items are particular to our design using conceptual graphs.

1.2 Conceptual graphs

The previous subsection presented our source of information, the AHFD, in which the knowledge is given by the dictionary sentences in natural language. We aim at building an LKB containing this knowledge in a more explicit form; a form that would be readable by human users and accessible by an NLP system.

A knowledge base relates to a specific universe of objects, here the set of objects defined in the AHFD. An object in the universe can be described in different ways depending on the representation formalism. For example, in a logical system an object can be defined by a list of predicates, or in a frame system by a list of attribute-values, or in a conceptual graph system by a list of relations and concepts. A knowledge representation language supports a method for specifying individuals or classes of individuals in terms of the functions and relations between them.

Any one of the numerous knowledge representation formalisms[5] could be adequate for building an LKB, given time to develop all the necessary tools to test and implement different heuristics and different manipulations on the knowledge stored.

We are using Conceptual Graphs [126, 128] in this research, as there is a large community of researchers working on CGs, developing tools that allow easy comparison of knowledge via graph matching. Comparison of knowledge will be very important to our research. Conceptual graphs present a logic-based formalism with the flexibility to express the background knowledge necessary for understanding natural language.

Here are some characteristics of Conceptual Graphs:

- Predicates and arguments from predicate logic are replaced by concepts and relations;
- Concepts are typed allowing selectional restriction; a relation's signature defines what concept types it can relate;
- Concepts allow referents to specify an individual or a set of individuals of a certain type;
- A type hierarchy can be defined on concepts;
- Different graphs can be related through a coreference link on an identical concept;
- Manipulation algorithms are defined: maximal join, graph generalization, which make use of the type hierarchy to determine concept similarity for finding common information between graphs;

[5] For an introduction to different formalisms used in Artificial Intelligence (logic, production rules, semantic networks, frame languages, parallel distributed processing), see [114]

- Quantifiers are dealt with: plural, forall, there exist, some, 9 cats, John and Mary, that man;
- Easy mapping to natural language is intended.

Figure 1.2 shows two sentences with their corresponding conceptual graph representations in graphical and linear forms. It also shows the result of a maximal common generalization and specialization performed on those graphs.

The maximal common generalization of two graphs extracts the largest generalization that they share. A generalization of a graph is a subgraph where all the concepts are identical or more general than the ones in the original graph. For example, a referent can be replaced by a general type, or a type replaced by a supertype.

If we specialize the concepts in the maximal common generalization to the most specific concept types found in the original graphs, and then add the extra information contained in each of the graphs, we build a maximal join.

1.2.1 Representing dictionary definitions

In his book *Conceptual Structures* [126], Sowa defines:

- A canonical graph (sect. 3.4) is a meaningful conceptual graph that represents a real or possible situation. It incorporates selectional restrictions imposed on permissible combinations of words.
- An abstraction (def. 3.6.1) is a canonical graph with one or more concepts designated as formal parameters.
- A type definition (def. 3.6.4) is an abstraction that introduces a new type defined with a genus containing the formal parameter connected to a graph called the differentia.
- A schemata (sect. 4.1) shows concepts and relations that are commonly associated with a particular concept type. Unlike type definitions, the relationships in a schemata are not necessary and sufficient conditions for defining that concept type.

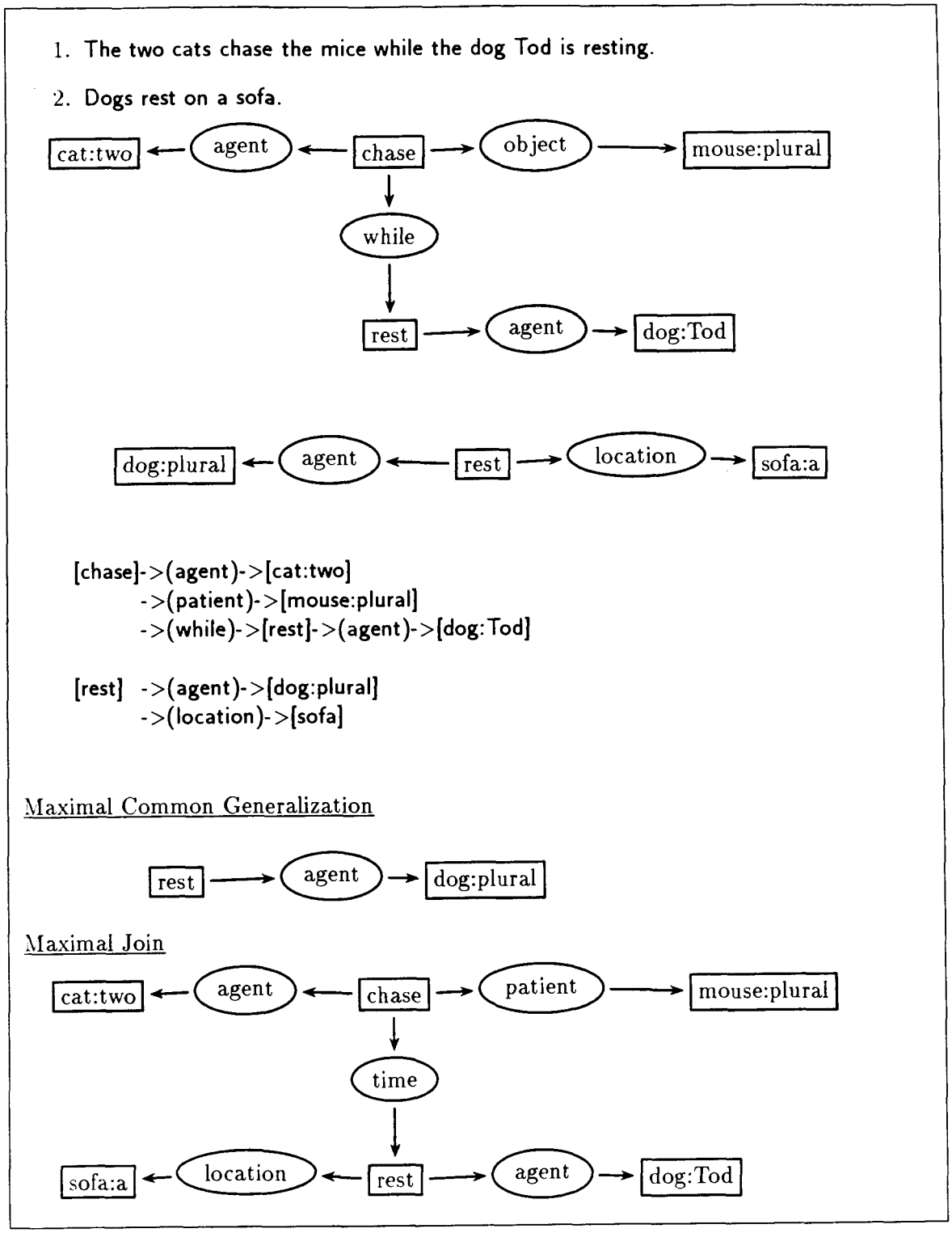


Figure 1.2: Conceptual graphs and their manipulation algorithms

The type definition at first seems to be the right structure for mapping dictionary definitions: one word, one type definition.

> *... a concept type may have at most one definition, but arbitrarily many schemata. ... Type definitions present the narrow notion of a concept, and schemata present the broad notion ... type definitions are obligatory conditions that state only the essential properties, but schemata are optional defaults that state the commonly associated accidental properties.* [126]

Unfortunately, it is more the rule than the exception to see a word in the dictionary with multiple senses. A concept type cannot correspond to a single word, nor could it correspond to a word sense since the number of word senses defined in the dictionaries is quite arbitrary, thus making us wonder what each sense really is. As well, a word sense can be defined by its usage and then its definition is closer to a schemata than to a type definition.

Furthermore, a major aspect of dictionary definitions prevents them from being type definitions: they do not always specify the necessary and sufficient conditions for defining a word. Dictionary definitions contain many inconsistencies, and different dictionaries emphasize different aspects of words. For example, the Webster's Seventh New Collegiate Dictionary (W7) and the AHD have different views on what a cuckoo is. In W7, a cuckoo is A largely grayish brown European bird noted for its habit of laying eggs in the nests of other birds for them to hatch. In AHD, a cuckoo is An old world bird with graying plumage and a characteristic two-note call.

In our work we will mainly use the term conceptual graph definition in the sense of a set of schemata used to define a word. Hence a CG definition is a group of CGs showing some aspects (description, usage) of a word, those aspects not necessarily constituting a set of necessary and sufficient conditions for defining the word.

1.3 The type hierarchy

By seeing a word's meaning as a CG definition which is a set of schemata, we emphasize the idea of meaning by usage, or meaning as the relation of a word to other

words. With respect to our hypothesis of building an LKB useful to an NLP system that will analyze texts using common language, we are more interested in knowing about a word's usage than having its detailed description.

Many dictionary definitions are written as descriptions, in the traditional way of giving the genus and differentia. In a noun definition, the genus specifies the class in which to put the noun, and the differentia specifies how different that noun is from the other nouns in the class.

Why should the genus differentia method of defining a word be the most favored one? We do not question the method itself (which goes back to Aristotle), but rather the practical use of such information in the context of building an LKB compared to other types of information that could be extracted from a dictionary.
A definition like:

> Aspirin: A white crystalline compound of acetylsalicylic acid, $C_9H_8O_4$, used to relieve pain and reduce fever.

make us certainly wonder about the value of the descriptive part for everyday knowledge (genus: **crystalline compound**) versus the explication of usage (**used to relieve pain**).

The first attempts at extracting knowledge from dictionary definitions [41, 34] were concentrating on the extraction of the genus word and the automatic construction of type hierarchies. A typical genus/differentia definition is for example: **A castle is a large building with high thick walls.** The word **building** is the head of the noun phrase, and therefore the genus. This is used in building the taxonomy, where the headword **castle** becomes a subtype of the genus **building.** This taxonomy based on the genus of the definitions, called an **is-a hierarchy**, is probably the most common one extracted from dictionaries.

Finding the genus of each definition is not always easy. The genus is not always the head of the noun phrase. There are patterns called empty heads where the genus is the word after the preposition **of**, as in <X **is a form of** Y>.

More recently, the large-scale project WordNet [23] created even more interest in the matter. WordNet is in fact much more than a large type hierarchy focusing

only on the hypernymy/hyponymy relations. The developers include a small number of relations that they believe are especially significant for the structure of the lexicon: synonymy, antonymy, hyponymy, hypernymy and three types of meronymy and holonymy. WordNet includes no definitions; instead, synsets (words having a similar meaning grouped together) are connected to other synsets by pointers representing the chosen relations.

All these relations are interesting, but they are not enough. All these -nymy relations bring together two words of the same part of speech. They all work at the paradigmatic level. However, we think it is absolutely necessary to look at the syntagmatic level, to explore the relations between the different parts of speech to find other ways to find intermediate links between words. For example, the fact that a goat lives on a farm, and a cow lives on a farm gives them a strong relation not expressed by any of the relations mentioned for WordNet.

Our goal is to find all these other relations by comparing the definitions of multiple words and finding what they have in common. We will create **covert categories**, which are categories without a label (a corresponding entry in an English dictionary), but that correspond to concepts. To take the same example as before, the concept of living on a farm does not correspond to a single English word, but it can be used to find similarity between other words.

These covert categories can be given type definitions as opposed to the actual English words which we decided to define through groups of schemata. We associate an arbitrary label (or type) to a unique conceptual graph, in which case using the label or the graph becomes totally equivalent. The graph is unique and it gives necessary and sufficient conditions for defining the label.

The increased interest in type hierarchies seems to overshadow a lot of other useful information that should be part of our concept world. The type hierarchy is often the only information used to establish the similarity (via a measure of semantic distance) between concepts. To find the semantic distance between two concepts C1 and C2, we must first find a concept C in the hierarchy which subsumes both of them. One way to establish the semantic distance is to calculate the path length going from C1 to C and from C to C2 [64]. Another way is to establish the degree of informativeness

of the concept C [116] that is based on its number of occurrences in a corpus of texts. The most frequent words are the less informative. Whatever the criteria is, it is based on a subjective structure. Type hierarchies are not absolute truths, they differ from one application to another, from one dictionary to another. Different lexicographers might define the same words using different genus, and therefore a path could be completely absent and the similarity between two concepts not found.

More fundamentally, many types of similarities are not found on the subclass/superclass axis. There are many more dimensions to this concept world. How are a clock and a watch similar? Or a pen and a pencil? Or a cardinal and a tomato? These are the dimensions we would like to capture and therefore augment the ontology so that it is not restricted to the type hierarchy.

1.4 The meaning of a word

As mentioned before, and in accordance with our third hypothesis for this thesis, we consider a word's definition as a group of schemata, giving descriptive and usage knowledge about the word. We continued to argue in section 1.3 that more than just the genus/differentia information given by single definitions should be seen as important information to be included in an LKB. Now, we go further. Why should a word's meaning be restricted to the information contained in its own definition?

Without getting into the large philosophical debate on lexical versus encyclopedic knowledge, it is important to note that, in a practical way, definitions are arbitrarily long or short, detailed or concise. As we said earlier by showing the cuckoo example, they emphasize different aspects depending on what was considered important by the lexicographer. One good way to overcome these differences is to combine information. The more information we have, the more different details we can gather, but as well, the redundancy found will augment our confidence in certain given facts. Additionally, the repetition of information can help solving anaphora present in the individual definitions. In our LKB we will build larger structures, called **concept clusters**, to better represent the meaning of a word using its interaction with other words.

When people have a conversation, or read a text about a particular subject, they

have in their mind an extended view of that subject that they can use to infer missing information in the text. For example, if one says the word "baseball" to another person, this word will trigger a group of related actions and concepts: that baseball is played with a bat and a ball (also called baseball), that many people attend major league games, that the players have to run on the bases, etc. All this information should be part of a concept cluster built around a word that we will call a trigger word. The idea is close to the idea of scripts presented in [121]. To build a particular cluster, we will use the definition of a trigger word in a dictionary and perform forward and backward searches to find related words in the dictionary and enrich the definition.

This idea of clustering is quite different from the many recent efforts in finding words that are semantically close which involve mostly statistical techniques [43, 142, 31, 108] to determine word clusters based on cooccurrences of words in text corpora or dictionaries.

Our idea of clustering is not based on finding groups of synonyms, or groups of words connected by any single relation. We try finding groups of words that help define each other, or that are used in a single situation and therefore are related to each other in different ways. As our knowledge representation is based on conceptual graphs, we will call these clusters **Concept Clustering Knowledge Graphs** (CCKGs). CCKGs give a CG representation for all the words in the cluster showing the relations among them and to other words part of the micro-domain represented.

1.4.1 Cruse's View

The research of Cruse [51] influenced many of the ideas in this dissertation, and the idea of extending a word's meaning from its definition to a cluster certainly was inspired by his work. Cruse describes how the meaning of a word is made of the meaning of other words and how it is influenced by the context:

> *The full set of normality relations which a lexical item contracts with all conceivable contexts will be referred to as its contextual relations. We shall say then, that the meaning of a word is fully reflected in its contextual*

> *relations; in fact, we can go further, and say that, for the present purposes, the meaning of a word is constituted by its contextual relations.*
>
> *We can picture the meaning of a word as a pattern of affinities and disaffinities with all the other words in the language with which it is capable of contrasting semantic relations in grammatical contexts.*
>
> *An extremely useful model of the meaning of a word, which can be extracted from the contextual relations, is one in which it is viewed as being made up, at least in part, of the meanings of other words.* [51]

All the work presented in his book is based on the idea of first defining lexical units (instead of talking of words) and then looking at many possible relations between these lexical units. A lexical unit should correspond minimally to a semantic constituent and a word.

At the syntagmatic level, Cruse considers as a semantic constituent, any constituent part of a sentence bearing a meaning that could be combine with the meaning of the other constituents to give a meaning to the sentence. He talks about words, idioms and collocations, prefixes and suffixes.

At the paradigmatic level, one lexical unit can have multiple senses. As we do not address the problem of finding idioms, a lexical unit will correspond to a single word which can have multiple senses.

Cruse introduces the interesting idea of sense modulation. A context can select a sense of a lexical unit if there are multiple senses, but a single sense can be modulated by the context as well.

Each sense of a lexical unit is made of semantic traits given by the other lexical units defining it. A semantic trait can have five statuses: criterial, expected, possible, unexpected and excluded. To modulate a sense, a context can highlight certain semantic traits of a lexical unit in a particular situation. For example, **the pregnant cat**, brings the female trait of **cat** from possible to criterial.

With our clustering process, we will see some lexical units having only one of their senses as part of a cluster which can be seen as a context. Another sense can be part of another cluster. When two senses are closely related (differing only in their part of

speech, for example "to mail" and "the mail") they might be found as interacting in the same cluster. As well, a lexical unit with a unique sense might be part of different clusters emphasizing different semantic traits of it.

The five statuses chosen for the necessity levels of semantic traits will be introduced later in our work as certainty factors on the facts given in the dictionary.

Once the lexical units are defined, Cruse introduces multiple lexical relations in which the lexical units can relate to each other. In particular he gives more information on taxonomies, meronimies, different types of opposites and synonyms. We will look into semantic relations in section 2.4.3.

1.4.2 Quillian's view

In Quillian's [113] work on semantic memories, he was trying to find the "larger word context". To him the meaning of a word was not only its definition, but also the definitions of all the words used to define it, and then all the definitions of the words used in those definitions and so on.

The problem with such a view is that the definition of a word becomes very large. There is no stopping condition, nor is there a procedure to focus the search in the right direction.

In a definition, not all words mentioned are of equal interest to pursue a search and include their definition. We will see in section 3.4.1 how we decide whether a word is *semantically significant* or not. If a word is too general, like the word **person**, it is certainly not very useful to include its definition as part of the definition of all the words that include it, which is probably half the dictionary.

In our view, a circular search is more appropriate than the expansion method of Quillian. There is information related to word X not only in the definitions of the words included in the definition of word X (forward search), but also in the definitions of the words that are defined using word X (backward search). We think of going forward then backward then forward again, then backward again, and so on, as a circular process.

We also want to ensure termination by including new words in the cluster only

if they already relate to that cluster, meaning that their definition overlaps with the larger cluster's definition. That prevents from going in all sorts of directions that give more and more information about more and more concepts. We want more information (more connections) but about a limited number of concepts.

1.5 Layout of this dissertation

The four previous sections presented four important aspects of the LKB that we aim to build: (1) the choice of children's dictionary as our source of information, (2) the choice of conceptual graphs as our representation formalism, (3) the expansion of the type hierarchy to include covert categories and (4) the formation of concept clusters around a trigger word found in the dictionary.

We want to present in the next chapters all the steps, ideas, processes, heuristics, leading to the construction of our LKB.

Chapter 2 will look at all the steps to transform one sentence found in the AHFD as part of a word's definition into a conceptual graph. The multiple steps will be presented: tagging and parsing, parse-to-CG transformations, structural disambiguation and semantic disambiguation. We will also show the construction of the type hierarchy, as we build the hierarchy automatically from the definitions. There is no interaction or links found between the graphs at this point, this will come in the next chapter.

This chapter will validate our first and second hypothesis, as the children's first dictionary will be transformed in a set of graph definitions containing general knowledge, and the partially built LKB will help process and disambiguate the graph definitions.

Chapter 3 will explore the actual construction of the LKB. We have all the individual graphs corresponding to each sentence part of all the nouns and verbs definitions. We can load them all in our environment and then experiment with them. We will explore the cluster formation, and the expansion of the type hierarchy to include covert categories.

This chapter will validate our third hypothesis as we will show the construction of clusters of words which give much information about a word through its relations

to other words. The second hypothesis is also validated as we create new structures in the LKB from existing ones. Covert categories are introduced and express some information contained in the graph definitions in a more explicit form.

Chapter 4 will demonstrate the system ARC-Concept (Acquisition, Representation and Clustering of Concepts) which is the software implementation of all the ideas presented in Chapter 2 and Chapter 3. We also present and evaluate results.

Chapter 5 will briefly explore the role of our LKB for text analysis. It will summarize the ideas presented in the thesis and give many suggestions for future research.

Chapter 2

FROM A DEFINITION TO A CONCEPTUAL GRAPH

By analyzing dictionary definitions, our goal is to find the knowledge contained in the sentences describing objects, qualities or actions, and to represent this knowledge in an explicit form that can be easily accessed, used and updated by a Natural Language Processing (NLP) system.

In the introduction chapter, we introduced our source of knowledge: the American Heritage First Dictionary (AHFD). We introduced as well our chosen representation formalism: Conceptual Graphs (CGs). In the present chapter, we take the reader through all the steps for transforming the definitions found in the AHFD into conceptual graphs (CGs).

The goal is to transform each sentence that is part of a definition into a CG, and to build a Lexical Knowledge Base (LKB) containing all these resulting CGs. The next chapter will then start from an LKB with all CG definitions and perform more operations on them to render the LKB richer in disambiguated structures and more informative to a Natural Language Processing (NLP) system.

The whole process described hereafter has two main goals: single out one graph out of the many possibilities that correspond to a sentence (structural disambiguation) and disambiguate the ambiguous information contained in the sentence (semantic disambiguation). Ambiguous information can take multiple forms: an unresolved anaphor

(a pronoun referring to some unknown noun), an unresolved semantic relation (an ambiguous preposition such as **with** that can have multiple meanings: accompaniment, instrument), or an unresolved sense (a word with multiple senses and we do not know which one is meant).

Figure 2.1 shows all the steps to transform a sentence into a single CG, and render it less and less ambiguous. We briefly present each step:

1. **Sentence to CG:** The first transformation of a sentence into a conceptual graph which contains surface semantics.

   Tagging: All words need to be tagged first.

   Parsing: Then a sentence is parsed using a chart parser that we developed for this application.

   Parse to CG: Once the parse trees are constructed, there is a set of rules to transform them into conceptual graphs containing multiple surface semantic relations.

2. **Building type hierarchy:** The type hierarchy is important for further manipulations and we make a first attempt at generating it from the graph representations we have so far by finding **is-a** relations.

3. **Structural Disambiguation:** Multiple possibilities for prepositional and conjunctional attachment are two large causes of structural ambiguity. We use a few heuristics to reduce the number of CGs. Some of these heuristics rely on the type hierarchy.

   Prepositional attachment: We use a statistical approach and another approach based on the LKB.

   Conjunctional attachment: We use an approach based on the type hierarchy.

4. **Semantic Disambiguation:** Four aspects are studied to render implicit information more explicit.

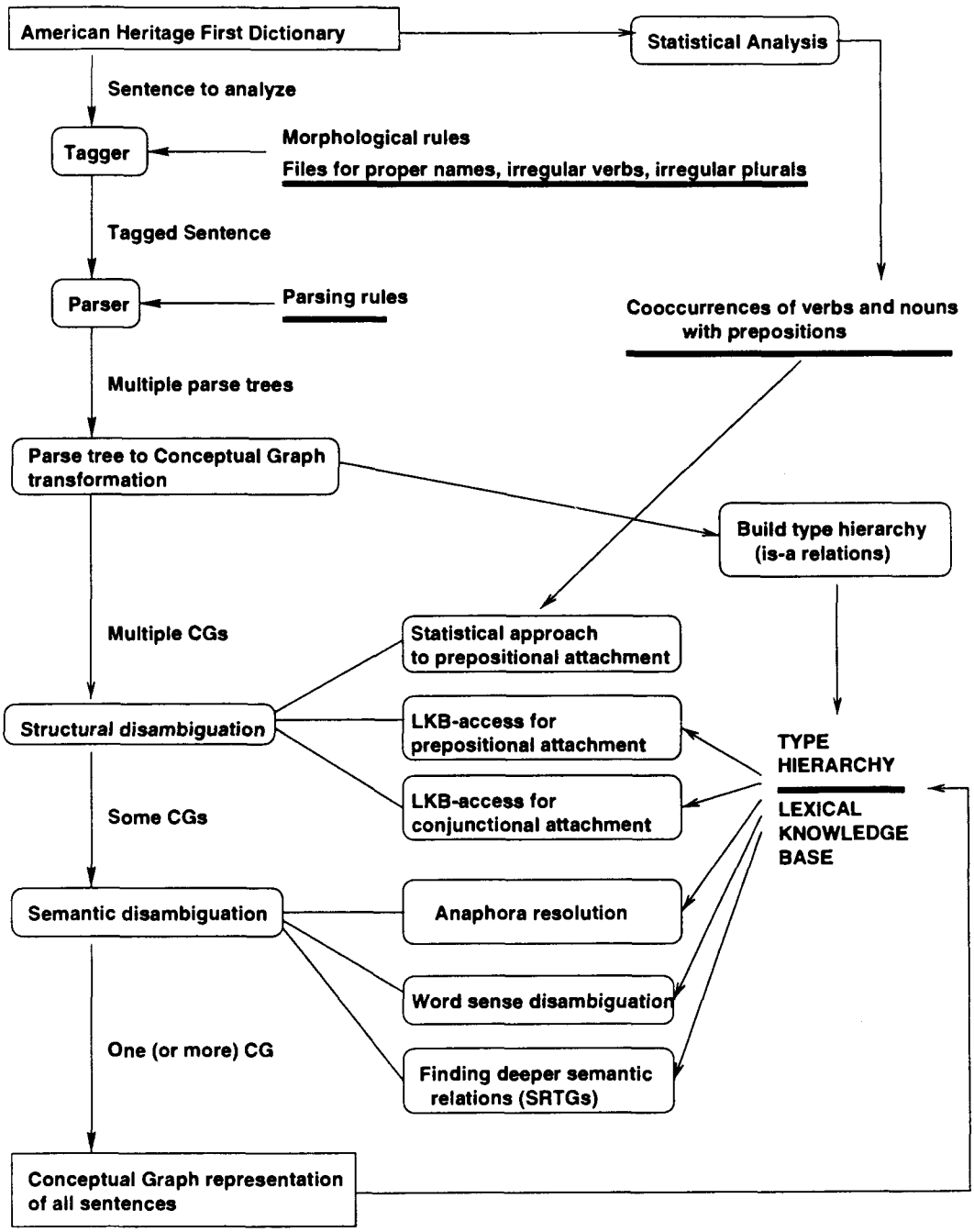


Figure 2.1: All steps from a sentence to a conceptual graph

Anaphora resolution: We use very basic anaphora resolution.

Word sense disambiguation: We try to assign the correct word sense to all the words within the graphs.

SRTGs: Semantic Relation Transformation Graphs are used to transform the surface semantic relations into deeper semantic relations.

Conjunction distributions: Some relations attached to a member of a conjunction can, in some cases, be distributed to the other members. We try simple transformations.

We try to explain afterward how all these disambiguation methods are not totally independent of each other, and that the whole disambiguation process is seen as a spiraling process. For example, at the beginning we have very little information to build the type hierarchy, and then we use it to perform other operations that might change the type hierarchy. For example, the word sense disambiguation will have modified some superclasses found (by giving the correct genus sense), and the conjunction distributions might have distributed some **is-a** relations. This affects the type hierarchy. But then, every time we modify the type hierarchy, the structural disambiguation that uses it might find better results, i.e. fewer graphs. We should therefore repeat some steps already done.

After all the preceding transformations, if we still have more than one graph per sentence, we can ask the user to manually choose one graph.

Many ideas presented in this chapter are not novel. But the exercise of performing a sentence to conceptual graph transformation allows the presentation of a good synthesis of multiple NLP issues. In some papers using CGs for NLP [61, 88], the authors assume the sentence to graph transformation is already done, but they never really explain how it is done, with what success and what effect how well the transformation is done has on subsequent results. The reader will see that a sentence to CG transformation is not an easy task. It must address many problems of text understanding such as anaphora resolution, structural disambiguation, word sense disambiguation. We are trying to understand text to transform the information it contains into an explicit representation.

For the present research, we think it is a very good way, as part of a doctoral dissertation, to get an overview of the multiple problems that computational linguists face, even if only attempting simple solutions to some problems, realizing that the door is open for a life-long exploration into the field.

When people analyze dictionaries, they usually concentrate on words that are part of the open set [51], i.e. the set of words containing the lexical roots. The closed set of words contains affixes, articles, conjunctions, prepositions, which are mostly defined by their function. We will see that the formulas or patterns discovered among definitions that lead to semantic relations are often constructed of words from the closed set. This assumes a knowledge of the closed set of words before we start our analysis of the dictionary.

In the AHFD, the part of speech of the words defined is not indicated in their definition. Our electronic version of the AHFD incorporates this information which allows us to distinguish between open set and closed set, and thus focus our analysis on nouns, verbs, adjectives and adverbs as the major constituents of the open set of words. We specifically concentrate on the definitions of nouns and verbs, as they account for three quarters of the words in the AHFD.

Their definition is usually given by a few sentences that contain up to three general types of information, as shown in the examples in Figure 2.2 (the words between brackets are our addition).

- **description:** This contains genus/differentia information. Such information is frequently used for noun taxonomy construction [34, 85, 142, 17].
- **general knowledge or usage:** This gives information useful in daily life, like how to use an object, what it is made of and what it looks like.
- **specific example:** This presents a typical situation using the word defined and it involves specific persons and actions.

An extract of the electronic version of the dictionary is shown in Figure 2.3. Details concerning the format of this dictionary are provided in Appendix A. It contains the same information as the book version of the AHFD, minus the accompanying pictures, but augmented with the information about parts of speech.

An **arm** is a part of the body. [description]
It is between the shoulder and the hand. [usage]
Amy used both **arms** to carry wood for the fireplace. [example]

Cereal is a kind of food. [description]
Many **cereals** are made from corn, wheat, or rice. [usage]
Most people eat **cereal** with milk in a bowl. [usage]

Ash is what is left after something burns. [usage]
It is a soft gray powder. [description]
Ray watched his father clean the **ashes** out of the fireplace. [example]

A **bag** is used to hold things. [usage]
It can be made of paper, plastic, cloth, or leather. [usage]
Jason brings his lunch to school in a paper **bag**. [example]

Figure 2.2: Example of definitions from AHFD

One original contribution of the research presented in the next few sections is the development of a grammar for children's simplified English that could be useful not only for this dictionary, but for other children's oriented text, and perhaps for tasks other than knowledge extraction.

2.1 Sentence to Conceptual Graph

Increasingly sophisticated analysis has been done to extract information from dictionary definitions. Chodorow *et al.* [41] performed string matching to find the genus of each definition and then build type hierarchies. Then Markowitz *et al.* [92] identified defining formulas showing patterns related to case relations. Alshawi developed the idea of using a hierarchy of phrasal patterns to identify defining formulas [5, 2]. Other researchers [24, 85, 101] prefer parsing the dictionary definition first, then do a search on the parse tree to locate defining formulas, and use some heuristics to find the words involved in the relations.

We follow the last approach. We parse a definition sentence before we transform it into a conceptual graph and then perform further steps at the graph level. Arguments

corner
1.n.
A corner is the place where two sides come together.
Squares and rectangles have four corners.

correct
1.adj.
Correct means without mistakes.
Jack's answer was correct.
2.v.
To correct means to check for mistakes in something.
The teacher corrects all our tests.

cost
1.v.
To cost means to have a price of some amount of money.
The book Susan wants costs a dollar, but she only has 50 cents.

costume
1.n.
A costume is a set of special clothes.
Stacy wore a costume in the school play.

cotton
1.n.
Cotton is soft, light, and gray.
It grows on a cotton plant.
It is made into cloth.
People wear clothes made from cotton in the summer.

could
1.aux.
Could is a form of can.
Tom can whistle.
He could whistle when he was five years old.

couldn't
1.abb.
Couldn't is a short way to say could not.
Last year A.J. couldn't read as well as Tyrone, but now he can.

Figure 2.3: Extract from electronic AHFD

in favor of the parsing method are presented in [101].

We must first tag each word, then parse the sentence and then transform the parse tree into a conceptual graph.

2.1.1 Sentence tagging

The first step in analyzing our sentence before the parse step is the tagging step. A sentence is made of tokens which can be:

1. a base word, such as a singular noun, adjective, non-conjugated verb (Appendix A shows all possible parts of speech.)

2. a different form of a word, which we can find via morphological rules (Appendix B shows all possible morphological rules.)

3. an irregular verb, an irregular plural form, a symbol or punctuation, or a proper name (Appendix B gives a list of these tokens.)

The tagging process finds a tag for most words of the dictionary definitions because almost all words in the dictionary are defined using words themselves defined in the dictionary. Table 2.1 shows 29 undefined words used in the 1800 words defined. Some of these words (interesting, jewelry, salty) might be found if we refine our morphological analysis more.

A word can also be defined as one part of speech but used in another definition in its base form as another part of speech. Table 2.2 shows the few cases where it happens.

The sentences also contain many proper names which can be identified during processing due to capitalization. The tagger interacts with the user concerning the recognition of proper names the first time an unknown capitalized word is encountered.

2.1.2 Sentence parsing

We implemented a chart parser which is a bottom up parser that works with unary and binary combination rules. For example:

Table 2.1: Words used but not defined in the AHFD

word	part-of-speech
ahead	adv
apartment	n
bus_stop	n
chain	n
classroom	n
couch	n
dad	n
flow	v
forth	adv
interesting	adj
jewelry	n
matter	v
none	pron
nor	conj
onto	prep
out_of	prep
pillow	n
press	v
respect	n
row	n
salty	adj
skirt	n
so_much	adv
someplace	pron
tab	n
thirty	adj_num
toast	n
treat	v
twelve	adj_num

Table 2.2: Words used but defined as different part-of-speech in the AHFD

word	part-of-speech used	part-of-speech defined
beat	n	v
dark	n	adj
equal	v	adj
help	n	v
human	n	adj
jump	n	v
look	n	v
open	adj	v
rise	n	v
round	adj	n
straight	adv	adj
turn	n	v
whatever	adj	pron

unary rule: vp → vi nil *verb phrase results from an intransitive verb and nil*

binary rule: vp → vt np *verb phrase results from a transitive verb and a noun phrase*

RULES

We use both a primary and a secondary set of rules which are provided in Appendix C. The parsing engine first tries parsing a sentence with the primary set of rules, which results in a parse for a large percentage of sentences. If no parse is found, then the engine adds one by one the rules from the second set of rules which are looser. For example we would expect a relative clause to start with a relative pronoun, therefore in the primary set we have:

np → np r2 *noun phrase results from noun_phrase + relative clause*

r2 → rel_pron vp *relative clause results from relative pronoun + verb_phrase*
(ex. the dog that runs)

r2 → rel_pron np_v *relative clause results from relative pronoun + incomplete*

np_v → np vt *sentence made of a noun_phrase and a transitive verb*
(ex. the cat that the dog chases)

In the secondary set we would add the following rule that skips the relative pronoun.

r2 → np_v nil **(ex. the cat the dog chases)**

The problem with looser rules like this one is that they generate a large set of wrong parses, but they are needed because they allow us to parse perfectly acceptable sentences present in the language and that should be allowed. That is why we decided to do the parsing in two steps instead of allowing all the rules to be tried at once. If only the first set of rules can be used, the number of parses is more limited, although it is often still more than one or two.

HEURISTICS

Most sentences when parsed result in more than one parse tree. Some heuristics can be used for reducing the number of parses, and we present a few hereafter. The heuristics presented are compared to the heuristics used in [81]. Unless stated otherwise, what we present is similar to their view.

Level difference: This idea is a variation on the theme of "right association" which says that postmodifiers prefer to be attached to the nearest previous possible head. Here we relaxed that preference a bit to allow not only the nearest head but sometimes the next nearest head too.

Each node in the parse tree has two children, its left child and its right child. When we build the parse tree, all the words correspond to leaves of the tree and are set at level 0. Every time we use a rule, the new node is at level n+1, where n is the highest level of both children nodes it is made from. The levels are indicated in parenthesis in the following example:

np(1) ->det(0)->the *np is at level 1, built from two level 0*

->n(0)->sky	*children*
p2(2) ->prep(0)->at	*prepositional phrase is at level 2 as*
->np(1)->"the sky"	*it is built from a preposition at level 0*
	and a noun phrase at level 1

What we call level difference is the difference in levels between the left child and the right child of a node. The subtraction goes from left to right.

If a parse is built sequentially from the last token in the sentence to the first one (as PARSE 1 shows in Figure 2.4), the level of the right child is always the highest. If a left child is at a much higher level than a right child it means we are building a large structure in the middle of the sentence and then try to attach to it a little word at the end. For example, in the sentence **A beach is an area of sand at the edge of a lake or an ocean** (definition of **beach** in the AHFD) there are many ways to decide which nodes are involved in the conjunctions and prepositions. One parse could find **ocean** and **area of sand at the edge of a lake** as members of the conjunction "or". PARSE 2 of Figure 2.4 shows that such an option would correspond to a large level difference of 5. The other possibility of having **ocean** and **lake** as members of the conjunction "or" is shown within PARSE 1 and PARSE 3 in Figure 2.4 and correspond to a level difference of -1.

A threshold on the level difference has been set experimentally to decide which parses to eliminate.

Argument versus modifier: Favoring arguments over mere modifiers would find the right interpretation of a sentence like **John bought a book from Mary.** The prepositional phrase **from Mary** is an argument of the verb **bought.** But for other examples, such as: **John borrowed a book from the library.** or **John eats a prepared dish from the freezer.** it becomes not so obvious to decide. We would like to keep both interpretations as opposed to [81]. To favor the argument is actually opposite to the right association principle, but because our level difference heuristic is looser, it allows both interpretations unless the verb's

```
PARSE 1:

s1(11)->np(1)->det->a
                n->beach
        vp(10)->vb->be
                 np(9)->np(1)->det->a
                              n->area
                        p2(8)->prep->of
                              np(7)->np(1)->n->sand
                                    p2(6)->prep->at
                                          np(5)->np(1)->det->the
                                                       n->edge
                                                p2(4)->prep->of
                                                      np(3)->np(1)->det->a
                                                                   n->lake
                                                            cnp(2)->conj->or
                                                                  np(1)->det->an
                                                                        n->ocean

PARSE 2:

s1(10)->np(1)->det->a
                n->beach
        vp(9)->vb->be
                np(8)->np(7)->np(1)->det->a
                             n->area
                             p2(6)->prep->of
                                   np(5)->np(1)->n->sand
                                         p2(4)->prep->at
                                               np(3)->np(1)->det->the
                                                            n->edge
                                                     p2(2)->prep->of
                                                           np(1)->det->a
                                                                 n->lake
                        cnp(2)->conj->or
                                np(1)->det->an
                                      n->ocean

PARSE 3:

s1(9)->np(1)->det->a
              n->beach
       vp(8)->vb->be
              np(7)->np(3)->np(1)->det->an
                                  n->area
                           p2(2)->prep->of
                                  np(1)->n->sand
                     p2(6)->prep->at
                           np(5)->np(1)->det->the
                                        n->edge
                                 p2(4)->prep->of
                                        np(3)->np(1)->det->a
                                                     n->lake
                                              cnp(2)->conj->or
                                                    np(1)->det->an
                                                          n->ocean
```

Figure 2.4: Example of parse trees showing difference in levels

direct object is modified intensively increasing the difference level to be above the set threshold. For example, we would not keep the argument interpretation in the following sentence: **John bought a book that was used by his school teacher in first grade from Mary.**

Infinitive complement versus adverbial: The complement interpretation of infinitives is favored over purpose adverbial interpretations. In the example **John wants his driver to go to Los Angeles.** the prepositional phrase **to Los Angeles** should modify **go** and not **want.** As the rule [vp → vp p2] would give the wrong interpretation, we must verify if vp does not have a child that is of type inf_vp before applying it, so to avoid structures like

```
vp -> vp -> vp
          -> inf_vp -> inf
                    -> vp
   -> p2
```

This alternate structure will be preferred:

```
vp -> vp -> vp
          -> inf_vp -> inf
                    -> vp -> vp
                          -> p2
```

Temporal prepositional phrases: The attachment of temporal prepositional phrases is reserved to verbs or event nouns. In example **John saw the President during the campaign** the prepositional phrase **during the campaign** should attach to **saw** as it expresses temporal information. We look for the preposition **during** before applying the rule [np → np p2] which would allow any prepositional phrase to modify a noun. Other prepositions that could express temporal information (such as at, in, by) can also express other types of information and are not restricted here.

Temporal noun phrases: In [81] they favor adverbial over object interpretations of temporal and measure noun phrases. In the example **John won one day in Hawaii.** the noun phrase **one day** is considered as an adverbial modifier to the verb instead of a direct object. In our opinion, the opposite interpretation seems as plausible in this case.

We could make use of the type hierarchy to see if a noun is a subtype of **time** before allowing the rule [vp → vt np]. But this is quite restrictive, as it does not allow any direct object to be a subtype of time.

We go the other way around, and for the rule [vp → vp adv_np] in which a noun phrase is considered an adverbial modifier, we only allow it if the noun is a subtype of time. So in the previous example, because **day** is a subtype of time both interpretations would be allowed. On the other hand if the sentence was **John won a kite in Hawaii.** only the direct object interpretation would be allowed.

At the beginning of our knowledge base construction, we start with a flat type hierarchy. Therefore, we do not know if any noun is a subtype of time or not. As we build the hierarchy, extracting the **is-a** relations from simple sentences, the word **day** will be put as a subtype of **time**. Then, in a second iteration of parsing we will allow new parses that were not allowed before.

Compound nominals: Temporal nouns are favored as adverbials over compound nominal heads. In the example **I saw the man Thursday.** the word **Thursday** is an adverbial and does not form a compound nominal with **man** as **man Thursday.** A reasoning similar to the previous heuristic is used in this case. We do not allow the rule [np → n n] to be used if the second noun is a temporal noun, a subtype of time.

Verb classes and alternations: We have six groups of verbs that are created at the beginning of the parsing process and the user can add verbs to these groups as he/she wishes. Each group is presented to the user through an example as follows:

1. I help/want/like her to find her shirt.
2. I made/help/watch Mary go to the store.
3. I feel/live/get better.
4. John decided/judged/saw who was right.
5. John mailed/gave Mary a letter.
6. I believe/think/say (that) Joe will win the race.

Each group allows for specific rules to be used that cannot be used by all verbs. For example, only group 5 allows for two noun phrases as direct object and indirect object [vp → vg np2], any other verb not part of group 5 would have to take the second noun phrase as an adverbial modifier. This takes care for most cases of the rule in [81] that favor predeterminers over separate noun phrases. Only the verbs in group 5 can have two separate noun phrases as arguments, in which case there is two possible interpretations and we allow both, such as in example **Send all the money.**

Only group 6 allows **that** as a complementizer which takes care of the rule in [81] favoring **that** as a complementizer rather than as a determiner. In example **I know that sugar is expensive.** the word **that** is not usually a determiner in front of **sugar**. Our heuristic favors the complementizer interpretation only in the case of certain verbs.

Even after the reducing process, there are often multiple parses left. In general, our heuristics tend to be looser than [81] as we believe that no decision is better than the wrong decision. For this reason, we keep the reduced set of possible parses and go on to the next step of transforming parse trees into conceptual graphs. We generate multiple conceptual graphs for each sentence and carry them along through more reduction and transformation processes performed at the graph level. We hope to perform structural disambiguation in a more informed way at a later stage as we will have access to some of the information from the LKB in construction.

2.1.3 From a parse tree to a Conceptual Graph

The parse trees generated during the previous steps show the syntactic level of sentence decomposition or analysis. We now want to take each parse tree and transform it into a Conceptual Graph (CG). For a first step, the syntax-semantic transformation will give us a CG containing surface semantics, that is a mix of syntactic and semantic relations, and therefore a semantic representation that is very close to the syntax. Section 2.4.3 will present graph transformations to find deeper semantic relations.

The relations first put in the CG come from two sources:

1. the set of closed class words, e.g.: **of, to, in, and**;
2. the relations that are extracted from the syntactic structure of a sentence, e.g.: **subject, object, goal, attribute, modifier.**

Figure 2.5 shows both types of relations. The set of closed class words includes prepositions, adverbs and conjunctions. We can see the prepositions **on** and **of** in sentence 1, the adverb **where** in sentence 2, and the conjunction **and** in sentence 3, all used as relations in the corresponding conceptual graphs. The second set of relations from syntactic structures can be seen by the relations **object** and **agent** in all three graphs.

The relations defined using the closed class words are often ambiguous. Example 2.1.1 shows the ambiguity generated by the preposition **of** as it expresses a relation of (1) **time**, (2) **material** and (3) **part-of.**

Example 2.1.1 -

(1) Birthday: The day or anniversary *of* a person's birth.
(2) Shelf: A flat, usually rectangular structure *of* a rigid material...
(3) Floor: The surface *of* a room on which one stands.

The ultimate goal of our translation process is to have a conceptual graph containing a restricted set of well-defined and non-ambiguous semantic relations. At this stage, we have no means of deciding what relation is meant by each preposition. By keeping the preposition itself within the graph, we delay the ambiguity resolution

process until we have gathered more information and we even hopefully avoid the decision process as the ambiguity might later be resolved by an integration process during which we combine multiple graphs.

Appendix D shows all the rules used to change a syntactic structure into one or multiple relations in the CG. The parse-to-CG rules are applied recursively until we have looked at the whole parse tree and generated a conceptual graph to represent the sentence.

Let us take the first simple sentence **A barn is a kind of building on a farm** from the example shown in Figure 2.5 to detail a few parse-to-CG rules. Under each rule there is an example.

1. **Noun with determinant:**

Syntactic Rule	CG
np → det n	[n:det]
Example:	
a barn	**[barn:a]**

It is not standard in the CG formalism to keep the determiner as a referent. We do not want to deal with quantifiers here, and therefore keep the referents as close as possible to the natural language form. In fact, because we are within a context which states generalities, facts about daily life, we assume by **a barn**, the author mean any barn, not a particular barn. This should be given by an existential quantifier, and in the CG formalism that is reflected by an empty referent field. The second sentence which says **the barn** should refer to a particular barn that was mentioned before in the text, but here it still means any barn. This is a complicated matter, and we prefer not to take any decision at this stage.

2. **Noun modified by prepositional phrase:**

Syntactic Rule	CG
np → np p2	[np]->(prep)->[np]
p2 → prep np	

WORD: BARN

A barn is a kind of building on a farm.
The barn is where the farm animals stay at night.
Farm machines and food for the animals are kept in the barns, too.

Parse for sentence 1:

```
s1(7)->np(1)->det->a
              n->barn
       vp(6)->vb->be
              np(5)->np(1)->det->a
                            n->kind
                     p2(4)->prep->of
                            np(3)->np(1)->n->building
                                   p2(2)->prep->on
                                          np(1)->det->a
                                                 n->farm
```

Conceptual Graph for sentence 1:

[barn:a]<-(agent)<-[be]->(object)->[kind:a]->(of)->[building]->(on)->[farm:a]

Parse for sentence 2:

```
s1(7)->np(1)->det->the
              n->barn
       vp(6)->vp(1)->vb->be
              quest_s1(5)->quest->where
                           s1(4)->np(2)->det->the
                                         n(1)->n->farm
                                                n->animal
                                  vp(3)->vp(1)->vi->stay
                                         p2(2)->prep->at
                                                np(1)->n->night
```

Conceptual Graph for sentence 2:

```
[place]<-(what)<-[be]->(agent)->[barn:the]
      <-(where)<-[stay]->(at)->[night]
                        ->(agent)->[animal:the]<-(modif)<-[farm]
```

Figure 2.5: Example of transformation from parse tree to CG

Parse for sentence 3:

```
s1(6)->np(5)->np(1)->n->farm
                n->machine
                cnp(4)->conj->and
                          np(3)->np(1)->n->food
                                    p2(2)->prep->for
                                              np(1)->det->the
                                                        n->animal
        vp(5)->vp(4)->vb->be
                          pp(3)->pp->keep
                                    p2(2)->prep->in
                                              np(1)->det->the
                                                        n->barn
                mod_ap(1)->pause->,
                                    adv->too
```

Conceptual Graph for sentence 3:

```
[keep]->(in)->[barn:the]
      ->(object)->[food]->(for)->[animal:the]
      ->(and)->[machine]<-(modif)<-[farm]
      <-(modif)<-[too]
```

Figure 2.5: Example of transformation from parse tree to CG (continued)

Example:	
building on a farm	[building]->(on)->[farm:a]

We keep the preposition as a relation. This is the first type of relation mentioned before, from the closed-set of words.

3. **Direct object to a verb:**

Syntactic Rule	CG
vp → vb np	[vb]->(object)->[np]
Example:	
is a kind of ...	[be]->(object)->[kind:a]->...

Transitive verbs (vt), and the verb **to be** (vb) have similar rules [vp → vb np] and [vp → vt np] to indicate the direct object.

4. **Sentence with subject:**

Syntactic Rule	CG
s1 → np vp	[vp]->(agent)->[np]
Example:	
a barn is a kind ...	[be]->(agent)->[barn:a]

Syntactic Rule	CG
s1 → np vp(passive)	[vp]->(object)->[np]
Example:	
the food is kept	[keep]->(object)->[food:the]

The subject becomes the agent of the verb if the form is active. In a passive voice it becomes the object.

This part of our work is in some ways similar to the work presented by Velardi *et al.* [133, 11]. Instead of going directly from the parse tree to a graph representation,

they have an intermediate representation as a list of *syntactic predicates*. A syntactic predicate is created from the analysis of the parse tree. For example a node with children (adj) and (np) leads to the predicate ATTRIBUTE(np,adj) and a node with children (np) and (p2) leads to the predicate NP_PP(np,prep,pp). Instead of using predicates, we directly generate the graphs **[np]->(attribute)->[adj]** and **[np]->(prep)->[np]**. For us, these syntactic predicates are the basis of our first graph constructions, before we go to the next level of transformation. It seems more appropriate to express everything with the same formalism instead of introducing another level of representation (predicates). As well, if their next step of transforming a predicate to a CG is ambiguous, it obliges them to make a choice, or to generate multiple CGs. In our case, we still have a unique CG with ambiguous relations that could be specified later on.

In the next section, we try to find deeper semantic relations and try to move away from how things were said and more toward what is the meaning of what was said.

Before we move to the next step however, one should note that it is possible to get the same CG from two parses. The redundancy due to the parse rules is usually called "spurious ambiguity". It is hard to design a perfect grammar and well beyond the scope of this thesis. Instead, we design a non_perfect grammar and remove the spurious ambiguities by finding identical graphs.

The way to find if two graphs (A and B) are identical is by doing a projection of A on B and then of B on A. If A is a subgraph of B, and B is a subgraph of A, then they are identical graphs.

2.2 Building a type hierarchy

We start with a flat hierarchy where all words are under the highest type **everything**. By using parts of speech, we can at least put nouns under **something**, verbs under **act**, and adjectives under **attribute**.

When we have words with multiple senses, they might be of different parts of speech. The type hierarchy will be transformed to have the general word under **everything** and each specific sense under **something**, **act** or **attribute** depending on its

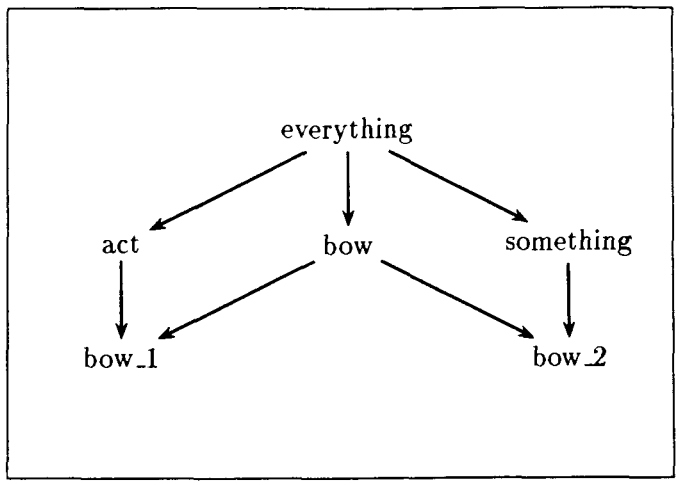


Figure 2.6: Type hierarchy modified to include word senses

part of speech.

Figure 2.6 shows an example with the word **bow** that has two senses, **to bow** and **a bow**. Both senses are put under the general term **bow**.

The hierarchy is also updated via the **is-a** relations found in all the graphs from all the definitions in the dictionary.

The (is-a) relation is only one of the multiple deeper semantic relations found via the Semantic Relation Transformation Graphs (SRTGs). These will be presented in section 2.4.3, along with a detailed analysis into the field of semantic relations.

Example 2.2.1 briefly illustrates how we extract is-a relations via SRTGs. This transformation is performed on all graphs in the LKB and then from all the is-a links found we can modify the type hierarchy.

Example 2.2.1 -

Sentence: **An ape is an animal that/with ...**
Graph representation: **[ape]<-(agent)<-[be]->(object)->[animal]<-(agent/poss) ...**
Defining formula *X1 is a X2*, typical formula to identify a is-a relation
Associated SRTG:
Before: [X1]<-(agent)<-[be]->(object)->[X2]

After: [X1]->(is-a)->[X2]

Result: **[ape]->(is-a)->[animal]**

2.3 Structural Disambiguation

We have already discussed the tagging of an input sentence, its analysis to find parse trees and the application of rules to transform the parse trees into conceptual graphs. If multiple parse trees were generated for a sentence, representing different syntactic analysis for that sentence, multiple graphs were generated and therefore the syntactic ambiguities are carried along as structural ambiguities given by different arrangement of concepts and relations in the graphs. Our first attempt at reducing the number of generated graphs was by finding identical graphs caused by spurious ambiguities.

We continue here with some graph manipulations that reduce the number of graphs by looking at different types of structural ambiguities. We look at some stand-alone and some LKB-dependent reductions. They, respectively, reach a decision without or with the help of the partially constructed LKB.

The problem with stand-alone reduction is that each graph is examined individually. This gives us only information that is within the graph, which is often not enough to make a decision about the graph itself. We must look at the larger picture. We are analyzing each graph with the goal of building a Lexical Knowledge Base. The information in the LKB can be used to disambiguate the graph we are looking at now. We must think of the whole process in terms of multiple iterations. For each word, we disambiguate as much as we can without accessing the LKB. When all words have been seen once, we can start again and disambiguate more in the second iteration as we have access to the information given by the definition of the other words.

2.3.1 Prepositions

One major contributor to structural ambiguities is the problem of prepositional attachment. When a sentence contains multiple prepositions and more than one possible

place of attachment (e.g. a verb and a noun), the number of parse trees grows exponentially with the number of prepositions.

We first describe a stand-alone method based on statistics, and then a method that uses information stored in the LKB.

Statistical method

Here we describe an heuristic that relies on a statistical measure to help us decide where a preposition should attach [77].

We calculate for all verbs and nouns, their co-occurrences with other words in all the definition sentences from the AHFD. Table 2.3 shows a few results.

Table 2.3: Examples of cooccurrences of words

word	possible following words with number of occurrences
run	fast 3, all 2, to 2, after 1, into 1, faster 1
fly	above 1, in 2, around 1, is 3, his 1, an 2, long 1, would 1, away 1
way	to 38, of 1, out_of 1, from 1, home 4, across 2
picture	of 8, on 6, or 1, that 1, can 1, with 1

To decide on the prepositional attachment, we give a global score to each graph based on its prepositions used as relations and the words involved in those relations. The ambiguity is never on what follows the preposition but on what it is attached to. In a structure like [a]->(of)->[b], we try to give a score to the combination [a]->(of), and for doing so, we look at noun and verb co-occurrence information in the data to find the noun or verb [a] followed by the word of. Each preposition in the graph is given a score depending on what concepts it is attached to. We add all the scores to give a global score to the graph. The graphs containing the highest score is kept.

Definition 2.3.1 shows part of the definition of ant for which there are two possible graph representations due to a prepositional attachment ambiguity.

Definition 2.3.1 -

ANT

Ants live in large groups in trees.

ant_1_B_A *first sense of ant (1), second sentence (B), first possibility (A)*

[live]->(agent)->[ant:plural]
->(in)->[group:plural]->(attribut)->[large]
->(in)->[tree:plural]

ant_1_B_B *first sense of ant (1), second sentence (B), second possibility (B)*

[live]->(agent)->[ant:plural]
->(in)->[group:plural]->(attribut)->[large]
->(in)->[tree:plural]

Differences:
[group:plural]->(in)->[tree:plural]
[live]->(in)->[tree:plural]

Our co-occurrence statistics give us that **live** cooccurs with **in** 28 times in the AHFD, and that **group** cooccurs with **in** only 1 time. This gives a score for ant_1_B_A of 29, where the first **in** is given 28, and the second **in** is given 1. Graph ant_1_B_B gets a score of 56, as both relations **in** are attached to **live** giving each 28. Graph ant_1_B_B is chosen.

The problem with the statistical measure is that our statistics are based on a pretty small corpus, the dictionary itself. It will help decide when the difference in score is very high (let say if we had to decide between **made of** ($occ > 100$) and **boat of** ($occ = 0$)) but it will be too risky to trust the statistics when the scores are close. To circumvent this problem we choose graph A against another graph B, if the score for graph A is at least twice the score as for graph B[1].

Graph comparison method

In section 2.1.3 we introduced the work of Velardi *et al.* [133]. Again, we present an approach similar to their work. We look into the definitions of the concepts involved

[1]There must be a penalty factor for small numbers as a score of 4 is twice a score of 2 but the difference is not significant.

in the prepositional attachment ambiguity to resolve the ambiguity [133, 61].

An example given from Velardi *et al.* [133] is the ambiguity in the phrase to produce wine in bottle. As they use syntactic predicates, they would generate two possible predicates: NP_PP(wine,in,bottle) and NP_PP(produce,in,bottle). They look into the definitions of bottle, wine and produce to find these possible predicates (they also call them triplets). From the most probable one, they generate the CG representation, that is [produce]->(object)->[wine]->(in)->[bottle].

Instead of predicates, we work directly at the CG level, and two possible parses would have generated two possible graphs:

Graph 1: [produce]->(object)->[wine]->(in)->[bottle]
Graph 2: [produce]->(object)->[wine]
->(in)->[bottle]

From these graphs, we identify subgraphs showing the differences:
[wine]->(in)->[bottle]
[produce]->(in)->[bottle]
We find the projections of these subgraphs onto the graph definitions for wine, bottle and produce to find the most probable graph representation for the phrase to produce wine in bottle.

Definition 2.3.2 shows part of the definition of drew for which there are two possible graph representations due to a prepositional attachment ambiguity.

Definition 2.3.2 -

DREW
Then he drew some squares with a blue crayon.
drew_1_B_A
[draw]->(object)->[square:plural]->(attribut)->[some]
->(with)->[crayon:a]->(attribut)->[blue]
->(agent)->[he:ref]

drew_1_B_B
[draw]->(object)->[square:plural]->(attribut)->[some]
->(with)->[crayon:a]->(attribut)->[blue]
->(agent)->[he:ref]

Differences:
sub_drew_1_B_A : [square]->(with)->[crayon]
sub_drew_1_B_B : [draw]->(with)->[crayon]

The only discrepancy between those two graphs is where the relation **with** is attached. In drew_1_B_A, it is attached to square, and in drew_1_B_B to draw.

We look into Definition 2.3.3 for square and definition 2.3.4 for crayon to find the maximal common subgraph with sub_drew_1_B_A. We look into definition 2.3.5 for draw and definition 2.3.4 for crayon to find the maximal common subgraph with sub_drew_1_B_A.

Definition 2.3.3 -

SQUARE
A square is a shape.
All four sides of a square are the same length.
square_1_A_A
[shape:a]<-(is-a)<-[square:a].
square_1_B_A
[length:the]->(attribut)->[same]
<-(is-a)<-[side:four]->(attribut)->[all]
->(of)->[square:a]

Definition 2.3.4 -

CRAYON
A crayon is a piece of colored wax.
It is used to draw and write with.
Crayons come in many colors.

crayon_1_A_A
[wax]<-(object)<-[color]
->(piece-of)->[crayon:a]
crayon_1_B_A
[crayon]<-(instrument)<-[write]
<-(instrument)<-[draw]
crayon_1_C_A
[come]->(in)->[color:plural]->(attribut)->[many]
->(agent)->[crayon:plural]

Definition 2.3.5 -

DRAW
To draw is to make a picture.
You can draw with pencils, pens, or crayons.
Abel likes to draw.
draw_1_A_A
[draw]->(equiv)->[make]->(object)->[picture:a]
draw_1_B_A
[draw]->(with)->[pencil:plural]
->(with)->[pen:plural]
->(with)->[crayon:plural]
<-(able)<-[you:ref]
draw_1_C_A
[Abel]<-(agent)<-[like]->(to)->[draw]

There is no common subgraph between sub_drew_1_A_A (square with crayon) and any of square_1_A_A, square_1_B_A, crayon_1_A_A, crayon_1_B_A or crayon_1_C_A. On the other hand, there are some common subgraphs between sub_drew_1_A_B (draw with crayon) and draw_1_B_A and crayon_1_B_A.

The maximal common subgraph between sub_drew_1_A_B and draw_1_B is:
[draw]->(with)->[crayon]

The maximal common subgraph between sub_drew_1_A_B and crayon_1_B is:
[draw]->(instrument)->[crayon]

These matches lead us to choose the graph drew_1_A_B. The second match gives us even more information by disambiguating the relation **with** to an **instrument** relation. In graph drew_1_A_B, we can replace **with** by **instrument**.

2.3.2 Conjunctions

Not only prepositions create structural ambiguities, but conjunctions as well. We look at conjunction attachment. The heuristic presented here relies on information already stored in the LKB as did the second heuristic presented for prepositional attachment.

Definition 2.3.6 shows part of the definition of jacket for which there are multiple graph representations due to conjunction attachment ambiguity.

Definition 2.3.6 -

JACKET
Jackets are good for spring and fall when the weather is cool.
jacket_1_B_A
[jacket]<-(agent)<-[be]->(attribute)->[good]->(for)->[spring]
->(and)->[fall]->(when)->[weather]...
jacket_1_B_B
[jacket]<-(agent)<-[be]->(attribute)->[good]->(for)->[spring]->(and)->[fall]
->(when)->[weather]...

To decide which concepts are part of a conjunction, we rely on the type hierarchy to give us an idea of the closeness of the items. We assume that closely related items have more chances of being joined by a conjunction. The idea is that if two words are put together in a conjunction they might have something in common.

> *the more similar two words are, the more informative will be the most specific concept that subsumes them both.* [115]

So here we have two possibilities: **be** and **fall**, or **spring** and **fall**. We look in the type hierarchy to find the lowest common superclass. This assumes that we have already done a full iteration through our dictionary and extracted from the **is-a** relations the information needed to construct our type hierarchy.

In fact, one sentence of the definition of **fall** is that **fall is a season**, which translates into the graph **[fall]->(is-a)->[season]**. The same is true for the word **spring**, **spring is a season**, which translates to the graph **[spring]->(is-a)->[season]**.

In the type hierarchy, we find **spring** and **fall** under **season**. The verbs **be** and **fall** are under a general superclass **act**. We choose the graph jacket_1_B_B as the superclass **season** is more semantically significant (or informative) than **act**. We explain in detail the idea of semantic significance in Section 3.4.1.

2.4 Semantic Disambiguation

The previous section took us through the steps of structural disambiguation. In this section, we explore four aspects of semantic disambiguation: anaphora resolution, word sense disambiguation, semantic relation disambiguation and conjunction distribution.

2.4.1 Anaphora resolution

Very simple techniques are used here to remove some pronominal referents from our graphs.

The first simple observation is that the pronoun **it** is very often used in definitions to refer to the object being defined. In a typical definition for a noun, the word defined is used in the first sentence, and then a pronoun **it** is used in the second sentence. By replacing the pronominal referent with the word being defined, the graph representation of each sentence becomes independent of all other graphs, as we eliminate the intersentential referents.

Definition 2.4.1 for the word **ash** shows the graph definitions after the anaphora

resolution.

Definition 2.4.1 -

ASH

Ash is what is left after something burns.

It is a soft gray powder.

ash_1_A_A

[ash]->(is-a)->[something]<-(object)<-[leave]

->(after)->[burn]->(agent)->[something]

ash_1_B_A

[ash]->(is-a)->[powder]->(attribute)->[soft]

->(attribute)->[gray]

In definition 2.4.1, replacing the word **it** by **ash** in the graph ash_1_B_A will have an effect on the construction of the type hierarchy.

We also process one simple type of intrasentential anaphora. We look for pronouns again, but this time not limited to **it**, but adding **they**, **she**, **he**, and **you**. Then we must find another concept in the graph that is compatible with the pronoun (gender and number, person or thing). We only resolve the referents if there is a single possibility for its anaphora.

2.4.2 Word sense disambiguation

At this point, multiple graph transformations and reductions have been done, trying to resolve structural ambiguities. Even if the number of graphs is reduced to one per sentence, there is still ambiguity, but this time within the concepts themselves. The graphs are made of concepts and relations, and so far the nouns, verbs and adjectives found in the sentences have been put directly as concepts within the graphs. But sometimes a word has multiple meanings, and we would like to identify which sense of the word is appropriate in a particular context.

We look into another type of language ambiguity: polysemy. The AHFD has a reduced polysemy problem as the number of senses given for each noun or verb is

quite minimal. We compiled that 83% of definitions of nouns and verbs contain only 1 sense, and 97% of definitions give at most 2 senses)[2] Even if the polysemy problem is not at the scale of an adult's dictionary, we address it here.

No research on knowledge extraction from MRDs gives a clear way to deal with polysemy. To disambiguate which sense of a word is used for defining another word, the decision is often based on probabilities or heuristics. For example the first sense of a word is assumed because it is the most probable [58].

For a word that has multiple senses, we use four different techniques to assign a sense to that word. We present them from the simplest to the most elaborate.

Heuristic 1: Assume word A is being defined, and it has multiple senses. When one sense of word A is defined, the word A is used as part of the defining sentences to mean the sense defined, unless word A is used twice or more in the sentence in which case we cannot make any easy assignment.

```
gas (sense 1)                    /* gas occurs once in the sense defined, gas_1 */
defining graph: [air]->(is-a)->[gas]
gas (sense 2)                    /* gas occurs twice in two different senses */
defining graph: [gas]->(group-of)->[gas]
```

Heuristic 2: For other words used in the definition, if they have different word senses corresponding to different parts of speech, they can be distinguished by looking at what types of relations they are involved in. For example,

```
[a]->(agent)->[b]                /* a must be the verb sense */
                                 /* b must be noun sense */

[a]->(attribute)->[b]            /* a must be the noun sense */
```

Each relation possesses a signature that gives what types of concepts are allowed to participate on each end of the relation. For example:

[2]Combining nouns and verbs, we have 1239 words defined, 1021 with 1 word sense, 182 with 2 word senses, 24 with 3 word senses and 12 with 4-6 word senses.

agent{Signature:2,act,something};
attribute{Signature:2,something,attribute};

By looking at the signatures of some relations we can decide whether it is the noun, verb or attribute sense of a particular word that is meant.

We also have this information about part-of-speech directly available from the parse tree, and we could keep it as we are doing the parse-to-CG transformation. By using the signatures, the process becomes independent from the parse tree. For example, if a graph was given as input, or if we construct new graphs from other graphs, we would still like to be able to perform this kind of disambiguation.

Heuristic 3: If a word X with multiple senses of the same part of speech appears in conjunction with another word Y, we find the maximal common subgraph between each sense of word X and word Y to disambiguate word X. Example 2.4.1 shows an example where the word **land** is defined using the word **earth** which has two senses but is in conjunction with word **ground** which has only one sense. Finding which sense of **earth** is most related to **ground** gives us which sense to put in the definition of **land**. By finding the maximal common subgraph between the two words involved in the conjunction, we indirectly find if they have the same superclass [39] (without looking in the type hierarchy) and we also find all other relations they have in common.

Example 2.4.1 -

LAND:	The land is the earth or ground that someone uses.
EARTH(1):	Earth means dirt.
	There is good earth in the garden for the plants to grow in.
EARTH(2):	The earth is our world.
	The earth is covered by land and oceans.
GROUND:	Ground is the earth.
	Plants grow out of the ground.

Maximal Common Subgraph between **earth(1)** *and* **ground.**

[grow]->(agent)->[plant:plural]

Number of common significant concepts: 2 (grow,plant)

Maximal Common Subgraph between earth(2) *and* ground

empty graph.

Number of common significant concepts: 0

Decision: *Assign sense 1 to earth in the definition of land.*

Heuristic 4: The most general case is when a word possesses two senses of the same part of speech and that word is not involved in any conjunction. Assuming word B is used in the definition of word A, and it has two possible senses. We try matching graph_A with graph_B_1 (sense 1) and then graph_A with graph_B_2 (sense 2) to find the largest common subgraph. By finding the pair that overlaps the most, it gives us which sense to use.

Example 2.4.2 shows an example where we disambiguate the word **bow** which has two senses and is used in the definition of the word **arrow** in its first sense.

Example 2.4.2 -

ARROW

You can shoot arrows from a bow.

arrow_1_B_A

[shoot]->(agent)->[you]

->(object)->[arrow_1]

->(from)->[bow]

BOW_1

A bow is a curved piece of wood.

A piece of string is tied to the ends of it.

Bows are used to shoot arrows.

bow_1_A_A

[bow_1]->(piece-of)->[wood]

->(attribute)->[curved]

bow_1_B_A
[piece]->(of)->[string]
<-(object)<-[tie]->(to)->[end:plural]->(of)->[it]

bow_1_C_A
[bow_1]<-(inst)<-[shoot]->(object)->[arrow]

BOW_2
A bow is also a knot made with ribbon or string.
It has two circles and two ends.
I tie my shoes with a bow.

bow_2_A_A
[bow_2]->(is-a)->[know]->(made-of)->[ribbon]
->(made-of)->[string]

bow_2_B_A
[it]<-(part-of)<-[circle:two]
<-(part-of)<-[end:two]

bow_2_C_A
[tie]->(agent)->[I]
->(with)->[bow_2]

Maximal Common Subgraph is found between arrow_1_B_A and bow_1_C_A:
[shoot]->(object)->[arrow]
Semantically significant concepts in common: shoot, arrow and bow.

2.4.3 Finding deeper semantic relations

We want to find deeper semantic relations to replace and disambiguate the surface semantic relations included at this point in our conceptual graphs.

Earlier, in section 2.1.3, we compared our work to the work of [133] in which they transformed their parse tree into a set of predicates. To continue the comparison, at this point, they go from their syntactic predicates to conceptual graphs through a process of "semantic verification" which tries to find the correct semantic relation for a predicate. For example, their predicate NP_PP(x,of,y) can be transformed into POSSESS(y,x) (eg. the book of Bill), PART-OF(y,x) (the pages of the book) or ARGUMENT(x,y) (the book of history).

We do not use predicates, but instead keep the ambiguous preposition in the conceptual graph itself. We have [x]->(of)->[y], and we need to replace the ambiguous relation **of** by either **possess**, **part-of**, or **about**. It seems more appropriate to express everything with the same formalism instead of introducing another level of representation (predicates). As well, if there is no help from the context and the transformation from a predicate to a CG is uncertain, they still have to choose a semantic relation or generate multiple CGs for multiple possibilities. With our method, we have a unique CG containing the ambiguous relation (of) until it can be specified later when we have more information about the concepts involved in the relation.

To help the disambiguation process, in [133], they assume the availability of a semantic lexicon of about 850 word-senses, for each there are 10-20 *surface semantic patterns*, which are the relations that a word-sense is involved in. Our work differs here, as they use the information stored in their lexicon to find the most probable semantic interpretation of the preposition, but we do not have access to a lexicon, as we are in the process of building it.

Working with natural language, you often find yourself tracing a circle that you have to break at some point by making assumptions. Assuming a preexistent source of lexical information to help the disambiguation of surface relations means that someone at some point had to create it by hand. Or if it was not created by hand, it had to be extracted from text, and to analyze that text it would have been useful to have

access to a source of lexical information, and we're back to starting point.

We prefer to view this circle as a spiral that expands with more and more information. We position ourselves near the starting point of that spiral by assuming no previous lexical knowledge, although we make some assumptions about the processing of sentences. As we build the lexical knowledge base, some ambiguities are left in, others are resolved by using stand-alone heuristics or by using the information from the partially constructed LKB. The ambiguities left in will eventually get resolved as we gather more and more information and as the LKB expands to include more words.

Therefore, at this point we have decided to leave the prepositions inside the graph as loosely defined relations that can be disambiguated later. Our way to show that the preposition **of** can indicate a relation of possession, part-of and argument will be through a manually established hierarchy on the relations used.

Before we establish the hierarchy connecting the closed set words to a set of semantic relations, we must investigate and choose a set of relations. We shall first look at research into semantic relations and then explain our choice of relations. Then we shall look at the implementation level where we transform the graphs obtained during the previous steps.

Choosing a set of semantic relations

Much of the work on semantic relations, from a perspective of extraction of information from a dictionary, is done via the analysis of **defining formulas**.

For a more general view on semantic relations, we refer the reader to a survey on lexical-semantic relations published in 1980 [59], that covers four different fields: anthropology, linguistics, psychology and computer science. Over fifteen years, part of the fields of linguistics, psychology and computer science have met and formed the interdisciplinary field of computational linguistics. The work on semantic relations became more stimulated by statistical corpora analysis than by the earlier methods of introspection and psychological experiments on human subjects.

Defining formulas correspond to phrasal patterns that occur often through the

dictionary suggesting particular semantic relations [3, 58]. For example, the relations **part-of, made-of, instrument** can be respectively found via the defining formulas <X1 is a part of X2>, <X1 is made of X2>, and <X1 is used to X2>.

Noun definitions are certainly the preferred vehicle to study defining formulas. The simplest defining formula leads to the most studied relation: the taxonomic relation between two nouns. The taxonomy relation is identified by looking at the genus of the noun definition. The genus which gives the class of the noun is usually the head of the noun phrase in an adult's dictionary, as for example **ape** would be defined as **Animal that/with** ... . In a children's dictionary, definitions are given by full sentences, including the noun to be defined in the sentence. Thus, the genus is normally the head of the noun phrase which follows a verb like **is** or **means**. For example, **An ape is an animal that/with** .... The defining formula <X1 is a X2> expressed in the CG formalism would be:

[X1]<-(agent)<-[be]->(object)->[X2]

leading to a taxonomic relation between X1 and X2 that could be expressed by the graph:

[X1]->(is-a)->[X2]

A quite frequent category of noun definitions has been called **empty heads** [41, 34]. They have the structure <X1 is a X2 of X3> where X2 does not give any strong semantic information, but primarily information concerning a relationship. The most frequent structure is X2=kind, <X1 is a kind of X3> that identifies again the **is-a** or taxonomic relation. There are many other empty heads that identify other types of relations. They are the key for the investigation into the sets of possible relations and have been analyzed by various researchers.

In [103], they chose manually 41 "function nouns" (what we just called empty heads) and grouped them into 8 link types: **is-a, part-of, member-set, action, state, amount, degree, form**, thereby reducing 41 syntactic structures into 8 semantic relations.

In [85] they ran a concordance program to find a list of words which occur before **of** in the structure <X1 of X2> among the 7 first words of the definition. They ran

Table 2.4: Formulas of type <N1 of N2>

N1 of	AHFD
kind of	164
part of	85
amount of	34
piece of	34
group of	25
month of	12
day of	8
set of	3
side of	3
foot of	2
form of	2
lot of	2
name of	2
path of	2

their test on the Webster's Seventh New Collegiate Dictionary (W7) and on LDOCE. We ran a similar test on the 1117 nouns in our dictionary, and we show in Table 2.4 the structures occurring more than once among the noun definitions.

Most X1 found by Klavans *et al.* that are present in the W7 are not present at all in the AHFD, such as **any of, state of, act of, one of, process of, instance of, member of, unit of, condition of, branch of, action of, system of, series of, practice of, period of, study of, use of, mass of, portion of, means of, lack of, number of.** Many of those words are not defined themselves in the AHFD, and therefore are not used to define other words. Among those, **act, state, process, instance, system** are quite abstract nouns not present in the child's world.

The AHFD's size is too small to analyze the semantic significance of the X1 which are part of patterns that occur only once compared to those occurring more than once. In [85] on a basis of about 70 000 words, they noticed that in patterns <X1 is a X2 of X3> where the X2 occurs only once in that position, the X2 has enough semantic value for the defined word to be a subclass of X2 and X3, and inherit from both genuses.

Our observation was that the N1s that were most frequent were most likely

Table 2.5: Noun definitions analysis from *Vossen_et_al89*

Type of kernel	description	Example
Link	hyperonym of entry word synonym of the entry word	cocktail: a mixed alcoholic drink abattoir: slaughterhouse
Linker	<N1 of N2> N1 giving a relationship	bourgeois: a member of the middle class
Link/Linker	<N1 of N2> N1 and N2 semantically significant	detonation: the noise of an explosion
Shunter	<N1 of VP> nominalization, where info in the VP <N1 rel_pron VP> noun is an argument of VP	adornment: the act of adorning camper: a person who camps
Link/Shunter	<N1 of VP> <N1 rel_pron VP> N1 and VP are semantically significant	angling: the sport of catching fish ... adjective: a word which describes ...
Nominalization	<N1 of N2> N1 represents an action	advent: the coming of Christ to the world

> *to signal special relationships that we would be interested in extracting for our lexical knowledge base; least frequent N1s were more likely to be in a IS-A relation with the genus term.* [85]

Another interesting work, is the study by [139]. They differentiate multiple types of definitions based on the properties of the syntactic kernel, the head of the noun phrase. Table 2.5 summarizes their work. The Link/Linker and Link/Shunter types have been added to reflect some comments given in their paper on intermediate types.

A kernel of type Link gives directly the genus of the word defined. A kernel of type Linker is what we called empty head before. It links the word defined to another word via a semantic relation. Sometimes the kernel could be a link and a linker, and we added the class Link/Linker to show kernels that give important information and could be considered the genus, but they also link to another noun phrase that could be a second genus. The Shunter kernel signals a change in part of speech by giving a very vague noun and relating it to a precise verb. We added the type Link/Shunter for Shunter types in which the noun is not that vague, and could be considered as an interesting genus as well as accentuating the verb it relates the noun to.

The same remark made in [85] about the frequency of occurrence of a kernel being an indication of a relationship is made in [139]. The real Linkers and Shunters have a high frequency. The Link/Linker kernels are the ones with a small number of

occurrences where both N1 and N2 can become superclasses of the word defined.

The Linker kernels are present in the AHFD (they have been shown in Table 2.4), and they lead to particular semantic relations. On the other hand, the Shunter kernels <N1 of VP> are not present. This syntactic structure is not used as a way of defining words in the AHFD. The nominalizations are in fact replaced by the verbs themselves, as in

AHFD :	**Attention is listening with care.**
Adult dictionary:	**Attention: the state of careful listening.**

On the other hand the Shunter kernels using a relative pronoun <N1 rel_pron VP> are sometimes used in the AHFD and correspond to definitions that we call outsiders in later chapters. Both types of Shunter kernels in fact lead to a relation between a noun and a verb, either the noun is surrounding the verb (state of VP, act of VP) or plays a case role for that verb (a person who VP).

An important work on relating defining formulas to semantic relations is by [3] who have built from the dictionary W7, a relation lexicon extracted by looking at defining formulas in the dictionary. Some of the noun relations and formulas they extracted are: AMOUNT (amount of), PART (a branch of, a portion of, a part of), SET (class of), METHOD (means of).

The work by [101, 58] also involves extracting semantic relations through the use of defining formulas. Their first step involves parsing the definitions of LDOCE entries using a broad-coverage grammar of English. The resulting parse structures are then subjected to a set of heuristic rules whose goal is to identify the occurrence of syntactic and lexical patterns which are consistently associated with some specific semantic relations, as instrument and location. They recognize about 25 relation types for verbs and nouns, some of which are: location, part-of, purpose, hypernym, time, (typical) subject, (typical) object, instrument, food, human, location, made-of, caused-by, causes, measure, means.

Another example of a set of semantic relations is the Appendix B of [126] which presents 37 relations used throughout the examples in the book. This set overlaps with the set in [101], and overlaps with the set in [3], but they are all slightly different,

giving different number of relations.

So who is right and who is wrong? Who has the magic set?

In our opinion, nobody is right and nobody is wrong. The number of relations that someone could include in their LKB is simply arbitrary. Each group works with a different dictionary (in Sowa's case he works from made up examples) and then finds different examples that they try to fit into their model. The model grows and adjusts as more examples are seen.

The number of relations might be arbitrary, but there needs to be structure among these relations. Therefore the important point we are trying to make is the necessity to structure relations into a hierarchy. The comparison among different relation sets would then resemble the comparison between the taxonomy of different languages. Some word exists in language A but not in language B where it might be replaced by a superclass word.

For now, we introduce our set of relations, and in the next chapter in which we show all parts of the LKB, we will show the relation hierarchy in section 3.3.

Now we introduce our set of relations which contains a different number of relations than the sets used in the works mentioned earlier. We have 51+ relations. The + stands for all the words in the closed set that we allow as relations but that will hopefully get changed into other semantic relation. We have conjunctions (and, or, pause/comma), prepositions (in, at, with, ...), and some adverbs (where, when, how). The size of our dictionary is too small to base our choice of relations solely on the analysis of the frequency of defining formulas, therefore, as we work with conceptual graphs we decided to start from the set of relations proposed by Sowa for conceptual graphs [126], and expand to add more relations, some based on defining formulas, others based on working with examples. Working from a set of defining formulas extracted from a corpus different from our own might not be appropriate. For example, the language used in an adult's dictionary (the W7 or LDOCE used by [3, 85]) is quite different than the language used in the AHFD.

In this research, we will often compare the CGs constructed to each other (as seen in Chapter 3). For comparison purposes, the more general the relations the more commonality exists between graphs, but we also risk generating meaningless

results. On the other hand if the relations are too specific, it will be hard to find common subgraphs between graphs. We need some kind of trade-off here. And again, establishing a hierarchy on the set of relations will be the key to solve this problem. It will allow us to compare graphs with more or less specific relations and decide which relations can be subsumed by others.

The relations presented fall into two main classes: objects or situations.

OBJECTS: We have in this group the relations found between objects, as well as the relation between a unique object and its properties or parts.

1. **part/whole relations:** This class looks at objects that can be segmented into a number of functional parts, or into smaller segments.
   Relations: **part-of, piece-of, area-of, amount-of, content**
2. **member/set relations:** This class contains all the relations of quantity of objects, whether we have none, one or many of the same or different types.
   Relations: **set-of, element-of**
3. **human/animal relations:** Some relations only apply to living entities having the capacity for decision, perception and feeling.
   Relations: **child-of, possession, home-for**
4. **comparison relations:** This class contains the relations for comparing the physical properties of two objects.
   Relations: **like, more-than, as, less-than**
5. **spatial relations:** The relations for comparing two objects with regard to their location.
   Relations: multiple prepositions such as **on, in, above, behind**
6. **word relations:** The relations of synonymy, opposite and taxonomy have been analyzed in great detail. To some extent, these relations are context independent. They are not part of an object or a situation, they are relations on the concepts that words represent instead of the physical entities

themselves. We could say they are intensional relations instead of extensional ones.

Relations: **opposite, synonymy, taxonomy**

7. **description relations:** This class gives the value of different attributes of objects through some formulas that describe the objects.

   Relations: **name, attribute, material, function, about**

SITUATIONS: This group deals with situations instead of objects, it therefore relates actions to participants, location, time. As well, it classifies the situations themselves as being states or events depending on the time they take to accomplish in comparison to the surrounding events.

1. **action modifiers:** General adverbial modifiers not yet classified as more precise

   Relations: **modif**

2. **case-role relations:** This is the largest class, it contains all the relations that can be subordinate to a verb.

   Relations: **instrument, method, manner, agent, experiencer, location, object, recipient, result, cause, transformation, reason, goal, accompaniment, direction, source, destination, path, during, point-in-time, frequency**

3. **agent involvement:** The agent of the action is a living entity with desires, feelings, goals. The involvement of the agent in a situation is an important factor to its progress.

   Relations: **ability, desired-act, intention**

4. **action relations:** This class contains different types of actions: event, state, process. The three relations form more of a continuum that three independent relations, as the distinction between them is subtle. It involves the ratio of elapsed time between the action itself and the other actions within a situation.

   Relations: **act, event, process, sequence**

Among the relations presented, the following are taken directly from Sowa [126]: **accompaniment, agent, attribute, cause, child, destination, duration, experiencer, frequency, instrument, location, manner, material, method, name, object, part, path, point-in-time, possession, amount(quantity), recipient, result, source, successor.**

Others are found in Table 2.4 (presented at the beginning of this section) which lists the most frequent defining formulas used in AHFD. Formulas <is a kind of>, <is a part of>, <is an amount of>, <is a piece of>, and <is a group of> are at the top of the list. We added the relations **is-a** (kind-of), **piece-of**, and **set-of** (group-of). Relations **part-of** and **amount-of** are already taken from Sowa.

The rest of the relations were added by looking through examples and trying to see which relations would be needed to generate an adequate conceptual graph representation.

All the relations presented are what we consider our "deeper semantic relations" in comparison to the surface semantic relations generated directly from the parse tree. The semantic relations will be used in the CG representation of the dictionary definitions, along with other surface semantic relations that are not yet disambiguated. The term **semantic relations** is used to cover all types of relations. There is often a distinction made [59] between syntagmatic and paradigmatic relations. In an association experiment, a response of type paradigmatic could be seen as a possible substitute for the cue word, as in **mother/father** or **orange/fruit**, where as the response of type syntagmatic would represent a precedence or following relation as in **mother/love** or **orange/eat**. Paradigmatic relations include the hierarchical, synonymy and meronymy relations. The syntagmatic relations include the restriction or modification relations, the case-type or argument relations as well as the derivational relations. Again, here, we talk in general terms of semantic relations. A CG is made of concepts and semantic relations.

As we are working at the CG level to identify the semantic relations, we need Semantic Relation Transformation Graphs (SRTGs) that can be projected onto our graph definitions to find out if some particular semantic relations corresponding to particular defining formulas are present. The SRTGs allow us to find the defining

formulas at the graph level and then transform the graph to contain the appropriate semantic relation. The graph level is much more flexible, as multiple string patterns lead to identical subgraphs. Example 2.4.3 shows different string patterns leading to graphs containing an identical subgraph **[use]->(to)->[B]** that we can find.

Example 2.4.3 -

<A is used to B>	[use]->(object)->[A] ->(to)->[B]
<A is used often to B>	[use]->(object)->[A] ->(to)->[B] <-(modif)<-[often]
<A is used by C to B>	[use]->(object)->[A] ->(to)->[B] ->(agent)->[C]

All three graphs contain the same subgraph that can be transformed into the instrument relation. We do not have to account for all the possible variations, as we would have to do at the string level [101].

Example 2.4.4 shows a few SRTGs that expresses defining formulas at the graph level. For a complete description, we present in Appendix E each of our relations, with a description, a few examples of possible defining formulas found in the AHFD, and the associated SRTGs.

Example 2.4.4 -

SRTG(part-of)
Before: [something:A]<-(agent)<-[be]->(object)->[part]->(of)->[something:B]
[something:B]<-(agent)<-[have]->(object)->[something:A]
After: [something:B]->(part-of)->[something:A]

SRTG(material)

Before: [something:A]<-(agent)<-[be]->(object)->[made]->(of)->[something:B]
[something:A]<-(agent)<-[be]->(object)->[made]->(from)->[something:B]
After: [something:A]->(material)->[something:B]

SRTG(instrument)
Before: [something:B]<-(object)<-[use]->(goal)->[act:A]
After: [act:A]->(instrument)->[something:B]

SRTG(reason)
Before: [act:B]->(to)->[show]->(what)->[act:A]
[act:A]->(because)->[act:B]
After: [act:A]->(reason)->[act:B]

SRTG(possession)
Before: [person/animal:A]<-(agent)<-[have]->(object)->[something:B]
After: [person/animal:A]->(poss)->[something:B]

Some of the defining formulas mentioned lead to multiple possible semantic relations and we face the problem of ambiguity mentioned earlier. For example, a simple verb like **have** can describe a part-of relation or a possession relation. In this case, the possessor being a person or an object provides the deciding key. This corresponds to the "signature" of the relation, meaning the selectional restrictions imposed on the concepts that it relates.

In other cases, mostly in the cases of prepositions indicating multiple possible relations, we keep the preposition in the graph until we have more information to disambiguate it. The only cases that would allow us to disambiguate the preposition arise when the signature of a particular deeper semantic relation is respected. For example, the signature of accompaniment could be accompanimentSignature:2,person,person. The preposition with can be refined into instrument, accompaniment, manner, etc. In this case, if it is joining two concepts of type person, we can make the transformation, as neither instrument nor manner have such a signature.

This disambiguation is quite unlikely, and the prepositions are more likely to get

disambiguated at a later stage of graph comparison and joining. They will be part of the relation hierarchy to show us which prepositions can be associated with which semantic relations when we need to compare graphs.

Instead of waiting until our clustering process (see section 3.4) to achieve more semantic relation disambiguation, we could use the technique proposed by [131] which looks into the definitions of the words involved in the ambiguous relation to see if one of the possible deeper semantic relation is present and involves compatible concepts.

2.4.4 Conjunction distribution

We want to distribute the information around conjunctions as it might be useful in further graph transformation.

In example 2.4.5 we look at the graph crayon_1_B_A presented earlier in section 2.3.1 within definition 2.3.4 at an intermediate stage of transformation that is after anaphora resolution and SRTGs were applied. Distributing the relations that **draw** is involved in to **write** will help us later if some information is required about **write.** tells us that we can not only draw but also write using the crayon as an instrument.

Example 2.4.5 -

CRAYON
    It is used to draw and write.
crayon_1_B_A
[draw]->(instrument)->[crayon]
    ->(and)->[write]
After conjunction distribution:
crayon_1_B_A
[draw]->(instrument)->[crayon]<-(instrument)<-[write]

We are being conservative here and only distribute a few relations, such as object, instrument, made-of, is-a. It is outside the scope of this thesis to do a full study on which relations are or are not distributive and in which cases the distribution can

apply. Therefore we only use this heuristic for a few cases. As well, we distribute relations only around the conjunction **and**. The conjunction **or** should not be dealt in the same way and we do not address this problem here.

2.5 Discussion

We have presented multiple graph transformation and heuristics to reduce the structural ambiguities, and hopefully reduce the number of graphs to one per sentence. Some heuristics presented were stand-alone, and others required the access to the LKB. We need to perform the whole process in multiple iterations, and at each iteration we construct an LKB that is more complete and contains less ambiguity. The heuristics needing access to the LKB will not be performed before the second iteration.

The work presented in [131] is similar to ours in the sense that it assumes a first pass through the whole dictionary to find **is-a** relations, but also other deeper semantic relations found via non-ambiguous defining formulas. They use this information as part of the LKB after their first iteration to help them into processes of conjunction and prepositional attachment (similar to our techniques that access the LKB) as well as to help them disambiguate some relations.

Again, we think of the whole process of constructing the LKB as a spiral. We start with not much information, build a little more from that, build even more from that, and so on. What we have at point A and are not able to disambiguate, might become different at point B when we have gathered more information.

We propose the following sequence, where each iteration means going through the whole set of nouns and verbs in the dictionary.

ITERATION 1.

1. Tag, parse and apply parse_to_cg rules.
2. Attempt anaphora resolution and word sense disambiguation on word defined.
3. Apply statistical method to prepositional attachment.
4. Apply ONLY the SRTG to find (is-a) relations.
5. Update hierarchy using parts of speech information and is-a relations.

This first iteration only uses stand-alone procedures and makes a first attempt at building a type hierarchy. The type hierarchy is important for the other heuristics used in the second iteration. The conjunction attachment relies directly on it. The LKB-based (not statistical) prepositional attachment relies on graph matching which is a procedure using the type hierarchy. Finally, some of the SRTGs need to know the difference between a concept of type person and of type something to transform the graphs correctly.

ITERATION 2.

1. Tag, parse and apply parse_to_cg rules.
2. Attempt anaphora resolution and word sense disambiguation on word defined.
3. Apply statistical method to prepositional attachment.
4. Apply the SRTG to find (is-a) relations.
5. Update hierarchy using is-a relations.
6. Perform conjunction attachment.
7. Perform prepositional attachment.
8. Rebuild hierarchy from scratch with reduced number of graphs.
9. Apply other SRTGs.
10. Distribute conjunctions.
11. Modify hierarchy.

Every time we update the type hierarchy, we want to parse the sentences again as some parse preference rules depend on the semantic categories of the nouns and verbs.

We must perform conjunction attachment before the conjunction distribution at the next iteration because if we distribute conjunctions there are no more conjunctions to decide the attachment of.

As we are reducing the number of graphs, with the conjunction and prepositional attachment, we regenerate the type hierarchy (not the part-of-speech transformations, only the is-a) as it might reduce some errors stored there. Then we apply the SRTGs. We waited to apply those after the type hierarchy is better set. A SRTG, might transform a graph like [X]->(is-a)->[piece]->(of)->[B] into [X]->(piece-of)->[B]. Therefore,

if we try to build the type hierarchy from the **is-a** links using the second graph after the SRTG, we have lost the information about concept X being a subtype of **piece**.

We also wanted to wait until all semantic relations are put in (after SRTGs) to perform conjunction distribution. This allows us to distribute some deeper semantic relations. Once we distribute the conjunctions, we might find new (is-a) links that got distributed, and we can update the type hierarchy. We do not rebuild the type hierarchy from scratch now because the SRTGs made us lose at the graph level some information that is already stored in the type hierarchy, and therefore we should not recreate the type hierarchy from those graphs.

ITERATION 3.

1. Steps 1 to 11 from iteration 2.

Now that the hierarchy has been modified, we can try to perform all the steps in iteration 2 one last time. This results in the final graph definitions to be part of the LKB, as well as the final type hierarchy.

To further modify the type hierarchy, we will use covert categories (as defined in section 3.2.7, as well as information extracted from new texts (see section 5.4.1).

The graph definitions will be further modified when we perform word sense disambiguation (see section 2.4.2). They might also be modified later as part of the clustering process presented in section 3.4.

After the last iteration of automatic disambiguation, we might still have more than one graph per sentence. At this point we can rely on a manual disambiguation performed by a user. By user, we mean someone familiar with the conceptual graph representation, who could distinguish between two graphs by looking at them. The user does not need any particular training, just a bit of familiarity with interpreting conceptual graphs in natural language.

Chapter 3

A LEXICAL KNOWLEDGE BASE

A Lexical Knowledge Base (LKB) should be a valuable source of information for a Natural Language Processing (NLP) system analyzing text for different goals, such as machine translation, information retrieval or text summarization. The LKB should contain information at different levels: morphology, syntax, semantic, pragmatics, that are more or less interdependent.

The preceding chapter presented the transformation steps to obtain conceptual graph representations of the sentences contained in the American Heritage First Dictionary (AHFD). To achieve these steps we presented the information from the LKB pertaining to morphology (for tagging) and syntax (for generating the parse tree). The fuzzy syntax-semantic boundary was crossed as we transformed parse trees into conceptual graphs and used some heuristics for multiple structural disambiguations. We introduced some semantic disambiguation methods for which we needed to access the information stored in the LKB about the definitions of the words.

This chapter continues at the semantic level, and shows different parts of the LKB that help structure the semantic information extracted from the definitions in a way to render implicit information explicit, to facilitate comparison among the information, and to facilitate access to this information. Let us look at all three aspects.

1. **explicitness of information**

   To render information explicit, we want the information from all definitions to interact via significant relations. First, let us mention the work of Calzolari [36] which aims at organizing a dictionary into a Lexical Database based on relationships more linguistically relevant than alphabetical relations. We look at the relations identified in her work and see how we integrate them into our conceptual graph representation.

   hierarchical relations: The hierarchical relation is shown by the **is-a** relation found in the conceptual graphs and extracted from the definitions via defining formulas such as <X is a Y> or <X is a kind of Y>. The construction of the type hierarchy depends on this relation. The type hierarchy is an important part of the LKB.

   synonymy relations: The conceptual graph may contain a **synonym** relation extracted via defining formulas such as <X is another word for Y> or <X is another way to say Y>. This is the direct synonym. Other synonyms are found by getting caught into a loop of definitions, where each word is defined by another word of the loop. We can think of these loops as forming synsets (sets of synonyms as called in the WordNet system, see section 3.2.5) which are often found at the root of the multiple subtrees which form the whole type hierarchy.

   derivational relations: Some languages have many suffixes and prefixes that have particular meanings. We are not investigating these very much here. We do a morphological analysis to find some base words, but they are usually suffixes that allow a change of part of speech. For example, the suffix *ly* changes an adjective into an adverb. We do have the mechanism in place to do further investigation. We actually see in the *er* suffix a change not only from a verb to a noun, but the noun becomes the agent of the verb, so the eater is an agent of eat, and we do put this information as part of the graph, eater becomes [eat]->(agent)->[person:*x]. These are

relations certainly worth looking at and we have the mechanism here to develop it further.

other taxonomies not organized on the is-a relation: A taxonomy that can be built is based on meronyms, or **part-of** relations. That relation is present in the conceptual graphs and can be extracted via defining formulas such as <X is a part of Y> or <X has Y>. Although the **part-of** relations are shown within the graph definitions, we do not extract that information to put it in a separate taxonomy, although all the mechanisms are in place to do so easily.

terminological sublexicons: This does not apply to our children's first dictionary. It is for larger dictionaries, where some codes are given in the definitions to relate a particular word sense to a particular domain. Those codes (present in dictionaries such as the LDOCE) are used to limit the disambiguation process.

restriction or modification relations: These are all the definitional patterns used to restrict the meaning of a genus term. They correspond to the many relations identified by defining formulas that are at the basis of our conceptual graph representation. All these relations are part of the graph representation of the definitions for individual words.

case-type or argument relations: These correspond to the syntagmatic relations such as **agent, location, object**; they are case roles for a verb. For example, for the verb **sell**, Calzolari will look at all the definitions where an agent is involved (**a person who sells**), or a location (**where they sell**), and use this information to assume some kind of template that should come with a verb to describe its possible arguments. These case relations are no different than other relations as part of the conceptual graph representation. We do not work with templates as we explain in more details in section 3.1.1.

Comparing our work to Calzolari's work emphasizes the fact that we include as part of our graph representation multiple types of relations representing a lot of

different information. The set of all graph definitions extracted from the AHFD make up the first large part of the LKB we are building.

2. **comparison of information**

Among all previous relations, the **is-a relation** is set apart and used to create a separate structure of the LKB, the type hierarchy. The type hierarchy is very important for concept comparison. The **is-a** relation is special as it permits certain kinds of inference to be made. In general, all the properties of a superclass hold for all members of the class.

We will come back to the importance of the **is-a** relation compared to other relations as we introduce covert categories in section 3.2.7. For human experiments, Collins and Quillian talk in terms of the "accessibility" of a relation in a person's memory.

> *In many cases, the superset is the most accessible property of a concept, though not always (e.g. it is probably not the most accessible property of a nose that it is an appendage or a body organ).* [45]

Another important part of the LKB, not often mentioned in work on LKB construction is the relation hierarchy. We mentioned in section 2.4.3 that the relation hierarchy will play an important role for comparing information within the LKB.

In this thesis we mostly insist on the importance of being able to easily compare information. The importance of concept comparison is noted by Collins and Quillian:

> *The process of identifying similar concepts with each other arises in many different aspects of language processing. ... Often the attempt to identify similar concepts turns into a question of whether the two concepts can be identified with each other in this particular case.* [45].

By emphasizing graph comparison, we take into account the context of comparison.

3. **access to information**

   Easy access to information in the LKB is very important if we want an NLP system to efficiently use the LKB for text analysis. Other researchers [91, 60] have worked on the problem of encoding conceptual graphs for easier retrieval, smaller storage and operational efficiency.

   This endeavor is outside the scope of this thesis, although the one last part to the LKB that we propose should help accessing the LKB. We propose the addition to the LKB of word clusters built from expanding the definition of a word to the definitions of related words.

   Clusters of words are important for many applications of computational linguistics, such as information retrieval [106, 147] and word sense disambiguation [145].

   What is interesting in this work, is that we do not only form clusters of words, but also give all the interactions (the ones we were able to extract from the dictionary) between those words. In term of accessibility to the LKB, clusters of words will direct the search into the LKB to the right cluster. For example, we will find a cluster containing a few of the words present in the text being analyzed. This will allow quicker comparisons from the text to the cluster of words, which in fact represent a fairly disambiguated merge of multiple definitions, instead of comparing the text to the individual definitions.

The LKB is divided into four parts which are detailed in the four sections of this chapter. Figure 3.1 shows all four parts with the connections between them.

1. **Graph definitions:** All the graphs constructed from the sentences of each word defined in the AHFD are gathered into this first part of the LKB. We try to obtain a single graph representation per sentence, via the algorithms presented in the previous chapter. Structural and semantic ambiguities are reduced as much as possible, and the relations found in the graphs are as specific as possible. The graph representation attempts at showing in a disambiguated form the

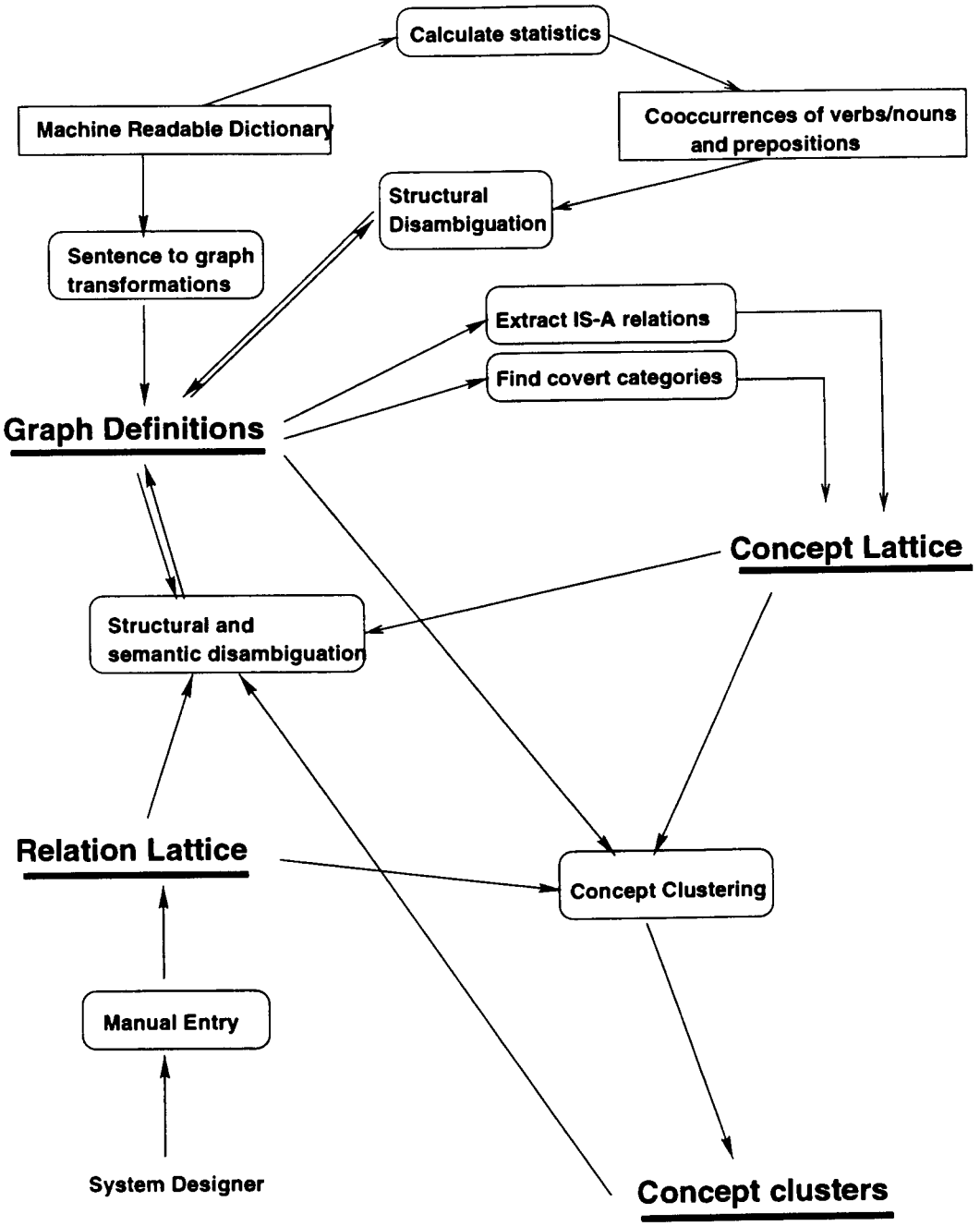


Figure 3.1: The Lexical Knowledge Base

ambiguous information hidden by the surface structure of the natural language sentences.

The sentences being represented in the LKB are extracted from a dictionary and represent facts about the world. These facts contain information that is valid all the time, or part of the time. We need a way to distinguish the absolute truths from the probable or uncertain facts. Certainty information is added to the facts represented in the LKB and we describe the process for doing so [16].

2. **Concept Lattice:** As it is the case in most research on LKB, the **is-a** relation is favored among all and a separate structure is built containing all nouns and verbs organized into a lattice of classes/subclasses. We have been talking in chapter 2 of a type hierarchy. The term concept lattice is more appropriate at this point. We will argue in section 3.2.2 that the hierarchy needs to be tangled and therefore becomes a lattice. Also, covert categories (see section 3.2.7), which do not correspond to a word or a word sense, will become part of the lattice, and therefore we prefer to talk in more general terms of concepts. The concept lattice is the main source of information to establish the similarity between concepts. We propose adding covert categories to this classification to expand the search space for finding similarities between concepts [17].

3. **Relation Lattice:** All relations used within the conceptual graphs are manually structured in a lattice to allow better graph comparisons. We defined multiple semantic relations in the previous chapter (see section 2.4.3), and we mentioned that all the prepositions are used as relations within the graphs until they can be disambiguated. One preposition can be refined to different more specific semantic relations. Organizing prepositions and semantic relations into a hierarchy will allow us to compare them in an easier way. We will see in section 3.3 that the hierarchy is tangled and we prefer to use the term lattice.

4. **Concept clusters:** Large conceptual graphs are built to show the connections between groups of related words that form clusters [18]. These clusters can be seen as groups of words defining a particular micro-world, or a particular

situation.

By constraining the language model, they will be helpful at different levels of text analysis [43]: at the signal analysis level for speech recognition or character recognition for choosing between different possible recognized words, at the syntactic level for disambiguating ambiguous syntactic structures, at the semantic level for choosing between multiple senses of a word.

When analyzing a text, if a few concepts in the text can be associated to one cluster, that particular cluster will explicitly give some information that is implicit in the text. By implicit, we mean that a text is directed toward a human reader, and much of that reader's life experience helps him/her understand that particular text which may contain references to actions or concepts not explicitly mentioned.

The word clusters will also be important for applications of information retrieval.

Moreover, the clustering is important as part of the construction of the LKB as it helps disambiguate the individual definitions. When we gather information from multiple definitions, the redundancy helps specialize certain general concepts, as well as resolve anaphora and perform word sense disambiguation. We will explore all these interesting side effects of clustering.

3.1 Graph definitions

The first major part of the Lexical Knowledge Base (LKB) is a list of all nouns and verbs with their graph definitions. We concentrate on nouns and verbs as they account for more than three quarters of the definitions in the American Heritage First Dictionary (AHFD). This observation was made for the W7 as well [6].

A noun or verb can have multiple senses, each one being defined via multiple sentences, and each sentence having its corresponding graph representation. Each graph definition is obtained from the sentence-to-graph process described in the previous chapter.

When we look at a book version of a dictionary, the entries are organized in a linear fashion according to their alphabetical order. The information is given in a localist way as all the information for each word defined is given locally in that word's entry. An LKB built from a dictionary should offer many more possibilities of organizing the world created by all the information present in the dictionary. We want to expand from the localist approach, and spread the information about one word to the entries for other words forming a large web showing multiple relations between words.

Through the eyes of our guide, the AHFD, we learn about different situations in life and build an interrelated ensemble of conceptual graphs. This ensemble is not an eclectic set, but rather a group of facts. Some facts are more important than others, and the guide emphasizes certain aspects of definitions and not others, it also gives general rules and exceptions to look for.

Keeping in mind the overall goal of using our LKB for natural language processing, we assume that further text analysis looking into the LKB for information will need to be aware of those differences about the certainty associated with each given fact. The text is better understood if we can relate some partial information given in it to more complete information given in the LKB. Therefore, a text processing task would constantly compare the text (in our case transformed into conceptual graphs) to the information stored in the LKB. We want to include the certainty information given about different facts at the graph level in a way that would ease further graph matching, as this is the general method of finding similar or partially compatible information.

Introducing certainty information as part of the conceptual graph representation barely opens the door into the area of knowledge consistency, belief and discourse analysis. The reader should view section 3.1.2 as an attempt to expand from our work into these fields which constitute large areas of study by themselves. Only a few ideas are explored.

3.1.1 Definitions as a set of interconnected nodes

Quillian [113] introduced the ideas of a **type node**, the defined concept, a **token node**, a concept used in other definitions, an **intraplane relation**, one concept with the others in its definition and an **interplane relation**, same concept from type to token. Under the non-localist approach, each token node is in an interplane relation with its type node defined somewhere else. A token node used in a definition points to its type node, and with these links all words are somehow connected.

Let us take for example, the definition of **zoo**.

> A zoo is a place where animals are kept. Many of them are kept in cages. You can see lions, tigers, and elephants at some zoos.

The type node is **zoo**, some of the token nodes are **place**, **animals**, **keep**, **cage**, **see**, **lion**, **tiger**. The intraplane relations are among the token nodes, such as **[keep]->(object)->[animal]**, and **[keep]->(in)->[cage]**. The interplane relations are pointers from each token node to its type node where it is defined.

> *Thus, a concept would be a set of interrelationships among other concepts. ... An interesting aspect of such a network is that within the system there are no primitive or undefined terms in the mathematical sense, everything is defined in terms of everything else.* [45]

For example, a system such as KL-ONE [29] which is based (as CGs are) on the idea of "structured inheritance networks", separates the world into two basic groups: primitives and defined.

We purposely do not make this distinction in our work. We mentioned in the introduction chapter (see section 1.1) an interesting aspect of the AHFD as being a closed world where all the words defined use in their definitions other words that are themselves defined. If the set of words is very large, as in an adult dictionary, we usually assume that all possible words are defined, and therefore we have a closed world. But in a small dictionary, only taking a subset of 2000 words from the large English vocabulary, we see that the authors of the AHFD tried to make the children as

part of a subworld that is consistent, and closed, having all the concepts interconnected and no primitives that have to be predefined before entering into that subworld.

The definitions are transformed to contain all the original information expressed in the sentences. We do not try to fit this information into a preexisting feature structure or "template" [50].

For example in Pustejovsky's work [26, 110, 8, 112], the Generative Lexicon Theory also rejects the characterization of a lexicon as a static listing of words, but his theory postulates a particular structure for representing semantic information. His qualia structure partitions the aspects of a noun's meaning into formal, constitutive, agentive, and telic roles.

- Constitutive Role: Relation to constituent parts
- Formal Role: Distinguishes the word within a larger domain
- Telic Role: Purpose and function
- Agentive Role: Whatever brings it about

Example 3.1.1 shows the qualia structure for two words, **food** and **bagel**.

Example 3.1.1 -

food(x)
[const: phys_obj(x)]
[formal:]
[telic: eat(E,z,x)]
[agentive: cook(w,x), prepare(y,x)]

bagel(x)
[const: dough(u)]
[formal: bread(x) & ring-shaped(x)]
[telic: ^food]
[agentive: bake(w,x)]

The qualia structure can be used in the selectional restrictions of verbs. Like for the verb eat which would select as a direct object nouns related to food, those nouns would have the eating process in their telic role.

One important aspect of the Generative Lexicon Theory is type coercion: a semantic operation that converts an argument to the type which is expected by a function, where it would otherwise lead to a type error. For example, an object can be taken to mean an event. This type coercion can be given in many cases by the telic role, as in the example **John left after the bagel.** where it means that **John left after the event of eating the bagel.**

The project ACQUILEX [37, 46, 136, 138] aims at extracting lexical (syntactic and semantic) from multiple machine dictionaries. An Italian, a Dutch a Spanish and two English monolingual dictionaries are used, as well as two bilingual dictionaries. The goal of ACQUILEX is the formalization of the basic general knowledge found in dictionaries in the form of concepts and semantic relations. They use an approach similar to the qualia structure of Pustejovsky but in a broader sense.

> *These four main roles (constitutive, formal, telic, agentive) on the one side do not cover the whole range of lexical notions which characterize nominals, and on the other side do not include other pertinent world-knowledge information which can be useful in many NLP tasks or applications and can be found in MRDs.* [37]

Therefore in ACQUILEX they use "meaning types" and associate a particular template to each one as a way to structure its semantic information. A template associated to a meaning type at the top-level of the taxonomy, such as **substance**, will be inherited by its subclasses, such as **liquid**. The meaning type **liquid** can have extra features as well. Each template for nominals contains four larger categories (function, property, constituency and source) which correspond to the four roles in the qualia structure (respectively, telic, formal, constitutive, agentive). Each larger category is subdivided into more specific relations (e.g. property into smell and color), and other larger categories are added such as location and cause-of. The templates for verbs are different and contain mostly the possible case roles.

The use of templates can have several advantages such as guiding the parsing of definitions toward finding the appropriate semantic relations and helping the comparison and merging of acquired information. The main problem though is to actually

construct those templates. It requires a large amount of human analysis, first to establish the taxonomy, then to decide on all the properties or features that will be part of each different template for all the different meaning types. The second problem is flexibility. Once the templates are constructed, all information that does not fit in has to be discarded.

Within the graph representation, all the roles (constitutive, formal, telic, agentive) from the qualia structure are described via the semantic relations. The constitutive role is given by relations such as **part-of, made-of, is-a**. The formal role is given by the attribute relations. The telic and agentive roles are shown by the object relations and will be even more emphasized via the covert categories (see section 3.2.7). All the relations used in CGs will cover the different features used in the more detailed templates of ACQUILEX.

Using a graph representation gives a much looser representation that can include all the information from the qualia structure and more. By putting as is all the information from each sentence into a conceptual graph, we do not distinguish between more important or more superfluous information. Having to put information within a template forces some information not to fit and to be discarded. That could possibly be an advantage for future comparison or analysis but it could also be a disadvantage if we made the wrong judgment on the importance of things. The problem is that the importance of a particular feature can depend on the context. For example in **They carried the piano**, and **He tuned the piano**, the verbs **carry** and **tune** do not refer to the same aspects of **piano**. Uniquely significant to each verb is that it is a **heavy object** and a **musical instrument**, respectively.

We do not, but we could at the conceptual graph level, establish a subset of relations that are considered more "important" than others and use them in priority to perform graph comparison and graph matching. This would still allow us to keep all the information given in the definitions.

We now introduce the large effort started at Microsoft [58, 56, 131, 118] to construct an LKB containing information extracted from the LDOCE. Their work shows similarity to our work by the fact that it is very much in the spirit of building an LKB made of a large interconnected network of words. We compare their choice of

representation, a semantic relation structure, to the Conceptual Graph formalism and come to the conclusion that their choice presents some severe limitations.

> *... the cluster of semantic relations extracted from a given definition actually forms a directed acyclical graph structure, referred to hereafter as a semantic relation structure. This structure has as its root node the headword of the entry containing the definition.* [118]

A CG is not an acyclical graph. It is actually very important to allow cycles as it is the only way to have multiple coreferences to a single concept. A CG does not have a root. Any CG is considered as a source of information for any concept that is part of it.

Definition 3.1.1 shows the definition of **motorist** given in [118] with the root node **motorist** (we constructed that one from our understanding of their work), then with the root node **car** (taken directly from [118]), and then we show our corresponding CG representation.

Definition 3.1.1 -

Definition of MOTORIST:

A person who drives, and usually owns, a car.

Root MOTORIST

```
motorist -< Tsub — drive
                      — Tobj >- car
        — Hyp >- person
                      -< Tsub — own
                                  — Tobj >- car
```

Inverted with root CAR

```
car -< Tobj — drive
               — Tsub >- motorist
                              — Hyp >- person
                                           -< Tsub — own
```

— Tobj >– car

Conceptual Graph Representation
[drive]->(object)->[car]<-(object)<-[own]->(agent)->[motorist *x]
->(agent)->[*x]->(is-a)->[person]

Only the CG representation allows the interpretation of a unique car which is owned and driven by the same person. The CG representation contains coreferents that refer to the same concept.

For our application, it is fine to have cyclical structures as part of the LKB. All the different investigations into the LKB to modify or restructure information are performed by the graph matching and graph joining algorithm that are not sensitive to cycles.

For his application, Richardson needs acyclical structures as he develops a whole notion of semantic similarity between words based on the path length and path weight joining them. He describes a whole process to invert a semantic relation structure. It creates a new structure with a different root node which can be stored as part of the information of that root node. He justifies the duplicated information as a trade-off between storage and retrieval efficiency. The process of inverting all the definitions in the dictionary for all the possible root nodes in each definition creates 180 000 inverted structures from 45 000 initial structures.

The CG representation is more adequate and flexible for our needs than the semantic relation structures. We do not have to invert any structure, and we base our notion of similarity between words on the similarity of their graph representation which can contain cycles.

The system DANTE [133, 11, 10] developed at the IBM Rome Scientific Center uses an approach more similar to ours than ACQUILEX or the Microsoft project. DANTE evolved into the PETRARCA system [132, 134]. Their subject and language is different from our application: they analyze short narrative texts in Italian about finance and economics. The similarity comes from their choice of conceptual graphs to represent the information they extract. They create a large interconnected network where all the concepts defined in these short texts are related. The system is later

CERTAINTY	DECISIVE SENTENCE IN NL	CONCEPTUAL GRAPH
Criterial trait:	It's a rose therefore it is a flower. [entailment]	[rose]->(is-a:criterial)->[flower]
Expected trait :	It's a bird, but it can fly. (odd) It's a bird, but it can't fly. (normal)	[bird]->(able:expected)->[fly]
Possible trait:	It's a dog, but it's brown. (why not) It's a dog, but it's not brown. (why should it be)	[dog]->(attribute:possible)->[brown]
Unexpected trait:	It's a dog, but it can walk on two legs. (normal) It's a dog, but it can't walk on two legs. (odd)	[dog]->(able:unexpected)->[walk].. ->(on)->[leg: @2]
Excluded trait:	It's a rose therefore it is not a tulip. (entailment)	[rose]->(is-a:excluded)->[tulip]

Figure 3.2: Interpretation of certainty levels

used to help analyzing more texts on finance and economics. We have contrasted their approach to ours with respect to sentence to CG transformation in section 2.1.3. We will present the idea of concept clustering in section 3.4 which is not presented in their approach.

3.1.2 Certainty Information

We base our notion of certainty of relations on Cruse's approach, where a semantic trait can be assigned five statuses of necessity to characterize the meaning of a lexical unit: criterial, expected, possible, unexpected and excluded [51]. Figure 3.2 shows (reflecting Cruse's ideas) how entailment helps assign a criterial/excluded certainty level, and how the oddity versus normality of some natural language sentences (shown as comments after each sentence) helps us understand the meaning of the other levels.

A semantic trait in the conceptual graph formalism is represented by a relation/concept pair. For example, [A]->(X)->[B], means that B is a semantic trait of type X for A, therefore [A]->(X:certainty level)->[B] represents the certainty level involved in a particular relation.

Cruse talks about **modulation** as the modification in multiple ways of a single sense, each context emphasizing certain semantic traits and obscuring others. By allowing context to modulate a word's meaning we allow it to play with the certainty

Table 3.1: Examples of criterial semantic traits

Defined word	Description	genus	criterial differentia
air	Air is a gas that people breathe.	gas	object: breath
ball	A ball is a round object.	object	round
bite	To bite means to cut with your teeth	cut	instrument:teeth
bake	To bake is to cook in the oven	cook	location:oven

levels associated with its semantical traits. In the example **A pregnant nurse attended us.**, the word **pregnant** promotes the trait of **female** from expected to necessary. A good example is the sentence **Arthur poured the butter into a dish.** Here, the trait **liquid** goes from possible, or unexpected to criterial (necessary).

As text is analyzed and compared to the representation in the LKB, information from the LKB can be brought into the text's environment and the certainty levels modified according to the context given by that particular text.

We will see how the three types of information contained in a dictionary definition (description, usage, specific examples), mentioned at the beginning of chapter 2, involve relations between concepts at different levels of certainty.

Description: Table 3.1 shows some descriptions of nouns and verbs where a particular attribute or case role is essential to the word defined. The genus in the definition gives the superclass of the object, and is used as mentioned earlier in building the concept lattice. The **is-a** relation between a word and its genus is criterial, as there is entailment from the word to the genus. The differentia part of the definition contains different attributes or case roles that differentiate the word defined from the other words in the same superclass. The presence of those semantic traits in the definition is usually essential to the meaning of the word defined and so they have a relation at the criterial level with that word.

Figure 3.3 shows the CG representation of the words from Table 3.1.

We can identify the description sentences by finding in their corresponding graph the relation **is-a**. The other relations found in the graph, which establish the differentia, will be given a criterial level of certainty.

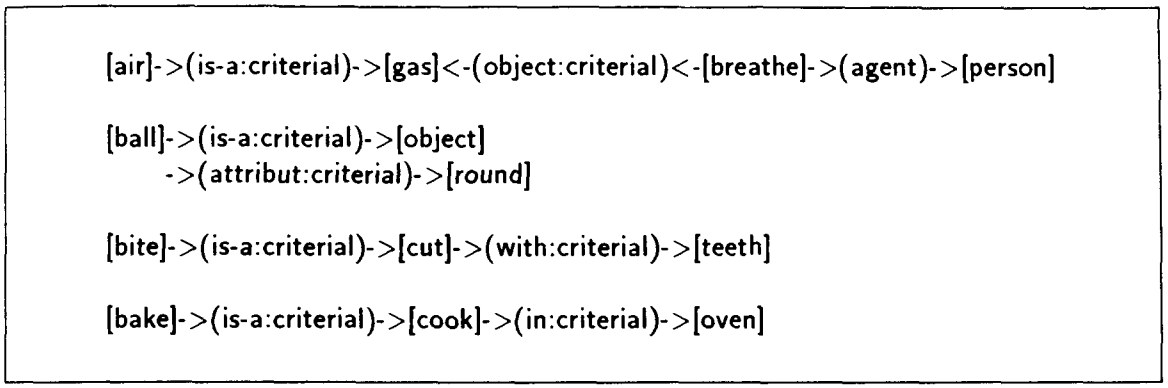


Figure 3.3: CG representation of sentences with criterial traits

Usage: The usage part of the definition presents information that can be assigned different levels of certainty. The generic information given suggests a typical usage to which we will assign an *expected* level of certainty. Some keywords present in the definition are good indicators of the certainty level to assign to that information. They usually express a position in a range of quantity (Table 3.2) or frequency (Table 3.3).

The frequency information can always be interpreted in terms of certainty, but we have to be more careful with the quantity information. We will interpret quantifiers like many, some, few, as certainty information only if they are modifying the concept acting as the agent to the verb, or as the object if no agent is present. So, in a sentence like **Many bears sleep all winter.** we will treat the keyword **many** as certainty information, while in a sentence like **Her father became a pilot many years ago.** we will not.

The CG representation will include the necessity level given by the keyword. An expected level should be assumed when no keyword is given, as these sentences are considered facts general enough to be stated as part of the AHFD.

Figure 3.4 shows a few CG representations taken from the examples in Table 3.2 and Table 3.3.

Specific examples : In the example part of the definitions, the concepts presented

Table 3.2: Certainty levels expressed by keywords of quantity

necessity	keyword	example
criterial	all	All people and animals have bodies. Cups, suitcases, and tools all have handles.
expected	most	Most birds can fly. Most births happen in hospitals.
	many	Many authors write books for children. Many bears sleep all winter. Many children go to a camp in the summer.
possible	some	Some people are afraid of the dark. Some astronauts have walked on the moon.
unexpected	few	... but few people live there (forest).
excluded	no	No trees grow there (field). A round object has no points or corners.

Table 3.3: Certainty levels expressed by keywords of frequency

necessity	keyword	example
criterial	always	The air inside (a greenhouse) is always warm. There are always guards in front of the palace.
expected	often	Knights often wore armor. Baskets are often shaped like bowls and have handles.
	usually	Cartoons are usually funny.
possible	sometimes	Birds and other animals are sometimes kept in cages. People sometimes cry when they are sad.
unexpected		
excluded	never	It (sun) never rises in the west.

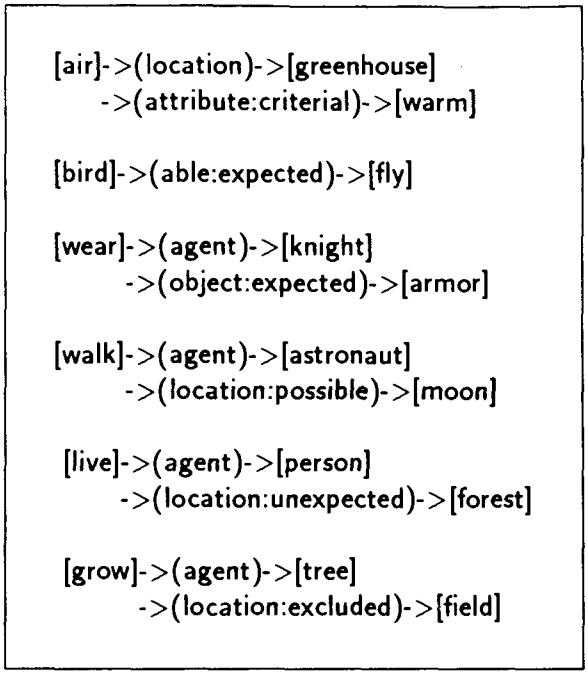


Figure 3.4: CG representations including different certainty information

are only possibly interacting with the noun or verb defined in a particular situation. The situations presented are very precise, and therefore the information given is of limited interest by itself but it might reveal some interesting aspects if we cumulate multiple examples to dynamically update our LKB. We present a few ideas hereafter that have not been implemented so far. There are not many examples given with the noun definitions, therefore we will look in Table 3.4 at the example part of the same two verbs presented in Table 3.1.

Table 3.4: Possible situations given by specific examples

word defined	example	possible roles
bite	She bites her sandwich.	agent: person object: sandwich
bake	John baked his pie for an hour.	agent: John object: pie duration: 1 hour

The examples can be seen as the different experiences that someone goes through

day after day. We keep some individual experiences in memory, but we also look for generalizing patterns. Within this LKB application when general patterns are found, we might discard the individual examples and keep the general pattern that encompasses all the separated examples. When we find one particular example that expresses a different view then the one we hold via our generalization pattern, we keep it as a contradictory example.

In the LKB, the individual examples extracted from the dictionary are kept separately from the generic information given by the usage part. When a generalization pattern is found, it is placed with the generic information, and the examples are discarded.

For doing so, we try to find a graph that subsumes multiple other graphs. We find all the concepts subsumed by a more general concept in the subsuming graph. For each case role attributed to the verb in the graph, the superclass of all unifiers establishes a selectional restriction.

For example, in Table 3.4, the object of **bake** is **pie** which is a subclass of **food**. In the dictionary, we find that **cakes**, **bread**, **loaves**, **cookies** and **pizza** are being baked, and they are all subclasses of food. We might therefore assign a selectional restriction to the object of [bake] by giving the relation an **expected** certainty level, here **[bake]->(object:expected)->[food]**.

This more general graph showing the selectional restriction, will replace the multiple specific example graphs we have.

Certainty Information in CG representations

Table 3.2 and table 3.3 showed a set of keywords expressing either a range in quantity modifying a noun, or in frequency modifying a verb.

In our CG representation, the frequency information on a verb will be identified via a **modif** relation, to which a frequency keyword concept is attached, and the quantity information on a noun will be identified via an **attribute** relation, to which a quantity keyword concept is attached.

To establish a certainty factor on one specific relation, we must first find the relation **modif** giving verb modifications, or the relation **attribute** giving noun modifications. The case of nouns is more complex as mentioned in the previous subsection. We only consider a noun as a candidate for certainty information if it is involved in an **agent** (or **object** in case **agent** is absent) relation to a verb. Here are the possible subgraphs found on which we can achieve a transformation.

```
[VerbX]->(modif)->[Frequency]
        ->...
[VerbX]->(agent)->[Noun]->(attribute)->[Quantity]
        ->...
[VerbX]->(object)->[Noun]->(attribute)->[Quantity]
        ->... ** no agent **
```

There are four possible cases to explore. The number of relations given to describe each case is not the total number of relations in the graph. It is the number of relation in which **VerbX** is involved (not counting the modif relation).

1. **Single relation:** VerbX is involved in one relation on which we assign the certainty level.

   Example 3.1.2 John never smiles.

   Graph 3.1.1 - *Starting Graph*

   ```
   [smile]->(agent)->[John]
           ->(modif)->[never]
   ```

   Graph 3.1.2 - *Resulting Graph*

   ```
   [smile]->(agent:excluded)->[John]
   ```

2. **Two relations including agent:** VerbX is involved in an agent relation and one other relation. The **agent** gives us the person or thing we are talking about in this sentence. The other relation give us details about what this agent did, where, or how. The certainty information is put on that relation.

Example 3.1.3 Some astronauts have walked on the moon.

Graph 3.1.3 - *Starting Graph*

```
[walk]->(agent)->[astronaut]->(attribute)->[some]
    ->(on)->[moon]
```

Graph 3.1.4 - *Resulting Graph*

```
[walk]->(agent)->[astronaut]
    ->(on:possible)->[moon]
```

3. **Two relations excluding agent:** The third case does not contain an **agent** relation, and we assume the **object** relation leads to a concept playing a role similar to the agent. Therefore we are in a situation similar to the previous case. We assign the certainty to the other relation that VerbX is involved in.

Example 3.1.4 Birds and other animals are sometimes kept in cages.

Graph 3.1.5 - *Starting Graph*

```
[bird]->(and)->[animal]->(attribute)->[other]
    <-(object)<-[keep]->(in)->[cage]
                    ->(modif)->[sometimes]
```

Graph 3.1.6 - *Resulting Graph*

```
[bird]->(and)->[animal]->(attribute)->[other]
    <-(object)<-[keep]->(in:possible)->[cage]
```

4. **More than two relations:** We assume agent is present, and if not, the object relation (as in previous case) takes the agent role. This case is more complex, and we will introduce a few ideas via an example.

Example 3.1.5 John often eats at the restaurant for lunch.

Graph 3.1.7 - *Starting Graph:*

[eat]->(agent)->[John]
<-(modif)<-[often]
->(at)->[restaurant]
->(for)->[lunch]

Graph 3.1.8 - *First interpretation: we see John and it's lunch time, so we expect the location to be a restaurant,*

[eat]->(agent)->[John]
->(for)->[lunch]
->(at:expected)->[restaurant]

Graph 3.1.9 - *Second interpretation: we see John and we are in a restaurant, so we expect the time to be lunch time*

[eat]->(agent)->[John]
->(at)->[restaurant]
->(for:expected)->[lunch]

Both interpretations are possible. But is there one more probable than the other? The following paragraphs are an attempt at answering this question. We look at the notion of informativeness of words.

At the core of the field of information theory is the rule by Shannon [109] stating that the most informative event is the least probable. The information given by an event x is inversely proportional to its probability. Shannon's theorem gives the information of x, I(x), as the following:

$$I(x) = log(\frac{1}{P(x)}) = -log(P(x))$$

where P(x) is the probability of the event x.

In our case, events are words. The probability of occurrence of a word in a corpus can be calculated by counting its number of occurrences in that corpus, and divide it by the total number occurrences for all words in the corpus.

Therefore, a word is considered informative (brings more surprise in the text) if it does not occur often in a corpus. The more common a word is (thing, person) the less we are surprised or informed by it. We hypothesize here that certainty information should be seen in relation to information given by each word in the graph. It should be put on the most informative word in the sentence.

In our example, the word restaurant occurs only one time in the AHFD (not counting in its own definition) and the word lunch is there 17 times. We would therefore put the certainty information on the word restaurant and favor the first interpretation, where we expect the location of the lunch to be at the restaurant.

Of course, the AHFD gives us a very small corpus to calculate statistics, but we present briefly this idea of informativeness from information theory to suggest a way to automatically decide which word is being emphasized by the certainty information.[1]

The information about certainty or necessity is usually given as a second order relation in the "standard" (from Sowa [126]) CG formalism. Let us look again at Example 3.1.5 given before, and its corresponding standard CG.

Graph 3.1.10 -

```
[Proposition: [eat]->(agent)->[John]
                 ->(at)->[restaurant]
                 ->(for)->[lunch]
]->(certainty)->[expected]
```

Our approach of attaching the certainty to a given relation within the graph surrounds more precisely where the certain or uncertain information is. By putting it outside the graph, it gives a certainty value to the whole proposition.

[1]The notion of informativeness of a word is taken without any given context. More precisely, the context becomes the whole dictionary used to calculate the probabilities of encountering each word, but each sentence when later analyzed is considered to be seen in isolation. In a text, when some sentences precede the one analyzed, the informativeness of a word will depend on whether it was mentioned before, and not only on how probable it is to occur in any context. Therefore its probability is modified and the interpretation of certainty information can change.

The standard representation has the advantage of not having to choose what is certain or uncertain. In our case, when no context is available, we use the rule from information theory and therefore assign the certainty information on the relation leading to the more informative concept. But again, we believe that if the sentence was taken in context, we would also have information about the salience of a word and would be able to point to the right relation where the certainty information should be added.

In our opinion, our method permits a more accurate representation. It has repercussions on the graph operations defined in [126] that allow us to compare or combine the information contained in the definitions.

Graph comparison One important operation, that we have been mentioning throughout this thesis is graph matching. Finding the maximal common subgraph between two graphs corresponds to finding what similar information is given by their representations. The certainty levels introduced in section 3.1.2 must be taken into consideration as we perform the graph matching operation, and we present here some ideas on how this may be done.[2]

We look at one simple case of comparing 2 graphs. We test if graph-1 can subsume graph-2, and if so, we indicate which certainty level should be kept in the subsuming graph. We assume the graphs contain identical concepts and relations, and one relation may contain a different level of certainty.

graph-1 : [eat] ->(object:expected)->[fruit]
->(agent)->[John]

graph-2 : [eat]->(object:possible)->[fruit]
->(agent)->[John]

The level **possible** is the most neutral level which can subsume all other levels. In this case, graph-1 can subsume graph-2 if its relation contains a more neutral (or less decisive) level of certainty. Table 3.5 shows the resulting certainty level when subsumption is possible. The use of natural language sentences helps to judge qualitatively if the results make sense. When we read a stronger statement, we cannot

[2]These ideas have not been explored to the point of being implemented in our system presented in chapter 4. They again open the door into the field of knowledge consistency, belief revision, and we do not pursue these issues in detail here.

Table 3.5: Case 1: Certainty level for A subsuming B

A / B	criterial	expected	possible	unexpected	excluded
criterial	criterial	–	–	–	–
expected	expected	expected	–	–	–
possible	possible	possible	possible	possible	possible
unexpected	–	–	–	unexpected	unexpected
excluded	–	–	–	–	excluded

then propose a weaker one. But we can start from a weaker statement and emphasize more toward a stronger statement.

- excluded cannot subsume/entail criterial
  John never eats fruits. Well in fact, **John always eats fruits.**
- possible can subsume/entail expected
  John sometimes eat fruits. Actually, **John often eat fruits.**
- unexpected cannot subsume/entail possible
  Cathy rarely drives the car. Even more, **Cathy sometimes drives the car.**
- unexpected can subsume/entail excluded
  Cathy rarely drives the car. To tell the truth, **Cathy never drives the car.**

If we assign numerical values between -1 and 1 to each certainty level, we have: criterial(1.0), expected(0.5), possible(0), unexpected(-0.5), excluded(-1.0). Relation A with certainty value C(A) can subsume relation B with certainty value C(B) if

$$C(A) * C(B) >= 0$$

$$|C(A)| <= |C(B)|$$

Things get more complicated when the concepts involved in the relation modified by certainty information are not identical. Then we have to understand the relation between the concepts, for example is one an hypernym or an opposite of the other, before we can decide on the relations between the graphs. Let us modify graph-2 to contain one concept more specific than graph-1.

graph-1: [eat] ->(object:expected)->[fruit]
->(agent)->[John]

graph-2: [eat]->(object:possible)->[apple]
->(agent)->[John]

In this case, talking about whether graph-1 can subsume graph-2 is not appropriate. We should talk more about their co-existence, do they contradict each other or can they both be part of the same graph?

Using the same scale as given for Case 1, we propose a condition for possible co-existence (non-conflict). Relation A with certainty value C(A) can co-exist relation B with certainty value C(B) if

$$C(A) >= C(B)$$

Again, we look at natural language sentences to get a feel for the validity of this result. This time, to be valid the second sentence is not a stronger statement of the same idea as in Case 1, but a precision given on that idea.

- expected can co-exist with excluded
  John often eats fruits. But, John never eats apples.
- unexpected cannot co-exist with expected
  John rarely eats fruits. But, John often eats oranges.
- possible can co-exist with excluded
  Cathy sometimes drives a car. But, Cathy never drives a Cadillac.
- possible cannot co-exist with criterial
  Cathy sometimes drives a car. But, Cathy always drives a Cadillac.

We are aware that we leave many questions unresolved. This is not the main focus of this thesis, and it will be very interesting as future research to explore in more details the few suggestions given in this section.

3.1.3 In Summary

We presented the idea of including certainty information into our graph representation of dictionary definitions. We presented some ideas inspired by the work of Cruse,

using five levels of certainty: criterial, expected, possible, unexpected, excluded to give weight to some relations in the graphs. We explained the repercussion on the standard conceptual graph matching algorithms. The finding of keywords and inclusion of certainty information in the conceptual graph formalism has been implemented in our ARC-Concept software presented in chapter 4, but the graph matching processes have not.

The certainty information presented has some similarity with the certainty factors used in MYCIN [14]. They use a confidence scale between -1.0 and +1.0, where -1.0 represents complete confidence that a proposition is false and +1.0 represents total confidence that a proposition is true. They say that standard statistical measures were rejected in favor of CFs because experience with clinicians had shown that clinician's reasoning patterns is more weighted by terms such as strong, weak in their decision making. Keeping the words themselves as part of the graphs allows us to express certainty information in a form closer to natural language.

Overall in this section, we presented the first part of the LKB, the graph definitions. All graphs are generated from the sentences found in the dictionary via the transformation process presented in chapter 2. By having each concept in a graph considered as a token node pointing back to its type node, where the concept is defined, we form a large interconnected set of facts and not only a list of definitions. Within these facts, we render explicit at the graph level the certainty information given in the sentences via different keywords.

3.2 Concept Lattice

In section 2.2, we presented very briefly the construction of the type hierarchy, to include the different word senses given in the AHFD, as well as the subclass/superclass relations found via the is-a links within the dictionary definitions. We explained that we needed the type hierarchy for further processing/reduction of the graph definitions.

In this section, we will go into more details into the construction of what we will call from now on a concept lattice, as we will see that a strict hierarchy is not sufficient to express all the subclass/superclass relations found in the dictionary.

First we will give some ideas from Cruse [51] in section 3.2.1 on what is supposed to be represented in a concept lattice. We will use another term, **taxonomy**, as interchangeable with concept lattice.

Then we will come closer to our subject of interest by reviewing some work on the building of a concept lattice from information extracted from machine readable dictionaries. This will lead us into presenting the taxonomy built from the AHFD. One important problem faced by researchers attempting automatic building of a taxonomy is how to determine which sense of each word is appropriate as a superclass of other concepts. We will introduce this problem of genus disambiguation and explain some heuristics for resolution.

As a point of comparison for our resulting taxonomy we look into WordNet, as it is the most talked-about taxonomy of the present days. Some WordNet-based applications are developed for sense disambiguation [115] and for establishing selectional restrictions [117]. A project, EuroWordNet, aims at building similar wordnets from existing resources for Spanish, Dutch and Italian [137].

We will pause to discuss what we have so far, what is part of the taxonomy, what is not, how we could use other methods to find more subclass/superclass relations as well as what is missing in the concept lattice. This leads us to the important subsection on **covert categories** in which we describe other types of classes to be added to the taxonomy.

Finally we have a short discussion on semantic distance, as it is often based on the concept lattice and it is important for our favorite operation of graph matching.

3.2.1 Defining a taxonomy

A taxonomy is a type of branching hierarchy which expresses a relation of hyponymy between daughter and mother nodes. An hyponym is a lexical unit which is the result of a necessary entailment from another lexical unit. For example, if a dog is a pet necessarily entails that a poodle is a pet., then poodle is an hyponym of dog.

A branching hierarchy must express two structural relations: a relation of dominance and a relation of difference [51]. The relation of dominance is the vertical

relation, and the relation of difference is the horizontal relation. The relation of dominance must be asymmetric in that it must have a direction, and it must be catenary, which is the capacity to form indefinitely long chains of elements. For example, the relation **husband-of** is asymmetric but not catenary, and the relation **father-of** is both, because A can be the **father-of** B, who can be the **father-of** C, who is not the **father-of** A. The relation of dominance can be transitive or intransitive. For example, $>$ represents a transitive relation, ($A > B$ and $B > C$) implies ($A > C$), but **friend-of** is intransitive (A **friend-of** B and B **friend-of** C) does not necessarily imply (A **friend-of** C).

In any type of hierarchy, the relation of dominance and the relation of difference must remain constant. The hyponymy hierarchy has a relation of dominance based on hyponymy, and a relation of difference based on incompatibility. A taxonomy, is a subtype of hyponymy hierarchy in which the relation of dominance is respected, daughter-nodes must be hyponyms of their mother-nodes, but the sister-nodes do not show a uniform relation of incompatibility. For example, **book** and **paperback** are hyponyms of **novel**, but they are not incompatible.

Natural taxonomies have been widely studied in different areas as anthropology, biology, etc. One characteristic is that they typically have no more than five levels. For example a chain, **[plant/bush/rose/hybrid tea/Peace]** corresponds to different levels, **[unique beginner/life-form/generic/specific/varietal]**.

The **generic level** is the most significant level of a taxonomy from the point of view of the speakers of the language.

> *This is the level of the ordinary everyday names for things and creatures: cat, oak, carnation, apple, car, church, cup, etc. ... most branches of taxonomic hierarchies terminate at the generic level. Items which occur at specific and varietal levels are particularly likely to be morphologically complex, and compound words are frequent.* [51]

3.2.2 Building a taxonomy from a dictionary

When people talk about building taxonomies from MRDs, they have a different notion of the word **taxonomy** than the formal definition just described in the previous section.

> *The notion of taxonomy that has been used in work on MRDs is essentially an informal and intuitive one: a taxonomy is the network which results from connecting headwords with the genus terms in their definitions, but the concept of genus term is not formally defined; however for noun definitions, it is in general taken to be the syntactic head of the defining noun phrase.* [48]

This definition is not as restrictive as the one defined by [51], as it does not require any consistency in the dominance or difference relations. There might be inconsistencies as well as circularity between the headwords and genus in the definitions.

Amsler was perhaps the first author within the research area of extracting information from MRDs to note the availability of taxonomic links [6]. Even if most of the work described in his thesis is actually done manually, Amsler had many insights about how things could be done if computers were more efficient at that time. Following his thesis, many articles from different groups present some aspect of automatisation of what Amsler was doing manually [35, 41, 34, 141]. As mentioned in section 2.4.3, most of the extraction techniques rely on finding defining formulas within the defining sentences. Formulas such as <A is a B> or <A is a kind of B> were mentioned as the most common ways to identify an **is-a** relation between concepts A and B, showing a subclass/superclass relation that can be added to the concept lattice.

Most research on taxonomies has been done on nouns, and sometimes verbs. This is well justified when we look at Amsler's analysis of the Merriam-Webster New Pocket Dictionary. Amsler found that it was composed of 66% of nouns, 22% verbs, 20% adjectives, 2% adverbs, and less than one percent all the other part of speech (prepositions, conjunctions, etc). The number of nouns in the dictionary far surpasses the number of any other part of speech.

> *Whereas a single definition or even a small set of definitions may fail to cover the complete meaning of the words they define, access to a complete*

> *set of taxonomically related definitions may be used to compose a nearly complete portrait of the meaning of their root concept.* [7]

He builds the taxonomy for the nouns and verbs based upon the hand-disambiguated genus words in their definitions. It confirms the expected structure of the lexicon to be a **tangled hierarchy**.

Allowing a concept to have more than one genus leads to a concept lattice instead of a tree. We can also talk in terms of multiple inheritance hierarchy, or tangled hierarchy. Having the noun taxonomy as a tree, in which any noun is in an upward relationship (is-a) to only one other noun, is preferable, according to some researchers [48] since it simplifies some operations, such as inheritance. Through the following paragraphs we will see that it is not always possible.

We face multiple possible genus when a definition includes a conjunction of genus, as seen in the following example from the AHFD: **An aquarium is a glass box or bowl that is filled with water.** If **aquarium** is a subclass of both **box** and **bowl**, it would inherit from both items. Alternatively, to keep the hierarchy as a tree, one could add an entry {**box or bowl**} that would inherit the intersection of the attributes of **box** and **bowl** [48].

If we put {**box or bowl**} under **box** and under **bowl** with some particular mechanism to mark it in a special way so it inherits the intersection of both concepts, we place this new concept as more specific than the two others separately. Which is not the case. The concept {**box or bowl**} is more general than **box** and than **bowl** so it should be put higher in the hierarchy. What we suggest here, is to find the lowest common generalization of **box** and **bowl**, let say it is **container** for this example, and put the node {**box or bowl**} under **container**, and place **box** and **bowl** under the new node. This keeps the specialization progression, as well as keeping the hierarchy to a tree structure as shown in Figure 3.5.

Example 3.2.1 shows a list of definitions that will lead to a tangled hierarchy as shown in Figure 3.6.

Example 3.2.1 -

A father is a man who has at least one child.

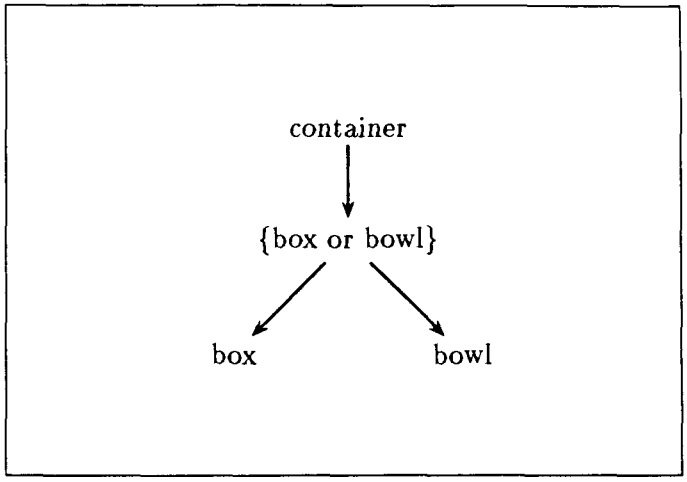


Figure 3.5: Tree hierarchy including sets.

A man is a grown male person.
A mother is a woman who has a child.
A woman is a grown female person.
A parent is a mother or father.

To add the node **parent** = {**mother**, **father**} in the hierarchy, we find the lowest common supertype of **mother** and **father** which is **person**. We then put **parent** under **person**, and **mother** becomes a subtype of **parent** but stays also a subtype **woman**, as well as **father** becomes a subtype of **parent** but stays also a subtype of **man**. This example shows the necessity of allowing tangled hierarchies.

3.2.3 Building a taxonomy from AHFD

We first start with a flat taxonomy where all words are leaf nodes and the only root is the supertype of all, that is type **everything**. We then refine the taxonomy in multiple steps:

1. Manually we assign a supertype to some pronouns often used in definitions, such as you, we, someone, somebody. We do not analyze pronoun definitions, therefore we cannot discover their superclass automatically.

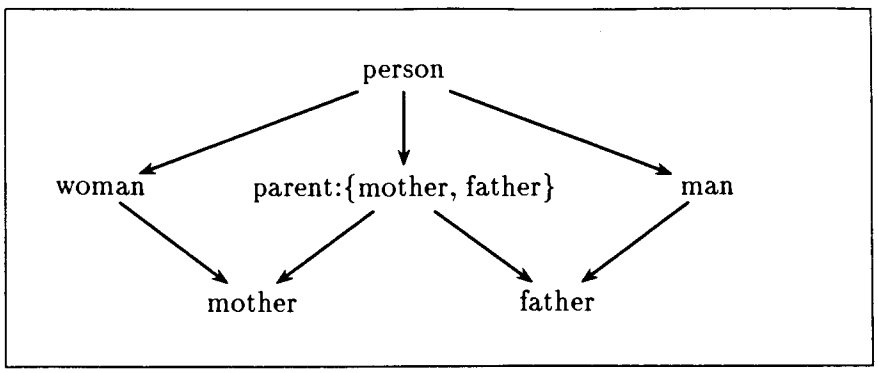
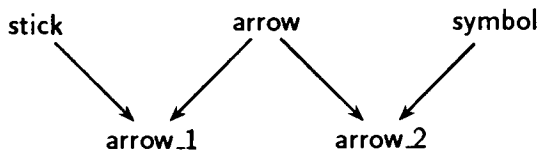


Figure 3.6: Tangled hierarchy including sets.

2. Words with multiple senses have their individual senses put under the word, so they will have multiple superclasses, but one of them will be their word. We can see in example 3.2.2 the definition of the word **arrow** and the relation between word senses (noted with an underscore followed by sense number) and their superclasses..

Example 3.2.2 -

arrow
1.n.
An arrow is a stick with a point at one end.
You can shoot arrows from a bow.
2.n.
An arrow is a symbol.
It is used to point in one direction.



3. Use the is-a relations found in the graphs to refine the noun and verb taxonomy (see section 2.2).

4. Only words that are still subtypes of **everything** after the two preceding steps are modified here. Depending on their part of speech, word senses are put under different categories. Verbs are put under **act**, adjectives under **attribute** and nouns under **something**. For example, the word **stick** is defined as **a piece of wood**, giving the graph **[stick]->(piece-of)->[wood]** in which there is no **is-a** relation, and therefore **stick** will not be updated from under **everything**. As it is a noun, we put it under **something**. On the other hand, we will not add the supertype **something** to arrow_1, as arrow_1 is subtype of stick and therefore also a subclass of **something**.

Figure 3.7 shows the subtree under **place** and figure 3.8 shows the subtree under **animal** as they were built automatically from the AHFD through the process of finding the genus of each definition. Each superclass in the figures points to its first and last subclass only so the figures are not too crowded. It is not our intention here to judge whether the IS-A hierarchy extracted from the AHFD is right or wrong. This is often a matter of long lasting debates. The only point we are trying to make is that the AHFD is structured in part on the basis of classification of nouns through a hierarchy of classes, subclasses and superclasses. The resulting hierarchy is certainly simpler than one that would be extracted from an adult's dictionary, but nonetheless informative, and adequate for our present task.

As we build the hierarchy, going upward, we find circularity at high levels. Under the main root **everything**, the whole taxonomy is a forest with multiple trees, each of which having at its root a group of words defined through a loop. Example 3.2.3 shows the circularity in the definitions of **place** and **animal** which are at the root of the two hierarchies presented in Figure 3.7 and 3.8.

Example 3.2.3 -

Animal: An animal is anything alive that is not a plant.
A thing is an object, animal, or plant.

Place: A place is somewhere for something to be.
Somewhere means any place.

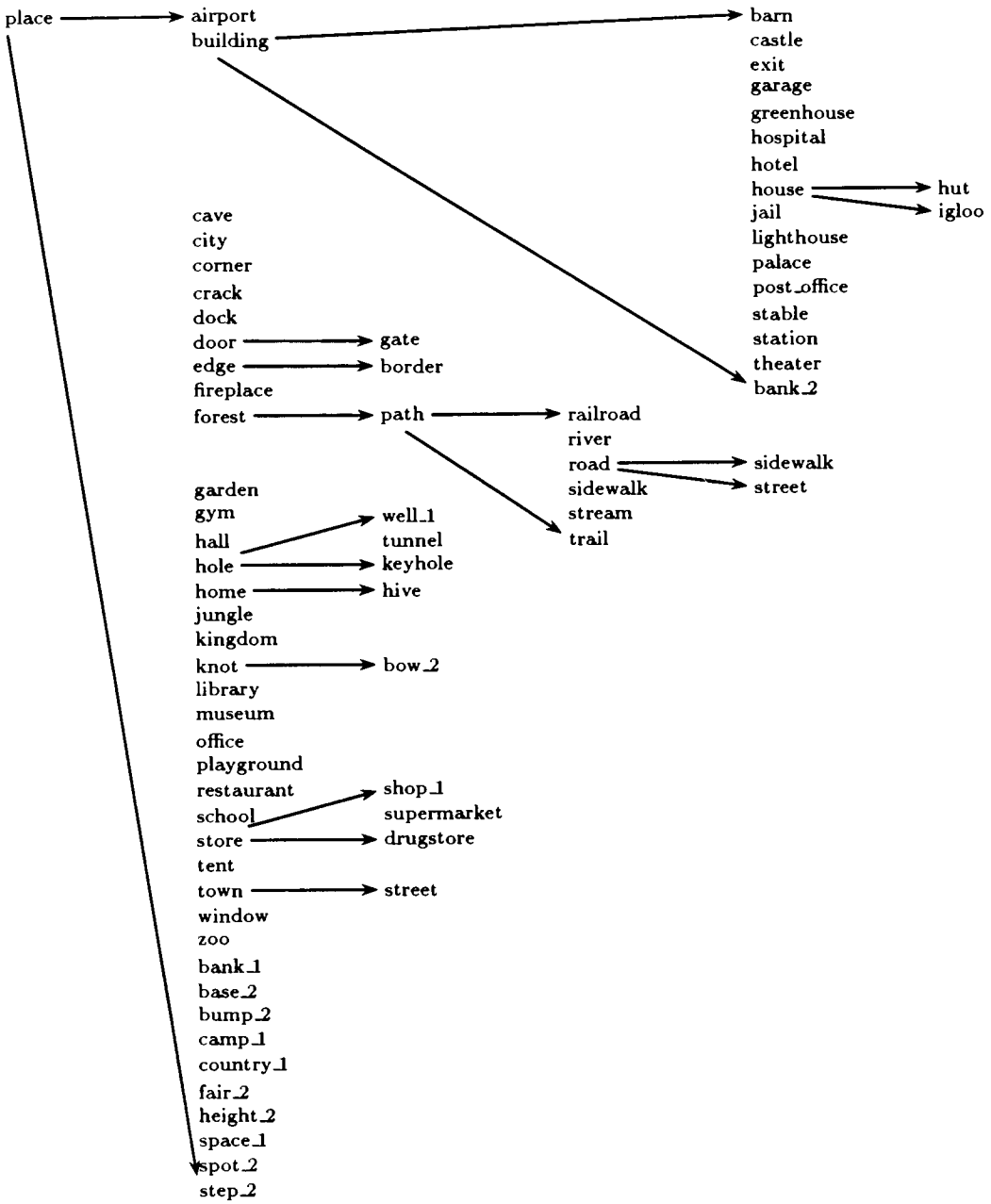


Figure 3.7: **is-a** hierarchy for root **place**

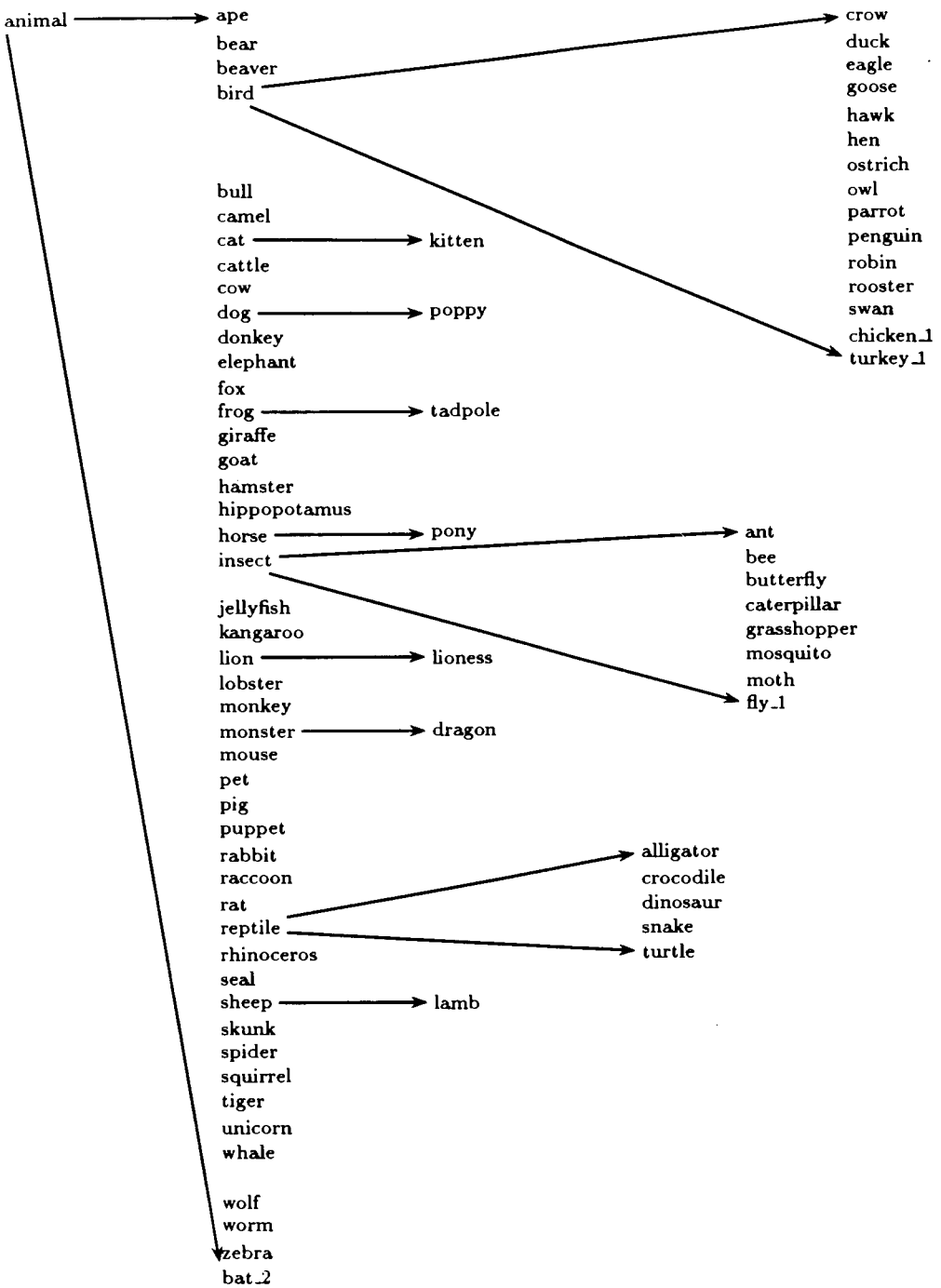


Figure 3.8: **is-a** hierarchy for root **animal**

3.2.4 Genus disambiguation

When a word with multiple senses is not a leaf node, we have to decide which sense is appropriate under which to put other nodes or leaves. This is the problem of **genus disambiguation**.

Even if the AHFD seldom gives multiple senses to a word (see section 2.4.2), it is important to be sure that when we build any network of relations, we actually put the correct word senses in relation with each other. Especially in an is-a relationship with inheritance along the links, we would not want one node to inherit from the word senses it is not related to [84]. In example 3.2.4, **dam** should not be related to the more specific sense **side of a room**, but instead to the more general sense, a wall being a separation between two things.

Example 3.2.4 -

DAM
dam is a wall across a river.
WALL_1
A wall is a side of a room.
WALL_2
A wall is something that is built to keep one place apart from another.

In Guthrie *et al.* [72], they analyze multiple possible relations between the genus and the entry word. They use the LDOCE dictionary, in which box codes (semantic information) and subject codes (pragmatic information) are given. The box codes use a set of primitives such as **abstract, concrete, animate**, organized into a hierarchy and the subject codes are a set of field names such as **engineering, electrical**, also organized into a hierarchy. Their **Genus Disambiguator** tries to find the genus sense whose semantic codes identically match with the headword, or that has a semantic category that is an ancestor to the semantic category of the headword. In case of a tie, they look into similarity of the subject code, and in case of another tie, they choose the most frequently used sense of the genus word (stated as the first sense in LDOCE). They give a success rate of about 80%.

In Bruce and Guthrie [32], they go further and analyze individually and in combination, each of the three factors they considered in their previous work (semantic code, pragmatic code, frequency of usage), to assign a weight to each of them.

In Klavans *et al.* [84], they work with the Webster's Seventh, in which no semantic codes are available. They propose two approaches of disambiguating the genus of words for building their taxonomy. The first one had been proposed by [90] previously. It consists of finding the number of overlapping words (excluding a stop list like **the, a, of** that are present in the definition of the headword, H, and the definition of the genus, G. The second approach consists of finding the number of overlapping words in the definition of H with synonyms of G, and of the synonyms of H with the synonyms of G. The second approach is considered more accurate, even if both give results of about 40% success. The difference is in their rating in terms of *full successes*, where all the proposed choices are good, as opposed to a *success* being when one of the genus proposed is the right one. In the first approach, 9% corresponds to a full success, and the second approach gives 19% full success. The remaining of the 40% are proposed sets of genus in which there are some successes and some omissions, or some successes and some erroneous choices.

In our study, we do not have access to any subject or box code, as the dictionary we use is not intended for any computational linguistic purposes as the LDOCE is. Statistics would not help either, as to determine what sense of a word is used the most often, you need a large corpus tagged by hand with word senses which we do not have either.

Therefore, we take a more general approach, that could be useful in cases where we deal only with raw text, not manually modified to add extra codes or tag some information in it. Our approach is based, as is all the rest of this thesis, on graph matching. By comparing information from different definitions and finding similarity we can solve some ambiguity problems.

For our purposes, we treat the genus disambiguation problem as part of our general word sense disambiguation process described earlier (see Section 2.4.2, method 4). The genus is just another word in the definition that needs to be disambiguated. Therefore using our general method involves performing a graph matching to find the maximal

common subgraph between the word defined and the different possible senses of the word in the definition, here the genus.

In a way it is similar to finding the number of overlapping words as in [84], except that we also consider the relations in which the words are involved. Example 3.2.5 shows a situation where an ambiguous word **chicken** (sense 1 for the animal and sense 2 for the meat) is used in a graph and needs to be disambiguated. If two graphs possess the word **chicken** in a disambiguated form they can help solving the ambiguity. In example 3.2.5, Graph 3.2.1 and Graph 3.2.2 have two isolated concepts in common: **eat** and **chicken**. Graph 3.2.1 and Graph 3.2.3 have the same two concepts in common, but the addition of a compatible relation, creating the common subgraph **[eat]->(object)->[chicken]**, makes them more similar. The different relations between words have a large impact on the meaning of a sentence.

Example 3.2.5 -

John eats chicken with a fork.

Graph 3.2.1 -

```
[eat]->(agent)->[John]
    ->(with)->[fork]
    ->(object)->[chicken]
```

John's chicken eats grain.

Graph 3.2.2 -

```
[eat]->(agent)->[chicken_1]<-(poss)<-[John]
    ->(object)->[grain]
```

John likes to eat chicken at noon.

Graph 3.2.3 -

```
[like]->(agent)->[John]
    ->(goal)->[eat]->(object)->[chicken_2]
                ->(time)->[noon]
```

The graph matching method gives a more precise answer for the choice of a genus, as it can look not only at concepts but at how their interaction is similar or not.

3.2.5 Comparing the children's taxonomy to WordNet

WordNet [96, 23, 97] is a large lexical knowledge base, hand built by lexicographers, and based on psycholinguistic principles. A small number of relations is used to structure the nouns in the lexicon. They are synonymy, antonymy, hyponymy (is-a), hypernymy (inverse of is-a), meronymy (part-of) and holonymy (inverse of part-of). The basic unit of WordNet is a set of synonyms called a **synset**. The synsets are organized into multiple hierarchies, with a set of semantic primes (a synset) at the root of each hierarchy. We compare in Table 3.6 their hierarchies to the hierarchy built from the AHFD. In WordNet, there are 25 semantic primes called unique beginners.

Some of the WordNet categories are defined in the AHFD. As they are very general concepts such as object, shape, time, animal, plant, food and place, they correspond to concept types in AHFD that are at the root of different subgraphs. Those subgraphs are joined together as all roots are subclasses of **everything**.

Some categories from WordNet are not directly defined in AHFD (that is the words representing the categories are not defined) but the concepts are present as they can be found via some defining formulas, such as event, feeling, possession, reason.

Some other categories from WordNet are not defined at all in AHFD, such as substance, relation, artifact, cognition. Those categories are not explored at all in the AHFD as they are probably too abstract for children.

Among the categories defined, many are not defined in a typical genus/differentia type of definition. We would expect that as these categories are very general and they are very near the unique root (everything) of the type hierarchy.

But there are many much more specific concepts in the AHFD that are not defined with the typical genus/differentia rule and they do not find a place in the AHFD hierarchy (besides being a subclass of **something** the superclass of all nouns). Some of these words are: **ash**, **gift**, **habit**, **interest**, **invention**, **lie**, **load**, **luck**, **mistake**, **nest**, **package**, **song**, **sugar**, **language**, **money**, **danger** and **heat**.

Classifying those nouns into the WordNet set of categories is not an easy task and it would probably not be any easier in any categorial system. That makes us wonder then why we would insist on doing such a task, and if it would be any useful

Table 3.6: WordNet hierarchies and AHFD's hierarchies

WordNet roots	Description in the children's AHFD
{act, action, activity}	Root for all verbs.
{animal, fauna}	Defined as Anything alive that is not a plant. Root of large subgraph containing all the animals.
{artifact}	Not defined. ** See natural object.
{attribute, property}	Not defined. Root for all adjectives.
{body, corpus}	Defined as The body of a person is the part you can see and touch. Body forms its own network of Part-of relations as many nouns are defined as part-of the body: arm, back, brain, chest, ear, foot, etc.
{cognition, knowledge}	Not defined.
{communication}	Not defined. Found via defining formula group-of words that have a purpose.
{event, happening}	Not defined. Found via defining formula is when A chase is when someone follows something quickly.
{feeling, emotion}	Not defined. Found via defining formula X is what you feel ...
{food}	Defined as what people or animals eat. Food forms a large hierarchy including: bread, butter, chocolate, meat, etc.
{group, collection}	A group is a number of people or things together. Found via defining formulas more than one, a group of
{location, place}	Defined as A place is somewhere for something to be. It forms a very large hierarchy including: airport, building, city, forest, jungle, etc.

Table 3.6: WordNet hierarchies and AHFD's hierarchies (continued)

WordNet roots	Description in the children's AHFD
{motive}	In AHFD, the only word corresponding to this category is reason. A reason is why something is so. Found via defining formula: is why
{natural object}	Defined as Anything that people can see or touch that is not alive The distinction between artifacts and natural objects is not made in AHFD. There is a large tree under object including: anchor, furniture, moon, rock, toy
{natural phenomenon}	This category is not distinguished from any general event.
{person, human being}	Defined as A person is a man, woman, boy, or girl. Person forms a large hierarchy including: artist, chief, engineer, guard, man etc.
{plant, flora}	Defined as anything alive that is not a person or an animal. Plant forms a large hierarchy including: bush, grass, mushroom, tree, etc.
{possession}	Not defined. Found via defining formula: X is what Y has ...
{process}	
{quantity, amount}	Defined as An amount is how much there is of something. Found via defining formulas: is an amount of.
{relation}	
{shape}	Defined as The shape of something is what it is like on the outside. Shape forms a small hierarchy including: cross, diamond, oval, round, star, etc.
{state, condition}	Not defined.
{substance}	Not defined.
{time}	Defined as how long it takes for something to happen. Time forms an average hierarchy including: day, hour, minute, week, etc.

to an LKB system. Inheritance and type generalization are some of the reason for building a type hierarchy. An NLP system needs help to analyze sentences that might be ambiguous. One way to disambiguate a word, if we cannot find any help within its description, is to look into the description of its superclass. However, there are nouns (as in the list above) that are difficult to classify. We can be sure that a text or discussion using such a noun would probably never use its superclass to talk about it. The text would use the noun itself, probably because it is the only term that could mean exactly what the author wants. A more general term would be far too general.

To have an idea of what an adult dictionary would say about the previous list of words, we looked in the American Heritage Dictionary and found the following genuses: ash/residue of combustion, gift/something, habit/pattern of behavior, interest/something, invention/{device,method,process}, lie/falsehood, load/material, luck(chance)/abstract nature, mistake(error)/unintentional deviation, nest/{container,shelter}, package(bundle)/something, song/composition, sugar/crystalline carbohydrates, language/use, money/commodity, danger/{cause,chance of harm}, heat/effect. The words in parenthesis are synonyms of the entry words for which the genus is found.

By qualitatively judging the interest of these genuses, we would agree that knowing about the superclass of each word in the list is not very useful. This leads us to the following section, in which we comment on other ways to access words in an LKB, and on the usefulness of the taxonomy.

3.2.6 Relative importance of hypernyms

One interesting aspect mentioned in Amsler [6], as well as in Copestake [48] is that taxonomies are limited to relationships between words or phrases having the same part of speech. It seems like dictionary definitions try to respect this rule, in the sense that they will define a noun by a noun phrase definition, or a verb by a verb phrase definition. In certain cases, this can make the definition more obscure by a choice of a meaningless genus. Some noun definitions [6] can be text descriptions of case arguments of verbs or relationships to other nouns. For example, vehicle : a means of carrying or transporting something. The noun vehicle exists mainly in a instrument

relation to a verb **carry, transport**. If we follow a strict noun taxonomy, a vehicle will be a subclass of a means.

In [139], as described in Table 2.5 in section 2.4.3, those examples correspond to the **shunter** type. There was two patterns <N1 of VP> and <N1 rel_pron VP>. The shunter using the relative pronoun corresponds here to a verb with a case argument (**a person who VP, a place where VP, an object that VP**). The shunter with the preposition **of** can also be a case argument, as shown in the previous example with **vehicle**.

To accomplish this task of explaining noun phrases by noun phrases often only renders the definitions more complex than they should be.

> *... the lexicographers must use nominalizations and other morphological transformations to incorporate sentences about the word being defined into the noun phrases used in the single definitional phrase.* [6]

In the AHFD, the classification of a noun via a genus is not always its most important feature. In some definitions, the first sentence describing the genus is completely missing; as if the lexicographers went to a more essential characteristic of the defined word, like its usage or purpose, something that would be more essential than trying to give a genus that might be confusing for a child.

A definition in which genus/differentia pattern is not present will put the noun in relation to other parts of speech, often as a case relation to a verb, as its typical object, agent or instrument. The genus is replaced by an over-general term, like **something** or **what** and the focus is on the action. Definition 3.2.1 shows a few sentences following that pattern.

Definition 3.2.1 -

Food is what people or animals eat.
A habit is something you do often.
Hair is what grows on your head.
A pan is something to cook in.
Sound is anything that you hear.

Table 3.7: Comparison of definitions from the AHFD and AHD

	AHFD: case relation	AHD: superclass
food	object(eat)	substance
habit	object(do often)	pattern of behavior
hair	agent(grows)	outgrowth
pan	instrument(cook)	container
sound	object(hear)	vibratory disturbance

Definition 3.2.2 shows how those nouns are defined in the adult version of American Heritage Dictionary (AHD). The one sense (written in parenthesis) the most similar to the AHFD definition was chosen.

Definition 3.2.2 -

Food(1): A substance taken in and assimilated by an organism to maintain life and growth; nourishment.

Habit(1): A pattern of behavior acquired by frequent repetition.

Hair(1): A fine, threadlike outgrowth, especially from the skin of a mammal.

Pan(1): A wide, shallow, open container for household purposes.

Sound(1a): A vibratory disturbance, with frequency in the approximate range between 20 and 20,000 cycles per second, capable of being heard.

The adult dictionary will usually provides a genus to the expense of getting into complicated sentence structures, as well as sometimes finding obscure nominalizations (e.g. vibratory disturbance). Table 3.7 shows the different emphasis given by both dictionaries. When we compare the complicated adult's definitions of nouns to the simple ones in the AHFD, we feel the AHFD gives a more understandable and useful definition as needed for most common daily dialogs. A good reason to build an LKB from this information is that it could be useful to an NLP system processing simple daily dialogues.

These cases lead us to the idea of covert categories identified through repeated phrasal patterns found in the AHFD. We present covert categories in the following

section.

3.2.7 Covert categories

Building the taxonomy using is-a relations gives us one representation of the concept world; one in which all words are compared to other words with respect to the hypernym relation. If two words have a common hypernym they will be close in the tree.

Here we introduce the idea of using **covert categories** in our LKB [17]. The notion of covert category is discussed in the book *Lexical Semantics* [51]. A covert category is really just an unnamed concept.

> *The lexical items in a taxonomic hierarchy may be considered to be labels for nodes in a parallel hierarchy of conceptual categories. Now while the existence of a label without a corresponding conceptual category must be regarded as highly unlikely, it is not impossible for what is intuitively recognized as a conceptual category without a label. Sometimes there may be clear linguistic evidence for the existence of the unlabeled category.* [51]

As linguistic evidence, Cruse suggests a sentence frame for a normal sentence containing a variable X where we determine a list of items that X could be. For example, given the sentence frame **John looked at the X to see what time it was.** we could generate the list **clock, watch, alarm clock** as possible values for X. Cruse calls these categories with no names, but for whose existence there is definite evidence, covert categories, and he mentions that they most frequently occur at the higher levels of a hierarchy.

Covert categories are often present in the AHFD, either because the label is not part of the vocabulary taught to the child, or because the label does not exist in the English language. Consider the **vehicle** category, which does have a label in English, but not in the children's world of the AHFD. A **vehicle** in the adult AHD is defined as **A means to carry people, loads and goods.** In the AHFD, to search for the concept of **vehicle**, we seek for an action of **carrying**. The sentence frame could be **X carries/carry**

Table 3.8: Definitions with pattern (X carry people/load)

An **airplane** is a machine with wings that flies in the air. **Airplanes** carry people from one place to another.
A **balloon** is a kind of bag filled with gas. Some **balloons** are huge and can carry people high into the sky.
A **boat** carries people and things on the water.
A **bus** is a machine. **Buses** carry many people from one place to another.
A **camel** is a large animal. **Camels** can carry people and things across the desert.
A **donkey** is an animal. **Donkeys** can carry heavy loads.
A **helicopter** is a machine. It carries people through the air.
A **ship** is a big boat. Large **ships** can carry many people across the ocean.
A **subway** is a train that travels through tunnels underground. **Subways** carry people through large cities.
A **train** is a group of railroad cars. **Trains** carry heavy loads from one place to another.
A **truck** is a machine. It is a very large car that is used to carry heavy loads.
A **wagon** is used to carry people or things from one place to another.

people/loads. The word goods is not in the AHFD so we will not use it. Table 3.8 shows the definitions from the AHFD that would match this sentence frame.

So there we have a category X including airplane, balloon, boat, bus, camel, donkey, helicopter, ship, subway, train, truck, wagon. All those lexical units have in common of carrying people or loads. Some of the vehicles found were already assigned a genus like airplane, bus, helicopter and truck which are machines. Some others like camel and donkey are animals. But others did not have any category assigned such as boat, train and wagon.

The notion of carrying people or loads corresponds to a covert category which is shown in Figure 3.9.

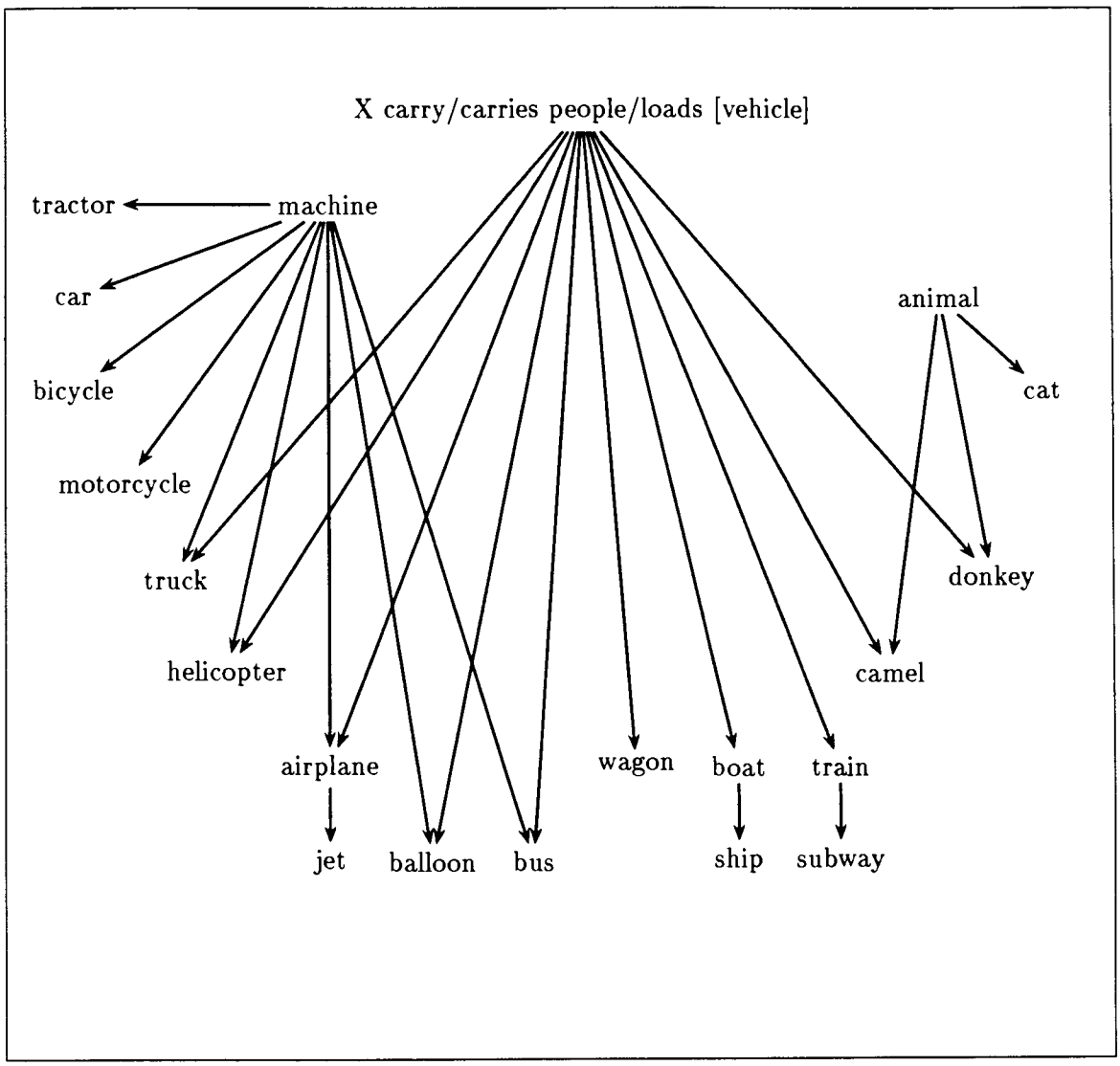


Figure 3.9: Part of the noun hierarchy including **vehicle**.

A few other nouns that we know as vehicles are defined in the AHFD, like **automobile, bicycle, jet, motorcycle, tractor, trailer** but they are not defined as **carrying people/loads.** **Jet** is a subclass of **airplane** so it will inherit the vehicle's properties through **airplane** which is a **vehicle.** The others are subclasses of **machine.** We see them in Figure 3.9.

Another covert category given by the sentence frame **X have engine**, gives us the following words: **boat, bus, car, ship, airplane, jet, motorcycle, tractor, tugboat.** Among these eight words, four of them are also in the covert category **X carry people/load.** The covert category given by the sentence frame **X have wheels**, gives us the following words: **bicycle, bus, car, roller skate, suitcase, tractor, wagon.** Among these eight words, three are also under **X carry people/load** and three are also under **X have engine.**

It is possible that an investigation into the overlaps between groups of words which partially share the same hypernyms could give us more insight into semantic distance. Let us briefly consider this in the context of the words we have been working with thus far, for which we have three sets:

Set A -	carry people/load:	**{airplane, balloon, boat, bus, camel, donkey, helicopter, ship, subway, train, truck, wagon}**
Set B -	have engine:	**{airplane, boat, bus, car, ship, jet, motorcycle, tractor, tugboat}**
Set C -	have wheels:	**{bicycle, bus, car, roller skate, suitcase, tractor, wagon}**

By partially sharing the same hypernyms, we mean that the intersection between two groups of words under different hypernyms is not empty. There is an intersection between Set A and Set B containing four elements **{boat, bus, airplane, ship}**. The fact that they share a large percentage of their elements shows some similarity between Set A and Set B, and their associated concepts.

In which case can we extend that second hypernym to cover the members of the first group as well? Can we assume that all members of Set A have an engine as well? How similar to each other are the members of group that do not share the hypernyms? How similar is a motorcycle (set B) to an helicopter (set A)? Or how similar is a roller skate (set C) to a train (set A)?

This leads us toward the idea of componential analysis [143] which divides the world into groups based on what attributes the members share. It would be a very

nice step to bring together the field of semantic networks [144, 127, 129] in which all words are not decomposed into features but are defined via their relations to other words, and the field of componential analysis in which words are not related as entities to other words but are rather seen as an agglomeration of attributes. Such an integration is beyond the scope of this thesis.

Finding covert categories within the Conceptual Graph representation

For building the taxonomy automatically, finding the covert categories requires more work than finding the genus of a definition. The categories are hidden through phrasal patterns which are repeated among many definitions with maybe some variants. Those patterns are not known in advance and have to be found through comparisons of the CG representations of sentences.

As those patterns are usually centered around a verb, we decided to take each verb and build a list of all possible immediate relations for each of them. This includes agent, object, location, instrument, time, as well as all the prepositions that might not have been disambiguated.

To do so we build a general graph (G1) around a verb with a relation r1 that will vary over all possible relations, and a very general concept **everything** that subsumes all concepts in the concept lattice and therefore could be match to any of them.

For example with the verb **carry**, we have:

G1 = [carry]->(r1)->[everything].

Using graph matching techniques, we project G1 onto all the graphs constructed from the dictionary definitions. As a result, we will have a list of all projections; meaning all subgraphs that are more specific than G1. We find seventeen occurrences of the projection where r1 is specialized to agent, and thirty-two occurrences where r1 is specialized to object. This generates two covert categories: **carry~agent** and **carry~object**. We call them level 1 categories.

If there are multiple occurrences with the same concept occupying a case role (as agent or object), we refine our graph and try the projection again. For example, we found the concept person in eight occurrences with r1 specialized to object. We

create graph (G2) that contains relation r1 that we can specialize again to all possible relations.

G2 = [everything]<-(r1)<-[carry]->(object)->[person].

The eight occurrences found when r1 is specialized to agent in G2 are: **airplane, balloon, boat, bus, camel, helicopter, ship, subway.**

This generates one more covert category: **carry~object~person~agent.** We establish a hierarchy on the covert categories as well. The new category is a subtype of category **carry~object**, it refines it to only carrying people and not anything. It is now a level 2 category.

When the number of occurrences of a projection exceeds a certain threshold (let us call it Covert Threshold, CT), we use it to define a covert category. Later we will be discussing different possible values for CT and the resulting covert categories. The λ-abstraction mechanism of CGs are used here to label and define a new type concept. For the two previous covert categories described, we generate two type concepts named **carry~agent** and **carry~object~person~agent.** Arbitrary names can be used as labels; we prefer to choose a name that is a good reminder of what the covert category expresses. The label could later correspond to an existing word, since we saw that some words are not yet part of the child's vocabulary. We assign each label to a λ-abstraction given by the projection we found.

carry~agent(*X) is
    [*X]<-(agent)<-[carry]

carry~object~people~agent(*Y) is
    [*Y]<-(agent)<-[carry]->(object)->[people]

The hierarchy is updated so that the new type concepts become hypernyms of the group of words they subsume.

Selectional restriction

We saw that finding a covert category consists of finding a graph that subsumes other graphs. By projecting Graph_A: [eat]->(object)->[everything] on all the graphs in the

LKB, we find multiple graphs subsumed by Graph_A that contain a specific concept subsumed by **everything**. The set of concepts subsumed by **everything** represents the labeled category **food** as all the individual concepts are subclasses of the class **food**. In that case, we can put a selectional restriction on a case role for the verb examined.

> *A selectional restriction is a "co-occurrence" constraint which is possessed by senses of words of certain parts of speech and which restricts the semantic classes of words with which it co-occurs.* [62]

Thus the selectional restriction of the **object** relation for **eat** is of class **food**. Furthermore, the class **food** is associated with the type concept labeled eat~object(*X) defined with the λ-abstraction **[eat]->(object)->[*X]**. We have a reciprocal relationship between the eating concept and the food concept, expressed through the interrelation between the selectional restriction and the covert category.

In [51], a selectional restriction is defined as a presupposition of a selector whose non-satisfaction leads to a paradox or incongruity.

For example the word **drink** could have a selectional restriction for its object relation to be a liquid. If we see **drink a liquid**, **liquid** is a tautonym of the head, as the combination of **drink** and **liquid** is pleonastic, its superordinates **substance** or **fluid** are also tautonym, but all hyponyms **beer**, **water** are philonyms, meaning that they lead to a normal reading.

Not mentioned in Cruse are the cases where some hyponyms of the selectional restriction reading would not give a normal reading, such as **drink gasoline**. **Gasoline** is an hyponym of **liquid**, but does not give a normal reading. In our finding covert categories, we find selectional restrictions by exhaustive search. Only concepts that appear as object of **drink** can become an hyponym of the covert category **[drink]->(object)->[]**. If all hyponyms of that covert category are hyponyms of liquid, the covert category will become itself an hyponym of liquid.

By indicating selectional restrictions, covert categories will help solve some structural ambiguities, such as prepositional attachment. For example, the covert category **see~with** can have as hyponyms: **eyes**, **telescope**, and **binoculars**. When encountering

the sentence He saw the girl with a scarf, there should be less likely to assign with a scarf to the action of seeing.

Selectional restriction is addressed in earlier work on discovering semantic relations among word sense [33]. It is also addressed in the context of acquisition from text corpora [69]. In LDOCE, or other adult dictionaries, the "typical" object, subject or instrument is sometimes (but not consistently) written in parenthesis within the definition. The selectional restriction for some verbs can be performed simply by looking at the word in the parenthesis.

3.2.8 In Summary

We presented some ideas from Cruse on what a taxonomy should be. Then we looked at some work on knowledge extraction from dictionaries which give a looser definition to the taxonomy, calling it an is-a hierarchy (concept lattice) and relating headwords with the genus of definitions. We have also shown part of the taxonomy extracted from the AHFD and talked about including sets in the hierarchy and the necessity of the hierarchy being tangled.

We presented covert categories as a way to find similarity between words [76]. We think they open more dimensions of comparison between words. Via the type concepts defined with λ-abstraction, it is possible to include these covert categories within the concept lattice already constructed by extraction of is-a relations.

We have mentioned numerous times the dependency of graph matching on the concept lattice, as we use the lattice to decide if two concepts in two different graphs or within the same graph are compatible. The concept lattice is our main source of information to establish the similarity between concepts.

Two concepts (C1,C2) are close if one subsumes the other in the type hierarchy. The type hierarchy can also be used to find a concept C that subsumes C1 and C2, and then the semantic distance is defined as the path length going from C1 to C to C2 [64] or as the informativeness of C [116]. If we are to use our concept hierarchy to establish the similarity between pen and crayon, we find that one is a subclass of tool and the other of wax, both then are subsumed by the general concept something. We

have reached the root of the noun tree in the concept hierarchy and this would give a similarity of 0 based on the informativeness notion.

These two concepts on the other hand have the similarity of both being instruments of writing. By sharing this information, a pen is more similar to a crayon than it is to a pumpkin (see [78] for concept similarity based on common features). The pumpkin is probably more similar to a carrot than it is to a broccoli. We can find similarity at many levels, and limiting ourselves to the type hierarchy does not seem adequate. By finding covert categories, we add more dimensions to the taxonomy, more ways to find similarities between objects.

Covert categories can be related to some extent to the work of [98, 99]. They use a method called MSG (Method of knowledge Structuring by Generalization) based on conceptual graph description of concepts. They start with a set of concepts, each defined with a graph definition and each put in a taxonomy. They find new categories by extracting the common features of multiple definitions and reorganize the type hierarchy by adding terms in it that represent the new established categories. We consider our covert categories as unnamed concepts and give them an arbitrary label, but they assume their new categories as possibly named by a knowledge expert. One main difference with our work is that before they use their generalization algorithm, the graph representation of each definition is first transformed into a set of triplets (Concept-Relation-Concept) and therefore their generalization process works only for one relation at a time. It corresponds to our Level 1 categories. They call their overall process "conceptual clustering". Other works [9, 28] are also using this term for a very similar approach.

We use the term concept cluster in section 3.4 for quite a different approach. What is interesting though about the work of Mineau and Allouche is the larger goal in which their research is situated which brings a new light on our work. They see such a process of vocabulary extension as facilitating vocabulary integration between different applications, and knowledge sharing between multi-agent processes. In our conclusion section, we mention future research into the area of lexical gaps. It is similar to non-compatibility between the vocabulary used by different agents wanting to communicate. For lexical gaps it is often the case that we are talking about

vocabulary from different languages.

Covert categories can also be related to the idea of establishing the similarity between items by finding the shortest path between them in a semantic network [79, 57]. Let us imagine that all our CGs for all the definitions become interconnected into a large network. Two hyponyms of **eat~object**, such as **fruit** and **potato** would be at distance 2 (or some proportionally related distance) as the path between them is:

[fruit]<-(object)<-[eat]->(object)->[potato].

The problem is that **fork** would be at the same distance from [fruit] because of **[fruit]<-(object)<-[eat]->(with)->[fork]**. Richardson [118] does an interesting investigation into finding particular paths which are indicators of similarity. He uses the thesaurus to define in advance similar words and then finds the best possible paths between them (he developed as well a measure of the weight of different paths). Among the 20 most frequent paths are: Hyp-HypOf (apple-fruit-orange), Tobj-TobjOf-Hyp (apple-eat-orange-fruit), Hyp-Hyp (castle-building-place). We see that the hypernym (Hyp as well as its opposite HypOf) plays an important role in these similarity path. In the CG formalism, they are taken care of by the concept hierarchy and are always considered as we perform graph matching. The covert categories at level 1 correspond to symmetric paths such as Tobj-TobjOf, or Manner-MannerOf. One advantage of the path similarity measure is that a path of any length can be considered, but its disadvantage is that a path does not branch out and therefore cannot find covert categories at level 2. For example, we are able to find the covert category **carry~instrument** at level 1, which has for hyponyms **arm, bucket, pipe, car, tube, wagon** and which could be found by a path Inst-InstOf. Now, we refine to level 2 and find the covert category **carry~object~liquid~instrument**. There is a closer similarity between **bucket, pipe** and **tube** which cannot be expressed by a path.

3.3 Relation taxonomy

In section 2.4.3 we briefly presented all the relations we are using in our LKB. Appendix E gives a complete detailed list of all relations. We grouped all relations into a few classes, such as case relations, agent involvement, comparison relations, etc. Within these groups, we listed the relations without giving any structure as to which relations are more closely related than others.

It is important to establish which relations are more similar, as it makes them more likely to be compatible during graph matching. We use graph matching for multiple purposes throughout this thesis, and the relation taxonomy will influence the matching process.

We present in Table 3.9 the compatibilities that we will accept as we perform graph matching. An arbitrary label is given to a relation that serves as a superclass of other semantic relations that we have defined earlier unless there is already a more general relation within the group that can be used as superclass.

We present the compatibilities as sets, but they will be represented in the taxonomy by having the given label as a superclass of all elements in each set.

Some superclass/subclass relations imply looking at the concepts in reverse order. For example [John]<-(agent)<-[eat], and [John]->(desire)->[eat] can be related if we match the opposite concepts. This is noted by (r) in Table 3.9.

In section 2.4.3 we mentioned that we have a set of 44+ semantic relations. The "+" is there for all the prepositions that are used as relations until we can later disambiguate them. A preposition that can be disambiguated into more specific semantic relations is considered in the relation taxonomy to be a superclass of these possible relations. Table 3.10 gives the prepositions that must be disambiguated into semantic relations. These prepositions are defined in the AHFD as having more than one sense. The examples given in Table 3.10 are taken directly from the definitions of the prepositions in the AHFD. We added a few more examples taken from other definitions in the AHFD when we felt that there were more possible meanings than the ones given in the preposition's definition. We do not intend to give a complete list of possible relations given by prepositions, as some more "abstract" semantic relations such as

Table 3.9: Compatible semantic relations

Subclass of relation	Grouping found
part/whole	$Group_1$ = {part-of, content, material}
	$Group_2$ = $Group_1$ ∪ {piece-of, area-of, amount-of}
human/animal	$Group_3$ = {possession, child-of}
comparison	$Group_4$ = {like,as,more-than,less-than}
description	$Group_{attribute}$ = {material, function, about}
action modifier	$Group_{modif}$ = {manner, frequency}
case roles	$Group_5$ = {method, manner}
	$Group_6$ = {agent, experiencer}
	$Group_7$ = {agent, instrument}
	$Group_{location}$ = {direction,source,destination,path}
	$Group_{time}$ = {point in time, frequency, during}
	$Group_8$ = {result,cause,transformation,goal}
agent involvement	$Group_{agent}$(r) = {ability, desired act, intention, obligation}
type of action	$Group_9$ = {act, event, process}

state or **circumstance** are not part of the children's world and therefore not part of the LKB built. For a review on the possible meanings of a set of twelve prepositions, see [55].

For each example in Table 3.10 we indicate from which word's definition a sentence is taken.

As for the hierarchy expressed in Table 3.9, some superclass/subclass relations between a preposition and a semantic relation require a reversal of the the order of the concepts. For example we have **[trunk]->(of)->[tree]**, that can subsume **[tree]->(part-of)->[trunk]**. We note these reversals in Table 3.10 with an (r).

Note that some prepositions such as **above, across, over**, are not generalizations of multiple semantic relations but instead are specializations of some relations mostly related to time and location, as shown in Table 3.11.

We have extracted the possible meanings of prepositions by looking through the AHFD. By no means do we assert that this is an exhaustive search through all the possible meanings of prepositions.

At its initial stage, the relation taxonomy is built manually but it should be dynamic, ready to be modified by new relations. Some argumentation in favor of a

Table 3.10: Ambiguous prepositions

PREPO-SITION	*SEMANTIC RELATION*	*EXAMPLE* (Word from which example is extracted in AHFD)
about	ABOUT	The story is about Lisa and David. (about_1)
	MODIF	That line is about four inches long. (about_2)
along	LOCATION	It is nice to walk along the beach. (along_1)
	ACCOMPANIMENT	Donna went to the store and her brother went along with her. (along_2)
at	LOCATION	She is at school now. (at_1)
	POINT IN TIME	School begins at nine o'clock. (at_2)
	DIRECTION	Jerry looked at the sky to watch for falling stars. (at_3)
before	SEQUENCE	Minny washes her hands before she eats. (before_1)
	POINT IN TIME	She had never been on a plane before. (before_2)
by	AGENT	The question was asked by Dana. (by_1)
	MANNER	We made a garden by planting some flowers. (by_2)
	POINT IN TIME	Nicole is usually hungry by supper. (by_3)
for	FUNCTION	A carpenter has a box for his tools. (for_1)
	RECIPIENT	I bought this book for you. (for_1)
	DIRECTION	People can reach for the sky, but they can't touch it. (for_2)
	DURING	We played baseball for two hours. (for_3)
from	SOURCE	Bob went from school to the library. (from_1)
	LOCATION	The moon is a long way from the earth. (from_2)
in	LOCATION	Fish swim in the water. (in_1)
	POINT IN TIME	Ted's birthday is in August. (in_2)
	MANNER	Ants live in large groups ... (ant)
	PART-OF (r)	The people in this story are Lisa and David. (about_1)
into	DIRECTION	My father drives the car into the garage. (into_1)
	RESULT	Caterpillars change into moths and butterflies. (into_2)
of	PART-OF (r)	Branches grow out from the trunk of a tree. (branch)
	CONTENT	Paul was carrying a pail of water. (of_2)
	POINT IN TIME	The time is ten minutes of four. (of_3)
	ABOUT	Steve drew a picture of his brother. (add_1)
	POSSESS (r)	You can see the work of many artists in a museum. (artist)
	MATERIAL	We made a border of stones around the garden. (border)
on	LOCATION	The dishes are on the table. (on_2)
	ABOUT	Thea has a book on dinosaurs. (on_3)
	POINT IN TIME	We play ball on Sundays. (on_4)
through	PATH	Jess walked through a field to get to school. (through_1)
	MANNER	Many people send messages through the mail. (message)
to	DIRECTION	Astronauts have flown to the moon. (to_1)
	POINT IN TIME	It(the store) is open from nine to six. (to_2)
	TRANSFORMATION	He changed the walls from yellow to white. (to_3)
with	CONTENT	Joe's hamburger came with onions on it. (with_1)
	PART-OF	A giraffe is an animal with a long neck. (with_2)
	INSTRUMENT	Brian dug a hole with a shovel. (with_3)
	MANNER	Attention is looking and listening with care. (attention)
	ACCOMPANIMENT	To march with someone means ... (march)

Table 3.11: Temporal and location subclasses

SEMANTIC RELATION	*PREPOSITION*	*EXAMPLE*
LOCATION	above	Airplanes fly above the ground.
	against	Kathy put her bicycle against the wall.
	around	Sandra looked around the room for her shoes.
	behind	Julio stood behind Sally.
	below	Roots grow below the ground.
	beside	Kim sits beside Don and Don sits beside Kim.
	between	It(wrist) is between your hand and your arm.
	inside	Then she went inside the house to get warm.
	off	Please take your books off the table before supper.
	over	A helicopter flew over our house.
	under	The roots of a plant grow under the ground.
	upon	The bird was sitting upon the branch.
PATH	across	A bridge was built across the river.
	beyond	She went beyond the fence.
DIRECTION	down	A big balloon came down in our garden.
	out	Joe went out the door and closed it behind him.
	toward	The ship seemed to grow bigger as it sailed toward us.
	up	We went up in a balloon.
SEQUENCE	after	After the ball game, we went home.
	before	February comes after January and before March.

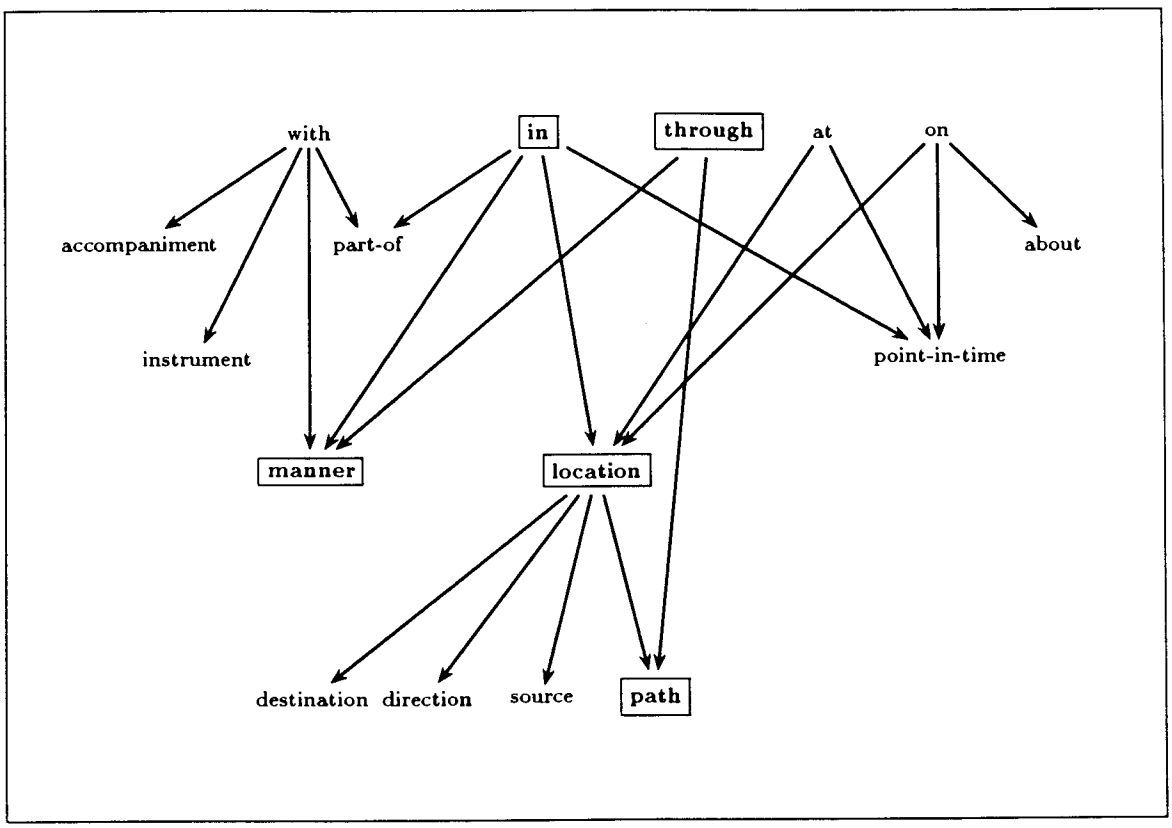


Figure 3.10: Small part of relation taxonomy.

non-static set of relations, arranged into a hierarchy, is given in [106]. They argue that it might be desirable at some point to add new relations that were perhaps not accorded much attention before. The level of detail for differentiating relations will depend on the domain of application, and if our relation set is organized into a hierarchy, any branch can be expanded at any time without affecting the overall system. For example, in [119] they position the case relations from Fillmore [63] in a taxonomy and refine each case as a more specific relation.

Building the hierarchy will allow us to compare graphs expressing similar ideas but using different sentence patterns that are reflected in the graphs by different prepositions becoming relations. If multiple prepositions can be specialized to the same deeper semantic relation, we can find this semantic relation as we try to join or compare graphs containing these different prepositions as relations connecting compatible concepts. This claim assumes that not all prepositions can represent all semantic relations. For example, if we see *in* **the mail** and **through the mail**, we can establish the similarity via the **manner** relation. They interact at **location** as well but more indirectly. Figure 3.10 shows a small part of the relation hierarchy, and we highlighted the compatibility between **through** and **in**.

3.4 Concept clusters

In this section, we describe word clusters, the last but very important part of our LKB. Word clusters allow the gathering of information about specific subjects. They are like "micro-worlds" in which the relations among multiple participants and actions are described [18].

Our idea of clustering is quite different from the many recent efforts in finding words that are semantically close. These other approaches [43, 142, 31, 108] determine word clusters based on co-occurrences of words in text corpora or dictionaries. The problem with such fully automatic and probabilistic approaches is that the linguistic plausibility of the resulting clusters cannot be evaluated [20]. Clusters of words are found, but it cannot be explained why the words appear in the same cluster and how they are related.

The term conceptual clustering has also been used to find groups of objects with similar features [99, 9, 28].

Our work on word clusters is in some respect similar to the work of Schank [122], where he introduces the idea of a Situational Memory (SM) containing information about specific situations. In his understanding process, information found in SM is used to provide the overall context for a situation, as well as the rules and standard experiences associated with a given situation in general.

Schank calls his memory structures at the situational memory level Memory Organization Packets (MOPs) A MOP is a bundle of memories organized around a particular subject that can be brought in to aid in the processing of new inputs. The key to understanding is the continual creation of MOPs which record the essential parts of the similarities in experience of different episodes. A MOP can help in the processing of a new event and it is itself affected by the new event.

We think the concept clusters will be helpful when we further analyze text and refer to our LKB (to find implicit information for example). The process of building the concept clusters is very important as well. The gathering of information plays an important role in disambiguating information contained in the LKB. The idea is that more is better; finding redundant information can lead to some clarifications.

In chapter 2, we showed the transformation from a sentence to a CG containing surface semantic and then we explored multiple structural and semantic disambiguation processes. In the end, ambiguity still remains, and sometimes the representation is still close to surface semantics. By gathering information from multiple definitions within the same small domain, we will find definitions sharing parts of information. One definition might be less ambiguous than the other and this will help us disambiguate the information in the other definition. For example, definition 1 contains an ambiguous word and definition 2 contains the same word with a sense assigned; if that word is part of the knowledge shared by both definitions, it allows us to disambiguate the word in the first definition.

The large graph that will result from merging the information from multiple definitions is a Concept Cluster Knowledge Graph (CCKG) which has the role of structuring as much information as possible about a particular topic. Each CCKG will start as

a CG representation of a trigger word and will expand following a search algorithm to incorporate related words and form a concept cluster. The concept cluster is interesting in itself as it identifies a set of related words, and these sets have been used in other research for tasks such as information retrieval or word sense disambiguation. Here the CCKG gives more to that cluster as it shows how the words are related. To go back to earlier work by Schank, we can say that a CCKG is in some ways similar to a script [121]. However, a CCKG is generated automatically and does not rely on primitives as Schank's work does but on a large number of concepts, showing objects, persons, and actions interacting with each other. This interaction will be set within a particular narrow domain (like a script), and the trigger word should be a keyword of the intended domain. For example, for the **shopping** domain, appropriate trigger words would include **shop, store, buy**. If we take a more general term such as **money**, we will not only go into the shopping domain, but also look at banks, jobs, types of money, etc.

This section describes how given a trigger word, we perform a series of forward and backward searches in the dictionary to build a CCKG containing useful information pertaining to the trigger word and to closely related words. The primary building blocks for the CCKG are the graphs built from the dictionary definitions of those words using our transformation process described in chapter 2 and presented as the first part of our LKB in section 3.1. Those graphs express similar or related ideas in different ways and with different levels of detail. As we will try to put all this information together into one large graph, we must first find what information the various graphs have in common and then join them around this common knowledge.

An important aspect of the clustering is the idea that some paths are worth exploring more than others. We think this decision should be based on how significant a concept is with relation to its surrounding context. The idea of finding Semantically Significant Words (SSWs) is an important factor at the base of the clustering process.

To build a CCKG and perform our integration process, we will rely on our two main knowledge structures described earlier as the second and third parts of the LKB, the concept lattice and the relation lattice. We use the graph matching procedure mentioned earlier, but it will be described in more detail here. Once the common

Table 3.12: SSWs and number of occurrences

Max number of occurrences to be considered a SSW	% of occurrences discarded	% words discarded
5076	10	0
2904	20	0
1095	30	0
623	40	0
264	50	1
126	60	2
62	70	6
42	75	8
35	77	10
27	80	13
18	85	19
11	90	29
5	95	48
4	97	60
2	99	80
1	100	100

subgraphs between two graphs are found, the joining of the remaining information particular to each graph is performed around that common information.

This section will process as follows. First we establish which words can be called semantically significant words. Second we give details of doing graph matching. Third, we present how to join two graphs around their maximal common subgraph. Fourth, we give an overview of our clustering process. We end with a discussion.

3.4.1 Semantically Significant Words

The semantic weight of a word or its informativeness can be related to its frequency as we saw earlier in section 3.1.2 (see also [116]). We should add that the semantic weight of a word is related to the context in which it is used, and that measuring the frequency of a word should be done with respect to a particular context.[3]

[3]Richardson [118] mentions the work of Salton *et al.* [120] who uses a term weight based on the term frequency and the inverse document frequency, which means that they assign a weigh to a term

Here we assume the context to be the simplified world described in the definitions of the AHFD, and we calculated the number of occurrences of each word within that context. In table 3.12, we can interpret the third column as giving us the percentage of different words needed to create the corresponding percentage of word occurrences shown in column 2. We find that 10% of the possible words in AHFD account for 77% of word occurrences. A small 1% of the possible words accounts for 50% of word occurrences, and 48% of the possible words account for 95% of all occurrences. Among the most frequent words in the AHFD, we have (with their number of occurrences in parenthesis): a(5106), the(1619), in(664), something(341), person(429), have(278), many(157), animal(116). And among the least frequent words, we have: haircut(3), palm(3), lumber(2), emerald(2), traffic(2), dentist(1), classroom(1).

We define as a semantically significant word (SSW), a word which occurs less often in the AHFD than a set threshold. Table 3.12 gives us the number of word occurrences, and the number of a different words that are eliminated for different thresholds on the SSWs. For example, if a word must occur less than 27 times to be considered semantically significant, and we use that criteria to restrict our search of words for our clustering process, than we are eliminating 80% of all word occurrences and 13% of the possible words in the dictionary on which the search will not be done. When trying to give more context to a word, the last thing we want is to investigate words such as **person** or **something** which are probably part of most definitions in the dictionary. By establishing a criteria on the possible words to be explored, we avoid such useless search.

The notion of a semantically significant word is used throughout the rest of this section.

3.4.2 Finding common subgraphs

Establishing overlap in information given by different definitions is performed by finding the common subgraphs between two graphs. This consists of finding a subgraph

(or word) with respect to a single document based on the term's frequency within that document compared to its frequency within a group of documents.

of the first graph that is isomorphic to a subgraph of the second graph. In our case, we cannot often expect to find two graphs that contain an identical subgraph with the exact same relations and concepts. Ideas can be expressed in many ways and we therefore need a more relaxed matching schema.

That relaxed matching schema consists of a looser approach in identifying matching concepts, we call them **compatible concepts**. We also use the possibility of not finding exact relation match but using the relation taxonomy to find **compatible relations**.

Two graphs can have multiple compatible concepts and multiple compatible relations, and the Maximal Common Subgraph (MCS) will be the largest connected subgraph that can be found among all these concepts and relations.

We describe our procedures for finding concept and relation compatibility in the following paragraphs. We also present a measure of similarity between two graphs expressed via their common subgraphs.

Compatible concepts Our approach is an extension to the graph matching procedure described in [126]. Two concepts A and B (noted in parenthesis, present in two separate graphs or within the same graph, are compatible in the following conditions:

- Condition 1: identical concept types
  [drive]->(object)->[car (A)]
  [wash]->(object)->[car (B)]

- Condition 2: concept A subsumes concept B
  [drive]->(object)->[vehicle (A)]
  [wash]->(object)->[car (B)]

- Condition 3: concept A subsumes a certain sense of concept B (which disambiguates B at the same time) With **message** and **letter**, **message** subsumes the second sense of **letter**, **letter_2**, the first sense **letter_1** is a symbol of the alphabet.
  [send]->(object)->[message (A)]
  [write]->(object)->[letter (B)]

- Condition 4: concept A is a pronoun or a very general concept (like something), it is linked via a relation X1 to concept C1; concept B is linked via a relation X2 to concept C2; X1 is compatible with X2 and C1 is compatible with C2

  Example:
  [send(C1)]->(object(X1))->[it (A)]
  [send(C2)]->(object(X2))->[letter_2 (B)]

- Condition 5: concept A does not subsume concept B, concept B does not subsume concept A, but both are subsumed by a third concept C, and C is semantically significant (C=fruit in the following example)
  [eat]->(object)->[banana (A)]
  [like]->(object)->[orange (B)]

- Condition 6: same as Condition 4, but concept A is not a pronoun or a very general concept type, in which case we do not automatically assign them as compatible, but we investigate the **semantic distance** between the two concepts.
  [write]->(instrument)->[crayon (A)]
  [write]->(instrument)->[pencil (B)]

  We describe the investigation hereafter.

We already talked about semantic distance in section 3.2.7. The concept lattice built via the is-a relations extracted from the dictionary is the main source of information about concept similarity. We look first into the concept lattice for subsumption relations. To the concept lattice, we added covert categories. We presented covert categories as another way to find similar concepts as it helps us find concepts that play the same case role to a particular verb. In [20], they look at the opposite categorization problem, finding groups of verbs depending on their relation to nouns. As we incorporate covert categories in the concept lattice, the subsumption process can be used as well.

In many cases, this should be sufficient, but when we have reasons to believe that two concepts might be similar (as in this case being related to the same concept by the same relation), we might want to go one step further and extend the subsumption notion to the graphs. Instead of finding a concept that subsumes two concepts, we will try finding a common subgraph that subsumes the graph representations of both concepts. In our example, *pen* and *crayon* have a common subgraph [write]->(instrument)->[] as part of their respective graph definitions.

The approach here is exactly the same as the one presented in section 3.2.7 for finding covert categories, groups of words that share a particular case role to a verb. We establish a Covert Threshold (CT) to decide if we create a new node in the type hierarchy to correspond to a new type defined via a λ-abstraction. In this particular case, the λ-abstraction [write]->(instrument)->[λ], labeled write~instrument can be found in the taxonomy as a covert category if it occurred more than CT times in the dictionary. If the subgraph [write]->(instrument)->[λ] does not occur often enough it does not become a covert category. But it still represents the common information between pen and crayon, it subsumes both their graph representations.

Compatible relations Two relations A and B are "compatible" in the following conditions:

- Condition 1: relations A and B are identical
  [eat]->(with (A))->[fork]
  [eat]->(with (B))->[knife]

- Condition 2: relation A subsumes relation B in the relation hierarchy
  [eat]->(with (A))->[fork]
  [eat]->(instrument (B))->[knife]

- Condition 3: relation A subsumes relation B with concepts involved in reversal order; we mentioned this factor before in section 3.3
  [branch]->(of (A))->[tree]
  [tree]->(part-of (B))->[branch]

Similarity expressed via common subgraphs Now that we have established the criteria for concept compatibility and relation compatibility, we can define the compatibility between two graphs as expressed by the list of all their compatible concepts, and the list of their compatible relations if these relations are connected to compatible concepts. Therefore the compatibility between two graphs is expressed by a disjoint set of subgraphs, some being unique concepts and others being larger structures of at least two concepts joined by a relation.

Normally, in the CG formalism, the largest common subgraph between two graphs is defined as the largest connected subgraph that subsumes the two graphs. Here we want to keep track not only of the largest connected subgraph, but also all the other isolated concepts that are present in both graphs. Two graphs can be more or less compatible depending on the amount and the significance of the information that they share. Their compatibility establishes their similarity.

Two measures are used: 1) the number of SSWs among the set of compatible subgraphs, and 2) the number of connected concepts in the Maximal Common Subgraph (MCS).

Let us take an example, where after the graph compatibility is performed between two graphs we obtain the following set of subgraphs.

[mail]	*isolated concept*
[post-office]	*isolated concept*
[send]->(from)->[person] ->(to)->[person]	*MCS with 3 connected concepts*

The set of common subgraphs, either isolated concepts or concepts connected via relations, has a total of three SSWs: {**send**, **mail**, **post-office**}. The MCS contains only one SSW **send** and the concept **person** which is not a SSW. Our first measure gives three.

The second measure is the number of concepts within the MCS. To be counted here a concept does not have to be semantically significant. In our example, that measure would also be three.

The total similarity measure is based on both factors: enough significant concepts in common, but as well some shared structure. $GS(a, b)$ gives the graph similarity measure. The two measures described are given by parameters a (number of SSWs in common) and b (number of concepts in the MCS).

3.4.3 Joining two graphs

The next section will describe how to direct our clustering into choosing which new word to add to the cluster. But independent of that choice, the process of expanding the cluster always boils down to joining the cluster-graph (Graph C) with the new-graph (Graph N).

We must find the common subgraphs between Graph C and Graph N using the algorithm proposed in the previous subsection. If the Graph Similarity (GS) between these graphs exceeds a fixed Graph Matching Threshold (GMT) then we perform the join between graphs C and N on the most significant concept of the MCS, and then we simplify the new graph by eliminating redundancy.

As we add more and more graphs to the cluster graph, we want to make sure that we are adding graphs that are related to the cluster graph. By that we mean that we expect them to share a significant part of knowledge. If, for example, they only have the concept **person** in common, that would be meaningless. We want the two graphs to share significant information.

The GMT is defined based on the GS measure given in the previous subsection. It is given in terms of the minimal number of SSWs present in the common subgraphs (parameter a in GS) and the minimal number of relations in the MCS (parameter b in GS).

When the GMT is satisfied between graph C and N, that is $GS_{C/N}(a, b) \geq GMT_{C/N}(c, d)$ ($a \geq c$ and $b \geq d$), the graph similarity between C and N exceeds the threshold, then we perform the join between the cluster-graph and the new-graph. The join is done around one SSW, but which one is not important as all the redundant information will be eliminated in the following simplification steps.

Definition 3.4.1 and definition 3.4.2 show the definitions of mail and stamp respectively. Then we can see their MCS in Result 3.4.1 and their maximal join in Result 3.4.2 which is performed on the concept mail_1.

Definition 3.4.1 -

MAIL_1
Many people send messages through the mail.

```
mail_1_B_A
[send]->(agent)->[person:plural]
      ->(object:expected)->[message:plural]
      ->(through)->[mail_1]
```

Definition 3.4.2 -

STAMP
People buy stamps to put on letters and packages they send through the mail.

```
stamp_1_A_A
[buy]->(agent)->[person:plural]
     ->(object)->[stamp:plural]
     ->(goal)->[put]->(on)->[letter:plural]->(and)->[package:plural]
                                          <-(object)<-[send]->(agent)->[they]
                                                       ->(through)->[mail_1]
```

Result 3.4.1 - *Common subgraph between mail_1_B_A and stamp_1_A_A*

```
[send]->(agent)->[person]
      ->(object)->[letter]
      ->(through)->[mail_1]
```

Result 3.4.2 - *Maximal join between mail_1_B_A and stamp_1_A_A*

```
Graph mail_1/stamp
[send]->(agent)->[person:plural]
```

```
    ->(agent)->[they]
    ->(object)->[message:plural]
    ->(through)->[mail_1 *x]
[send]->(object)->[letter:plural]->(and)->[package:plural]
                                <-(on)<-[put]<-(goal)<-[buy]->(agent)->[person:plural]
                                                          ->(object)->[stamp]
    ->(through)->[mail_1 *x]
```

Simplification within a graph We present two simplification methods that will eliminate redundant information generated by the joining of two graphs.

internal join: if a SSW is present twice in the resulting graph, possibly in two compatible forms, it can become one. This happens when both Graph N and Graph C contain a concept X, but the join is performed on another compatible concept from the MCS, and therefore the resulting graph still contains concept X twice. After the join of **mail_1** and **stamp** which resulted in the graph presented in Result 3.4.2, we have the concept **send** present twice, and also the concept **letter** is present in two compatible forms **letter** and **message**. These compatible forms actually allow us to disambiguate **letter** into **letter_2**. It is **letter_2** that is a subclass of **message**. The other sense **letter_1** is a subclass of **symbol**. Result 3.4.3 shows the result of performing an internal join on **letter** with **message** and on **send**. We can see that only concepts are joined, relations are not eliminated. They become redundant (twice **object** and twice **through**) and are eliminated at the next step.

Result 3.4.3 - *After internal join on graph mail_1/stamp*

```
[letter_2:plural]->(and)->[package:plural]
                <-(object)<-[send *z]
                <-(object)<-[*z]->(agent)->[they]
                                ->(agent)->[person:plural]
                                ->(through)->[mail_1:*x]
                                ->(through)->[*x]
```

<-(on)<-[put]<-(goal)<-[buy]->(agent)->[person:plural]
->(object)->[stamp]

reduction: if two **compatible** concepts are linked to the same concept with compatible relations, the concepts can be joined and the less specific relation eliminated. If a unique concept is joined with two compatible relations to another concept, the less specific relation is eliminated. From Result 3.4.3, the reduction allows us to eliminate the redundant relations **through** present twice between **mail_1** and **send** and **object** present twice between **send** and **letter_2**. Both **agent** relations attached to concept **send** lead to compatible concepts: **they** and **person**. This gives us an anaphora resolution for pronoun **they** in graph stamp_1_A_A which could have referred to **letters**, **packages**, **people** or **stamps**, but now is set to **people**. The concepts **they** and **person** are not SSWs and therefore could not be joined during the previous simplification step. At the reduction step, we have more evidence through the compatible relations that the two concepts refer to the same thing.

Result 3.4.4 - *After reduction of graph mail_1/stamp*

[letter_2:plural]->(and)->[package:plural]
<-(object)<-[send]->(agent)->[person:plural]
->(through)->[mail:*y]
<-(on)<-[put]<-(goal)<-[buy]->(agent)->[person:plural]
->(object)->[stamp]

3.4.4 Clustering procedure

In the preceding subsections, we have given ideas on how to determine semantically significant words, described finding common subgraphs between two graphs, attempted at a measure of graph similarity, looked at how to join graphs as well as reduce the resulting graph. We summarize three important definitions hereafter:

Semantically Significant Word (SSW): A word occuring less often in the dictionary than a certain threshold (set empirically);

Graph Similarity (GS): Similarity measure based on the number of shared SSWs between two CGs, as well as on the size of their maximal common subgraph

Graph Matching Threshold (GMT): Threshold put on the GS measure to decide whether a graph will be joined to the CCKG (graph representation of the cluster) or not.

Now that we have presented all the elements needed for the clustering procedure, we present the strategy for performing the clustering itself. We start from a trigger word and build a Concept Cluster Knowledge Graph (CCKG) around it. The CCKG will start as the CG representation of the trigger word and will grow from there by being joined to the CG representation of other words. We present our method for directing which word should be joined to the CCKG. We will explore some words, and among them some will be joined and some will not. To limit our exploration, we investigate only SSWs. When we explore a new word, it is not automatically joined to the CCKG. Its CG must have a graph similarity with the CCKG that is equal or more than the GMT. The GMT will vary depending on the phase (trigger or expansion) and has been set from experimentation. When we investigate a SSW that has multiple sense, we will add a penalty to the GMT to avoid joining word senses that are not appropriate. That penalty is set at GMT(a+1,b).

TRIGGER PHASE. We start with a keyword, a word central to the subject of interest, that becomes the trigger word. The CG representation of the trigger word's definition forms the initial CCKG. The GMT is set low for the trigger phase: GMT(1,0). The graph to be joined must have one SSW in common with the CCKG.

Trigger forward: Find the SSWs part of the CCKG, and attempt a join with their respective CG to the initial CCKG.

Trigger backward: Find all the words in the dictionary that use the trigger word in their definition and attempt a join with their respective CG to the CCKG. If

the trigger word is ambiguous, the backward search might find multiple words using it in their definition but in the wrong sense. We add a penalty to GMT in that case.

Instead of a single trigger word, we now have a cluster of words that are related through the CCKG. Those words form the **concept cluster**.

EXPANSION PHASE. We try finding words in the dictionary containing many concepts identical to the ones already present in the CCKG but perhaps interacting through different relations allowing us to create additional links within the set of concepts present in the CCKG. Our goal is to create a more interconnected graph rather than sprouting from a particular concept. For this reason, we establish a GMT higher than for the trigger phase. GMT(2,1) means that the graphs to be joined must have at least two SSWs in common as well as a relation. The relation in common must be part of a common subgraph. The two SSWs do not have to be the concepts that are part of the subgraph.

Expansion forward: Find the SSWs part of the CCKG, and attempt a join with their respective CG to the initial CCKG.

Expansion backward: Find all the words in the dictionary that use any word part of the concept cluster in their definition and attempt a join with their respective CG to the CCKG.

Repeat: Continue forward and backward expansion until no changes are made.

We do not present an example of the clustering process at this stage because in chapter 4, a large example is presented in section 4.5 which illustrates all the steps.

3.4.5 In Summary

We presented our clustering mechanisms, and showed all the steps and sub-processes necessary to expand the graph representation of a trigger word into a Concept Cluster Knowledge Graph.

If that process is done for the whole dictionary, we would obtain an LKB divided into multiple clusters of words, each represented by a CCKG. Then during text processing for example, a portion of text could be analyzed using the appropriate CCKG to find implicit relations and help understanding the text. All the mechanisms are in place to perform the clustering operations as part of our ARC-Concept software presented in chapter 4. We have not performed the clustering on all possible trigger words in the AHFD but only on some randomly chosen words to show some results.

Not only do we think the clusters will help further text analysis, but they also play an important role within the construction of the LKB as they allow for some semantic disambiguation as they gather more information on a same subject. Redundancy leads to disambiguation.

When we integrate different graphs together using the dictionary, we assume that the information is valid information to be put in a knowledge base. It is not one person's view on a particular story, but mostly generalities about daily life situations. We can therefore use that kind of trust in the knowledge to perform some inferences that we might not do if we were not certain of our source.

There are two good side effects to our knowledge integration process. By putting more information together, we can find redundancy that helps disambiguate single examples.

1. Anaphora resolution: if a pronoun and a concept C1 are involved in a compatible relation to another concept C2 then the pronoun can be coreferred to C1. We saw an example with **they** and **person** (plural) during the reduction process presented in section 3.4.3.

2. Word sense disambiguation: a match between concept A and concept B, where concept A subsumes a certain sense of concept B, will disambiguate B at the same time. We saw an example with **letter** and **message** as part of the simplification process presented in section 3.4.3.

3.5 Discussion

This chapter presented all four parts of our Lexical Knowledge Base (LKB).

1. Graph definitions: the set of all nouns and verbs defined in the AHFD with their graph representations;

2. Concept lattice: all the is-a links found in the graph definitions are extracted to form a separated structure, the concept hierarchy; we add covert categories to expand the search space for finding similar concepts;

3. Relation lattice: built by hand, it puts together all deeper semantic relations and the prepositions;

4. Concept clusters: from the graph definitions we can build larger structures around particular words; a trigger word becomes part of a cluster, which is a group of words representative of a particular context or micro-world; it gives a larger view to the meaning of that starting word.

Here is a more formal way to describe the elements part of the LKB. An LKB is a tuple (G, L_c, L_r, CL) where

- G is a set of Conceptual Graphs;
  a Conceptual Graph is defined as a tuple (C, R), where
  - C is a set of possible concepts
  - R is a set of possible relations
- L_c is a concept lattice in which all concepts from set C are positioned
- L_r is a relation lattice in which all relations from set R are positioned
- CL is a set of clusters;
  each cluster is defined as a tuple $(CL_{trigger}, CL_{words}, CL_{graph})$, where
  - $CL_{trigger}$ is a concept from the subset C_{ssw} which gives all the semantically significant words from set C, $C_{ssw} \subset C$;

- CL_{words} is the set of all significant words that were included within the cluster via the clustering process, $CL_{words} \subset C_{ssw}$;
- CL_{graph} is a Conceptual Graph centered around a single concept $CL_{trigger}$ and containing a set of concepts $C_{cluster}$, we have $C_{cluster} \subset C$, and $CL_{words} \subset C_{cluster}$.

Chapter 4

IMPLEMENTATION AND RESULT ANALYSIS

In this chapter, we take the reader through a demonstration of our system **ARC-Concept** (Acquisition, Representation and Clustering of Concepts). We refer to the two preceding chapters, as all the ideas presented in them are tested via our ARC-Concept system, and we are able to present and evaluate results at this stage.

ARC-Concept runs on a UNIX/Linux platform and is written in C++. It contains modules for parsing, transforming a parse tree into a representation using the Conceptual Graph (CG) formalism, performing structural and semantic disambiguation at the CG level, clustering definitions from the dictionary, and discovering covert categories as present in the dictionary.

Making use of a grammar file, a lexicon with parts of speech and definitions, a file of cooccurrences of words, and a set of files (verb exceptions, plural exceptions) used for the morphological analysis, it provides the user with facilities to acquire, represent and cluster concepts in a Lexical Knowledge Base. The user interacts with the system through a series of menus which can cause modification of the LKB and can display aspects of the LKB, such as the type hierarchy associated with any word.

The user interface (menu system) was not designed with a general user in mind, instead it is a custom menu system which is tailored to the needs of the research of this thesis. It would be possible to replace the menuing system with a more general

purpose arrangement of menus.

Menu 4.0.1 shows the first menu presented to a user of ARC-Concept.

Menu 4.0.1 -

Initialize.
Work on a specific word.
Work on all words of the dictionary for building the LKB.
Discover covert categories.
Clustering.
Show type hierarchy.
Others.
Termine.

The first section introduces briefly the CoGITo system [73] (**Initialize**) which we used as a starting platform to develop parts of ARC-Concept dealing with the storage and manipulation of Conceptual Graphs.

The second section presents all the steps for converting a sentence into a conceptual graph representation (**Work on a specific word**) using a few examples: the parsing, the parse-to-cg transformation, the different heuristics for structural and semantic disambiguation.

These steps from a sentence to a conceptual graph are performed on all nouns and verbs in the dictionary (**Work on all words of the dictionary**). The third section shows results such as the average number of parse trees per sentence and the reduction factor associated with each step of structural disambiguation.

The fourth section looks into covert categories (**Discover covert categories**). Quantitative results are given as to how many covert categories are found depending on the Covert Threshold used. Then we give qualitative interpretation and evaluation of the relevance of some covert categories found.

The fifth section gives one detailed example of the construction of concept clusters (**Clustering**). It then looks at the effect of varying the threshold for semantically significant words. Results for clusters built around different trigger words are then presented.

4.1 CoGITo System

The CoGITo system [74] was developed at Université Montpellier and was demonstrated at ICCS'95, the third International Conference on Conceptual Structures. Since it was the most promising system at that time, we used it as a basis for our research. In addition, it was free, giving us access to the source code in case we wanted to modify some routines. Although many new systems are now available[1], we are staying with CoGITo, now that we have a lot of software already developed and since it satisfies our needs.

In CoGITo, before a graph can be manipulated it must be part of an "environment" which is based on a "Support". In the Support, we must give all the concepts types, all the relations and all the referents (possible individuals of a certain type) that will be used in the graphs. This is a bit limiting, as everything must be known in advance. An unknown concept type cannot be used in a graph that we are trying to load in the environment. This means that we could not directly use our system on graphs generated by another system. We can use it though on a text containing unknown words, as for each word that cannot be tagged we ask the user to tag it (or eventually we could attempt automatic tagging [95]) and we put it in the Support. A Support can be modified after we have started working in an environment.

The Support does not only state all possible concept types, but also the subtype/supertype relations between them. Therefore the Support is the host for the type hierarchy and it can be easily modified as we refine the hierarchy via the is-a relations found in the graph representations of definitions. The Support is the host for the relation hierarchy as well. Both the concept and the relation hierarchies will be used when we try to find common information between two graphs.

Graphs can be read from a file (that has a particular format) and put in memory, or saved on a file from memory. Each graph in memory is part of an environment, and it has a unique name. Multiple graphs can be put in an environment so that operations can be performed on them.

CoGITo has an implementation of the standard operations performed on graphs,

[1] see web site http://maths.une.edu.au/ cgtools, where they are listing 25 known CG-based tools

such as projection (whether a graph A can be found as a subgraph of graph B) and join (making a larger graph A by joining it to graph B on a compatible concept type).

4.2 From a sentence to a Conceptual Graph

From the main menu, Work on a specific word gives a user a large submenu for all the possible operations on a single word (menu 4.2.1). We present three examples involving the words **doughnut**, **bat**, **piano** to go through most of the important operations.

Menu 4.2.1 -

Assign word to work with.
Find and show its definition.
Create CG file from definition.
Load its CG and update hierarchy (category and multiple sense).
Eliminate identical graphs.
Anaphora on word defined.
Anaphora on other words.
Word sense on word defined.
Prepositional attachment (statistics).
Update support (is-a relations).
Eliminate some graphs based on the superclasses of conjunctions.
Prepositional attachment (based on LKB).
Applying Semantic Relation Transformation Graphs.
Distribute conjunctions.
Try to assign word sense to each word.
Certainty.
Manual reduction.
Save graph to graph file.
Show its super and sub classes from hierarchy.
Show CG.

4.2.1 Example 1: DOUGHNUT

Assign word to work with must be chosen first. For this first example, the used typed in doughnut.

The selection Find and show its definition then prompts ARC-Concept to open the entire dictionary file (**dict.dat** as described in Appendix A) and retrieve the word given by the user from this dictionary.

Result 4.2.1 -

```
WORD: doughnut

sense_number: 1
cat: n

A doughnut is a small round cake.
Many doughnuts have a hole in the center.
Some have jelly in the center.
People like to eat doughnuts for breakfast.
```

Next, entering Create cg file from definition causes ARC-Concept to tag all words (see section 2.1.1) from the sentences contained in the definition, then apply parse rules (see section 2.1.2) to find possible parse trees for these sentences. The parser makes use of the rules provided in Appendix C. According to our grammar, the parser finds two parses for each of the four sentences. The parses are shown in Result 4.2.2. In each parse we can see the words themselves as leaf nodes and the different grammatical categories used as intermediate nodes. The root of each tree is **s2** which means a whole sentence. The numbers in parenthesis correspond to levels as seen in section 2.1.2 in the subsection on heuristics, as useful for the level difference heuristics. A leaf is considered as level 0, and each rule applied creates a new node that is one level higher than the highest level of its children nodes.

The trees are transformed into conceptual graphs (see section 2.1.3) and all saved in a file **doughnut.cg** in a specific format that can be read by the CoGITo system which will put the graphs in memory within its environment for further operations. Since there are eight parse trees, we will end up with eight CGs for doughnut. The CGs are also saved in files so they can be used for subsequent ARC-Concept sessions.

Result 4.2.2 -

```
nb_parses: 2
s2(6)->s1(5)->np(1)->det->a
                     n->doughnut
              vp(4)->vb->be
                     np(3)->det->a
                            n(2)->adj->small
                                   n(1)->adj->round
                                          n->cake
       ep->.
s2(6)->s1(5)->np(1)->det->a
                     n->doughnut
              vp(4)->vb->be
                     np(3)->det->a
                            n(2)->adj->small
                                   n(1)->n->round
                                          n->cake
       ep->.

nb_parses: 2
s2(6)->s1(5)->np(2)->n(1)->adj->many
                           n->doughnut
              vp(4)->vt->have
                     np(3)->np(1)->det->a
                                    n->hole
                            p2(2)->prep->in
                                    np(1)->det->the
                                           n->center
       ep->.
s2(5)->s1(4)->np(2)->n(1)->adj->many
                           n->doughnut
              vp(3)->vp(2)->vt->have
                            np(1)->det->a
                                    n->hole
                     p2(2)->prep->in
                            np(1)->det->the
                                    n->center
       ep->.

nb_parses: 2
s2(6)->s1(5)->np(1)->pron->some
              vp(4)->vt->have
                     np(3)->np(1)->n->jelly
                             p2(2)->prep->in
                                    np(1)->det->the
                                           n->center
       ep->.
s2(5)->s1(4)->np(1)->pron->some
              vp(3)->vp(2)->vt->have
```

```
                        np(1)->n->jelly
                  p2(2)->prep->in
                        np(1)->det->the
                              n->center
      ep->.

nb_parses: 2
s2(8)->s1(7)->np(1)->n->person
             vp(6)->vp(1)->vi->like
                    inf_vp(5)->prep->to
                               vp(4)->vt->eat
                                      np(3)->np(1)->n->doughnut
                                             p2(2)->prep->for
                                                    np(1)->n->breakfast
      ep->.
s2(7)->s1(6)->np(1)->n->person
             vp(5)->vp(1)->vi->like
                    inf_vp(4)->prep->to
                               vp(3)->vp(2)->vt->eat
                                             np(1)->n->doughnut
                                      p2(2)->prep->for
                                             np(1)->n->breakfast
      ep->.
```

Load its CG and update hierarchy (category + multiple sense) prompts ARC-Concept to open file **doughnut.cg** and load all graphs in the CoGITo environment. The graphs become accessible via a specific data structure built around the word **doughnut** which contains its part-of-speech, its definitions, its graph representations, etc.

A concept type is created from each word and put in the type hierarchy under "something" or "act" depending on the part-of-speech of its associated word. If the word contains multiple senses, a concept type is created for each sense and is put in the type hierarchy as a subclass of the concept type associated with the word.

Once the graphs are loaded in the CoGITo environment, Show CG can be called at any time to show the conceptual graphs associated with the word in usage. Result 4.2.3 shows the graphs for **doughnut**. Some attribute names and values are in French, since the original CoGITo environment was developed in France and made use of French. We have eight graphs, two possible graphs for each of the four sentences in the definition. The name of each graph indicates the word defined, the sense of the

word, the sentence number, and the possible graph for that sentence. For example, doughnut_1_C_B indicates we are working with the word **doughnut**, sense **1**, third sentence **C**, and second possible graph for that sentence **B**.

Result 4.2.3 -

```
Linear output for :
graph:doughnut_1_A_A;
nature:fait;
set:ens;
[be]-
  {
  (object)->[cake:a]-
    {
    (attribut)->[round];
    (attribut)->[small];
    };
  (agent)->[doughnut:a];
  }.

Linear output for :
graph:doughnut_1_A_B;
nature:fait;
set:ens;
[be]-
  {
  (object)->[cake:a]-
    {
    (attribut)->[round];
    (attribut)->[small];
    };
  (agent)->[doughnut:a];
  }.

Linear output for :
graph:doughnut_1_B_A;
nature:fait;
set:ens;
[have]-
  {
  (object)->[hole:a]->(in)->[center:the];
  (agent)->[doughnut:plural]->(attribut)->[many];
  }.

Linear output for :
graph:doughnut_1_B_B;
nature:fait;
```

```
set:ens;
[have]-
  {
  (object)->[hole:a];
  (in)->[center:the];
  (agent)->[doughnut:plural]->(attribut)->[many];
  }.

Linear output for :
graph:doughnut_1_C_A;
nature:fait;
set:ens;
[have]-
  {
  (object)->[jelly]->(in)->[center:the];
  (agent)->[some:ref];
  }.

Linear output for :
graph:doughnut_1_C_B;
nature:fait;
set:ens;
[have]-
  {
  (object)->[jelly];
  (in)->[center:the];
  (agent)->[some:ref];
  }.

Linear output for :
graph:doughnut_1_D_A;
nature:fait;
set:ens;
[like]-
  {
  (goal)->[eat]->(object)->[doughnut:plural]->(for)->[breakfast];
  (agent)->[person:plural];
  }.

Linear output for :
graph:doughnut_1_D_B;
nature:fait;
set:ens;
[like]-
  {
  (goal)->[eat]-
    {
    (object)->[doughnut:plural];
```

```
   (for)->[breakfast];
   };
  (agent)->[person:plural];
  }.
```

At this point we can choose any one of a number of operations to be performed on the eight graphs. We illustrate a few of the options which affect the graphs for **doughnut**. We start with three options that do not need to access the other graphs stored by ARC-Concept to do their task: **eliminate identical graphs, anaphora on word defined, prepositional attachment (statistics).**

Option **Eliminate identical graphs** finds graphs that are identical. Our grammar generates spurious ambiguities which lead to identical graphs. The two graphs **doughnut_1_A_A** and **doughnut_1_A_B** are identical. The word **round** can be considered as a noun or an adjective leading by two different ways to the noun phrase **round table** which correspond to the same graph where **round** is an attribute of **table**. ARC-Concept eliminates arbitrarily the second graph.

Option **Anaphora on word defined** looks for a pronoun related concept such as *it* or *some*. If the graph does not contain the concept for the word defined (in our case, if it does not contain the doughnut concept) then the pronoun related concept can be replaced by the concept for the word defined (see section 2.4.1). Notice in Result 4.2.4 that **[some:ref]** has been replaced by **[doughnut]->(attribut)->[some]**.

Result 4.2.4 -

BEFORE:

```
Linear output for :
graph:doughnut_1_C_A;
nature:fait;
set:ens;
[have]-
  {
  (object)->[jelly]->(in)->[center:the];
  (agent)->[some:ref];
  }.
```

AFTER:

```
Linear output for :
```

```
graph:doughnut_1_C_A;
nature:fait;
set:ens;
[have]-
  {
  (object)->[jelly]->(in)->[center:the];
  (agent)->[doughnut]->(attribut)->[some];
  }.
```

Prepositional attachment (statistics) calculates a score for each graph (see section 2.3.1). It looks at each relation in the graph one by one, if that relation is a preposition, it goes to files **verbs_cooccur.dat** and **nouns_cooccur.dat**[2]

to find out how many times over the whole dictionary did that particular preposition occur after the verb or noun to which it is attached.

Result 4.2.5 -

```
score 0 --- doughnut_1_A_A : (no prepositions)
score 8 --- doughnut_1_B_A : [hole]->(in)->[center]
score 1 --- doughnut_1_B_B : [have]->(in)->[center]
score 2 --- doughnut_1_C_A : [jelly]->(in)->[center:the]
score 1 --- doughnut_1_C_B : [have]->(in)->[center:the]
score 1 --- doughnut_1_D_A : [doughnut:plural]->(for)->[breakfast]
score 0 --- doughnut_1_D_B : [eat]->(for)->[breakfast]
```

When two or more graphs are competing for the representation of a single sentence, we take them two at a time and choose the one with the highest score if that score is more than twice the score for the other candidate. If one score is small (score1¡5), the other graph's score must be twice the first score + 5 ($score2 > (score1 * 2 + 5)$) to choose it[3]. Following these criteria, in Result 4.2.5 only the graph **doughnut_1_B_B** can be eliminated.

Update support (is-a relations) first transforms some graphs to include the **is-a** relation and then updates the type hierarchy with these relations (see section 2.2). Result 4.2.6 shows one modified graph. The type hierarchy will establish **doughnut** as a subclass of **cake** as shown in graph **doughnut_1_A_A**.

[2]these two files are created via a C++ program which counts for each noun and verb in the dictionary (in a base form, so after performing the tagging process) the number of times that any other words occurs after it.

[3]these arbitrary thresholds are chosen from experimentation, the penalty (+5) is needed in case of small scores to make sure the difference in scores is somewhat significant

Result 4.2.6 -

BEFORE:

```
Linear output for :
graph:doughnut_1_A_A;
nature:fait;
set:ens;
[be]-
  {
  (object)->[cake:a]-
    {
    (attribut)->[round];
    (attribut)->[small];
    };
  (agent)->[doughnut:a];
  }.
```

AFTER:

```
Linear output for :
graph:doughnut_1_A_A;
nature:fait;
set:ens;
[cake:a]-
  {                                  ------------------------------------
  (attribut)->[round];               * IS-A relation found               *
  (attribut)->[small];               * doughnut becomes subclass of cake *
  (is-a)<-[doughnut:a];              ------------------------------------
  }.
```

Let us now consider some options that need to access the other graphs stored by ARC-Concept as well as the type hierarchy.

Prepositional attachment (based on LKB) looks at differences between graphs that compete for the same sentence (see section 2.3.1), rather than looking at statistics. It looks for differences like those seen in CASE 1 and CASE 2.

CASE 1:

subgraph1: [ConceptA]->(preposition)->[ConceptB]

subgraph2: [ConceptA]->(preposition)->[ConceptC]

CASE 2:

subgraph3: [ConceptA]->(preposition)->[ConceptB]
subgraph4: [ConceptC]->(preposition)->[ConceptB]

The idea is to prefer a CG for a sentence if it has a subgraph that projects onto a related CG from another word.

In CASE 1, ARC-Concept searches for subgraph1 in the graph representation of the definitions of ConceptA and ConceptB and searches for subgraph2 in the graph representation of the definitions of ConceptA and ConceptC.

Similarly, in CASE 2, ARC-Concept searches for subgraph3 in the graph representation of the definitions of ConceptA and ConceptB and searches for subgraph4 in the graph representation of the definitions of ConceptC and ConceptB.

We are looking for subgraphs that are similar, differing in only one concept. With the last sentence of the definition of **doughnut** we have an example of the second case. Result 4.2.7 shows graphs **doughnut_1_D_A** and **doughnut_1_D_B**. They vary by the attachment of the word **breakfast** to either **doughnut** or **eat**. ARC-Concept searches for subgraph1 [doughnut]->(for)->[breakfast] in the graph representations of **breakfast** and **doughnut** and searches for subgraph2 [eat]->(for)->[breakfast] in the graph representations of **eat** and **breakfast**. In this particular case, looking at the word **doughnut** is not informative as it is the word we are working on. We look at the other two words **breakfast** and **eat**. The graph representations for **breakfast** shown in Result 4.2.8 do not contain either subgraph1 or subgraph2. The graph representations for **eat** shown in Result 4.2.9 contain the subgraph [eat]->(for)->[lunch]. Both **lunch** and **breakfast** are subclasses of **meal** and therefore compatible concepts. Graph **doughnut_1_D_B** is chosen because subgraph2 can be projected on **eat_1_C_A** and no projection for subgraph1 was found.

Result 4.2.7 -

```
Linear output for :
graph:doughnut_1_D_A;
nature:fait;
set:ens;
[like]-
  {
  (goal)->[eat]->(object)->[doughnut:plural]->(for)->[breakfast];
```

```
  (agent)->[person:plural];
  }.

Linear output for :
graph:doughnut_1_D_B;
nature:fait;
set:ens;
[like]-
  {
  (goal)->[eat]-
    {                              ------------------------------------------------
    (object)->[doughnut:plural]; * Graph chosen against previous doughnut_1_D_A *
    (for)->[breakfast];          * [eat]->(for)->[breakfast] can be projected   *
    };                           * onto graph eat_1_C_A                         *
  (agent)->[person:plural];      ------------------------------------------------
  }.


DIFFERENCES:
subgraph1: [doughnut]->(for)->[breakfast]
subgraph2: [eat]->(for)->[breakfast]
```

Result 4.2.8 -

```
      WORD: breakfast

      sense_number: 1
      cat: n

      Breakfast is a meal.
      It is the first meal of the day.
      At breakfast Bud usually eats cereal with milk.

Linear output for :
graph:breakfast_1_A_A;
nature:fait;
set:ens;
[meal:a]<-(is-a)<-[breakfast].

Linear output for :
graph:breakfast_1_B_A;
nature:fait;
set:ens;
[meal:the]-
  {
  (attribut)->[first];
  (is-a)<-[breakfast]->(of)->[day:the];
  }.
```

```
Linear output for :
graph:breakfast_1_C_B;
nature:fait;
set:ens;
[eat]-
  {
  (object)->[cereal]->(with)->[milk];
  (agent)->[person:Bud];
  (at:expected)->[breakfast];
  }.
```

Result 4.2.9 -

```
      WORD: eat

      sense_number: 1
      cat: v

      To eat means to take food into the body through the mouth.
      People eat when they feel hungry.
      I ate some soup for lunch.

Linear output for :
graph:eat_1_A_A;
nature:fait;
set:ens;
[eat]->(equiv)->[take]->(object)->[food]->(into)...
                  ->[body:the]->(through)->[mouth:the].

Linear output for :
graph:eat_1_A_C;
nature:fait;
set:ens;
[eat]->(equiv)->[take]-
  {
  (object)->[food];
  (into)->[body:the]->(through)->[mouth:the];
  }.

Linear output for :
graph:eat_1_B_A;
nature:fait;
set:ens;
[eat:*1]->(when)->[feel]-
  {
  (attribut)->[hungry];
  (agent)->[person:plural]<-(agent)<-[*1];
```

```
  }.


Linear output for :
graph:eat_1_C_A;
nature:fait;
set:ens;
[eat]-
  {
  (object)->[soup];                ----------------------------------------
  (for)->[lunch];                  * subgraph [eat]->(for)->[breakfast]     *
  (agent)->[I:ref];                * from doughnut_1_D_B is compatible here *
  }.                               ----------------------------------------
```

Result 4.2.10 shows that we have dealt with all the structural ambiguity, we have obtained a single graph for each sentence of the definition of **doughnut** after all the following options have been used.

Assign word to work with.
Find and show its definition.
Create cg file from definition.
Load its CG and update hierarchy (category + multiple sense).
Anaphora on word defined.
Prepositional attachment (statistics).
Update support (is-a relations).
Prepositional attachment (based on LKB).

We found identical graphs representing the first sentence. We used the statistics to resolve the structural ambiguity of the second sentence, preferring [hole]->(in)->[center] to [have]->(in)->[center]. We used the LKB-based method for the third and fourth sentences, preferring respectively [jelly]->(in)->[center] to [have]->(in)->[center] and [eat]->(for)->[breakfast] to [doughnut]->(for)->[breakfast].

Result 4.2.10 -

```
Linear output for :
graph:doughnut_1_A_A;
nature:fait;
set:ens;
[cake:a]-
  {
  (attribut)->[round];
  (attribut)->[small];
```

```
  (is-a)<-[doughnut:a];
  }.

Linear output for :
graph:doughnut_1_B_A;
nature:fait;
set:ens;
[have]-
  {
  (object)->[hole:a]->(in)->[center:the];
  (agent)->[doughnut:plural]->(attribut)->[many];
  }.

Linear output for :
graph:doughnut_1_C_A;
nature:fait;
set:ens;
[have]-
  {
  (object)->[jelly]->(in)->[center:the];
  (agent)->[doughnut]->(attribut)->[some];
  }.

Linear output for :
graph:doughnut_1_D_B;
nature:fait;
set:ens;
[like]-
  {
  (goal)->[eat]-
    {
    (object)->[doughnut:plural];
    (for)->[breakfast];
    };
  (agent)->[person:plural];
  }.
```

The Certainty option reorganizes the certainty information within the graph (see section 3.1.2). Result 4.2.11 shows the transformation of doughnut_1_B_A to include the certainty level **expected** due to the presence of **many**.

Result 4.2.11 -

```
Before:

Linear output for :
```

```
graph:doughnut_1_B_A;
nature:fait;
set:ens;
[hole:a]-
  {
  (in)->[center:the];
  (part-of)<-[doughnut:plural]->(attribut)->[many];
  }.

After:

Linear output for :
graph:doughnut_1_B_A;
nature:fait;
set:ens;
[hole:a]-
  {
  (in)->[center:the];
  (part-of:expected)<-[doughnut:plural];
  }.
```

4.2.2 Example 2: BAT

With this second example, we look into a few more options that we did not get at with the first example on **doughnut**. The word **bat** has two senses. This allows us to show the option for word sense disambiguation on the word defined. We also examine options for conjunction attachment, semantic relation transformation and word sense disambiguation of other words.

Find and show its definition gives the following result. We have two senses of **bat** each having three sentences in their definition.

Result 4.2.12 -

```
WORD: bat

sense_number: 1
cat: n

A bat is a thick stick.
It is used to hit a ball.
Bats are made of wood, metal, or plastic.
```

```
sense_number: 2
cat: n

A bat is a small animal.
A bat has a body like a mouse and wings.
Bats sleep during the day and fly around at night.
```

Create cg file from definition gives the following results. For the first sense, only the third sentence is ambiguous, receiving two parses. For the second sense, the first sentence is unambiguous, the second sentence receives three parses and the third sentence receives two. So, we have a total of ten parse trees.

Result 4.2.13 -

```
SENSE 1:

nb_parses: 1
s2(5)->s1(4)->np(1)->det->a
                     n->bat
              vp(3)->vb->be
                     np(2)->det->a
                            n(1)->adj->thick
                                   n->stick
       ep->.

nb_parses: 1
s2(6)->s1(5)->np(1)->pron->it
              vp(4)->vp(1)->vb->be
                            pp->use
                     inf_vp(3)->prep->to
                                  vp(2)->vt->hit
                                         np(1)->det->a
                                                n->ball
       ep->.

nb_parses: 2
s2(10)->s1(9)->np(1)->n->bat
              vp(8)->vb->be
                     pp(7)->pp->make
                            p2(6)->prep->of
                                   np(5)->np(1)->n->wood
                                          cnp(4)->conj(1)->pause->,
```

```
                                                    np(3)->np(1)->n->metal
                                                          cnp(2)->conj(1)->pause->,
                                                                          conj->or
                                                                 np(1)->n->plastic
       ep->.
s2(9)->s1(8)->np(1)->n->bat
              vp(7)->vp(1)->vb->be
                           pp->make
                     p2(6)->prep->of
                           np(5)->np(1)->n->wood
                                 cnp(4)->conj(1)->pause->,
                                        np(3)->np(1)->n->metal
                                              cnp(2)->conj(1)->pause->,
                                                              conj->or
                                                       np(1)->n->plastic
       ep->.


SENSE 2:

nb_parses: 1
s2(5)->s1(4)->np(1)->det->a
                     n->bat
              vp(3)->vb->be
                     np(2)->det->a
                           n(1)->adj->small
                                 n->animal
       ep->.

nb_parses: 3
s2(8)->s1(7)->np(1)->det->a
                     n->bat
              vp(6)->vt->have
                     np(5)->np(1)->det->a
                                 n->body
                           p2(4)->prep->like
                                 np(3)->np(1)->det->a
                                              n->mouse
                                       cnp(2)->conj->and
                                               np(1)->n->wing
       ep->.
s2(7)->s1(6)->np(1)->det->a
                     n->bat
              vp(5)->vp(2)->vt->have
                           np(1)->det->a
                                 n->body
                     p2(4)->prep->like
                           np(3)->np(1)->det->a
```

```
                                        n->mouse
                                    cnp(2)->conj->and
                                            np(1)->n->wing
        ep->.
s2(7)->s1(6)->np(1)->det->a
                    n->bat
              vp(5)->vt->have
                    np(4)->np(3)->np(1)->det->a
                                        n->body
                                    p2(2)->prep->like
                                        np(1)->det->a
                                                n->mouse
                              cnp(2)->conj->and
                                      np(1)->n->wing
        ep->.

nb_parses: 2
s2(7)->s1(6)->np(1)->n->bat
              vp(5)->vp(4)->vp(3)->vp(1)->vi->sleep
                                    p2(2)->prep->during
                                        np(1)->det->the
                                                n->day
                              cvp(3)->conj->and
                                      vp(2)->vp(1)->vi->fly
                                              mod_ap(1)->prep->around
                    p2(2)->prep->at
                              np(1)->n->night
        ep->.
s2(7)->s1(6)->np(1)->n->bat
              vp(5)->vp(3)->vp(1)->vi->sleep
                              p2(2)->prep->during
                                    np(1)->det->the
                                            n->day
                    cvp(4)->conj->and
                             vp(3)->vp(2)->vp(1)->vi->fly
                                              mod_ap(1)->prep->around
                                     p2(2)->prep->at
                                            np(1)->n->night
        ep->.
```

Load its CG and update hierarchy (category + multiple sense) followed by Show CG gives in Result 4.2.14 the ten CGs corresponding to the ten parse trees given in Result 4.2.13.

Result 4.2.14 -

```
Linear output for :
graph:bat_1_A_A;
nature:fait;
set:ens;
[be]-
  {
  (object)->[stick:a]->(attribut)->[thick];
  (agent)->[bat:a];
  }.

Linear output for :
graph:bat_1_B_A;
nature:fait;
set:ens;
[use]-
  {
  (goal)->[hit]->(object)->[ball:a];
  (object)->[it:ref];
  }.

Linear output for :
graph:bat_1_C_A;
nature:fait;
set:ens;
[make]-
  {
  (of)->[wood]->(pause)->[metal]->(or)->[plastic];
  (object)->[bat:plural];
  }.

Linear output for :
graph:bat_1_C_B;
nature:fait;
set:ens;
[make]-
  {
  (of)->[wood]->(pause)->[metal]->(or)->[plastic];
  (object)->[bat:plural];
  }.

Linear output for :
graph:bat_2_A_A;
nature:fait;
set:ens;
[be]-
  {
```

```
  (object)->[animal:a]->(attribut)->[small];
  (agent)->[bat:a];
  }.

Linear output for :
graph:bat_2_B_A;
nature:fait;
set:ens;
[have]-
  {
  (object)->[body:a]->(like)->[mouse:a]->(and)->[wing:plural];
  (agent)->[bat:a];
  }.

Linear output for :
graph:bat_2_B_B;
nature:fait;
set:ens;
[have]-
  {
  (object)->[body:a];
  (like)->[mouse:a]->(and)->[wing:plural];
  (agent)->[bat:a];
  }.

Linear output for :
graph:bat_2_B_C;
nature:fait;
set:ens;
[have]-
  {
  (object)->[body:a]-
    {
    (like)->[mouse:a];
    (and)->[wing:plural];
    };
  (agent)->[bat:a];
  }.

Linear output for :
graph:bat_2_C_A;
nature:fait;
set:ens;
[sleep]-
  {
  (during)->[day:the];
  (and)->[fly]->(modif)->[around];
  (at)->[night];
```

```
  (agent)->[bat:plural];
  }.

Linear output for :
graph:bat_2_C_B;
nature:fait;
set:ens;
[sleep]-
  {
  (during)->[day:the];
  (and)->[fly]-
    {
    (modif)->[around];
    (at)->[night];
    };
  (agent)->[bat:plural];
  }.
```

Graph **bat_1_C_A** and **bat_1_C_B** are identical. Since our grammar generates spurious ambiguities which lead to identical graphs, Eliminate identical graphs can be used to arbitrarily remove one (it removes the second one).

Anaphora on word defined (see section 2.4.1) finds the pronoun related concept *it* in graph **bat_1_B_A** and replaces it with the **bat** concept.

Result 4.2.15 -

BEFORE:

```
Linear output for :
graph:bat_1_B_A;
nature:fait;
set:ens;
[use]-
  {
  (goal)->[hit]->(object)->[ball:a];
  (object)->[it:ref];
  }.
```

AFTER:

```
Linear output for :
graph:bat_1_B_A;
nature:fait;
set:ens;
[use]-
```

```
{
(goal)->[hit]->(object)->[ball:a];
(object)->[bat];
}.
```

Word sense on word defined (see section 2.4.2) puts within each graph the sense of **bat** being defined in that sentence. So, each occurrence of **bat** in Result 4.2.14 is replace by either **bat_1** or **bat_2**. For example, the CG from Result 4.2.15 is converted as shown in Result 4.2.16.

Result 4.2.16 -

```
Linear output for :
graph:bat_1_B_A;
nature:fait;
set:ens;
[use]-
  {
  (goal)->[hit]->(object)->[ball:a];
  (object)->[bat_1];
  }.
```

Prepositional attachment (statistics) (see section 2.3.1) cannot make any decision between **bat_2_C_A** over **bat_2_C_B** as the scores are too close.

Result 4.2.17 -

```
score 3 --- bat_2_C_A : [sleep]->(during) (score 1)
                        [sleep]->(at)     (score 2)
score 1 --- bat_2_C_B : [sleep]->(during) (score 1)
                        [fly]->(at)       (score 0)
```

Prepositional attachment (LKB-based) will choose (wrongly) the graph **bat_2_C_A** as it finds in the definition of **sleep** the subgraph **[sleep]->(at)->[night]**. The subgraph **[fly]->(at)->[night]** is not found in the definitions of **night** or **fly**.

Eliminate some graphs based on the superclasses of conjunctions (see section 2.3.2) can be used next for further structural disambiguation using the type hierarchy.

It plays a role eliminating two candidates for the sentence **A bat has a body like a mouse and wings**. The three possible graphs put in conjunction different elements.

Both graphs **bat_2_B_A** and **bat_2_B_B** have [mouse]->(and)->[wings], and graph **bat_2_B_C** has [body]->(and)->[wings] in conjunction. The superclass for **mouse** and **wing** is **something** and the superclass for **body** and **wing** is **part**. As **part** is more specific than **something** in the type hierarchy, ARC-Concept eliminates two graphs and to keep only **bat_2_B_C** to represent the sentence.

Result 4.2.18 -

```
Linear output for :
graph:bat_2_B_A;
        nature:fait;                  ---------------------------------
        set:ens;                      * Superclass of mouse and wing  *
[have]-                               *         SOMETHING             *
  {                                   ---------------------------------
  (object)->[body:a]->(like)->[mouse:a]->(and)->[wing:plural];
  (agent)->[bat_2:a];
  }.

Linear output for :
        graph:bat_2_B_B;
        nature:fait;                  ---------------------------------
        set:ens;                      * Superclass of mouse and wing   *
[have]-                               *         SOMETHING              *
  {                                   ---------------------------------
  (object)->[body:a];
  (like)->[mouse:a]->(and)->[wing:plural];
  (agent)->[bat_2:a];
  }.

Linear output for :
graph:bat_2_B_C;
nature:fait;
set:ens;
[have]-
  {
  (object)->[body:a]-                 ---------------------------------
    {                                 * Superclass of body and wing   *
    (like)->[mouse:a];                *          PART                 *
    (and)->[wing:plural];             ---------------------------------
    };
  (agent)->[bat_2:a];
  }.
```

Result 4.2.19 shows that we have dealt with all the structural ambiguity, we have

obtained a single graph for each sentence of the definition of **bat** after all the following options are used.

Assign word to work with.
Find and show its definition.
Create cg file from definition.
Load its CG and update hierarchy (category + multiple sense).
Anaphora on word defined.
Word sense on word defined.
Prepositional attachment (statistics).
Update support (is-a relations).
Eliminate some graphs based on the superclasses of conjunctions

For sense 1, the first and second sentences were already unambiguous, the third sentence generated two identical graphs and one was arbitrarily removed. For sense 2, the first sentence was unambiguous, the second sentence was disambiguated via the conjunction attachment option preferring to put **body** and **wing** in conjunction rather than **mouse** and **wing**, the third sentence was (wrongly) disambiguated via the prepositional attachment (LKB-based) option preferring to have [sleep]->(at)->[night] rather than [fly]->(at)->[night].

Result 4.2.19 -

```
Linear output for :
graph:bat_1_A_A;
nature:fait;
set:ens;
[stick:a]-
  {
  (attribut)->[thick];
  (is-a)<-[bat_1:a];
  }.

Linear output for :
graph:bat_1_B_A;
nature:fait;
set:ens;
[use]-
  {
  (goal)->[hit]->(object)->[ball:a];
  (object)->[it:ref];
  }.

Linear output for :
```

```
graph:bat_1_C_A;
nature:fait;
set:ens;
[make]-
  {
  (of)->[wood]->(pause)->[metal]->(or)->[plastic];
  (object)->[bat:plural];
  }.

Linear output for :
graph:bat_2_A_A;
nature:fait;
set:ens;
[be]-
  {
  (object)->[animal:a]->(attribut)->[small];
  (agent)->[bat:a];
  }.

Linear output for :
graph:bat_2_B_C;
nature:fait;
set:ens;
[have]-
  {
  (object)->[body:a]-
    {
    (like)->[mouse:a];
    (and)->[wing:plural];
    };
  (agent)->[bat:a];
  }.

Linear output for :
graph:bat_2_C_A;
nature:fait;
set:ens;
[sleep]-
  {
  (during)->[day:the];
  (and)->[fly]->(modif)->[around];
  (at)->[night];
  (agent)->[bat:plural];
  }.
```

Applying Semantic Relation Transformation Graphs tries to find deeper semantic relations in the graphs as discussed in section 2.4.3. ARC-Concept looks for

particular subgraphs that lead to deeper semantic relations. It finds the relation **instrument** and **material** respectively in graphs bat_1_B_A and bat_1_C_A shown in result 4.2.20.

Result 4.2.20 -

BEFORE:

```
Linear output for :
graph:bat_1_B_A;
nature:fait;
set:ens;
[use]-
  {
  (goal)->[hit]->(object)->[ball:a];
  (object)->[bat_1];
  }.

Linear output for :
graph:bat_1_C_A;
nature:fait;
set:ens;
[make]-
  {
  (of)->[wood]->(pause)->[metal]->(or)->[plastic];
  (object)->[bat_1:plural];
  }.
```

AFTER:

```
Linear output for :
graph:bat_1_B_A;
nature:fait;
set:ens;
[hit]-
  {
  (object)->[ball:a];
  (instrument)->[bat_1];
  }.

Linear output for :
graph:bat_1_C_A;
nature:fait;
set:ens;
[wood]-
```

```
{
(pause)->[metal]->(or)->[plastic];
(material)<-[bat_1:plural];
}.
```

Try to assign word sense to each word finds two word sense disambiguations (see section 2.4.2), one for word **stick** and the other for word **fly** based on the part of speech used (shown by different types of relations that a word is linked to in the graph). Nothing is found for **night** as it is not ambiguous. And the word **day** which has two senses in the AHFD is left ambiguous. Result 4.2.21 shows the two modified graphs.

Result 4.2.21 -

```
Linear output for :
graph:bat_1_A_A;
nature:fait;
set:ens;
[stick_1:a]-
  {
  (attribut)->[thick];
  (is-a)<-[bat_1:a];
  }.

Linear output for :
graph:bat_2_C_A;
nature:fait;
set:ens;
[sleep]-
  {
  (during)->[day:the];
  (at)->[night];
  (agent)->[bat_2:plural]<-(agent)<-[fly_2]->(modif)->[around];
  }.
```

4.2.3 Example 3: PIANO

With the word **piano** we show an example of word sense disambiguation looking into the knowledge base. First the option Find and show its definition gives the following result.

Result 4.2.22 -

```
WORD: piano

sense_number: 1
cat: n

A piano is an instrument.
It has 88 white and black keys.
You push down on the keys with your fingers to make music.
```

Result 4.2.23 shows the resulting graphs after applying all the following options.

Assign word to work with.
Find and show its definition.
Create cg file from definition.
Load its CG and update hierarchy (category + multiple sense).
Eliminate identical graphs.
Anaphora on word defined.
Applying Semantic Relation Transformation Graphs.

Result 4.2.23 -

```
Linear output for :
graph:piano_1_A_A;
nature:fait;
set:ens;
[instrument:an]<-(is-a)<-[piano:a].

Linear output for :
graph:piano_1_B_A;
nature:fait;
set:ens;
[key:num]-
  {
  (attribut)->[white];
  (attribut)->[black];
  (part-of)<-[piano];
  }.

Linear output for :
graph:piano_1_C_A;
nature:fait;
set:ens;
[push]-
```

```
  {
  (modif)->[down];
  (on)->[key:plural]->(with)->[finger:your];
  (goal)->[make]->(object)->[music];
  (agent)->[you:ref];
  }.

Linear output for :
graph:piano_1_C_C;
nature:fait;
set:ens;
[push]-
  {
  (modif)->[down];
  (on)->[key:plural];
  (with)->[finger:your];
  (goal)->[make]->(object)->[music];
  (agent)->[you:ref];
  }.
```

Option Try to assign word sense to each word (see section 2.4.2 general case) identifies the sense for **key** by looking into its graph representation. Graph **piano_1_B_A** will be modified to include the second sense of **key**. It has a common subgraph with **key_2_A_A**. The definition and graph representation for **key** are shown in result 4.2.24. Other graphs **piano_1_C_A** and **piano_1_C_C** also contain the concept **key** but it cannot be disambiguated within these graphs as there is no common subgraph with any sense of **key**.

Result 4.2.24 -

```
WORD: key

sense_number: 1
cat: n

A key is a piece of metal.
It opens a lock.
People use keys to open the doors of their homes and cars.


sense_number: 2
cat: n
```

```
      A key is also a part of a piano.
      The keys are where you put your fingers to play.
      There are white keys and black keys on a piano.


Linear output for :
graph:key_1_A_A;
nature:fait;
set:ens;
[metal]->(piece-of)->[key_1:a].

Linear output for :
graph:key_1_B_A;
nature:fait;
set:ens;
[open]-
  {
  (object)->[lock:a];
  (agent)->[key_1];
  }.

Linear output for :
graph:key_1_C_A;
nature:fait;
set:ens;
[use]-
  {
  (object)->[key_1:plural];
  (goal)->[open]->(object)->[door:plural]-
    {
    (of)->[home:their];
    (of)->[car:plural];
    };
  (agent)->[person:plural];
  }.


Linear output for :              ----------------------------------------
       graph:key_2_A_A;          * Subgraph [key_2]<-(part-of)<-[piano] *
       nature:fait;              * in common with piano_1_B_A           *
       set:ens;                  ----------------------------------------
[also]<-(modif)<-[key_2:a]<-(part-of)<-[piano:a].

Linear output for :
graph:key_2_C_A;
nature:fait;
set:ens;
[be:plural]-
```

```
{
(object)->[key:plural]->(attribut)->[white];
(agent)->[there:ref];
(object)->[key:plural]-
  {
  (attribut)->[black];
  (on)->[piano:a];
  };
}.
```

4.3 Results over the whole dictionary

Until now, we have been looking at augmenting our Lexical Knowledge Base (LKB) one word at a time. This is important for exploring different definitions and observing the behavior of the many operations in isolation. Now, we want to consider the wide spread application of many operations to the words from the dictionary file to build the LKB.

Work on all words of the dictionary for building the LKB from the main menu gives the following submenu.

Menu 4.3.1 -

Generate the CG files for each word from a dictionary file.
Statistics on number of graphs.
Iteration 0: Update Support with part-of-speech and word senses.
Iteration 1: stat. prepositional attachment + is-a relations.
Iteration 2. all structural disambiguation.
Iteration 3: Same as Iteration 2.
Certainty.
For all graphs, disambiguate words used in definitions having multiple senses.
Look at cg files and manually choose the graphs to keep.

Let us now consider each of these options in detail.

Generate the cg files for each word from a dictionary file looks at all nouns and verbs in the dictionary and for each one creates a file "X.cg" that corresponds to the conceptual graph representations of all sentences defining the word "X" that ARC-Concept

was able to parse and transform into a CG representation. When all these files are loaded into memory in the CoGITo environment, they form the first large part of the LKB, the graph definitions of all words.

Option Iteration 0: Update Support with part-of-speech and word senses loads all graphs in CoGITo and updates the type hierarchy with respect to the part of speech of the words defined as well as the different word senses possible for that word. We want to create a type hierarchy containing all possible concepts. The initial support contains one concept for each word, but we want to have one concept for each sense. We also eliminate spurious ambiguities generated by the chart parser by eliminating identical graphs.

This iteration 0 corresponds to the following operations presented in the previous section for a single word. Now they are performed on all nouns and verbs. (see section 2.5 for a description of all iterations)

Load its CG and update hierarchy (category + multiple sense).
Eliminate identical graphs.

Statistics on number of graphs calculates for all sentences their number of corresponding graphs. We aim toward one graph per sentence, which would mean that we dealt with all the structural ambiguity in the sentences. The statistics will allow us to see the effect of each step of structural disambiguation toward that goal. After iteration 0, the number of corresponding graphs is in fact equivalent to the number of parses generated by the chart parser minus the spurious ambiguities taken away via the Eliminate identical graphs option. There was 20% of the parses eliminated as spurious ambiguities.

Table 4.1 shows the results in terms of parses, but you could as well read it in number of graphs. There is a total of 3742 sentences, a total of 6900 parses and therefore an average of 1.84 parse/sentence.

Option Iteration 1: stat. prepositional attachment + is-a relations follows iteration 0, and looks into anaphora resolution of the word defined and other words, as well as word sense disambiguation for the word defined. We attempt structural disambiguation via the statistical prepositional attachment heuristic. We also look for is-a

Table 4.1: Statistics on number of parses

number of parses	number of sentences	% of sentences
0	299	8.0
1	1748	46.7
2	1110	29.7
≥ 3	585	15.6
≥ 5	256	6.8
≥ 10	38	1.0
≥ 20	6	≈ 0

relations to create a type hierarchy that will be useful for further structural and semantic disambiguation processes making use of ARC-Concept's ensemble of graph definitions and the type hierarchy.

Iteration 1 includes the following operations presented in the previous section for a single word. Now they are performed on all nouns and verbs.

Anaphora on word defined.
Anaphora on other words.
Word sense on word defined.
Prepositional attachment (statistics).
Update support (is-a relations).

The only operation which actually reduces the structural ambiguity is the prepositional attachment heuristic. There is an 18% reduction in the number of graphs, as we go from 6900 total number of graphs(parses) to 5655 graphs. Now that we made a first attempt at updating the type hierarchy, we start over from the parsing step. The number of parses can be different at each iteration as some parsing heuristics are based on the type hierarchy.

Iteration 2. all structural disambiguation, we go through the following steps which are previously discussed.

Table 4.2: Number of graphs after steps of iteration 2

Step	Number of graphs	Average nb graphs per sentence
Load graphs	8078	2.16
Eliminate identical graphs	6716	1.84
Statistical prepositional attachment	5558	1.49
Superclass of conjunction	5469	1.46
Prepositional attachment based on LKB	4905	1.31

Create cg file from definition.
Load its CG and update hierarchy (category + multiple sense).
Eliminate identical graphs.
Anaphora on word defined.
Anaphora on other words.
Word sense on word defined.
Prepositional attachment (statistics).
Update support (is-a relations).
Eliminate some graphs based on the superclasses of conjunctions
Prepositional attachment (based on LKB).

Table 4.2 shows the resulting number of graphs after each step of the structural disambiguation. We have started at 1.84 graph/sentence after eliminating identical graphs and we have reduced to 1.31 graph/sentence after all steps of structural disambiguation. This represents a reduction of almost 30%. Currently, we do not perform any more structural disambiguation; we can use manual reduction at this point if we want to decide on a single graph per sentence.

Applying Semantic Relation Transformation Graphs takes care of semantic disambiguation by trying to find deeper semantic relations within the graphs (as described in section 2.4.3), and Distribute conjunctions distributes the relations in which some concepts involved in a conjunction are also involved in (as described in section 2.4.4). They are the two last steps of iteration 2.

The results of these steps, along with other options which could be performed on all nouns and verbs such as Try to assign word sense to each word and Certainty are more difficult to present.

We can see quantitative results for the structural disambiguation process in terms

of number of graphs corresponding to one sentence. For semantic disambiguation, the evaluation would be trickier. To decide whether the correct sense of a word has been assigned to an occurrence of that word in a particular sentence, we must ask a human reader to be the judge of the task. The same is true as for judging if the correct semantic relation has become part of a graph. As the task of human judging would be very time consuming if applied on the whole dictionary, we would have to take a sample of sentences and evaluate those. We decided against that option mostly for time reason, as even a sample size of limited significance (5looking at each word for sense disambiguation and each relation for SRTGs.

4.4 Covert categories

So far we have seen how the definitions from a dictionary can be individually transformed into Conceptual Graphs and how these CGs can be used to build the type hierarchy. Now, we want to examine another important aspect of the LKB: the discovery of covert categories. **Covert categories** are discovered among the definitions and are used to complement the type hierarchy in its role for establishing concept similarity. Covert categories are seen as concepts without words, such as "writing instrument" or "device giving time". They show grouping of words based on different criteria than a common hypernym.

From the main menu, Discover covert categories brings up the following menu:

Menu 4.4.1 -

Find covert categories level 1.
Find covert categories level 2.
Build lambda abstractions and add covert categories.
Calculate average number of occurrences for a covert category.
Retour au menu principal.

We try finding covert categories (see section 3.2.7) within the set of resulting graphs after iteration 1, and iteration 2 shown in the previous section. To find a covert category, we create a subgraph

[A]->(REL)->[everything]

where A varies over the range of all verbs in the dictionary and **REL** varies over the range of all possible relations. We look at verbs because the usual nature of a covert category is as a case role to a verb.

We have a total of 267 verbs and a total of 51 deeper semantic relations + 65 prepositions, conjunctions, adverbs that can be used as relations. This gives over 30 thousand possible combinations. Although, this is without considering that many of those combinations are not possible, as for example a verb cannot be related to a "part-of" relation. There are 16 relations that we can eliminate as they cannot apply to verbs. This still leaves us with over 25 thousand possible combinations.

We establish a Covert Threshold (CT) for the number of times the particular subgraph must be present among all graph definitions in ARC-Concept to be considered as a new covert category and be assigned a λ-abstraction. We present results with a CT of 2, 5, 10 and 20.

The results for finding the covert categories demand equality of relations instead of compatibility. Looking at compatible relations will lead to multiple covert categories and we will not be able to identify what is the predominant relation within that category.

Option Find covert categories level 1 discovers multiple covert categories and save them into a file. Option Calculate average number of occurrences for a covert category then gives us some quantitative results based on the file generated at the previous step.

The results presented are not exact. They contain some errors due to the fact that we do not have a single graph representation for each sentence, but an average after iteration 1 of 1.8 graphs per sentence. This means that a pattern found 3 times might not come from three occurrences in the dictionary but from three possible graph representations for the same occurrence. One obvious way to solve this problem is to manually chose for all 1200 words, that is 3742 sentences, which graph should represent that sentence. That is a long task that we will not do here, and we will compensate by using the 1.8 graph/sentence as a reduction factor. For example, to

Table 4.3: Covert categories found after iteration 1, level 1

Covert Threshold (with factor)	Number of covert categories		
	Concept Subsumption(CS): YES Factor: 1.0	CS: YES Factor: 1.84	CS: NO Factor: 1.84
2 (3.7)	866	592	393
5 (9.2)	508	256	227
10 (18.4)	256	123	117
20 (36.8)	121	55	51

Table 4.4: Covert categories found after iteration 2, level 1

Covert Threshold (with factor)	Number of covert categories		
	Concept Subsumption(CS): YES Factor: 1.0	CS: YES Factor: 1.31	CS: NO Factor: 1.31
2 (2.6)	804	638	504
5 (6.5)	411	290	283
10 (13.1)	195	136	136
20 (26.2)	76	53	53

be counted as 5 occurrences, a pattern should occur more than 5*1.8, that is 9 times among the graph representations.

Table 4.3 and Table 4.4 show results respectively after iteration 1 and iteration 2 of the number of covert categories found. The first column shows the results as they are, then column two takes the reduction factor into account. In the third column, we show results that add one constraint on the comparison process. Concept subsumption using the type hierarchy is usually used for graph projection. The constraint added to the projection of the simple graph onto all graphs in the dictionary is to not allow for subsumption. Therefore the projection is possible only when all the concepts in both graphs are identical.

The covert categories extracted are at Level 1. They contain only one concept type and one relation and have one variable within the λ-abstraction.

Result 4.4.1 shows a small part of file **covertB.dat** corresponding to covert categories found at Level 1 after iteration 2. The name of each covert category is generated as a unique label from its graph representation. For example, blow~agent, means that we projected the graph [blow]->(agent)->[everything] on all graphs in the dictionary. We found three occurrences of **wind** as the agent of **blow**, five occurrences of **person**, one of **horn_3** and one of **instrument**.

Build lambda abstractions and add covert categories. takes each covert category one by one, creates a type concept and defines it via a λ-abstraction. The new type concept is put in the type hierarchy, and the λ-abstraction associated with that type concept is put in the CoGITo environment.

Result 4.4.2 shows λ-abstractions corresponding to the covert categories from Result 4.4.1. Each λ-abstraction is represented as a graph and is equal to the graph that was projected on the whole dictionary to establish the covert category in the first place.

Result 4.4.1 -

In file COVERTB.DAT

```
blow~agent:
wind 3
person 5
horn_3 1
instrument 1

clean~with:
brush 1
soap 1

sew~object:
button 1
cloth 1
patch_1 8
pocket 1
dress 2
clothes 3
```

Result 4.4.2 -

```
LAMBDA
```

```
Linear output for :
graph:blow~agent;
nature:;
set:;
[blow]->(agent)->[everything].


LAMBDA
Linear output for :
graph:clean~with;
nature:;
set:;
[clean]->(with)->[everything].


LAMBDA
Linear output for :
graph:sew~object;
nature:;
set:;
[sew]->(object)->[everything].
```

We then continue at Level 2 where we have two concept types, two relations and one variable. The level 2 covert categories are based on those of Level 1. If a covert category found at Level 1 contains many occurrences (surpasses a certain threshold) of a particular concept type, we fix it and try to vary on a second relation.

For example, assume we have the following Level 1 covert category:

Type WordX~relationA(λ) is
    [WordX]->(relationA)->[λ]

If the concept type WordY is found 5 times among the graphs of ARC-Concept to replace the λ, then we go on to Level 2 with a subgraph :

[WordX]->(relationA)->[WordY]
    ->(REL)->[everything]

We project that subgraph on all graphs in the LKB and again use the Covert Threshold to establish more covert categories.

Option Find covert categories level 2 discovers multiple covert categories and saves them into a file. Option Calculate average number of occurrences for a covert category

Table 4.5: Covert categories found after iteration 1, level 2

Covert Threshold (with factor)	Number of covert categories		
	Concept Subsumption(CS): YES Factor: 1.0	CS: YES Factor: 1.84	CS: NO Factor: 1.84
2 (3.7)	1867	997	256
5 (9.2)	741	290	125
10 (18.4)	290	129	56
20 (36.8)	118	44	22

Table 4.6: Covert categories found after iteration 2, level 2

Covert Threshold (with factor)	Number of covert categories		
	Concept Subsumption(CS): YES Factor: 1.0	CS: YES Factor: 1.31	CS: NO Factor: 1.31
2 (2.6)	1171	811	393
5 (6.5)	415	255	180
10 (13.1)	154	88	66
20 (26.2)	44	31	24

then gives us some quantitative results based on the file generated at the previous step. We present again the results after iteration 1 and iteration 2 in Table 4.5 and Table 4.6.

Result 4.4.3 shows a small part of file **covert2B.dat** corresponding to covert categories found at Level 2 after iteration 2. Then Result 4.4.4 shows the corresponding λ-abstractions.

Result 4.4.3 -

In file COVERT2B.DAT

```
play^agent^person^object:
music 2
baseball 1
game 5
outside 2
secret 2
```

```
instrument 1
sport 2
movie 1
time_3 1
song 1
well_2 1
violin 1

rise~agent~sun~in:
east 1
morning 1

send~agent~person~object:
card_2 1
message 1
person 1
package 2
letter 1
```

Result 4.4.4 -

```
LAMBDA
Linear output for :
graph:play~agent~person~object;
nature:;
set:;
[play]-
  {
  (agent)->[person];
  (object)->[everything];
  }.


LAMBDA
Linear output for :
graph:rise~agent~sun~in;
nature:;
set:;
[rise]-
  {
  (agent)->[sun];
  (in)->[everything];
  }.


LAMBDA
Linear output for :
```

```
graph:send~agent~person~object;
nature:;
set:;
[send]-
  {
  (agent)->[person];
  (object)->[everything];
  }.
```

We present now two examples of how the type hierarchy gets modified by the addition of covert categories. Figure 4.1 presents the addition of the Level 1 covert category **live~in** and of multiple Level 2 categories that become subclasses of **live~in** as they are different specializations depending on the **agent** relation. In parenthesis after a leaf node we display its superclass as extracted from the **is-a** relations. The superclasses **area** and **place** were already grouping many of the concepts together, but other concepts such as **water**, **castle**, **ground** were not grouped together, and the covert category **live~in** brings them all together.

Figure 4.2 shows the addition of the covert category **wear~agent~person~object**. It is interesting to note a large part of its elements are also subclasses of **clothes**. But the two classes are not equivalent, there are also **boots** and **hats** and other things that a person can wear and they are not considered as clothes. The left part of Figure 4.2 shows the hierarchy as extracted from the **is-a** relations. We note that the similarity between **ring** and **skate**, or **ski** and **clothes**, or **coat** and **suit** could not be established. The smallest superclass common to each pair is the very general concept **something** under which all nouns are. Adding the covert category **wear~agent~person~object** establishes a similarity between all the words and things that a person can wear.

It is interesting to compare our results with *An exploration into graded set membership* [93]. Using 76 adults, Markowitz performed some experiments, asking the subject how much a concept is prototypical of a class. The subject has to give a rank from 1 to 10. For example, where do you rank **shoe**, **ice skate** and **slippers** as being representative of **footwear**. Or how do you rank **robin**, **eagle**, **chicken**, **ostrich**, **bat** as representative of **bird**. She found that:

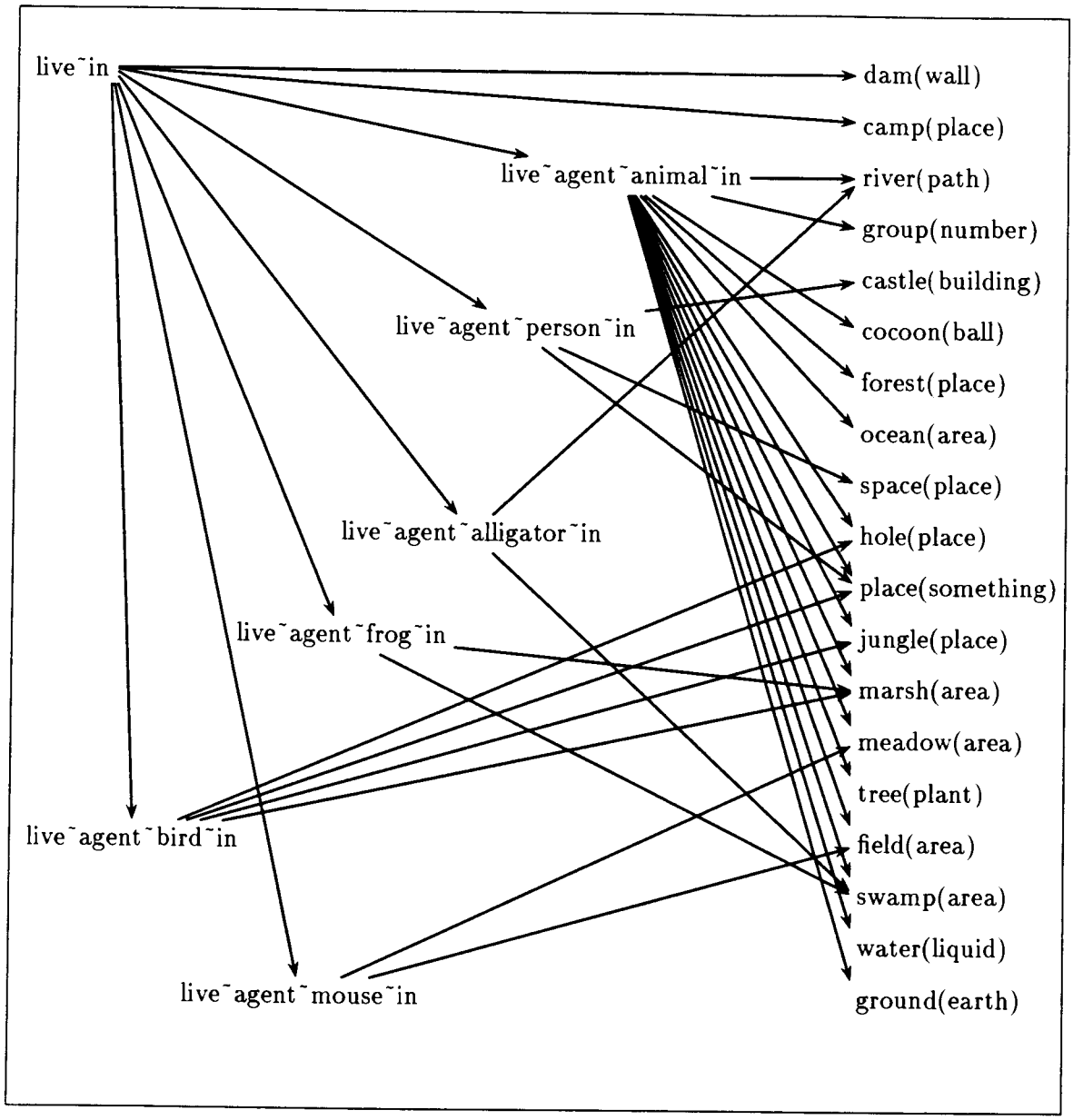


Figure 4.1: Example of covert categories involving **live~in** in the hierarchy.

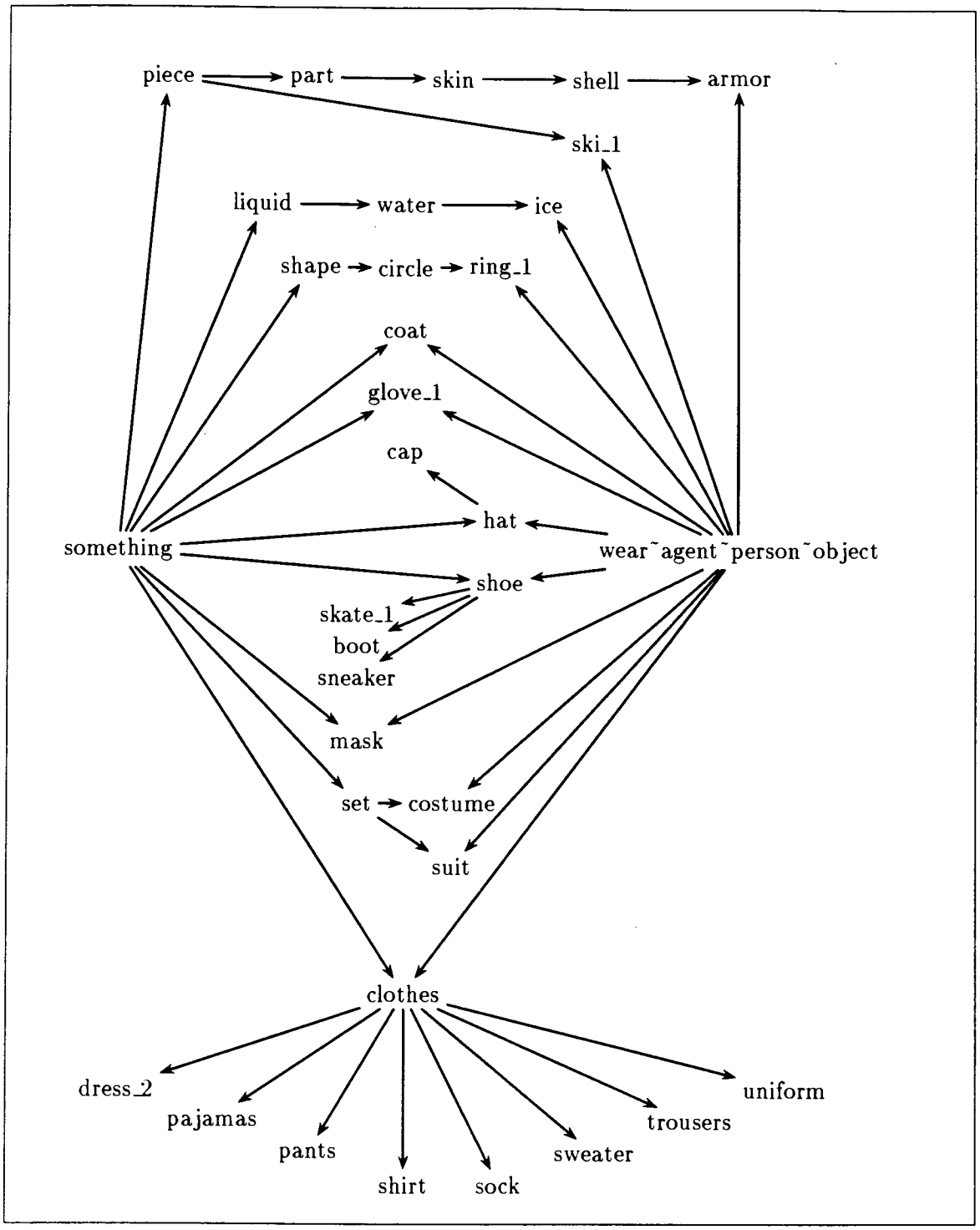


Figure 4.2: Covert category **wear~agent~person~object** part of the hierarchy.

> *... the semantic relations that played the most important role in the ranking and restriction of membership were modification, part-whole, function, agent and object.* [93]

She also mentions location and instrument as being used sometimes in the ranking but not as much.

Table 4.7 shows the distribution of the relations that are part of the covert categories found. To be able to compare with the work of Markowitz, we must look at Level 1 categories as they contain only one relation. We present results without the adjusting factors as they can be calculated on the raw data. Using the adjusting factors would mean to divide all the resulting numbers by a constant which would not change any proportion.

It is interesting to note that for both iterations, about 20% of the covert categories are based on an agent case role, and as well around 20% are based around an object case role. Around 11% are based on a modification to a verb.

With these three relations, modif, agent and object, we cover 50% of all covert categories found. These three are among the five noted by [93] as the most used by humans for ranking objects. The other two relations, function and part-whole are not present here among our covert categories as we are only looking at case roles for verbs[4]. The **function** relation combines a noun with a verb (but in the reverse order to a case role) and the **part-whole** relation combines two nouns.

The location relation mentioned in [93] as being of some importance, is harder to isolate as it is distributed among prepositions and adverbs (in, on, to, from, at, into, where, through, by) that are present in 209 covert categories (in:70, on:40, to:23, from:20, at:16 into:13, where:12, through:9, by:6), that is 25% of all covert categories. This 25% contains relations of manner, agent, time, etc, as the prepositions are ambiguous and we have not analyzed in how many cases they really did express a location relation.

The last relation mentioned in [93] is the instrument relation, and it is present here in 3.6% of the covert categories.

[4]All the mechanisms are in place to find covert categories around nouns as well as around verbs, we just decided to emphasize the grouping of nouns around verbs.

Table 4.7: Relations found within the covert categories

Relation	Number of covert categories			
	Iteration 1		Iteration 2	
	nb	%	nb	%
agent	180	20.8	182	22.6
object	162	18.7	166	20.6
modif	97	11.2	93	11.6
in	70	8.1	75	9.3
on	41	4.7	34	4.2
with	31	3.6	25	3.1
goal	27	3.1	25	3.1
and	26	3.0	6	0.7
to	23	2.7	20	2.5
from	20	2.3	17	2.1
of	18	2.1	15	1.9
at	16	1.8	12	1.5
for	14	1.6	11	1.4
into	13	1.5	14	1.7
is-a	13	1.5	14	1.7
what	13	1.5	13	1.6
when	12	1.4	13	1.6
where	12	1.4	13	1.6
through	9	1.0	7	0.9
like	7	0.8	6	0.7
by	6	0.7	6	0.7
about	5	0.6	5	0.6
others	51	5.9	31	3.9
total	866	100	804	100

4.5 Concept Clustering

The final aspect of our LKB, which augments the information found in dictionary definitions is the concept clusters.

Clustering from the main menu leads us to menu 4.5.1 which shows all possible operations needed for clustering.

Menu 4.5.1 -

Assign Trigger word.
Trigger forward.
Trigger backward.
Expansion forward.
Expansion backward.
Show word cluster.
Show resulting CCKG.
Set threshold for SSWs.

In this clustering section, we first present one large example of clustering, detailing all the necessary steps. The Trigger Word used for that example is **post-office**. The threshold, for a word to be considered a Semantically Significant Word (SSW), is set at 42, meaning that for a word to be considered a SSW, it must occur less than 42 times in the dictionary (see section 3.4.1). This corresponds to eliminating 75% of all word occurrences, and 8% of all possible words. The Graph Matching Threshold (GMT) is set at GMT(1,0) for the trigger phase and GMT(2,1) for the expansion phase (see section 3.4.3). This means that the graphs to be joined must have at least one SSW in common for the trigger phase, and at least two SSWs in common plus a relation in common for the expansion phase.

We then show how that particular cluster built around **post-office** would be modified if we changed the threshold on SSWs.

Continuing with that same cluster around **post-office**, we show the effect of using other words that resulted in the cluster as trigger words. When we start with **post-office**, we obtain **send**, **stamp**, **mail**, **message** in the cluster, but what would happen if we use **stamp** or **send** as trigger words, would we obtain a similar cluster?

Finally we briefly present clustering results with more words used as trigger words: needle, sew, kitchen, farmer, stomach, plane, elephant, soap, wash. We show the resulting clusters for different thresholds on SSWs, and different GMTs.

4.5.1 Example of integration: POST-OFFICE

Assign Trigger word is the first option that must be chosen. Here we assign **post-office** as our trigger word.

Show word cluster can be called at any time to show what are the words included in the cluster at any particular step. Similarly Show resulting CCKG shows the resulting Concept Cluster Knowledge Graph (CCKG) at any time. Result 4.5.1 shows the initial cluster for **post-office** which comes from its definition.

Result 4.5.1 -

```
CLUSTER: post_office /

Linear output for :
graph:cluster_post_office_0;
nature:fait;
set:ens;
[post_office]<-(to)<-[go]-
  {
  (agent)->[it:ref];
  (when)->[mail_2]-
    {
    (object)->[letter:a];
    (agent)->[you:ref];
    };
  }.
```

Now we go forward in the dictionary to find SSWs that come from the definition of **post-office** and backward to find SSWs whose definitions contain **post-office**. As each SSW is examined, we try to join some of their defining graphs to the CCKG (see section 3.4.4). There are a few steps done every time a new word is looked at for joining its graphs with the cluster graph (see section 3.4.3):

1. Find common subgraphs between NewGraph and ClusterGraph

2. Verify that they contain enough concepts and relations to continue (threshold changes depending on the step backward or forward, trigger or expansion)

3. If so, perform maximal join between both graphs

4. Then, perform an internal join in case some concepts were duplicated during the join

5. Finally, perform a reduction looking for compatible concepts that are joined via compatible relations to the same concept

Trigger forward is the first step of clustering. It finds all SSWs from the cluster graph and looks into their definitions one at a time. Result 4.5.2 shows the whole process. First, two SSWs are identified, **mail_2** and **letter**. When a word with multiple senses is disambiguated, we only try to join the graphs for the appropriate sense to the cluster graph. Here, the graphs for **mail_2** are joined. The Graph Matching Threshold (GMT) is set quite low when we are in the Trigger Phase, that is the first step away from the trigger word. We have GMT(1,0). We must have one SSW in common between both graphs.

The second SSW is **letter**, and it is not disambiguated. As we do not want to join in some word senses that are not appropriate, we raise the GMT in a case of sense ambiguity, it is brought up to GMT = (2,0). This higher threshold requires the graphs to have two SSWs in common and in the present example it prevents both senses **letter_1** and **letter_2** from being joined to the graph.

Result 4.5.2 -

```
SEMANTICALLY SIGNIFICANT WORDS: mail_2 / letter /

nom: mail_2

Graph to join:
Linear output for :
graph:mail_2_A_A;
nature:fait;
set:ens;
[mail_2]-
  {
```

```
  (object)->[something:ref];
  (object)->[it:ref];
  (through)->[mail_1:the];
  }.

New graph has : 1 concepts in common and : 0 relations.
GMT(a,b): 1 concepts and 0 relations.
```

```
------------------
* possible join  *
------------------
```

```
COMMON GRAPH:
Graph:CopyOf2_A_A/post_office_0;
Nature:fait;
Set:ens;
Concepts:
c1=[mail_2:*];
Relations:
Edges:
EndGraph;

AFTER MAXIMAL JOIN :
Linear output for :
graph:cluster_post_office_0;
nature:fait;
set:ens;
[something:ref]<-(object)<-[mail_2]-
  {
  (object)->[it:ref];
  (through)->[mail_1:the];
  (object)->[letter:a];
  (agent)->[you:ref];
  (when)<-[go]-
    {
    (to)->[post_office];
    (agent)->[it:ref];
    };
  }.
```

```
----------------------------------------
* mail_2 has three object relations     *
* that will be eliminated by the        *
* internal join and reduction procedure *
----------------------------------------
```

```
AFTER REDUCTION:
Linear output for :
graph:cluster_post_office_0;
nature:fait;
set:ens;
[mail_1:the]<-(through)<-[mail_2]-
  {
  (object)->[letter];
  (agent)->[you:ref];
  (when)<-[go]-
```

```
    {
    (to)->[post_office];
    (agent)->[it:ref];
    };
  }.


Nom: letter_1

Graph to join:
Linear output for :
graph:letter_1_A_B;
nature:fait;
set:ens;
[symbol:plural]-
  {
  (object)<-[use]-
    {
    (goal)->[write]->(object)->[word:plural];
    (agent)->[person:plural];
    };
  (of)<-[letter_1:a];
  }.

New graph has : 1 concepts in common and : 0 relations.
GMT(a,b): 2 concepts and 0 relations.

COMMON GRAPH:
Graph:CopyOf1_A_B/post_office_0;
Nature:fait;
Set:ens;
Concepts:
c7=[letter_1:a];
Relations:
Edges:
EndGraph;

Nom: letter_2

Graph to join:
Linear output for :
graph:letter_2_A_A;
nature:fait;
set:ens;
[also]<-(modif)<-[letter_2:a]<-(object)<-[write]-
  {
  (on)->[paper];
  (agent)->[you:ref];
```

```
--------------------
* No join possible *
--------------------
```

```
  }.

New graph has : 1 concepts in common and : 0 relations.
GMT(a,b): 2 concepts and 0 relations.

COMMON GRAPH:
Graph:CopyOf2_A_A/post_office_0;
Nature:fait;
Set:ens;
Concepts:
c6=[letter_2:a];
Relations:
Edges:
EndGraph;
```

```
--------------------
* No join possible *
--------------------
```

Result 4.5.3 shows the result of the **trigger forward** phase.

Result 4.5.3 -

```
CLUSTER: post_office / mail_2 /

Linear output for :
graph:cluster_post_office_0;
nature:fait;
set:ens;
[mail_1:the]<-(through)<-[mail_2]-
  {
  (object)->[letter];
  (agent)->[you:ref];
  (when)<-[go]-
    {
    (to)->[post_office];
    (agent)->[it:ref];
    };
  }.
```

The second step is the option **Trigger backward**. It finds all SSWs in the AHFD that use in their definition the trigger word **post-office**. There are only two such words in the AHFD: **address** and **package**. As we are still in the trigger phase, the GMT is still low at GM(1,0) that is only one SSW in common between the graph to join and the cluster graph to allow the join. In Result 4.5.4 we can see that the first sentence of **address** leading to the graph **address_1_A_B** does not contain the word

post-office and therefore is not join to the cluster. Its second sentence leading to the graph **address_1_B_E** does contain the trigger word and therefore is joined to the cluster graph. A word (such as in this case for address) can be added to the cluster even if not all its defining graphs are joined.

Result 4.5.4 -

```
SEMANTICALLY SIGNIFICANT WORDS: address / package /

nom: address

Linear output for :
graph:address_1_A_B;
nature:fait;
set:ens;
[place:a]<-(of)<-[address:an].


New graph has : 0 concepts in common and : 0 relations.      --------------------
GMT(a,b): 1 concepts and 0 relations.                        * No join possible *
                                                             --------------------

Linear output for :
graph:address_1_B_E;
nature:fait;
set:ens;
[put]-
  {
  (object)->[address:an];
  (on)->[letter:a];
  (goal)->[tell]-
    {
    (who)->[post_office:the];
    (what)->[place:quest]<-(where)<-[send]->(object)->[it:ref];
    };
  (agent)->[you:ref];
  }.

New graph has : 2 concepts in common and : 0 relations.      -----------------
GMT(a,b): 1 concepts and 0 relations.                        * join possible *
                                                             -----------------

COMMON GRAPH:
Graph:CopyOf1_B_E/post_office_0;
Nature:fait;
Set:ens;
Concepts:
c9=[post_office:*];
```

```
Relations:
Edges:
EndGraph;


AFTER MAXIMAL JOIN :
Linear output for :
graph:cluster_post_office_0;
nature:fait;
set:ens;
[put]-
  {
  (object)->[address:an];
  (on)->[letter:a];
  (goal)->[tell]-
    {
    (who)->[post_office:the]<-(to)<-[go]-
      {                                  ----------------------------------------
      (agent)->[it:ref];                 * SSW letter is duplicated from the join *
      (when)->[mail_2]-                  * internal join will put them together   *
        {                                ----------------------------------------
        (through)->[mail_1:the];
        (object)->[letter];
        (agent)->[you:ref];
        };
      };
    (what)->[place:quest]<-(where)<-[send]->(object)->[it:ref];
    };
  (agent)->[you:ref];
  }.


AFTER REDUCTION:
Linear output for :
graph:cluster_post_office_0;
nature:fait;
set:ens;
[put:*1]-
  {
  (object)->[address:an];
  (on)->[letter]<-(object)<-[mail_2:*2]-
    {
    (where)->[place:quest]<-(what)<-[tell]-
      {
      (who)->[post_office:the]<-(to)<-[go]-
        {
        (agent)->[it:ref];
        (when)->[*2];
```

```
        };
      (goal)<-[*1];
      };
    (through)->[mail_1:the];
    (agent)->[you]<-(agent)<-[*1];
    };
  }.


nom: package

Graph to join:
Linear output for :
graph:package_1_A_A;
nature:fait;
set:ens;
[tie:*1]-
  {
  (modif)->[up];
  (object)->[package:a]<-(object)<-[send]-
    {
    (through)->[mail_1:the];
    (agent)->[you:ref]<-(agent)<-[*1];
    };
  }.

New graph has : 2 concepts in common and : 0 relations.
GMT(a,b): 1 concepts and 0 relations.      ---------------------------------
                                           * possible join, first sentence *
                                           * from definition of package    *
COMMON GRAPH:                              ---------------------------------
Graph:CopyOf1_A_A/post_office_0;
Nature:fait;
Set:ens;
Concepts:
c6=[package:*];
Relations:
Edges:
EndGraph;

AFTER REDUCTION:
Linear output for :
graph:cluster_post_office_0;
nature:fait;
set:ens;
[tie:*1]-
  {
  (modif)->[up];
```

```
  (object)->[package:a]<-(object)<-[mail_2:*2]-
    {
    (where)->[place:quest]<-(what)<-[tell]-
      {
      (who)->[post_office:the]<-(to)<-[go]-
        {
        (agent)->[it:ref];
        (when)->[*2];
        };
      (goal)<-[put]-
        {
        (object)->[address:an];
        (on)->[letter]<-(object)<-[*2];
        (agent)->[you]-
          {
          (agent)<-[*1];
          (agent)<-[*2];
          };
        };
      };
    (through)->[mail_1];
    };
  }.


Graph to join:
Linear output for :
graph:package_1_B_F;
nature:fait;
set:ens;
[bring]-
  {
  (object)->[package:plural];
  (to)->[post_office:the];
  (goal)->[mail_2];
  (agent)->[person:plural];
  }.

New graph has : 3 concepts in common and : 0 relations.
GMT(a,b): 1 concepts and 0 relations.   ----------------------------------
                                        * possible join, second sentence *
                                        * from definition of package     *
COMMON GRAPH:                           ----------------------------------
Graph:CopyOf1_B_F/post_office_0;
Nature:fait;
Set:ens;
Concepts:
c5=[post_office:*];
```

```
Relations:
Edges:
EndGraph;

AFTER REDUCTION:
Linear output for :
graph:cluster_post_office_0;
nature:fait;
set:ens;
[bring:*1]->(object)->[package:*2]<-(object)<-[tie]-
  {
  (modif)->[up];
  (agent)->[you:*3]-
    {
    (agent)<-[*1];
    (agent)<-[put:*4]-
      {
      (object)->[address:an];
      (on)->[letter]<-(object)<-[mail_2:*5]-
        {
        (goal)<-[*1];
        (object)->[*2];
        (where)->[place:quest]<-(what)<-[tell]-
          {
          (who)->[post_office:the]-
            {
            (to)<-[*1];
            (to)<-[go]-
              {
              (agent)->[it:ref];
              (when)->[*5];
              };
            };
          (goal)<-[*4];
          };
        (through)->[mail_1];
        (agent)->[*3];
        };
      };
    };
  }.
```

The resulting cluster from the trigger backward phase is {post_office, mail_2, address, package}. Its CCKG is the last graph from Result 4.5.4.

The next step is the Expansion forward. Now we look at all SSWs that are in part of the cluster graph and that are not yet in the Cluster. During the expansion

phase the GMT is a bit higher, as we are looking at more definitions and chances are that they are less related to the trigger word. We set GMT = (2,1), at least 2 common concepts and 1 common relation. When there is sense ambiguity the GMT is incremented to GMT = (3,1). In Result 4.5.5 we can see that the words **bring**, **tie**, **letter**, **send** and **mail_1** are candidates to be joined to the cluster. In all the definition graphs of **bring**, **tie** and **letter** there is only the word **package** in common with the cluster graph, so no join is possible. It is important during the expansion phase to put the GMT high enough that we do not expand in directions more specific to one particular word in the cluster (here bring and tie relating more to package) because that does not increase the coherence of the cluster but contrarily expands it by adding new concepts in all directions. The case of **send** is quite different. Both sentences defining **send** were not parsed and therefore no graphs were generated and no join can be envisaged. Finally **mail_1** has a GS(2,1) which is at the threshold level and it can be joined.

Result 4.5.5 -

```
SEMANTICALLY SIGNIFICANT WORDS: bring / tie / letter / send / mail_1 /

nom: bring

Graph to join:
Linear output for :
graph:bring_1_A_A;
nature:fait;
set:ens;
[bring]->(equiv)->[take]-
  {
  (object)->[something:ref];
  (with)->[you:ref];
  }.

New graph has : 1 concepts in common and : 0 relations.
GMT(a,b): 2 concepts and 1 relations.

COMMON GRAPH:
Graph:CopyOf1_A_A/post_office_0;
Nature:fait;
Set:ens;
Concepts:
```

```
c4=[you:ref];
Relations:
Edges:
EndGraph;

Nom: tie_1

Graph to join:
Linear output for :
graph:tie_1_A_A;
nature:fait;
set:ens;
[tie_1]->(equiv)->[hold]-
  {
  (object)->[something:ref];
  (modif)->[together];
  (with)->[string];
  (with)->[rope];
  }.

New graph has : 1 concepts in common and : 0 relations.
GMT(a,b): 3 concepts and 1 relations.

COMMON GRAPH:
Graph:CopyOf1_A_A/post_office_0;
Nature:fait;
Set:ens;
Concepts:
c1=[tie_1:*];
Relations:
Edges:
EndGraph;

Nom: tie_2

Linear output for :
graph:tie_2_A_A;
nature:fait;
set:ens;
[tie_2]->(equiv)->[make]->(object)->[knot:plural]...
   ->(in)->[string]->(or)->[rope]->(or)->[ribbon].


New graph has : 1 concepts in common and : 0 relations.
GMT(a,b): 3 concepts and 1 relations.

COMMON GRAPH:
Graph:CopyOf2_A_A/post_office_0;
```

```
---------------
* No join     *
---------------
```

```
------------
* No join  *
------------
```

```
Nature:fait;
Set:ens;
Concepts:
c1=[tie_2:*];
Relations:
Edges:
EndGraph;

Graph to join:
Linear output for :
graph:tie_2_B_A;
nature:fait;
set:ens;
[tie_2]-
  {
  (object)->[bow:plural]-
    {
    (attribut)->[pretty];
    (on)->[present:plural]->(attribut)->[all];
    };
  (agent)->[joy];
  }.

New graph has : 1 concepts in common and : 0 relations.      ------------
GMT(a,b): 3 concepts and 1 relations.                        * No join  *
                                                             ------------

COMMON GRAPH:
Graph:CopyOf2_B_A/post_office_0;
Nature:fait;
Set:ens;
Concepts:
c1=[tie_2:*];
Relations:
Edges:
EndGraph;

Nom: letter_1

Graph to join:
Linear output for :
graph:letter_1_A_B;
nature:fait;
set:ens;
[symbol:plural]-
  {
  (object)<-[use]-
    {
    (goal)->[write]->(object)->[word:plural];
```

```
    (agent)->[person:plural];
    };
  (of)<-[letter_1:a];
  }.

New graph has : 1 concepts in common and : 0 relations.      ------------
GMT(a,b): 3 concepts and 1 relations.                        * No join  *
                                                             ------------

COMMON GRAPH:
Graph:CopyOf1_A_B/post_office_0;
Nature:fait;
Set:ens;
Concepts:
c7=[letter_1:a];
Relations:
Edges:
EndGraph;

Nom: letter_2

Graph to join:
Linear output for :
graph:letter_2_A_A;
nature:fait;
set:ens;
[also]<-(modif)<-[letter_2:a]<-(object)<-[write]-
  {
  (on)->[paper];
  (agent)->[you:ref];
  }.


New graph has : 1 concepts in common and : 0 relations.      -----------
GMT(a,b): 3 concepts and 1 relations.                        * No join *
                                                             -----------

COMMON GRAPH:
Graph:CopyOf2_A_A/post_office_0;
Nature:fait;
Set:ens;
Concepts:
c6=[letter_2:a];
Relations:
Edges:
EndGraph;

nom: send   ------------------------
            * No sentences parsed *
            ------------------------
```

```
nom: mail_1

Linear output for :
graph:mail_1_A_I;
nature:fait;
set:ens;
[be]-
  {
  (how)->[send]-
    {
    (object)->[letter:plural]->(and)->[package:plural];
    (from)->[place:one]->(to)->[another:ref];
    (agent)->[we:ref];
    };
  (agent)->[mail_1:the];
  }.


New graph has : 2 concepts in common and : 1 relations.      ----------------
GMT(a,b): 2 concepts and 1 relations.                        * Possible join *
                                                             ----------------

COMMON GRAPH:
Graph:CopyOf1_A_I/post_office_0;
Nature:fait;
Set:ens;
Concepts:
c2=[mail_2:*];
c3=[letter:plural];
Relations:
r2=(object);
Edges:
r2,c2,1;
r2,c3,2;
EndGraph;

AFTER REDUCTION:
Linear output for :
graph:cluster_post_office_0;
nature:fait;
set:ens;
[be:*1]->(how)->[mail_2:*2]-
  {
  (from)->[place:one]->(to)->[another:ref];
  (goal)<-[bring:*3]->(object)->[package:plural*4]-
    {
    (and)<-[letter]-
      {
```

```
(on)<-[put]-
  {
  (object)->[address:an];
  (goal)->[tell]-
    {
    (who)->[post_office:the]-
      {
      (to)<-[*3];
      (to)<-[go]-
        {
        (agent)->[it:ref];
        (when)->[*2];
        };
      };
    (what)->[place:quest]<-(where)<-[*2];
    };
  (agent)->[we]-
    {
    (agent)<-[*3];
    (agent)<-[tie]-
      {
      (modif)->[up];
      (object)->[*4];
      };
    (agent)<-[*2];
    };
  };
(object)<-[*2];
};
(through)->[mail_1:the]<-(agent)<-[*1];
}.
```

The resulting cluster from the Expansion forward phase is {post_office, mail_2, address, package, mail_1}. Its CCKG is the last graph from Result 4.5.5.

The next step is Expansion backward in which we look in the dictionary for all SSWs that contain in their definition any of the words in the Cluster. The GMT is GMT = (2,1) again for the expansion phase. One SSW found is mail, meaning any sense of the word, but as ARC-Concept looks into each sense, it finds out that both mail_1 and mail_2 are already in the cluster, so it continues with the second SSW stamp. There are many concepts and relations in the second graph for stamp in

common with the cluster graph.

Result 4.5.6 -

```
SEMANTICALLY SIGNIFICANT WORDS: mail / stamp /

Nom: mail_1        ---------------------------------------
Nom: mail_2        * Present individually in the cluster *
                   ---------------------------------------
nom: stamp

Graph to join:
Linear output for :
graph:stamp_1_A_A;
nature:fait;
set:ens;
[small]<-(attribut)<-[stamp:a]<-(piece-of)<-[paper]-
  {
  (with)->[word:plural];
  (with)->[number:plural];
  (with)->[picture:a]->(on)->[it:ref];
  }.


New graph has : 1 concepts in common and : 0 relations.      -----------
GMT(a,b): 2 concepts and 1 relations.                        * No join *
                                                             -----------
COMMON GRAPH:
Graph:CopyOf1_A_A/post_office_0;
Nature:fait;
Set:ens;
Concepts:
c4=[number:plural];
Relations:
Edges:
EndGraph;

Graph to join:
Linear output for :
graph:stamp_1_B_E;
nature:fait;
set:ens;
[buy:*1]-
  {
  (object)->[stamp:plural];
  (goal)->[put]->(on)->[letter:plural]<-(object)<-[send]-
    {
    (through)->[mail:the];
```

```
    (agent)->[person:plural]<-(agent)<-[*1];
    (object)->[package:plural];
    };
  }.
```

```
New graph has : 4 concepts in common and : 4 relations.
GMT(a,b): 2 concepts and 1 relations.
```

```
----------------
* Possible join *
----------------
```

```
COMMON GRAPH:
Graph:CopyOf1_B_E/post_office_0;
Nature:fait;
Set:ens;
Concepts:
c4=[letter:plural];
c5=[package:plural];
c6=[mail_2:*];
c7=[mail_1:the];
c8=[we:plural];
Relations:
r2=(object);
r3=(through);
r4=(agent);
r8=(object{certainty:;});
Edges:
r2,c6,1;
r2,c4,2;
r3,c6,1;
r3,c7,2;
r4,c6,1;
r4,c8,2;
r8,c6,1;
r8,c5,2;
EndGraph;
```

```
AFTER REDUCTION:
Linear output for :
graph:cluster_post_office_0;
nature:fait;
set:ens;
[buy:*1]-
  {
  (object)->[stamp:plural];
  (goal)->[put]->(on)->[letter:plural*2]->(and)->[package:plural*3]...
      <-(object)<-[bring:*4]->(to)->[post_office:the*5]<-(who)<-[tell:*6]...
      ->(what)->[place:quest]<-(where)<-[mail_2:*7]-
    {
```

```
    (from)->[place:one]->(to)->[another:ref];
    (how)<-[be]->(agent)->[mail_1:the]<-(through)<-[*7];
    (goal)<-[*4];
    (agent)->[we:plural]-
      {
      (agent)<-[*1];
      (agent)<-[*4];
      (agent)<-[tie]-
        {
        (modif)->[up];
        (object)->[*3];
        };
      (agent)<-[put]-
        {
        (object)->[address:an];
        (on)->[*2];
        (goal)->[*6];
        };
      };
    (object)->[*3];
    (object)->[*2];
    (when)<-[go]-
      {
      (to)->[*5];
      (agent)->[it:ref];
      };
    };
  }.
```

The resulting cluster of the Expansion backward phase is {post-office, mail_2, address, package, mail_1, stamp}. It is interesting to note that only one sentence from stamp was pulled in, not the details about what a stamp looks like, but its usage.

At this point we can repeat the Expansion forward and Expansion backward steps until there are no more changes in the cluster. We continue with the same GMT, as we are still in the expansion phase.

Result 4.5.7 -

```
Expansison Forward:

SEMANTICALLY SIGNIFICANT WORDS: bring / tie / letter /
```

```
   ------------------------------------------------------------
   * All graph representations are tried an no join is possible *
   ------------------------------------------------------------

Expansion Backward:

SEMANTICALLY SIGNIFICANT WORDS: mail /

   ------------------------------------------------------------------------
   * All senses of mail are already in the cluster so there is no addition here *
   ------------------------------------------------------------------------
```

Result 4.5.8 shows the final cluster.

Result 4.5.8 -

```
CLUSTER: post_office / mail_2 / address / package / mail_1 / stamp /

Linear output for :
graph:cluster_post_office_0;
nature:fait;
set:ens;
[buy:*1]-
  {
  (object)->[stamp:plural];
  (goal)->[put]->(on)->[letter:plural*2]->(and)->[package:plural*3]...
    ... <-(object)<-[bring:*4]->(to)->[post_office:the*5]<-(who)<-[tell:*6]...
    ... ->(what)->[place:quest]<-(where)<-[mail_2:*7]-
    {
    (from)->[place:one]->(to)->[another:ref];
    (how)<-[be]->(agent)->[mail_1:the]<-(through)<-[*7];
    (goal)<-[*4];
    (agent)->[we:plural]-
      {
      (agent)<-[*1];
      (agent)<-[*4];
      (agent)<-[tie]-
        {
        (modif)->[up];
        (object)->[*3];
        };
      (agent)<-[put]-
        {
        (object)->[address:an];
        (on)->[*2];
        (goal)->[*6];
```

```
      };
    };
  (object)->[*3];
  (object)->[*2];
  (when)<-[go]-
    {
    (to)->[*5];
    (agent)->[it:ref];
    };
  };
}.
```

4.5.2 Changing the threshold

The option Set threshold for SSWs allows the user to decide on the threshold for establishing Semantically Significant Words. It gives us the following menu.

Menu 4.5.2 -

Compute total.
Compute thresholds.
Return to main menu.

Result 4.5.9 shows the result from Compute total based on the file **SSW_occ.dat** which contains the number of occurrences of all words in the AHFD.

Result 4.5.9 -

```
total number of word occurrences: 58354
total number of words: 2073
```

There are 2073 words used in total in the dictionary, and there are 58354 occurrences of all these words among all the sentences.

Option Compute thresholds allows the user to specify how many words and word occurrences he/she wants to discard. Result 4.5.10 shows the dialogue about changing the threshold.

Result 4.5.10 -

Table 4.8: Different thresholds

Max number of occurrences to be considered a SSW	% of occurrences discarded	% words discarded
5076	10	0
2904	20	0
1095	30	0
623	40	0
264	50	1
126	60	2
62	70	6
42	75	8
35	77	10
27	80	13
18	85	19
11	90	29
5	95	48
4	97	60
2	99	80
1	100	100

```
ARC-Concept: What percent of occurrences to discard?
User: 50

ARC-Concept: To eliminate 50% of the occurrences, that is 29177 occurrences,
             we must eliminate 29 words, that is 1% of the words.

             We consider SSW, a word with less than : 264 occurrences.
```

Table 4.8 shows for different thresholds, how many words and occurrences of words get discarded. It is interesting to note that only 1% of the words account for 50% of all occurrences of words in the AHFD. And only 10% of the words account for 77% of all occurrences.

The threshold 42 that we used in the previous subsection for the example with post-office corresponds to excluding 75% of the occurrences and 8% of the words. Figure 4.3 shows the resulting cluster for the same trigger word for different thresholds.

Sometimes the resulting cluster is the same but relaxing the threshold allows more SSWs to be explored and more iterations need to be done. At each step during the clustering process, we show the SSWs to be explored and we emphasize in bold the words that become part of the cluster.

For the trigger word **post-office**, a threshold 42 seems a good threshold as we do not explore multiple words that never get brought in, but we do get a large cluster. The cluster stays the same if we loosen the threshold to 126 but we need to explore more words and therefore the process takes longer to reach the same conclusion.

We try with another trigger word **farmer** to show the influence of the threshold for SSW on the resulting cluster. For the word **farmer** as we can see in Figure 4.4, the threshold 42 make us explore many, many words that do not get joined in. But even with this long exploration the resulting cluster is qualitatively quite good. When we lower the threshold to 18 (making it more difficult to be a SSW), the exploration is much more limited and we only lose the words **rose_2** and **morning**. Lowering the threshold even more reduces the exploration so much that we lose almost everything.

A threshold of 18, that was way too limiting for the trigger word **post-office**, is fine for the trigger word **farmer**. We will come back to this remark in section 4.5.4.

4.5.3 Changing the trigger word within the cluster

From the resulting cluster: {**post-office**, **mail_2**, **address**, **package**, **mail_1**, **stamp**} we performed an experiment in which all the different words in the cluster were used as trigger words themselves and we generated clusters around them. From the concepts present in the cluster from **post-office** we formed a set of clusters. That particular set was an extremely stable set of clusters. Result 4.5.11 shows all the resulting clusters and shows the coherence of a subgroup of concepts by the number of time they occur in the clusters.

Result 4.5.11 -

```
Cluster:

post-office:  {address, mail_1, mail_2, package, post-office, stamp}
mail_2:       {mail_2, mail_1, package, stamp}
```

Threshold = 42
% occurrences discarded = 75

Trigger word:	post-office
Trigger forward:	mail_2, letter
Trigger backward:	address, package
Expansion forward (1):	bring, tie, send, mail_1
Expansion backward (1):	mail, stamp
Expansion forward (2):	bring, tie, send
Expansion backward (2):	mail

Resulting cluster: {post-office, mail_2, address, package, mail_1, stamp}

Threshold = 126
% occurrences discarded = 60

Trigger word:	post-office
Trigger forward:	mail_2, letter
Trigger backward:	address, package
Expansion forward (1):	bring, tie, up, put, tell, letter, send, mail_1
Expansion backward (1):	mail, stamp
Expansion forward (2):	buy, send, bring, tie, up, tell, put, letter
Expansion backward (2):	land, mail

Resulting cluster: {post_office, mail_2, address, package, stamp, mail_1 }

Threshold = 18
% occurrences discarded = 85

Trigger word:	post-office
Trigger forward:	NONE
Trigger backward:	address
Expansion forward (1):	NONE
Expansion backward (1):	NONE

Resulting cluster: {post_office, address}

Figure 4.3: Varying the SSW threshold for clustering around post-office

THRESHOLD = 42, GMT(1,1) for expansion

Trigger word:	farmer
Trigger forward:	farm, start, early, morning
Trigger backward:	been, field(1), land, patch, raise(2), tractor, wall
Expansion forward (1):	help, prepare, rise, sun, first, start, early, area, vegetable
Expansion backward (1):	barn, been, brush, bulldozer, continue, cool, country, cow, cowboy, cowgirl, early, field, goat, horse, last, mad, o'clock, patch, pig, raise, ring, rise, rooster, rose(2), sleep, spend, stable, sunrise, trailer, wake, wall
Expansion forward (2):	o'clock, rise, wool, help, prepare, first, start, early, vegetable, milk, sun
Expansion backward (2):	barn, been, brush, bulldozer, continue, cool, country, cow, early, field, horn, horse, last, mad, milk, o'clock, patch, pig, raise, ring, rise, rooster, rose, sleep, spend, stable, sunrise, trailer, wake, wall

CLUSTER: { farmer, farm, morning, field_1, raise_2, tractor, cowboy, cowgirl, goat, rose_2 }

THRESHOLD = 18, GMT(1,1) for expansion

Trigger word:	farmer
Trigger forward:	farm, early
Trigger backward:	patch, raise(2), tractor, wall
Expansion forward (1):	prepare, early
Expansion backward (1):	barn, bulldozer, cow, cowboy, cowgirl, goat, patch, pig, raise, stable, trailer, wall
Expansion forward (2):	wool, prepare, early
Expansion backward (2):	barn, bulldozer, cow, horn, patch, pig, raise, stable, trailer, wall

CLUSTER: {farmer, farm, raise_2, tractor, cowboy, cowgirl, goat }

THRESHOLD = 11, GMT(1,1) for expansion

Trigger word:	farmer
Trigger forward:	NONE
Trigger backward:	been, field, land, patch, raise, tractor, wall
Expansion forward (1):	prepare, early
Expansion backward (1):	bulldozer, patch, trailer

CLUSTER: {farmer, tractor}

Figure 4.4: Varying the SSW threshold for clustering around farmer

```
address:      {address, mail_1, mail_2, package, post-office, stamp}
package:      {address, bring, mail_1, mail_2, package,  post-office, stamp}
mail_1:       {address, mail_1, mail_2, package, post-office, stamp}
stamp:        {address, letter_1, mail_1, mail_2, package, post-office, stamp}

Concepts present with their number of occurrences:

   address      (5)
   mail_1       (6)
   mail_2       (6)
   package      (6)
   post-office  (5)
   stamp        (6)
   bring        (1)
   letter_1     (1)
```

The subgroup: address, mail_1, mail_2, package, post-office, stamp forms a very coherent set.

We perform the same task with **farmer** as our trigger word. The threshold is lowered to reduce the search space, but the graph matching threshold is lowered as well to (GMT(1,0) trigger, GMT(2,0) expansion) to allow more matches.

It generates a much less stable set of clusters as shown in Result 4.5.12.

Result 4.5.12 -

```
Cluster:

farmer:  {cowboy, cowgirl, farm, farmer, goat, raise_2, tractor}
farm:    {barn, cattle, cow, cowboy, cowgirl, farm, farmer, farmer, goat, pig,
          raise_2, stable}
raise_2: {cowboy, cowgirl, farm, farmer, raise_2}
tractor: {engine, farmer, prepare, tractor, tugboat}
cowboy:  {cattle, cowboy, cowgirl, farm, farmer}
cowgirl: {cattle, cowboy, cowgirl, farm, farmer}
goat:    {chin, cotton, farm, goat, hair, lion, lioness, raise_2, stable, sweater,
          wizard, wool, yarn}

Concepts present with their number of occurrences:

   farm         (6)
   farmer       (6)
   cowboy       (5)
   cowgirl      (5)
   raise_2      (4)
```

```
goat        (3)
cattle      (3)
tractor     (2)
stable      (2)

barn, cow, pig, chin, cotton, hair, lion,   (1)
lioness, sweater, wizard, wool, yarn
```

Subgroup: {farm, farmer, cowboy, cowgirl, raise_2} forms a coherent set. It gets harder to see the place of the subgroup {goat, cattle, tractor, stable}. In such a case where it is less clear than with the previous example, we might want to continue our search and start with some of these in-between words as trigger words, and then look over the larger set of clusters. In this particular case, it means looking at cattle and stable as the other two words goat and tractor have been used as trigger words already. The new results are shown in Result 4.5.13.

Result 4.5.13 -

```
Cluster:

farmer:  {cowboy, cowgirl, farm, farmer, goat, raise_2, tractor}
farm:    {barn, cattle, cow, cowboy, cowgirl, farm, farmer, farmer, goat, pig,
          raise_2, stable}
raise_2: {cowboy, cowgirl, farm, farmer, raise_2}
tractor: {engine, farmer, prepare, tractor, tugboat}
cowboy:  {cattle, cowboy, cowgirl, farm, farmer}
cowgirl: {cattle, cowboy, cowgirl, farm, farmer}
goat:    {chin, cotton, farm, goat, hair, lion, lioness, raise_2, stable, sweater,
          wizard, wool, yarn}

cattle:  {cattle, meat, beef, cowboy, cowgirl, ham}
stable:  {stable, farm, raise_2, goat}

Concepts present with their number of occurrences:

   farm        (7)
   farmer      (6)
   cowboy      (6)
   cowgirl     (6)
   raise_2     (5)
   goat        (4)
   cattle      (3)
   stable      (3)
   tractor     (2)
```

```
ham, beef, meat (1) *** new ones ***
barn, cow, pig, chin, cotton, hair, lion,   (1)
lioness, sweater, wizard, wool, yarn
```

There is still no clear cut distinction between the words that should definitely be part of the cluster, and those that should not. The ones occurring only once should be eliminated. A word occurring twice that was used as a trigger word (as tractor) in fact only occurs once in another word's cluster. We can have the resulting cluster as {farm, farmer, cowboy, cowgirl, raise_2, goat, cattle, stable}.

4.5.4 Results on other trigger words

This section shows a few more results of clusters starting from different words. Table 4.9 shows clusters for different thresholds for establishing the Semantically Significant Words and different thresholds for the Graph Matching criteria. Again, we judge the resulting clusters in a qualitative manner. The resulting clusters tend to get less focused as we loosen the threshold for SSW. Instead of setting it, the clustering program should adjust the threshold automatically so that it would not explore more than X number of words at each step. Actually the trigger phase is less problematic, and the threshold could be looser at that phase. We are only looking at one word, the word it uses in its definition (trigger forward) and the words that are defined using it (trigger backward). When we enter the expansion phase, especially the backward phase, the search space can become quite large, as we are trying to find all the words in the dictionary that are defined using one of the words presently in the cluster. The larger the cluster becomes, the larger the search space becomes, and the lower the threshold should be.

The effect of the Graph Matching Threshold is less dramatic. We tried three variations for the expansion phase, GMT(1,1), GMT(2,1), GMT(3,0). GMT(1,1) means that there must be one SSW in common and a subgraph containing two concepts and one relation in common. The SSW could be part of the subgraph or not. GMT(2,1) GMT(1,1) means that there must be two SSWs in common and a subgraph containing

two concepts and one relation in common. The SSWs could be part of the subgraph or not. The third possibility GMT(3,0) means that you need three SSWs in common not necessarily part of any structure.

4.6 Discussion

First, the CoGITo system was briefly introduced, that is the CG platform used for our CG development.

All the steps presented in chapter 2 for converting a sentence into a Conceptual Graph were presented using three examples: doughnut, bat, piano. We saw the first transformation steps, tagging, parsing, parse-to-cg transformations, as well as all steps for structural and semantic disambiguation.

Then a few results were presented on applying all heuristics from chapter 2 to all the words in the dictionary. We showed the average number of parse per sentence and the reduction achieved by each step of structural disambiguation.

Some results on covert categories were shown. We presented the number of categories extracted when varying the Covert Threshold. Two examples, **live~in** and **wear~agent~person~object**, were chosen to exemplify the modification introduce by covert categories within the type hierarchy. We were able to give a sense of how much more dimensions we need to explore to address the problem of similarity between words.

The last section showed results from clustering. We gave a large detailed example of all the clustering steps (trigger forward and backward, expansion forward and backward) starting from the trigger word **post-office**. Loosening the SSWs threshold allows an expansion of the search space, but it might still result in the same cluster and spend more time to find it. On the other hand, tightening the SSWs threshold might result in missing a lot of important words that should be part of the cluster. We suggest that the threshold should be adaptive and vary as the clustering process is going on.

Testing ideas is always interesting as it opens avenues not thought of before. Trying a clustering process around trigger words that are part of the resulting cluster from

Table 4.9: Multiple clusters from different words

Trigger word	Threshold for SSW	Threshold for expansion phase	Cluster
needle_1	42	GMT(2,1)	{needle_1, sew, cloth, thread, wool, handkerchief, pin, ribbon, string, rainbow}
needle_1	12	GMT(1,1)	{needle_1, thread}
needle_1	18	GMT(3,0)	{needle_1, sew, thread}
sew	42	GMT(2,1)	{sew, cloth, needle_1, needle_2, thread, button, patch_1, pin, pocket, wool, ribbon, rug, string, nest, prize, rainbow}
sew	18	GMT(1,1)	{sew, needle_1, needle_2, thread, button, pocket}
sew	18	GMT(3,0)	{sew, needle_1, needle_2, thread, button, pocket}
kitchen	42	GMT(2,1)	{kitchen, stove, refrigerator, pan, box}
kitchen	12	GMT(2,1)	{kitchen, stove, refrigerator, pan}
kitchen	12	GMT(1,1)	{kitchen, stove, refrigerator, pan, pot, clay}
kitchen	18	GMT(3,0)	{kitchen, stove, refrigerator, pan}
stove	18	GMT(1,1)	{stove, pan, kitchen, refrigerator, pot, clay}
stove	18	GMT(3,0)	{stove, pan, kitchen}
stomach	42	GMT(1,1)	{stomach, kangaroo, pain, swallow, mouth}
airplane	18	GMT(1,1)	{airplane, wing, airport, fly_2, helicopter, jet, kit, machine, pilot, plane}
airplane	18	GMT(3,0)	{airplane, wing, airport, helicopter, jet, kit, pilot, plane}
elephant	12	GMT(1,1)	{elephant, ear, zoo}
elephant	42	GMT(1,1)	{elephant, skin, trunk_1, ear, zoo, bark, leather, rhinoceros}
soap	8	GMT(1,1)	{soap, bath, bubble, suds, wash, boil, steam}
soap	18	GMT(1,1)	{soap, dirt, mix, bath, bubble, suds, wash, boil, steam}
soap	18	GMT(3,0)	{soap, dirt, mix, bath, bubble, suds, wash, boil}
soap	42	GMT(1,1)	{soap, help, dirt, mix, bath, bubble, suds, wash, clean_2, boil, anchor, steam}
wash	16	GMT(1,1)	{ wash, soap, bath, bathroom, suds, bubble, boil, steam}
wash	42	GMT(1,1)	{ wash, soap, bath, bathroom, suds, bubble, boil, clean_2, steam}

another word opened the door into a new dimension of analysis for the clusters. We could see the coherence of the set as it is more or less variant when we start from different words. We started to investigate the idea of having a subset of words that is more central to the cluster and some other words are more on the edge.

Chapter 5

DISCUSSION AND CONCLUSION

This dissertation presented a method for transforming a machine readable dictionary into a Lexical Knowledge Base (LKB) made from Conceptual Graphs (CGs).

In chapter 2 we showed how to convert a definition made of a few natural language sentences into a conceptual graph representation that is as structurally and semantically disambiguated as possible. The CGs for all the nouns and verbs in the American Heritage First Dictionary (AHFD) formed the first major part of our LKB which is seen in its entirety in chapter 3. The LKB contains four parts: the **graph definitions**, the **concept lattice**, the **relation lattice** and the **concept clusters**. Chapter 4 presented the implementation of the ideas seen in the previous two chapters. We took the reader through numerous examples to show all the different operations possibly performed by **ARC-Concept**, our **A**cquisition, **R**epresentation and **C**lustering system which allows us to build all four parts of the LKB.

The present chapter will develop as follows. First, section 5.1 summarizes the research presented. Second, section 5.2 emphasizes the major contributions from this research. Third, section 5.3 looks at a few specific problems that were encountered. Fourth, section 5.4 offers numerous suggestions for future research. Finally, section 5.5 provides some general conclusions.

5.1 Dissertation summary

In the introduction, we stated the following problem to be addressed: How can one, from existing information, (semi-)automatically create a substantial lexical knowledge base that is useful for processing, disambiguating and understanding sentences. This is an important problem because manual construction of an LKB can be labour intensive, error-prone and less systematic that an automatic process.

For our research, the existing information is a children's first dictionary of which we claimed as our first hypothesis that it would be a good source of information to build an LKB if we focus toward NLP applications that need to understand sentences used in a non-technical, daily usage type of context.

We showed in chapter 2 all our processes to extract the information from the AHFD and build a set of graph definitions. The graph definitions represent in a more disambiguated form the information given by the sentences in the AHFD. We build the graph definitions through multiple iterations, and by using the partially built LKB to better analyze and disambiguate the sentences in the AHFD we validated not only our first hypothesis but also our second, which said that the processes developed to automatically build the LKB, can also be used to augment and restructure the information contained in that LKB.

Our third hypothesis was that the LKB can be structured so that words are "defined" in terms of their relationship to other words in the dictionary. The covert categories and the concept clusters presented in chapter 3 show interesting structures resulting from the definition of words in terms of other words.

To elaborate on the points made above, we suggested the use of a children's first dictionary, the American Heritage First Dictionary (AHFD) for its emphasis on daily usage of words giving us a "naive" view on the world. When we look at the simple definitions of the AHFD, it is amazing to see how much information they actually provide through the usage and examples, and how this information is often what we mostly need to understand a non-technical daily conversation. By using the AHFD as our source of lexical information, we were able to restrict our vocabulary to result in a project of reasonable size, dealing with general knowledge about day-to-day concepts

and actions.

The conceptual graph formalism was used throughout the construction of the LKB. The graph matching procedures, finding maximal common subgraphs and joining graphs, are the basic procedures for the construction and update of our LKB.

All the steps from dictionary definitions to conceptual graph representations were presented: the morphological analysis, the syntactic analysis to generate a parse tree from the sentence, the set of parse-to-CG rules to transform the parse tree into a surface semantic conceptual graph. We looked into structural disambiguation, investigating conjunction and prepositional attachment. Finally we looked at semantic disambiguation, trying to find deeper semantic relations, performing some anaphora resolution and word sense disambiguation.

We saw how the different parts of a definition, description, usage and example, are assigned a different weight and play a different role in the LKB. The description usually gives information essential to the definition of the word, and it also contains the information used for building the concept hierarchy. The usage part is rich in information expressing generalities about the world. It uses keywords that can be interpreted as certainty information. The specific examples give the best illustration of the dynamic aspect of the LKB; processing them can make us change our expectations about typical situations and take note of exceptions to rules.

As an important part of our LKB, taxonomic relations were found through the analysis of the graph definitions from the AHFD. We reviewed how most research on knowledge extraction from dictionaries looks for taxonomic relations by finding the genus of the definition. The AHFD most often gives a taxonomic link in the first sentence of a definition using the genus/differentia structure. We discussed the necessity of allowing tangled hierarchies to be able to represent all the information found in those definitions.

We explored the idea covert categories and showed that we should include those unlabeled classes in the concept hierarchy. These covert categories are often superclasses whose subclasses occupy the same case relation to a verb. We saw the dual relation between covert categories and selectional restrictions. The covert categories

are created by processing information already in the LKB and can be used to complement the concept hierarchy in semantic similarity measures. The facet of a human's lexical knowledge that has received the most attention in recent efforts to build LKBs is the set of lexical relations to a given word, most frequently the -nym relations (synonym, antonym, hyponym, meronym) [33, 23, 65, 103]. This exploration of the covert categories made us question the importance accorded to those particular relations, especially (as it is the most widely used) the importance given to the traditional taxonomy, the type/supertype information. Sometimes in the AHFD, a word is not defined via the genus/differentia structure. This differs from the adult dictionaries, where a word of a certain part of speech is always defined through another word of the same part of speech. The adult dictionaries will find an abstract or complicated nominalization in order to assign a genus. In the AHFD, a noun can be put into a relationship to another part of speech, the most frequent case being that a noun is given as a case relation to a verb.

We presented another important part of the LKB, the building of concept clusters. We showed the multiple steps leading to the building of **Concept Clustering Knowledge Graphs** (CCKGs). Those knowledge structures are built within the LKB and can be seen as special structures integrating multiple parts of the LKB around a particular concept. The CCKGs could be either permanent or temporary structures depending on the application using the LKB. For example, for a text understanding task, we can build beforehand the CCKGs corresponding to one or multiple keywords from the text. Once built, the CCKGs will help us in our comprehension and disambiguation of the text. The graph operations (maximal common subgraph and maximal join) defined on conceptual graphs, play an important role in our integration process toward a final CCKG. Finding semantically significant words limits the search for the expansion of the cluster, and putting a threshold on the graph matching process limits the expansion itself, focusing on the addition of words who share some information with the CCKG. Clustering is often seen as a statistical operation that puts together words "somehow" related. Here, we give a meaning to their clustering: we find and show the connections between concepts, and by doing so, we build more than a cluster of words - we build a knowledge graph where the concepts

interact with each other giving explicitly important information that will be useful for natural language processing tasks.

We showed the LKB as a dynamic entity, that can and should be modified constantly through the acquisition of more information. Humans are constantly adapting their models of the world, often using NL to achieve this goal. To mimic this process, the LKB should be dynamic and easily modified through the processing of additional NL input. In our project, the dictionary is the starting point for building the LKB. The same method used to process the dictionary definitions can be used to process new text, assuming this text is written for children or it is made of sentences containing a simple vocabulary and written in a simple way. Further research has to be done to establish a reasonable updating mechanism of the information by comparing and modifying the certainty levels assigned in the LKB.

Finally, in a larger perspective, our LKB could be used for NLP applications for children, or as the core of a larger LKB, on which new information coming from dictionaries and corpora could be added. The new information does not necessarily have to be more complex or at a different level of detail from what we have so far. In the introduction, we mentioned that the AHFD was a good source of "shallow lexical knowledge" as described by Dahlgren [53] to be an independent and sufficient layer of naive semantics useful for text analysis. As our LKB is built on the AHFD, it becomes a source of shallow lexical knowledge. From there, we can try to find sources of information of the same type we have now (small descriptions, words put in typical contexts) but for more words.

5.2 Major contributions

We present this section as five different contributions made over the course of this research. The first three are more general contributions that apply to the overall perspective or approach promoted during this research and the last two are more specific contributions that make this work unique and original.

1. Our use of Conceptual graphs as a unifying formalism
2. Graph similarity used for text processing

3. LKB construction as a catalyst for future research
4. Covert Categories
5. Clustering

5.2.1 Our use of Conceptual Graphs as a unifying formalism

By unifying formalism, we mean two things: (1) CGs can be used for most of the structures constructed during the development of the LKB, (2) our use of CGs allows the coexistence of ambiguous and non-ambiguous information within the LKB.

We start with natural language sentences and first perform tagging and parsing. We obtain parse trees which are not structures represented by CGs[1], and then we transform the parse trees into CGs. From then on, all the operations are performed on CGs and all the new information extracted by comparison and combination of known information is also stored as CGs.

Unique to this work is the idea of including prepositions within the relation taxonomy. This is an important factor in our ability to show, within the same representation formalism, information that is at different level of ambiguity. The CG representation of a definition which contains a preposition as a relation shows a surface semantic representation, that is close to the syntax of the sentence. As we have access to more information, the representation can be disambiguated to contain more specific relations, but we still use the same representation formalism. This is an important difference with the work of [133] in which they introduce the use of predicates as an intermediate structure between their parse tree and their Conceptual Graph representation. A flexible structure must be able to account for ambiguous concepts and relations. A concept with multiple senses can be part of a CG as an ambiguous concept. Each word has a concept type associated with it, even if that concept does not correspond to any graph representation. Each sense of that word also has a concept type associated with it, which does correspond to the graph representation of its definition sentences. All concept types of the different senses of a word are considered

[1] A parse tree is still a directed graph, and therefore could be represented by a conceptual graph. The words that are nodes and leaves of the parse tree become CG concepts, and the CG relations are reduced to a single possible relation "child".

subclasses of the concept type associated with the word. Therefore, putting in the CG the concept type associated with the word corresponds to putting a more general (therefore more ambiguous) concept that can hopefully be disambiguated later.

The CG formalism makes use of a type hierarchy and a relation hierarchy. Those two structures are essential for comparing concepts. We introduced the idea of sets of words (found from conjunction) that can be assigned a label and be put in the type hierarchy (see section 3.2.2). A set can simply be represented by a CG containing isolated concepts with no relations between them. We are also able to introduce covert categories, represent them by a structure described in the CG formalism, the λ-abstractions, and include them in the type hierarchy (section 3.2.7).

The algorithms described for finding common subgraphs are useful as we attempt structural and semantic disambiguation which is based on comparison of information contained in the LKB, as well as when we build our clusters. For the clustering process, we make use of the maximal join algorithm from the CG formalism. We again represent the resulting cluster within the CG formalism. Its corresponding CCKG is a large CG containing all the relations among the cluster words as well as their relation to other words in the specific context. The cluster itself is a smaller CG containing the set of words as isolated concepts.

We intend to do text analysis using the CG formalism as well. We can transform a sentence in a text into a CG, and then compare that CG to the information contained in our LKB, all stored as CGs.

5.2.2 Graph similarity for text processing

The importance of concept comparison is noted by Collins and Quillian:

> *We only want to point out here that it is quite basic to the understanding of language processing to find out how people decide whether two concepts can be identified in a particular case.* [45].

During this research, we use graph comparison for many different tasks. Graph comparison and graph subsumption are used everywhere. Comparing graphs is based

on comparing concepts. It is more elaborate as it demands comparing relations as well, and identifying identical structures, not just isolated identical or compatible concepts and relations. The way to find if a graph is compatible with another graph, is to attempt a projection of the first graph on the second graph, or of the second graph on the first graph. The way to find if two graphs share a common subgraph is to attempt the projection of the different subgraphs of the first graph onto the second graph and vice versa. Similarity or compatibility of information is the key to multiple ambiguity resolution.
Graph comparison is used in the following tasks:

Finding is-a relations: To build the type hierarchy, we must find **is-a** relations within graphs. We project a simple subgraph **[everything]->(is-a)->[everything]** onto all graphs in the LKB and update the type hierarchy from our findings.

Prepositional attachment: To decide between two possible graphs for the same sentence that differ on their prepositional attachment, we generate two subgraphs (one for each possible attachment) that we project on the definitions on the words involved in the attachment. If a projection is found for one subgraph and not the other, we choose the graph containing it.

Word sense disambiguation: To decide on the sense of a word, we try finding the largest common subgraph between the graph in which it is used and the defining graphs corresponding to its different senses.

Finding deeper semantic relations: A defining formula found in a sentence that corresponds to a particular semantic relation can be found at the graph level by projecting a particular subgraph corresponding to the formula onto all graphs in the LKB.

Covert categories: Finding covert categories consists of projecting simple subgraphs built around verbs onto all graph representations of the LKB.

Clustering: The whole clustering process is based on finding common subgraphs between the graph representation of different words and then joining more information around them.

5.2.3 LKB construction as a catalyst for future research

This research investigates a large problem: the construction of an LKB from a children's dictionary. In addressing such a large problem, it allows subproblems to be examined in context rather than in isolation. This dissertation presented solutions to some associated subproblems and established some grounds to resolve other subproblems. In addition, it allows previously proposed solutions to be integrated with other solutions, again in context.

The creation of an LKB by automatically extracting information from text demands that some part of the project be oriented toward text analysis. Yet, text analysis is in fact the problem for which the LKB is needed.

We attempted to break the circle here by starting with a simple text source that would be easier to analyze so we could build an LKB and then enrich that LKB later with other texts. Although we did start with a children's dictionary in which there is no notion of part-of-speech, we used a tagger and parser to analyze the sentences. This is one area that could be further expanded. There is much exciting research on language acquisition and automatic grammar learning [52].

We transformed our parse trees into conceptual graphs, and applied multiple structural disambiguation heuristics. At that point as well there is much on-going research on finding the right prepositional attachment or the right conjunction attachment. But, whatever method is used, we can always see it as a reduction of the number of graphs assigned to each sentence. The goal is to produce one graph per sentence, and more advanced techniques can be added to help that task.

Then we briefly touched on the idea of semantic disambiguation, looking at very interesting problems such as anaphora resolution and word sense disambiguation. Again there is a large open door to explore these avenues more. Anaphora resolution can further lead us into exploring discourse analysis, which will be useful if we attempt

to augment our LKB with information from free-form text and not from dictionary definitions.

Once we obtain a single graph per sentence, we are only starting our exploration into the construction of the LKB. We looked into the construction of the type hierarchy, explored the idea of covert categories and mentioned the importance of these structures for finding similarity between concepts. Semantic similarity is another fascinating aspect of language analysis. A more detailed exploration into semantic distance could only ease the construction of some internal structures to the LKB (such as the clusters), as well as ease the comparison of new text with the information within the LKB. This last issue is an important research avenue, how to use new texts to augment the LKB, and how the LKB is useful to analyze these texts. In the present thesis we consider only the initial LKB. Using this LKB for further text analysis will illustrate and increase the value of the LKB.

5.2.4 Covert categories

This research allowed us to put in perspective the importance given to the taxonomic link compared to other kinds of information given in a dictionary. Even if the taxonomic link is often considered to represent the most important relation between words, we saw many examples where that is not the case. This work gives us an insight into the importance of not only the taxonomic relation but also all the other relations often considered less important or ignored.

By trying to find covert categories, that is, concepts with no corresponding word, we in fact find groups of words that are related by the fact that they can act as the same case role for a particular verb. For example, two concepts might share the property of being possible objects of the verb **drink**. That category does not have a name (we could come up with "drinkable liquid" which in fact expresses in an implicit manner the underlying case role we are trying to get at and express explicitly). It is certainly a subclass of **liquid**, but it is more specific. The concepts **gasoline** or **oil** are subclasses of **liquid** but they are not drinkable.

The covert categories give more dimension to the similarity of words than looking

only at hyponyms. Within the conceptual graph formalism, we are able to give a label to each covert category, and define it through a λ-abstraction. This allows us to consider the new labels as concept types that are put inside the type hierarchy and on which we can use subsumption as we do on the other concept types created to correspond to words and word senses.

We believe an LKB must include the taxonomic relation, as well as many other ones represented by covert categories, to contain as much information as possible for a natural language processing system.

5.2.5 Concept Clustering

The clustering process, which was explained in detail in section 3.4, is concerned with an issue that we have talked about since the start of the thesis: the importance of not looking at words in isolation. We have mentioned the work of [45] on semantic memory and the work of [51] in which they emphasize the importance of looking at the surrounding context, specifically at the surrounding words to be able to find the correct meaning of a word within a particular sentence and paragraph (giving a larger context).

Most clustering techniques are based on statistical methods [43, 142, 31, 108]. Although our clustering technique does involve statistical measures to decide which words are semantically significant, the primary mechanism is not statistical. Our technique is also used on a machine readable dictionary and not on raw texts. We make use of the fact that the words used in a sentence are being defined elsewhere and we can go look up their own definitions.

Our technique makes used of forward and backward references in the dictionary to find definitions of words which share some common information. The results are not only groups of words, as is found by all statistical techniques, but as well a large set of relations that describe the relations between all the words in the cluster, and between them and other words in the near context. We do not only know that a set of words share something in common, but we also know *what* it is that they share. Each word that is part of the cluster is therefore put in a larger context, and we have

a more extended view of its relations to other words.

One computational linguistic area which makes good use of clustering is **information retrieval**. Information retrieval uses keywords instead of understanding the words in the request. It can be improved by adding terms to the query related to the original term by lexical relations like synonymy, taxonomy, set-membership, or part-whole. The cluster might give some extra information worth taking into account. If we are trying to find documents about **mailing** for example, we probably would also like documents that talk about **sending letters** even if the query only mentions the word **mail**. On the other hand we would not want documents about **sending flowers** to be retrieved. So only knowing that **send** can be a synonym of **mail** is not enough. The cluster gives us a few semantically significant words that we can look for. We can have some measure of how appropriate is a document based on the number of concepts it contains that overlap with the cluster. Just looking at the words might not be enough, and therefore we will need the structure given by the CCKG of the relations between concepts.

> *... the system can use clusters of relevant words to identify the desired senses of at least some of them, the lexicon can help filter unwanted references as well as help find wanted ones that might otherwise be missed.*
>
> *There are huge numbers of possible path definitions, only some of which will ever be useful. Which paths aid effective query expansion for information retrieval, for instance, is an open research topic.*
>
> *Without disambiguation, expanded indexing would create garbage indexes (expanded from wrong sense of words in the text), thus aggravating the problem of retrieving documents the user does not want.* [106]

We think one possible path for query expansion is through the semantically significant words (section 3.4.1) in a cluster and the relations that give them a sense in a larger structure.

The cluster can also be used to help the user specify one from the multiple senses of a particular keyword he/she gave to start a search. If the keyword is vague (non

semantically significant) or ambiguous (multiple senses) it might lead to an enormous number of retrieved documents. If that keyword is part of multiple clusters, the set of words within each cluster can be presented to the user to help him/her refine the search [42].

5.3 Specific problems

We address in this section a few specific problems encountered along the development of the ideas in the thesis.

Use of thresholds: It is always delicate to base calculations and heuristics on defined thresholds, as we have to be careful of the way each threshold is either calculated or decided. There are three thresholds we use.

1. **SSW Threshold:** One important threshold is the one that differentiates between semantically significant words and non-significant ones. We calculated that threshold based on the small corpus of the American Heritage First Dictionary. As our application stays within the use of the AHFD the statistics are adequate, but if we envisage adding new information in from text corpora, we should revise our statistics. We also calculate these statistics with respect to words as entities and not word senses. This will affect our clustering procedure. A word might contain two senses, one which is frequently used and the other not. Overall that word is not a SSW. However, in a cluster that contains the second sense, the word should be considered a SSW and we should explore its definition for expansion. In AHFD, only the few most common used word senses are presented, in which case, the problem is less noticeable.

   We saw during section 4.5 how the threshold on SSW influences the cluster built. A threshold adequate for one trigger word is not necessarily adequate for another trigger word (we gave an example with post-office and farmer). We believe an adaptive threshold, as mentioned in section 4.6, would be the solution to this problem, as it would keep the search space constant.

2. **Covert Threshold:** Another threshold is used for deciding if a covert category becomes part of the system and is assigned a λ-abstraction and put in the type hierarchy. Again, our corpus is small, and it becomes difficult to decide if four occurrences is enough compared to three, or whether a pattern really corresponds to a covert category, or just happened to be repeated a few more times than usual in that particular corpus. We must consider as well that there is error included in the results as we do not have a single graph per sentence. If the corpus were larger, covert categories that need to be put in the type hierarchy would be discovered by finding a much larger number of occurrences.
3. **Graph Matching Threshold:** The third threshold, the graph matching threshold, is used for deciding whether we join a new graph to the cluster graph during the clustering process. This threshold causes less problem as it would not vary with the size of the corpus. It is representative of how much information we think should be shared between two structures before we decide to join them. But still, it is a decision based on empirical testing and observations, and therefore subject to criticism.

Separation of stages parse/parse-to-CG/reduction : It is non-optimal to build all the parse trees and then transform all parse trees into Conceptual Graphs, and then apply some structural disambiguation techniques to reduce the number of graphs. These three stages should be more intertwined, so we can reduce the number of parses as we go.

Limitations of CoGITo: The CG platform that we use has some important limitations which would render it less adequate if our system ARC-Concept were to become much larger. One such limitation is that the type hierarchy is implemented as a large matrix. If there are N concept types in the type hierarchy, CoGITo needs a matrix of size N^2 in which it stores the most immediate superclass for each combination pair of concept types. This saves in time when the information is needed, but it adds in time at the beginning of a session to calculate all this information, and mostly it adds in memory demand. At this

point, we have less than 3000 concept types, but already a matrix of 9 million integers.

The second limitation of CoGITo is that it does not allow nested context. So, we can not represent, for example, a sentence like *John said: "I like skiing"* which should look like:

```
[say]->(agent)->[John *x]
    ->(what)->[Proposition: [like]->(agent)->[I *x]
                                ->(object)->[skiing]]
```

During our implementation, we have overlooked this limitation by interpreting the CGs with an ordering to the relations used. In this last example, if we decide that the relation **what** has less priority than all other relations, the following graph,

```
[say]->(agent)->[John *x]
    ->(what)->[like]->(agent)->[I *x]
                    ->(object)->[skiing]
```

without the **proposition** concept being itself defined by a CG, would be equivalent to the first graph. The relations **agent** and **object** have priority over **what** and therefore we regroup *I like skiing* into a subgraph before joining it to the rest by the relation **what**. It is not standard practice in the CG formalism to give an ordering to the relations. CGs usually allow for nested contexts, in which one concept type, such as proposition or a situation, can be itself defined by another CG.

5.4 Future research

As we said earlier, this research has a broad coverage, and it forms a good base for further explorations. We suggest here a few directions for future research, but there are certainly even more. We give an overview of some suggestions, and then reserve a

sub-section to future research in text analysis as it would be the next immediate step for testing and augmenting our LKB.

Salience of relations: We compare graphs for multiple purposes, and we base some decisions (such as whether or not to include a new graph in a cluster) based on the common subgraphs found between two graphs. The number of SSWs contained in the common subgraphs and the size of the maximal common subgraph (hopefully a structure with one or more relations and attached concepts) are used to establish the similarity between two graphs. One problem not addressed in this research and noted in Delugach [54] is that not all semantic features found via our semantic relations are of equal importance. For example the fact that two concepts share the same color might be less important then the fact that two concepts share the same instrument for performing an action. A further exploration into semantic relations to establish some kind of relative importance among them should be part of the future research agenda.

Finding similarities between word senses: Each word in the dictionary has multiple senses, and each sense contains multiple sentences. One major component of my research proposal presented two years ago [15] was the integration of the multiple senses of a word into a larger broader sense, or at least trying to find subgraphs present in multiple senses to find connections.

The problem of dictionary definitions is well expressed by Atkins:

> *The word meaning is often divided into discrete senses (and sometimes subsenses), which are then analyzed and recorded as though they had a life of their own, with little to link them except the coincidence of their being expressed by the same string of characters pronounced in the same way.* [13]

As mentioned in our introduction (see section 1), our objective is a non-localist representation of the lexicon. This non-locality should first be present at the word level, by relating the multiple word senses of a word. Polysemy is handled

quite arbitrarily by lexicographers giving multiple senses according to more or less specific usage or as being part of different contexts. The idea of regular polysemy [12], lexical rules [47] or Lexical Implication Rules (LIRs) [107] sees relations between word senses that are more than accidental. The relations are present in multiple words and can be extracted and formalized.

This regular polysemy could hopefully be expressed with conceptual graphs, using λ-abstraction that can be seen as transformation graphs. For example, one LIR says that a count noun of class **vehicle** can become an intransitive verb, meaning traveling using the noun as an instrument.

RULE 1: Type Vehicle(λ) is
[Travel: λ]->(instrument)->[Vehicle]

As these rules assume that we can transform one part of speech into another, they must be accessible at the parse level.

Regular polysemy is an interesting subject by itself. We continue hereafter on the idea that some senses could be related in an arbitrary manner but that we could still find connections between them.

Let us look at the 3 senses of *door* from the American Heritage Dictionary where it seems obvious that some connections should be established between word senses. We could be talking about the entranceway, or the panel blocking it, but those two things are related. By establishing links between senses, we hope to reflect the meaning of a word in a more coherent manner.

- **Door(1):** A movable panel used to open or close an entranceway.
- **Door(2):** An entranceway to a room, building, or passage.
- **Door(3):** A means of approach or access.

The research proposal suggested to establish links between word senses, as again the non-localist approach should start at that stage. It is after the proposal that we decided to work with the American Heritage First Dictionary (AHFD). In

the AHFD, most words have a unique sense, and when there are multiple senses they usually have very different meanings. Still, sometimes there is a relation between the verb and noun versions of the same word, and we could try finding relations at that level.

In support of our idea, the work of Dolan [56] aims at finding similarity between word senses. His work is part of the large Microsoft effort at building an LKB from the LDOCE dictionary and using this LKB to help in text analysis. Dolan suggests that the idea of assigning a particular word sense to a word found in a text might not always be appropriate. It might become too specific, and the choice is often arbitrary as some senses are so closely related that either sense would be fine in the text or, better, a more general sense that includes these two senses. To establish similarity between word senses he makes use of the semantic and domain codes given in the LDOCE. But (in a spirit more similar to ours) he looks into the semantic relations in which both word senses are involved to compare their values.

The ideas presented in this dissertation are close in spirit to the ones in the proposal, as instead of joining word senses, we decided to join different words with our clustering process. So even if our clustering is on multiple words, the technique of graph matching and graph joining can be tried to see what it does on the joining of multiple senses of one word. In the large example we presented in section 4.5.1, we built a cluster around the trigger word **post-office**, in which are present two senses of **mail**. One side effect of clustering is to bring together related word senses and show the relation between them.

Parsing: There are two options (quite opposite in spirit) for future research related to the parsing done in this research. One is to parse better, and the other one is not to parse at all.

1. The ARC-Concept system makes use of its own parse, which allows the exploration of many aspects of the NLP, understanding the many types of ambiguity. The grammar and chart parser developed for ARC-Concept are

certainly adequate for analyzing the simple sentences in the AHFD, and it would certainly be adequate for analyzing texts coming from children's story. Its potential is not certain for adult's texts. There are many commercial and non-commercial parsers available, and we could certainly look into those to replace the parsing module. Our parser and grammar are quite robust, finding a parse for 92% of all sentences present in the AHFD.

2. The AHFD was augmented in the electronic version with parts of speech information. Exploring grammar learning or children's language acquisition was outside the scope of this dissertation, so we had no choice other than to tag and parse our sentences to obtain the graph representation needed to go on with our explorations into covert categories, clustering, etc. But as future research, it would be fascinating to try to generate the conceptual graph representations without doing this parsing process, or by learning how to parse by starting with two-word sentences, then three-word sentences and so on. There is certainly much interesting research to do in lexical and grammar acquisition in children of different ages [140].

Text Segmentation: A text in more than a sequence of words. Each sentence is related or not to the preceding and following one. Identifying the structure of the text consists in finding smaller units showing a certain lexical cohesion. In [86], they talk about a process called **lexical cohesion profile** which locates segment boundaries in a text. They define lexical cohesiveness as word similarity computed by spreading activation on a semantic network [87]. Their network is made up of words definitions extracted from the LDOCE.

We believe that our clustering process base on definitions, could be triggered dynamically by semantically significant words found in the sentences of a text. When there is cohesion between a sequence of words, then we could find common subgraphs between the definitions of these words. When a word cannot be added to the present cluster, it can become the starting point of a second cluster. When a new word comes along we try clustering it to the first and second cluster, if it does associate with the second cluster, we are probably at a boundary.

Language generation: Some work has been done on generating sentences from a CG representation [104]. As part of the KALIPSOS project mentioned earlier, Nogier and Zock [105] developed a generation module. It would be very interesting to express in a paragraph in natural language the result of our clustering process. Starting with multiple sentences from multiple definitions, we could bring all the information together into a coherent, non-redundant unit. Expressing the information with natural language sentences would be like giving a summary of our discovery.

We could investigate whether some of our analysis processes are reversible and can be used as a generation process. For example, we have multiple rules for parse-to-CG transformations which could probably be used as cg-to-parse rules as well.

Second language acquisition: When you look at books for second language acquisition, one method to make a person learn a language is to put him/her into a particular situation and learn the elements, objects, actions, typical questions and answers that are related to that situation. For example, in the Berlitz Essential German [1], some of the chapters are: meeting a new person and presenting yourself, going on a vacation trip, being at the office, being at the hotel, talking about the weather, having breakfast, going on a picnic, talking about the family, shopping in a large department store.

Similarly, Schoelles and Hamburger talk about the role of NLP in CALL (Computer-Assisted Language Learning) and mention the idea of "microworlds" as a hierarchy of objects with their description, properties, associated plans and actions [123].

This is reminiscent of Schank's scripts [121], and for us this seems like a good application for our clustering. We can start from a trigger word representative of the subject matter for a particular lesson, such as **breakfast**, **store**, **picnic**, **family**, and from there we create a cluster around that word showing all the related concepts possibly present in that situation and all the relations that link all these concepts together.

To interact with the learner, we need to be able to transform his/her questions/remarks into conceptual graphs and compare these graphs with the information we have stored in the appropriate cluster. We would need as well a Natural Language (NL) generator that can start from a CG description and explain in NL to the learner what is present in the cluster, what the important elements and relations are. As well, the rules the NL generator uses to go from a CG description to a NL sentence could be explained to the learner as a mechanism for him/her to generate correct sentences from mental representations of the situation.

Machine translation: The clustering would be interesting to explore in a multilingual environment. Let us assume we work with two languages, French and English. We could perform clustering starting from a particular concept for which we know the word in both languages (for example we start with **post-office** and **bureau de poste**), and then compare the resulting clusters. As there is a lot of information present in both clusters, we have more chances of finding corresponding subgraphs (which would show compatible relations between concepts) and connections between words.

It is usually hard to find matching on single definitions as they might emphasize different aspects and therefore often no connections are found between the definitions in the two languages. The more information we gather the more chances we have to find overlapping information. As well, the clustering process on both sides will have disambiguated some words having multiple senses, and the French-English correspondence between two words might be reduced to finding a correspondence between word senses.

Another interesting aspect of matching cluster graphs from two languages is that it might help disambiguate some relations. On each side (example the English and the French side) there is a relation taxonomy that shows the superclass/subclass relations between deeper semantic relations and more general prepositions. For example we have the two following corresponding subgraphs:

[send]->(in)->[mail]
[envoyer]->(par)->[poste]

assuming we were able to find the correspondence between **mail** and **poste** and between **send** and **envoyer**. The French preposition **par** is a supertype for more than one deeper semantic relations:

agent:	j'ai appris la nouvelle **par** Jacques
	I heard (learned about) the news from Jacques
path:	je regarde **par** la fenêtre
	I look out the window
manner	elle est venue **par** avion
	she came by plane

The English preposition **in** is a supertype for four semantic relations:

point-in-time:	Ted's birthday is **in** August
location:	fish swim **in** the water
manner:	ants live **in** large group
part-of:	the branch **in** the tree

In this particular case, we have only one possibility of convergence of the two prepositions, it is on the semantic relation **manner**. We therefore disambiguate the relation in both languages. In other cases, for example if it had been the English sentence **send through the mail**, we could still have reduced the French ambiguity, but from four to two possible semantic relations, here **manner** and **path**.

Investigation into Lexical Gaps: Each language is a reflection of a whole culture behind it, it has a history, it evolved over time, sometimes into strange or inexplicable constructions. When we look at pairs of languages we find many *lexical gaps* [89], that is words in a source language that don't have any equivalent in a target language. In relation to CGs, we show part of an example with the taxonomy of vehicles in English and Chinese given in [130].

vehicle —> bicycle (zixingche)
—> qiche —> car
—> taxi (chuzuqiche)
—> bus (gonggongqiche)
—> truck (kache)
—> train (huoche)

Our investigation into covert categories might be helpful for looking at lexical gaps. In fact a covert category is a concept in one language that does not have a corresponding word. We saw for example, that the concepts of a "writing instrument" or a "drinkable object" can be present in our mind without a single word to correspond to it. These covert categories as we saw often correspond to case roles to particular verbs. When we try to compare two type hierarchies coming from different languages, maybe the covert categories in one language will actually be known words in the other language.

This observation certainly deserves more investigation for different languages. We might look into other types of covert categories that could be related to nouns and not just centered around verbs.

5.4.1 Using the LKB for text processing

The LKB is built for one main objective: helping Natural Language Processing (NLP) systems for their task of text understanding. NLP systems need a great deal of information about words. By doing a detailed analysis on word definitions and a large part of the disambiguation process as we are building the Lexical Knowledge Base (LKB) from a Machine Readable Dictionary, we aim toward a more coherent view of a lexicon containing as much information as possible and in the most explicit form. We want to give a larger amount of information surrounding each word, putting it in context. For applications like machine translation, information retrieval or question-answering, the LKB aims to ease the tasks of disambiguation, finding related concepts or finding implicit information.

In its present state, the LKB can be used to help us analyze more dictionary definitions, or other text input. In return, the new inputs will help us modify and update the LKB. It is a constant loop of refining the LKB to be more accurate, perform better text analysis and then refine the LKB even more [19].

We can think of the constructed LKB as a core that contains general and basic information about day to day life. As mentioned in the introduction chapter, the AHFD is the second dictionary in a series of four, meant to contain information for children of different ages. The AHFD contains around 2000 words, and the next one in the series, the American Heritage Children's Dictionary contains about 30000 words. It is aimed at children of age 8-11. This is when in primary school children start learning about more specialized fields, they go into more details about history, science, geography, etc. It would be interesting to expand our LKB by learning about these different fields from text books. As well as text books on particular subjects, it would be very interesting to take children's stories and try to update the information stored in our LKB.

The expansion of the LKB means the addition of more concepts and more links, and one concern is to what extent the processes presented during this research are able to scale up, if instead of 2000 words, we now have 40000 words. The sentence to graph transformation will need some adaptation. It works on individual sentences, therefore the number of sentences to process is not an issue. On the other hand, if the texts used to acquire information come from different sources, the parser will need to be adapted. The parse-to-CG module will also need some adaptation as it is based on the discovery within the AHFD of frequently used patterns that lead to deeper semantic relations. Once the CGs are obtained, the rest of the process will be identical. The prepositional attachment disambiguation module makes use of the information in the LKB but it only looks at the words related to the preposition, and therefore having more words in the LKB will not be an issue.

The finding of covert categories is the process that will be influenced the most by the size of the LKB. To establish a covert category, we must iterate through all the verbs present, build subgraphs around those verbs and then do multiple projections on all the graph definitions within the LKB to find if the subgraphs (that will possibly

become covert categories) are present a sufficient number of times to be considered covert categories. The threshold will need to be adapted and the process will take much longer. The main operation is the projection operation which is known to be NP-complete [44]. Cogis and Guinaldo [44] list multiple methods proposed by different researchers to restrict the problem in such a way that the complexity is reduced to be polynomial and therefore not create combinatory explosion.

The clustering process will not really be influenced by the size of the LKB. Increasing from 2000 to 40000 words will not have a dramatic effect on the number of words used in a single definition. The notion of semantically significant words (which will scale up as it is a statistical measure on a larger corpus) will still constrain and direct the search for words to be added just as it does in the AHFD.

One concern is to use the LKB for text analysis. If we perform sentence disambiguation by accessing information in the LKB and comparing it to the information in the sentence to be analyzed, we will need a good indexing method on the words stored in the LKB, so we can access the definition of a particular word quickly and also access the clusters in which it is present in a quick manner.

The use of time and storage is one technical concern, but another more fundamental concern is knowledge consistency. With respect to the update mechanism, research should be oriented toward ways of establishing consistency, or non-conflict of information (as briefly introduced using the certainty levels). The LKB only perpetuates a general view of things, but if we start analyzing texts we will have to enter into the field of knowledge belief, and individual representation of the world as similar or different from what is in the LKB.

The following subsections show how the LKB might be updated through the processing of textual input (in addition to the AHFD) and how the LKB can help in the processing of additional input.

Modifying the LKB

As we said earlier, the LKB is built dynamically and should continue to evolve as we process new text. Two main parts of the LKB, the concept hierarchy and the graph

definitions, are subject to modification by new incoming information. To continue updating our LKB, after our definitions have all been analyzed, we will look for simple texts, such as children's stories. We assume we are processing simplified text, like that written for children that only includes concepts from the dictionary, or new concepts that we can link to the existing ones via the concept hierarchy or definitions. The sentences should be short and simple like the examples found in the dictionary. So, the process of updating the LKB from text is essentially the same as the one that updates the LKB as dictionary definitions are processed.

Concept hierarchy When analyzing a text that contains concepts not already in the knowledge base, the certainty levels established on the relations can help us find a place for a new concept in the concept hierarchy. For example, if we have the fact **[eat]->(instrument:expected)->[utensil]** in the knowledge base, and the graph **[John]<-(agent)<-[eat]->(instrument)->[chopsticks]** is extracted from a text, we can infer the following taxonomic link: **[chopstick]->(is-a)->[utensil]** and place **chopstick** as a subclass of **utensil** in the type hierarchy. For much research on lexical acquisition, see [146].

In general, it is important to make a hierarchy as rich as possible to aid in word sense disambiguation and anaphora resolution when analyzing text. To further augment the concept hierarchy from the analysis of texts, we can use the method of pattern finding presented in [75]. Patterns such as *<NP {,NP} *{,} or other NP>* (e.g. apples, pears or other fruits), or *< NP{,} especially {NP,} * {or/and} NP>* (e.g. fruits, especially apples and pears) can be used to determine the classes of words.

Definitions The certainty levels that we introduced within the CG representation will play an important role in the changes to occur in the LKB. As we said earlier, the examples can be seen more or less as the different experiences that someone goes through day after day. We keep the general pattern that encompasses all the individual examples, and when we do find one particular example that expresses a different view, it is kept as a contradictory example.

In the example from section 3.1.2 we had the verb **bake** where the various concepts

that could fill the **object** relation were all of class **food**. Therefore we assigned a selectional restriction on the object of **bake**, expressed by an **object** relation at the certainty level *expected*:

[bake]->(object:expected)->[food]

As we process new texts, if we then encounter an example saying: *Susan baked her clay animals in the oven* which does not follow the norm, we include it in the LKB as a specific example:

[Susan]<-(agent)<-[bake]->(object)->[animals]->(made-of)->[clay]

We can establish a threshold for the number of examples to be encountered before we actually move the information from the example part to the usage part of the LKB. This means deleting multiple examples, and adding a generic information with a selectional restriction expressed as a *possible* or *expected* relation. For example, if we see more examples, in which we bake clay pots, then clay figurines, we might update to:

[bake]->(object:expected)->[food]
->(object:possible)->[object]->(made-of)->[clay]

and then we can discard the examples we were keeping.

Analyzing text

> *From the perspective of natural language processing, the value of any lexical knowledge base is ultimately to be judged by the degree of support it offers to tasks like syntactic analysis and semantic interpretation.* [25]

An important case of syntactic ambiguity that can make use of semantic information to resolve the ambiguity is prepositional attachment. We presented in section 2.3.1 how to eliminate structural ambiguity trying to find the correct prepositional attachment by looking into the graph definitions in the LKB. We can use the exact same method for analyzing new texts and achieve with our system a similar disambiguation process as described in [83].

We see a typical prepositional attachment problem in example 5.4.1.

Example 5.4.1 - I eat my fish with a fork.

Graph1:
[eat]->(object)->[fish]->(with)->[fork]
Graph2:
[eat]->(object)->[fish]
->(with)->[fork]

We will look into the graph definitions of **fish** and **fork** to find the subgraph **[fish]->(with)->[fork]** and we will look into the graph definitions of **eat** and **fork** to find the subgraph **[eat]->(with)->[fork]**. We can choose the best structural interpretation depending on the similarity between our graph definitions.

Finding similarity with graph definitions can also help in the case of anaphora resolution [88].

One interesting research based on an LKB of Conceptual Graphs is the project KALIPSOS described in [61] which performed some natural language understanding tasks using CGs. They differentiate between two aspects of a word, its canonical graph and its meaning by use. Their LKB consists of a list of words; each word is associated to a canonical graph, which gives the "basic" meaning of the word, and a set of schemata called *schematic cluster* which represents the meaning by use. For example the entry for the word **key** and **open** might look as follows (from [61]):

'key'
concept type: KEY < PHYS_OBJECT
schematic cluster:
(s1) [KEY]<(INST)<-[OPEN]->(OBJ)->[DOOR]
(s2) [KEY]<-(PART)<[KEYBOARD]
<-(OBJ)<-[PRESS]->(AGT)->[PERSON]

'open'
concept type: OPEN < ACT
canonical graphs:
(c1) [PERSON]<-(AGT)<-[OPEN]->(OBJ)->[PHYS_OBJECT]

(c2) [PERSON]<-(AGT)<-[BEGIN]->(OBJ)->[COMMUNICATION_PROCESS]

We intentionally do not distinguish between the canonical graph and the meaning by use graph in our system as we think such a division is arbitrary and does not help further processing. But at this point we want to look at their text analysis using the LKB and not the structure of their LKB.

Their analysis of a sentence is based on the compositionality principle. The semantic representation of an entire sentence can be built by combining the semantic representations associated with its components. The meaning of a sentence is therefore the maximal join of the conceptual graphs associated with the words in the sentence. Those words correspond to the leaves of the parse tree resulting from a syntactic analysis. For a word, the conceptual graph chosen could be its canonical graph or some schemata of its schematic cluster.

In their paper, it does not seem obvious why they would chose to look at the graphs of certain words and not others in the parse tree. They use, as an example, the sentence **John opened the session by pressing the enter key.** They look at two possible schemata of **key** and two possible canonical graphs of **open** to find the right interpretation of the sentence, where the right interpretation contains the appropriate sense of each ambiguous word.

Looking into the graph definitions (for them canonical graphs and schemata) to disambiguate a new sentence seems the right thing to do. But they do not mention how they decide to look into **key** and **open**, and not into **session**, **pressing** or **enter**. One way to reduce the search would be to limit the chosen words to be nouns or verbs and to be Semantically Significant Words (SSWs). In which case, for their example, we would limit the search to **open**, **session**, **press** and **key**. Some of these words might not be SSWs. The prior syntactic analysis is useful to eliminate the word **open** as it is not used as a verb but rather as a modifier to the noun **key**.

Our clustering process allows us to perform in advance multiple maximal join and create larger graphs representing different situations. When we analyze a new sentence, we look at the SSWs in the sentence and form a test cluster that we can

intersect with the clusters built within our LKB. If there is such an intersection (containing a certain minimal number of words), then the cluster graph will be our guide to interpret the sentence, maybe disambiguate some words, help in anaphora resolution, etc. On the other hand, if such a cluster does not exist, we can use the same method as presented in [61] and build a larger graph from the individual definition graphs of the words in the sentence. Here, we would use the same idea of only looking at SSWs in the sentence. If there are NbWords in the sentence that are SSWs, and each word W_i possesses $NbSenses(W_i)$, there will be M possible maximal joins.

$$M = \prod_{i=1}^{NbWords} NbSenses(W_i)$$

The largest of all these maximal joins will likely contain the right sense for all words in the sentence that could be assigned multiple senses. One interesting aspect of this approach to word sense disambiguation is that it addresses the ambiguities of all the words in the sentence at the same time. More often we see access to dictionary definitions to try to disambiguate a single word in a sentence [90, 71]. An immediate context is established around a word and that word is disambiguated by looking up the definitions of the words in its surrounding context to find overlapping information. For simultaneous disambiguation using dictionary definitions, Veronis and Ide [135] propose a method based on spreading activation in neural networks. As well, Cowie *et al.* [49] propose a method based on simulated annealing which is a technique we see used in the associative memory types of neural networks. None of these methods looks into the relations between words, they only look into sets of overlapping (or activated) words extracted from each definition.

5.5 Conclusion

We conclude and summarize the ideas presented in this dissertation by answering some interesting and very pertinent questions raised in [22]

1. **What can be learned from texts and from dictionaries?**

A lot. We believe that much information can be learned from texts and dictionaries, and that the manual effort of encoding information for a system should be kept to a minimum. We did not look at acquisition from text in this thesis, but we looked into much detail at the acquisition and encoding of information from dictionaries. More particularly we looked at a children's dictionary, the American Heritage First Dictionary (AHFD). The AHFD has been our guide and it certainly made us learn a lot about the world, about day to day objects, actions and situations. Through a process of sentence analysis, and transformation into conceptual graphs we were able to encode all the information from the AHFD into a Lexical Knowledge Base (LKB). Not only did we learn about the definitions of each word, but we were able to relate all these words together into a large interconnected network of information. We were also able to extract a type hierarchy from the definitions of the AHFD. The one thing we did manually was to build a relation hierarchy. Both the concept and relation hierarchy are used intensively for comparing information, finding similarities between graph representations. We were also able to construct clusters, to rearrange the information into larger structures to make us understand the meaning of groups of words within a larger context.

2. **Are domain-specific corpora better than general purpose texts and dictionaries for the purpose of extracting data useful to automatic language processing?**

   We would say that domain-specific corpora have their place in the construction of an LKB, but not as a first source of information. What would be very fascinating is to build an LKB from nothing, as a child starts from nothing. He/she acquires knowledge from day to day experiences, from being told about objects and actions by people around. We do not have the luxury of spending years on knowledge acquisition, but studying the process would be interesting. With this thesis, we feel like we have started at least closer to the core by acquiring knowledge from a children's first dictionary than if we had acquired information on a specific subject via corpora of very specialized texts. The

AHFD is a very good source of information about daily situations and it can be used a the core of an LKB that can later grow via the analysis of other more advanced general purpose texts and dictionaries and eventually via domain-specific corpora.

3. **What is the relative contribution of statistical and AI-based techniques (syntax and semantics) in the analysis of textual databases?**

   We could not say the exact contribution of each but we did, during our research, touch both approaches. We in fact used more AI-based techniques, that we called heuristics, to solve multiple problems that came along. In fact, the most general technique used throughout the thesis is graph matching. It is good and satisfying to reduce all problems to be solved by a singular approach. We did use statistical techniques, although we know that the size of our corpora is much too small to consider the numbers found to be reliable. One very important statistical measure is to establish whether a word is semantically significant or not. That measure is used throughout the thesis as part of our graph comparison process.

   We could in fact say that most heuristics in our thesis are based on two general principle, one AI-Based, and one statistically based. From AI-Based Conceptual Graph formalism, we constantly use the graph matching algorithm to compare graphs to find similarities. From statistical techniques, we use a simple calculation of number of occurrences of a word within a corpora to establish its semantic significance. The semantic significance is then used within the graph matching process so that matches on significant words is more significant then matching on non-significant words. The graph matching based on SSWs is used to solve some NL problems such as prepositional attachment and word sense disambiguation.

4. **How much of the process of acquiring lexical knowledge can actually be automated?**

Much of it can be automated, and it should. There is no sense in doing manual work that can be done automatically, especially with the tremendous amount of information that is present in electronic form these days.

For this thesis, we do have structures that were entered manually, as well as all the heuristics were decided and manually encoded, such as the morphological rules and tagging process, the parse rules and chart parser, the parse-to-CG transformation rules, the multiple structural disambiguation heuristics, the semantic disambiguation heuristics, the process to find covert categories, and the process to build clusters.

It is mostly processes and not data that were entered manually, and therefore, once they are in place, they can be used on all different data. Certainly some investigations into machine learning [52] might show some processes as learnable from data, such as morphological rules, part of speech tagging, parsing rules, or parse-to-CG transformation rules. But that assumes large set of sentences tagged, parsed, transformed into cg so that we can learn the process from them. It is probably feasible, but was not the goal of this thesis. The process of discovering covert categories requires minimal code to find much information within the data source that we have. It could be used on any data source transformed into CGs. The clustering process can be applied on any starting word and will automatically generate a cluster around it, so again minimal process encoding that can be used on large data sets.

Finally, it is good to note that many of all the processes described are based on a more fundamental process, graph matching. Throughout the thesis, we emphasized the importance of graph comparison, the importance of finding similarity between concepts. We based all our heuristics, and our explorations into different avenues on that simple idea, and it lead us quite far.

Appendix A

Electronic version of AHFD

Input: American Heritage First Dictionary (AHFD)

Description: Contains 1800 entries and pictures for English vocabulary addressed to children from age 6 to 8.

Format: .

Word_X
Defining sentence 1.
Defining sentence 2.

Word_Y
1.Defining sentence 1.
Defining sentence 2.
2.Defining sentence 1. Defining sentence 2.

Process: We produced an electronic version of the American Heritage First Dictionary (FD)[1]. We add information about the parts of speech which are not present in the AHFD.

- The chosen parts of speech (tags) associated with the words are:

[1]Copyright ©1994 by Houghton Mifflin Company. Reproduced by permission from THE AMERICAN HERITAGE FIRST DICTIONARY.

  - abb: abbreviation
  - adj: adjective
  - adj_num: numerical adjective
  - adj_c: comparative adjective
  - adj_s: superlative adjective
  - adv: adverb
  - aux: auxiliary verb
  - conj: conjunction
  - det: determinant
  - m: verb (mean) for definitions
  - n : noun
  - prep: preposition
  - pron: pronoun
  - pron_pos: possessive pronoun
  - rel_pron: relative pronoun
  - than: word than always used in special conditions
  - v : verb transitive or intransitive
  - vi : verb intransitive only
  - vb : verb (be)
  - quest: words used for questions (how, what)

- More information is added when known:
  - to verb entries (v,vi,vb): tense, person and number
  - pronoun entries: person, number, gender and position occupied in the sentence (subject, object)
  - noun entries and determinants: number (only if plural for nouns)
- Multiple senses: a word can have multiple senses, each sense defined via a few sentences. We give a tag to each sense, and an integer represents the word sense.

- Multiple tags: Sometimes a word sense is defined in the AHFD using examples that suggest multiple parts of speech. For example, the word "after" is defined as:

  After means following.
  After the ball game, we went home.
  Steve ran fast when his brother came after him.

  In this case, we would add two tags, one for conjunction and one for preposition and separate them with a "/" in the file. Each tag gets an integer as if it was a different word sense even if they are defined using the same set of sentences.

Output: File **dict.dat**.

Format: .

Word_W
1.conj./2.adv.
Defining sentence 1.
Defining sentence 2.

Word_X
1.vb.tense.person.number.
Defining sentence 1.
Defining sentence 2.

Word_Y
1.pron.number.person.gender.position.
Defining sentence 1.
Defining sentence 2.

Word_Z
1.n.
Defining sentence 1.
2.n.

Defining sentence 1.

Defining sentence 2.

Appendix B

Irregular words and morphological rules

Input files or data:

- Sentence to tag
- **dict.tags** : file of possible words and their tags, it is created automatically from the dictionary file dict.dat
- **Verb_list.dat, plurals.dat, abb.dat** : files containing respectively:
  - a list of irregular verbs

    beat/beat/beaten/
    become/became/become/
    begin/began/begun/
    bend/bent/bent/
    bite/bit/bitten/
    blow/blew/blown/
    break/broke/broken/

  - a list of irregular plurals

child/children/
foot/feet/
goose/geese/
mouse/mice/
person/people/
tooth/teeth/

- a list of abbreviations.

aren't/are not/
can't/can not/
cannot/can not/
couldn't/could not/
didn't/did not/
doesn't/does not/
don't/do not/
Dr./doctor/
hadn't/had not/
haven't/have not/
I'll/I will/

- **people_names.dat** : as the tagging process goes, when an unknown word is found, the tagger stops and asks the user to enter the base word and its tag, if the tag happens to be "pn" for a person's name, we add the name into this file

Lisa pn f
David pn m
Tom pn m
Ann pn f
Steve pn m
Timmy pn m
Kathy pn f
John pn m
Sarah pn f
Kate pn f
Beth pn f

Marcel pn m

- **symbols.dat**: created by hand is a file containing a list of symbols, it contains individual letters "a-z" (tag: let_min), "A-Z" (tag: let_maj), "0-9" (tag: n), "+=<>" (tag: s), "." (tag: ep (end phrase)), "," (tag: pause), "?" (tag: eq (end question)), """ (tag: dquote,equote)

A let_maj /
...
Z let_maj /
a let_min /
...
z let_min /
0 n /
...
9 n /
+ s /
- s /
= s /
s /
/ s /
> s /
< s /
. ep /
, pause /
: p /
; p /
! ep /
? eq /
@ o /
` dquote /
" equote /

- **mots_composes.dat**: this file is created manually, although in a large dictionary it should be done automatically, and it contains a list contains the entries from AHFD that are compounds. If a word is added by the user (was not defined in dictionary but used in a definition) and it is a compound word it will be added to the list.

  each other
  fire engine
  hot dog
  ice cream
  peanut butter
  post office
  roller skate
  string bean
  teddy bear
  bus stop
  so much
  out of

Morphological rules:

- Noun plural ending in "men", singular "man".
- Words ending in "s", "es": noun plural, or verb in third person singular
- Words ending in "ies": noun plural or verb 3.p.s. ending in "y"
- Noun plural ending in "ves" from singular ending in "f"
- Past tense verb, ending in "ed"
- Continuing present tense verb, ending in "ing"
- Comparative or superlative adjective ending in "er" and "est"
- Adverb made from adjective ending in "ly"

- Noun made from verb ending in "er"
- Adjective made from noun ending in "al"

Appendix C

Parse rules

RULE.DIC

```
s2 s1 ep                /* full-sentence(s2) -> sentence(s1) + "."(ep) */
s2 sq eq                /* full-sentence(s2) -> question-sentence(sq) + "?"(eq) */
sq do s1                /* question-sentence(sq) -> "do" + sentence(s1) */
s1 s1 cs                /* sentence(s1) -> conjunction of multiple sentences */
cs conj s1 s1 sm p_s1   /* sentence(s1) -> adverbial phrase(sm) + ","(pause) + s1 */
                        /* ex: when you go to the store, you buy groceries */
s1 p2 p_s1              /* sentence(s1) -> prepositional phrase(p2) + pause + s1 */
                        /* ex: at the store, you buy groceries */
sm adv s1               /* adverbial phrase(sm) -> adverb(adv) + s1 */
p_s1 pause s1
vp vp sm                /* verb phrase(vp) -> verbphrase(vp) + adverbial phrase(sm) */
                        /* ex: (John) ((goes to the store)vp (when the sun goes down)sm)vp) */
s2 if_s1 ep             /* full sentence(s2) -> if + s1 + pause + s1 */
                        /* ex: If you go to the movies, I'll go with you. */
if_s1 if_s p_s1
if_s if s1
p_s1 s1 nil             /* pause can be omitted in previous cases */
s1 np vp                /* sentence(s1) -> noun phrase(np) + verb phrase(vp) */
vp vp if_s              /* verb phrase(vp) -> vp + if-sentence */
                        /* (I) will go to the movies if you go */
np np cnp               /* noun phrase can be a conjunction of multiple noun phrases */
cnp conj np
np np r2                /* noun phrase(np) -> np + relative clause(r2) */
```

```
r2 r2 cr1              /* a relative clause can be a conjunction of relative clauses */
cr1 conj r2
r2 rel_pron vp         /* relative clause(r2) -> relative pronoun
+ verb phrase(vp) */
                       /* ex. the ladies (who sing) */
r2 rel_pron np_v       /* relative clause(r2) -> relative pronoun + np */
                       /* + non-complete sentence (np_v) */
np_v np vt             /* non-comple sentence(np_v) -> noun phrase + transitive verb */
                       /* ex. the shirt that (you like) */
np_v np vt_inf         /* np_v -> noun_phrase + transitive verb modified by infinitival phrase */
vt_inf vt inf_vp       /* ex. the shirt that (you like to wear) */
np_v np vt_sm          /* np_v -> noun_phrase + transitive verb modified by adverbial phrase */
vt_sm vt sm            /* ex. the shirt that (I wore when you came) */
np_v np vi_p           /* np_v -> noun_phrase + verb phrase + prep */
vi_p vp prep           /* ex. the chair that (you are sitting on) */
np_v np vst            /* np_v -> noun_phrase + verb phrase (want/like) +
                       /* np + inf + verb_phrase + prep */
vst vs s_vp            /* ex. the chair that (you want John to sit on) */
s_vp np inf_vi_p
inf_vi_p inf vi_p
adj adj_cause that_s1  /* adjective(adj) -> so + adj + that + sentence */
adj_cause so adj       /* a painting (so marvellous that your eyes blink when you see it) */
that_s1 that s1
np np pp_inf           /* noun phrase(np) -> np + past participle + infinitival phrase */
pp_inf pp inf_vp       /* the shirt hung to dry */
np np p2               /* np -> np + prepositional phrase (p2) */
p2 prep np             /* the shirt in the drawer */
p2 p2 cp1              /* p2 can be a conjunction of prepositional phrases */
cp1 conj p2
inf_vp inf_vp cinf     /* infinitival phrase can be a conjunction of infinitival phrases */
cinf conj inf_vp
cp1 conj inf_vp        /* prepositional phrases and infinitival phrases */
cinf conj p2           /* can be mixed in conjunctions */
np det n               /* np -> determinant(det) + noun */
np pron_pos n          /* np -> possessive pronoun + noun */
                       /* ex. my shirt */
np adj_num n           /* np -> numerical adjective + noun */
```

```
                    /* ex. twelve shirts */
np number n         /* np -> number + noun */
                    /* ex. 12 shirts */
np n nil            /* np -> noun */
np pn nil           /* np -> proper noun */
np pron nil         /* np -> pronoun */
n nc np             /* np -> noun + possessive mark(poss) + noun phrase(np) */
nc n poss           /* ex. the lady's hat */
nc pron poss        /* ex. someone's hat */
nc pn poss          /* ex. John's hat */
np np adj           /* np -> np + adjectival phrase */
np np pp            /* np -> np + past participle */
n adj n             /* noun -> adjectival phrase + noun */
pron adj pron       /* pronoun -> adjectival phrase + pronoun */
adj_num adj_num cadj_num
                    /* numerical adjective can be a conjunction of numerical adjectives */
cadj_num conj adj_num
adj adv adj         /* adj -> adverb + adjectival phrase */
                    /* ex. the very large room */
adj adj p2          /* adj -> adjectival phrase + prepositional phrase */
                    /* ex. afternoons are short in winter */
adj adj_s nil       /* adj -> superlative adjective */
adj adj_c nil       /* adj -> comparative adjective */
adj adj cadj        /* adj can be a conjunction of adjectival phrases */
cadj conj adj
adv adv cadv        /* adverb can be a conjunction of adverbs */
cadv conj adv
vp vs inf_s         /* verb phrase -> (want/like)vs + infinitival sentence */
inf_s np inf_vp     /* ex. John wants Mary to come with him */
inf_s inf_s ci      /* infinitival sentence can be a conjunction */
ci conj inf_s       /* of multiple infinitival sentences */
vp vw sx            /* vp -> "watch" + sentence */
sx np xx_vp         /* ex. John watched Mary go to the store */
xx_vp vp nil
vp vq quest_s1      /* vp -> (decide/judge)vq + (where/what)quest + vp */
quest_s1 quest vp   /* ex. John judged who did the best drawing */
vp vs inf_vps       /* vp -> (help/make)vs + infinitival sentence */
```

```
inf_vps inf vp          /* ex. John helped (to) bring the tree inside */
inf_vps np inf_vps      /* ex. John helped Mary bring the tree */
inf_vps vp nil
inf_vp inf vp           /* infinitival phrase -> "to" + verb phrase */
vp vp how_s1            /* vp -> vp + how phrase */
                        /* ex. know how heavy it is */
how_s1 how_adj s1
how_adj how adj
how_adj how_adj n       /* ex. know how many pounds this weighs */
p2 prep vp_ing          /* prepositional phrase -> preposition + ing verb phrase */
                        /* for eating */
vp_ing vp_ing cing      /* ing verb can be a conjunction of ing verbs */
cing conj vp_ing
vp_ing vt_ing np        /* ing verb phrase (vp_ing) -> ing transitive verb + np */
vp_ing vi_ing nil       /* ing verb phrase (vp_ing) -> ing intransitive verb */
n vi_ing nil            /* any ing verb can become a noun */
                        /* ex. the eating, the coming */
vp vp adv_vp            /* verb phrase -> verb phrase + adverbial phrase */
vp adv_vp vp            /* verb phrase -> adverbial phrase + verb phrase */
adv_vp quest inf_vp     /* adverbial phrase -> quest + infinitival phrase */
                        /* where to go */
vp vp cvp               /* verb phrase can be a conjunction of verb phrases */
cvp conj vp
vp vp p2                /* verb phrase -> vp + prepositional phrase */
                        /* eat at the table */
vp vp inf_vp            /* verb phrase -> vp + infinitival phrase */
                        /* eat to get bigger */
vp vi nil               /* verb phrase -> intransitive verb */
vp vt np                /* verb phrase -> transitive verb + noun phrase */
vp vg np2               /* verb phrase -> (give/mail)vg + 2 noun phrases */
np2 np np               /* give Mary the book */
vp vb nil               /* verb phrase -> verb to be */
vp vx r3                /* verb phrase -> (think/believe) + relative clause
                        /* made of a complete sentence */
r3 rel_pron s1          /* I believe that John will win the race */
vp vf adj               /* verb phrase -> (stay/feel) + adjectival phrase */
                        /* ex. I feel bad.
```

```
vt vt cvt              /* a transitive verb can be a conjunction of transitive verbs */
cvt conj vt
vt vb vt_ing           /* transitive verb -> verb to be(vb) + ing form of transitive verb */
vi vb vi_ing           /* intransitive verb -> verb to be(vb) + ing form of intransitive verb */
vb vb not              /* vb -> vb + negation(not) */
vp vb np               /* vp -> vb + np */
                       /* ex. John is a carpenter. */
vp vb p2               /* vp -> vb + p2 */
                       /* ex. John is in a good mood */
vp vb adj              /* vb -> vb + adjectival phrase */
                       /* ex. John is tall. */
vp vb pp               /* vb -> vb + past participle */
                       /* ex. is gone */
pp pp p2               /* past particple -> past participle + prepositional phrase */
                       /* left on the couch */
pp pp cpp              /* past participle can be a conjunction of multiple past participles */
cpp conj pp
vt mod vt              /* modifiers before verbs, as auxiliaries or adverbs */
vi mod vi
vb mod vb
mod aux nil
mod adv nil
mod aux not
vp vp mod_ap           /* modifiers after the verb */
vt vt mod_ap
vb vb mod_ap
mod_ap adv nil
conj pause nil         /* conjunction -> ","(pause) */
conj pause conj        /* conjunction -> pause + conjunction */
vp vp comp_np          /* verb phrase -> verb phrase + comparative noun phrase */
comp_np comp np        /* comparative noun phrase -> comparative adjective + than + np */
comp adj_c than
comp as adj_as         /* comparative noun phrase -> as + adjective + as */
adj_as adj as
m m not                /* verb mean or is can be negated in the definition */
                       /* a car is not a ... */
m adv m                /* verb mean or is can be modified by an adverb */
```

```
                        /* a car also means a ... */
s1 inf_vp m_inf         /* verb phrase definitions: to xxx is to yyy */
m_inf m inf_vp          /* infinitival phrase (means/is) infinitival phrase */
np adj_num nil          /* noun phrase can be a number */
                        /* you add two and two */
/*n symbx nil           /* a noun can be a symbol or a conjunction of symbols */
/*symbx symb csymb
/*symbx symb nil
/*csymb conj symbx
/*symb let_maj nil      /* the only symbol defined so far is a capital letter */
s1 np m_np              /* noun can be defined by a sentence */
m_np m s1               /* ash is what is left after ... */
m_np m how_s1           /* the age of something is how old it is */
vp va quote             /**/ vp -> va pause equote s2 equote
quote pause s_quote     /**/ vp -> va : equote s2 equote
quote p s_quote         /**/
s_quote equote s2x      /**/
s2x s2 equote           /**/
vt vh_a pp              /* vt/vi -> auxiliary have + pp */
vi vh_a pp              /* ex. astronauts have walked on the moon */
vh_a vh nil             /* add adverb between auxiliary and pp */
vh_a vh adv
```

RULES_EXTRA.DIC

```
adv_vp adj np          /* adverbial phrase -> adjective + noun phrase */

r3 s1 nil              /* skip the relative pronoun */
                       /* John believes (that) Mary will win the race */

r2 np_v nil            /* skip the relative pronoun */
                       /* Mary's skirt (that) I washed yesterday */

mod_ap prep nil        /* use a preposition as a modifier after a verb */

n n n                  /* a noun can be made of two nouns */
                       /* the metal head */

s2 sm ep               /* a full sentence can be an adverbial sentence */
                       /* When the leaves fall. */

s2 vp ep               /* a full sentence can be a verb phrase */
```

Appendix D

Parse to Conceptual Graph rules

Here is some information to help the understanding of the rules presented:

- The word in parenthesis next to a category indicates a variation. That variation shows information that has been carried up into the parsing from the morphological analysis. The words in the parse tree correspond to the base forms found. For example, if the word "eater" was analyzed as an agent variation on a verb "eat", in the parse tree we will have (n->eat), but in the data structure we keep information about part-of-speech, here noun, and variation, here agent. These variations will affect the resulting graph.
- We put letters in parenthesis next to the categories to avoid confusion when there are two items of the same part of speech involved
- The referent "ref" indicates an anaphora that will have to be resolved.
- The referent "quest" indicates a question is asked, so the referent is not resolved, but it is not in the sentence
- The relation "quest" is a superclass for all relations (where, when, what, who). We have no way to decide now which it is, and we will try at the graph level to decide by looking at the superclass of the object. ex. [go]->(quest)->[home], and we find that "place" subsumes "home", we can specialize the relation to "where", [go]->(where)->[home]

parse to CG rules: LEAF LEVEL	
SYNTACTIC RULE(S) GRAPH	EXAMPLE SENTENCE EXAMPLE GRAPH
pn [person:pn]	John [person:John]
pron [pron:ref]	he [he:ref]
n(agent) [n]->(agent)->[person]	eater [eat]->(agent)->[person]
adj []->(attribut)->[adj]	beautiful ring [ring]->(attribut)->[beautiful]
adv [adv]->(modif)->[]	runs fast [fast]->(modif)->[run]
not [not]->(modif)->[]	will not go [not]->(modif)->[go]
n/v/others [n]	chair [chair]

parse to CG rules: CONJUNCTIONS	
SYNTACTIC RULE(S) GRAPH	EXAMPLE SENTENCE EXAMPLE GRAPH
cnp conj np []->(conj)->[np]	cat and dog [cat]->(and)->[dog]
conj pause conj []->(conj)->[np]	cat, or dog [cat]->(or)->[dog]
conj pause nil []->(pause)->[np]	cat, dog [cat]->(pause)->[dog]

parse to CG rules: PREPOSITIONS	
SYNTACTIC RULE(S) GRAPH	EXAMPLE SENTENCE EXAMPLE GRAPH
?? ?? p2 p2 prep np []->(prep)->[np]	 cat on mat [cat]->(on)->[mat]
p2 p2 cp1 cp1 conj p2 []->(prep1)->[np(a)] ->(prep2)->[np(b)]	cat is playing with a ball and on the mat [play]->(with)->[ball] ->(on)->[mat]

parse to CG rules: VERB FORMS AND MODIFIERS	
SYNTACTIC RULE(S) GRAPH	EXAMPLE SENTENCE EXAMPLE GRAPH
v? vb v?_ing [v?_ing]	is going [go]
v? v? mod_ap [mod_ap]->(modif)->[v?]	drive fast [fast]->(modif)->[drive]
v? vb pp [pp]	is gone [go]
v? vh_a pp [pp]	have walked [walk]

parse to CG rules: VERB COMPLEMENT	
SYNTACTIC RULE(S) GRAPH	EXAMPLE SENTENCE EXAMPLE GRAPH
vp vt np [vt]->(object)->[np]	brush your teeth [brush]->(object)->[teeth]
vp vg np2 np2 np(a) np(b) []->(to)->[np(a)] ->(object)->[np(b)]	give Mary the book [give]->(to)->[Mary] ->(object)->[book]
vp vp inf_vp inf_vp inf vp []->(goal)->[vp]	eat to grow [eat]->(goal)->[grow]
vp vs inf_s inf_s np inf_vp []->(what)->[vp]->(agent)->[np]	wants Mary to come [want]->(what)->[come]->(agent)->[Mary]
vp vw sx sx np xx_vp xx_vp vp nil []->(what)->[vp]->(agent)->[np]	watch Mary go home [watch]->(what)->[go]->(agent)->[Mary]
vp vp sm sm adv s1 []->(adv)->[s1]	when you go []->(when)->[go]->(agent)->[you]
vp vp sm adv_np np nil []->(quest)->[np]	go every day [go]->(quest)->[day]->(attribut)->[every]
vp vp comp_np comp_np comp np comp adj_c than []->(modif)->[adj]->(than)->[np]	sing better than you [sing]->(modif)->[better]->(than)->[you]
comp as adj_as adj_as adj as []->(modif)->[adj]->(equal)->[np]	sing as good as you [sing]->(modif)->[good]->(equal)->[you]

parse to CG rules: VERB COMPLEMENT (continued)	
SYNTACTIC RULE(S) GRAPH	EXAMPLE SENTENCE EXAMPLE GRAPH
vp va quote quote pause s_quote s_quote equote s2x s2x s2 equote []->(what)->[s2]	John says, "xxx". [say]->(what)->[xxx]
vp vp how_s1 how_s1 how_adj s1 how_adj how adj []->(what)->[s1]->(attribut)->[adj:how]	you know how heavy this is [know]->(what)->[is]->(attribut)->[heavy:how]
how_adj how_adj n []->(what)->[s1]->(attribut)->[n:how_many]	you know how many pounds this weighs [know]->(what)->[weigh]->(attribut)->[pound:how_many]
vp vp quest_s1 quest_s1 quest vp []->(what)->[]<-(quest)<-[vp]	John decided who did ... [person:quest]<-(what)<-[decide] <-(who)<[do]
quest_s1 quest s1 []->(what)->[]<-(quest)<-[s1]	I know where the animal are [place:quest]<-(what)<-[know] <-(where)<-[be]->(agent)->[animal]
quest_s1 ques inf_vp []->(what)->[]<-(quest)<-[vp]	I know how to fight [manner:quest]<-(what)<-[know] <-(how)<-[fight]

parse to CG rules: NOUN MODIFIERS	
SYNTACTIC RULE(S) GRAPH	EXAMPLE SENTENCE EXAMPLE GRAPH
np det n [n:det]	the cat [cat:the]
np pron_pos n [n:pron_pos]	my cat [cat:my]
np number n [n:number]	12 cats [cat:12]
np np pp [pp]->(object)->[]	broken car [break]->(object)->[car]
np nc np nc poss np [np(a)]->(poss)->[np(b)]	John's book [John]->(poss)->[book]
np n n [n(a)]->(modif)->[n(b)]	metal head [metal]->(modif)->[head]
adj adj_cause that_s1 adj_cause so adj that_s1 that s1 []->(attribut)->[adj]<-(modif)<-[so] ->(cause)->[s1]	light so bright that ... [light]->(attribut)->[bright] .....<-(modif)<-[so] ->(cause)->[...]

parse to CG rules: SENTENCE FORMATION	
SYNTACTIC RULE(S) GRAPH	EXAMPLE SENTENCE EXAMPLE GRAPH
s1 np vp [vp]->(agent)->[np]	the animals stay [stay]->(agent)->[animals]
s1 np vp(passive) [vp]->(object)->[np]	food is kept [keep]->(object)->[food]
if_s1 if_s p_s1 if_s if s1 p_s1 pause s1 [s1(a)]->(if)->[s1(b)]	 if you go, I'll go [go]->(agent)->[I] ->(if)->[go]->(agent)->[you]
if_s1 if_s then_s1 then_s1 then s1 [s1(a)]->(if)->[s1(b)]	if you go, then I'll go if you go, then I'll go [go]->(agent)->[I] ->(if)->[go]->(agent)->[you]

parse to CG rules: SPECIFIC DEFINITION SYNTAX	
SYNTACTIC RULE(S) GRAPH	EXAMPLE SENTENCE EXAMPLE GRAPH
s1 np m_np m_np m s1 [np]->(equiv)->[s1]	 ash is what is left [ash]->(equiv)->[what]<-(object)<-[leave]
s1 inf_vp m_inf m_inf m inf_vp [vp(1)]->(equiv)->[vp(2)]	 to lie is to say [lie]->(equiv)->[say]

parse to CG rules: RELATIVE CLAUSES	
SYNTACTIC RULE(S) GRAPH	EXAMPLE SENTENCE EXAMPLE GRAPH
np np r2 r2 rel_pron vp []<-(agent)<-[vp]	 the cat that runs [run]->(agent)->[cat]
r2 rel_pron np_v np_v np vt_mod vt_mod vt nil [vt]->(object)->[] ->(agent)->[np]	 the car that I like [like]->(object)->[car] ->(agent)->[I]
vt_mod vt sm vt_mod vt adv_np vt_mod vt inf_vp	add modifier to vt
np_v np vi_p vi_p vp prep [vp]->(agent)->[np] ->(prep)->[]	 the chair that you are sitting on [sit]->(agent)->[you] ->(on)->[chair]
np_v np vst vst vs inf_vt [vs]->(what)->[vt]->(object)->[] ->(agent)->[np]	 the shirt that you like to wear [like]->(agent)->[you] ->(what)->[wear]->(object)->[shirt]
vst vs s_vp s_vp np inf_vi_p inf_vi_p inf vi_p vi_p vp prep [vs]->(agent)->[np(1)] ->(what)->[vp]->(agent)->[np(2)] ->(prep)->[]	 the chair that you want John to sit on [want]->(agent)->[you] ->(what)->[sit]->(agent)->[John] ->(on)->[chair]

Appendix E

Semantic Relation Transformation Graphs

OBJECT RELATIONS: 1. **Part-whole relations**

PART-OF: part of an object (part-of)

Defining formulas: an arm **is a part of** the body
pines **have** needles on their branches

SRTG : [something:B]->(part-of)->[something:A]

Before :

[something:A]<-(agent)<-[be]->(object)->[part]->(of)->[something:B]

[something:B]<-(agent)<-[have]->(object)->[something:A]

PIECE-OF: piece of an object (piece-of)

Defining formulas: a block is **a piece of** wood

[something:A]<-(agent)<-[be]->(object)->[piece]->(of)->[something:B]

SRTG: [something:B]->(piece-of)->[something:A]

AREA-OF: area of an object (area-of)

Defining formulas: a beach is **an area of** sand

[something:A]<-(agent)<-[be]->(object)->[area]->(of)->[something:B]

SRTG: [something:B]->(area-of)->[something:A]

AMOUNT-OF: amount of some mass noun (amount-of)

Defining formulas: a breath is **an amount of** air

SRTG: [something:B]->(amount-of)->[something:A]

Before:

[something:A]<-(agent)<-[be]->(object)->[amount]->(of)->[something:B]

CONTENT: the content of an object (content)

Defining formulas: Paul was carrying a pail **of** water.

SRTG: [something:B]->(content)->[something:A]

Before:

[something]->(of)->[something:B]

2. Member/set relations

SET-OF: group, set, pile of something (set-of)

Defining formulas: an army is a large **group of** people
a costume is **a set of** special clothes
dust is tiny **pieces of** dirt
a pile is **a lot of** something

SRTG: [something:B]->(set-of)->[something:A]

Before:

[something:A]<-(agent)<-[be]->(object)->[group]->(of)->[something:B]

[something:A]<-(agent)<-[be]->(object)->[set]->(of)->[something:B]

[something:A]<-(agent)<-[be]->(object)->[piece:plural]->(of)->[something:B]

[something:A]<-(agent)<-[be]->(object)->[lot]->(of)->[something:B]

ELEMENT-OF: one of many (element-of)

Defining formulas: a letter is **one of** the symbols

SRTG: [something:B]->(element-of)->[something:A]

Before:

[something:A]<-(agent)<-[be]->(object)->[one]->(of)->[something:B]

3. Human/animal relations

CHILD-OF: the child of a human or animal (child-of)

Defining formulas: your cousin is **the child of** your aunt or uncle
a lamb is **a young** sheep

SRTG: [person/animal:B]->(child-of)->[person/animal:A]

Before:

[person/animal:A]<-(agent)<-[be]->(object)->[child]->(of)->[person/animal:B]

[person/animal:A]<-(agent)<-[be]->(object)->[person/animal:B]
->(attribut)->[young]

POSSESSION: the possession of someone (poss)

Defining formulas: men and women who **have** children are parents
many people **have** dogs as pets

SRTG: [person/animal:A]->(poss)->[something:B]

Before:

[person/animal:A]<-(agent)<-[have]->(object)->[something:B]

HOME: someone's home or area of living (home-for)

Defining formulas: a hive is **a home for** bees
horses **live** on farms

SRTG: [person/animal:B]->(home-for)->[something:A]

Before:

[something:A]<-(agent)<-[be]->(object)->[home]->(for)->[person/animal:B]
[person/animal:B]<-(agent)<-[live]->(in)->[something:A]

4. Comparison relations

LIKE: indicate similarity between objects (like)

Defining formulas: a rainbow **looks like** a ribbon ...
rubbers are **like** shoes
baskets are often **shaped like** bowls and ...

SRTG: No one to find, the preposition like will be kept as is,
and considered to mean similar

MORE-THAN: a exageration of an attribute (more-than)

Defining formulas: So far limited with the verb to be
John is **better than** you (comparative adj)

SRTG: No one to find, established at parse time

[be]->(agent)->[person]
->(attribut)->[attribute]->(more-than)->[person]

AS: an equality in attribute (as)

Defining formulas: Again so far limited with to be
John is **as good as** you

SRTG: No transformation for as if found as is in sentence,
but negation of more-than, and less-than leads to as

[be]->(agent)->[something:A]
->(attribut)->[attribute:B]->(as)->[something:B]

Before:

[be]->(agent)->[something:A]
->(attribut)->[attribute:B]->(more-than)->[something:B]

<-(modif)<-[not]

[be]->(agent)->[something:A]
->(attribut)->[attribute:B]->(less-than)->[something:B]
<-(modif)<-[not]

LESS-THAN: a diminution of an attribute (less-than)

Defining formulas: Same again, limited with **to be**
John is **not as good as** you

SRTG: negation of **as**

[be]->(agent)->[something:A]
->(attribut)->[attribute:B]->(less-than)->[something:B]

Before:

[be]->(agent)->[something:A]
->(attribut)->[attribute:B]->(as)->[something:B]
<-(modif)<-[not]

5. **Spatial relations**

Prepositions: All possible prepositions, left as they are until further disambiguations

Defining formulas: Many prepositions are possible (on,in,above,...)

SRTG: They are kept as relations, put as is in graph.
When used with the verb **to be**, we can transform

[something:A]->(preposition)->[something:B]

Before:

[something:A]<-(agent)<-[be]->(preposition)->[something:B]

6. **Word relations**

OPPOSITE: the contrary of something/action (opposite)

Defining formulas: back is **the opposite of** front
to break means **not** to work

SRTG: In the case of the verb definition, the correct relation is established directly from the parse tree.

[something:A]->(opposite)->[something:B]

Before:

[something:A]<-(agent)<-[be]->(object)->[something:B]
<-(modif)<-[not]

SYNONYMY: two similar objects or actions (equiv)

Defining formulas: automobile **is another word for** car
earth **means** dirth **with no modifiers**
to beat **means** to hit **with no modifiers**

SRTG: In the case of the verb definition, the correct relation is established directly from the parse tree.

[something:A]->(equiv)->[something:B]

Before:

[something:A]<-(agent)<-[be]->(object)->[word]->(attribut)->[another]
->(for)<-[something:B]

[something:A]<-(agent)<-[mean]->(object)->[something:B]

TAXONOMY: something/act being a subclass of another something/act (is-a)

Defining formulas: an acorn **is a** nut that ...
an apple **is a kind of** fruit
a date is **any one** day
dogs, cats, birds, and insects **are all** animals
pictures, poems, and musics **are kinds of** art
to crawl **is to** move on your hands and knees

SRTG: [something:A]->(is-a)->[something:B]

Before:

[something:A]<-(agent)<-[be]->(object)->[something:B]

[something:A]<-(agent)<-[be]->(object)->[kind]->(of)->[something:B]

7. Description relations

NAME: the name of an object (name-of)

Defining formulas: an address is the **name of** a place

SRTG: [something:B]->(name-of)->[something:A]

Before:

[something:A]<-(agent)<-[be]->(object)->[name]->(of)->[something:B]

ATTRIBUTE: any attribute to an object (attribut)

Defining formulas: an adjective modifying a noun

SRTG: Extracted from parse tree

MATERIAL: what an object is made-of (material)

Defining formulas: glass is what windows are **made from**
a hose is a tube **made of** rubber

SRTG: [something:A]->(material)->[something:B]

Before:
[something:A]<-(agent)<-[be]->(object)->[made]->(of)->[something:B]
[something:A]<-(agent)<-[be]->(object)->[made]->(from)->[something:B]

FUNCTION: the function of an object (function)

Defining formulas: a pen is **a tool to** write
a fence is **made to** keep two places ...

SRTG: [something:A]->(function)->[act:B]
Before:
[something:A]<-(agent)<-[be]->(object)->[made]->(to)->[act:B]

ABOUT: the subject matter (about)

Defining formulas: preposition about, or some adjectives

SRTG: we will need more information to decide to transform an
attribute relation into an about relation,
as in the history book, [book]->(attribute)->[history]

SITUATION RELATIONS: 1. Action modifier

MODIF: general adverbial modifier (modif)

Defining formulas: All adverbial modifiers

SRTG: Extracted from parse tree
[act:A]->(modif)->[B]

2. Case-role relations

INSTRUMENT: instrument involved in act (instrument)

Defining formulas: a bag is **used to** hold things
to bite means to cut **with** your teeth
to climb means **to use** your hands and feet

SRTG: the formula used to can be transformed, but the
preposition with is ambiguous and we need
further information to transform it into the instrument relation
[act:A]->(instrument)->[something:B]
Before:
[something:B]<-(object)<-[use]->(goal)->[act:A]
[something:B]<-(agent)<-[be]->(object)->[tool]
->(goal)->[act:A]

METHOD: method used to achieve act (method)

Defining formulas: a trap is **a way to** catch wild animals

to blow means to make a sound **by** pushing **ing form** air

SRTG: [act:A]->(method)->[something:B]

Before:

[something:B]<-(agent)<-[be]->(object)->[way]
->(goal)->[act:A]

[act:A]->(by)->[act:B]

MANNER: manner that the act is performed (manner)

Defining formulas: to giggle is to laugh **in a silly way**

SRTG: [act:A]->(manner)->[attribute:B]

Before:

[act:A]->(in)->[way]->(attribute)->[attribute:B]

AGENT: the person/animal who performs the action (agent)

Defining formulas: an author is **someone who** writes a story
a barber is **a person who** gives haircut

The question was asked (past part) **by** Dana.

SRTG: Most agent relations are found directly for the parse tree, either the subject of the verb, or object if in passive form

[act:A]->(agent)->[something:B]

Before:

[act:A]->(by)->[something:B]

EXPERIENCER: person experiencing a state (experiencer)

Defining formulas: Mary **is cold** (adj)

SRTG: [person:A]->(experiencer)->[attribute:B]

Before:

[person:A]<-(agent)<-[be]->(attribut)->[attribute:B]

LOCATION: where the action takes place (location)

Defining formulas: an airport **is a place where** ...
a bathroom **is a room where** ...
a bank is a safe **place to** keep your money
a direction is **somewhere** you can look
to arrive is to come to **a place**

SRTG: [act:A]->(location)->[place:B]

Before:

[something:B]<-(agent)<-[be]->(object)->[place]
->(to)->[act:A]

[something:B]<-(agent)<-[be]->(what)->[place/somewhere]<-(where)<-[act:A]
[act]->(in)->[way]->(attribut)->[attribute:B]

OBJECT: direct object of the action (object)

Defining formulas: Direct object of transitive verb

SRTG: Put into graph directly from transformation of parse tree

[act:A]->(object)->[something:B]

RECIPIENT: recipient of act (recipient)

Defining formulas: to pass means to give **to someone** with your hands

SRTG: [act:A]->(recipient)->[person:B]

Before:

[act:A]->(to)->[person:B]
->(object)->[something]

RESULT: an entity that results from an action (result)

Defining formulas: ash is **what is** left (**past part.**) after something burns
smoke is **made by** things that burn

SRTG: [act:A]->(result)->[something:B]

Before:

[something:B]<-(object)<-[act:A]->(after)->[act]
[something:B]<-(object)<-[make]->(by)->[act:A]

CAUSE: an action/state resulting from another action/state (cause)

Defining formulas: to kill is **to cause to** die
to pour is **to make** liquid go from one place to another
the window broke **when** a baseball went through it

SRTG: The relation **when** will need more analysis to be changed
into **result**, it could also mean a time.

[act:B]->(cause)->[act:A]

Before:

[act:A]->(equiv)->[cause/make]->(goal)->[act:B]
[act:B]->(when)->[act:A]

TRANSFORMATION: the new state of an entity because of an action (transform)

Defining formulas: to die means **to become** dead

SRTG: [act:A]->(transform)->[attribute:B]

Before:

[act:A]->(equiv)->[become]->(attribute)->[attribute:B]

REASON: why the act is performed (reason)

Defining formulas: many people hug each other **to show that** they are glad
to laugh is to make a sound **that show that** something is funny
Olivier asked for more soup **because** he was hungry

SRTG: [act:A]->(reason)->[act:B]

Before:

[act:B]->(to)->[show]->(what)->[act:A]

[act:A]->(because)->[act:B]

GOAL: goal of an action (goal)

Defining formulas: to chase means to run after something **to** try to catch it
to explain means to tell about something **so that** ...

SRTG: The goal relation will be put in the cg from the parse tree

[act:A]->(goal)->[act:B]

ACCOMPANIMENT: multiple agents cooperating to a unique act (accompaniment)

Defining formulas: Bill and Jane swam **together**

SRTG: [person:A]->(accompaniment)->[person:B]

Before:

[person:A]->(and)->[person:B]
<-(agent)<-[act]<-(modif)<-[together]

Before:

DIRECTION: (direction)

Defining formulas: to bow means to bend the body **forward**

SRTG: [act:A]->(direction)->[forward/backward]

Before:

[act:A]->(modif)->[forward/backward]

SOURCE/DEST: (source)/(dest)

Defining formulas: to move is to go **from** one place **to** another

SRTG: [place]<-(source)<-[act:A]->(destination)->[place]

Before:

[place]<-(from)<-[act:A]->(to)->[place]

[act:A]->(from)->[place]->(to)->[place]

PATH: .

Defining formulas: to eat means to take food into the body **through** the mouth
to fly is to travel **through** the air

SRTG: [act:A]->(path)->[something]

Before:

[act:A]->(through)->[something]

DURING: (during)

Defining formulas: to dream means to imagine stories **while** you sleep
Jeff breathes fast **when** he runs

SRTG: when is ambiguous and will need more processing before a transformation to during.

[act:A]->(during)->[act:B]

Before:

[act:A]->(while)->[act:B]

[act:A]->(when)->[act:B]

POINT IN TIME: specific time (at-time)

Defining formulas: birth is **the moment when** a person is born

SRTG: [act:A]->(at-time)->[time:B]

Before:

[time:B]->(when)->[act:A]

FREQUENCE: (frequence)

Defining formulas: to practice is to do something **many times** so that ...

SRTG: [act:A]->(frequence)->[time:B]

Before:

[act:A]->(quest)->[time:B]

3. Agent involvement relations

ABILITY : (able)

Defining formulas: power is **being able to** do work
to know is to **be able** to say who they are

SRTG: [person:A]->(able)->[act:B]

Before:

[person:A]<-(agent)<-[be]->(attribute)->[able]
->(to)->[act:B]

[person:A]<-(agent)<-[act:B]<-(modif)<-[can]

DESIRED ACT: (desire)

Defining formulas: to hurry is **to try** to go quickly

SRTG: [person:A]->(desire)->[act:B]

Before:

[person:A]<-(agent)<-[want]->(to)->[act:B]

INTENTION (intention)

Defining formulas:

SRTG: [person:A]->(intention)->[act:B]

Before:

[person:A]<-(agent)<-[try]->(to)->[act:B]

OBLIGATION (obligation)

Defining formulas: have to

SRTG: [person:A]->(obligation)->[act:B]

Before:

[person:A]<-(agent)<-[have]->(to)->[act:B]

4. **Action relations**

ACT (act)

Defining formulas: attention is looking (**verb**) and listening with care
exercise is running (**verb**) and jumping

SRTG: [something:A]->(act)->[act:B]

Before:

[something:A]->(equiv)->[act:B]

EVENT (event)

Defining formulas: a chase **is when** someone follow ... (**with a sentence**)
an earthquake **happens when** ...

SRTG: [something:A]->(process)->[act:B]

Before:

[something:A]<-(agent)<-[is/happen]->(when)->[act:B]

PROCESS (process)

Defining formulas: to roll is **to keep** turning over and over
to twist is to turn around and around **repetition**

SRTG: [act:A]->(continuous)->[act:B]

Before:

[act:A]->(equiv)->[keep]->(object)->[act:B]

[act:A]->(equiv)->[act:B]->(modif)->[C]->(and)->[C]

SEQUENCE: (sequence)

Defining formulas: to dip means to put something in liquid **and then** to ...

After the ball game, we went home.

SRTG: [act:A]->(succ)->[act:B]

Before:

[act:A]->(then)->[act:B]

[act:B]->(after)->[act:A]

Bibliography

[1] *Essential German.* Berlitz Publishing Company, Inc, 1992.

[2] E. Agirre, X. Arregi, X. Artola, A. Diaz de Ilarraza, and K. Sarasola. Lexical Knowledge Representation in an Intelligent Dictionary Help System. In *Proc. of the 15 th COLING*, Kyoto, Japan, 1994.

[3] T. Ahlswede and M. Evens. Generating a relational lexicon from a machine-readable dictionary. *International Journal of Lexicography*, 1(3):214–237, 1988.

[4] H. Alshawi. Processing Dictionary Definitions with Phrasal Pattern Hierarchies. *Computational Linguistics*, 13:203–218, 1987.

[5] H. Alshawi. Analysing the dictionary definitions. In Bran Boguraev and Ted Briscoe, editors, *Computational Lexicography for Natural Language Processing*, chapter 7, pages 153–170. Longman Group UK Limited, 1989.

[6] R.A. Amsler. The Structure of the Merriam-Webster Pocket Dictionary. Technical Report TR-164, University of Texas, Austin, 1980.

[7] Robert A. Amsler. A taxonomy for English nouns and verbs. In *Proc. of the 19 th ACL*, pages 133–138, Stanford, CA, June 1981.

[8] P. Anick and S. Bergler. Lexical Structures for Linguistic Inference. In J. Pustejovsky and S. Bergler, editors, *Lexical Semantics and Knowledge Representation : First SIGLEX Workshop*, chapter 9, pages 121–136. Springer-Verlag, 1992.

[9] N. Anquetil and J. Vaucher. Extracting hierarchical graphs of concepts from an objects set: comparison of two methods. In *Proceedings of Workshop on Knowledge Acquisition Using Conceptual Graph Theory*, pages 26–45, University of Maryland, College Park, MD, USA, August 1994.

[10] F. Antonacci, M.T. Pazienza, M. Russo, and P. Velardi. Analysis and Generation of Italian Sentences. In T.E. Nagle, J.A. Nagle, L.L. Gerholz, and P.W.Eklund, editors, *Conceptual Structures: Current Research and Practice*, chapter 22, pages 437–459. Ellis Horwood, 1992.

[11] F. Antonacci, M. Russo, M.T. Pazienza, and P. Velardi. A system for text analysis and lexical knowledge acquisition. *Data and Knowledge Engineering*, 4(1):1–20, 1989.

[12] Ju. D. Apresjan. Regular Polysemy. *Linguistics*, 142:5–32, 1974.

[13] B.T.S. Atkins. Building a Lexicon: The Contribution of Lexicography. *International Journal of Lexicography*, 4(3):168–203, 1991.

[14] A. Barr and E.A. Feigenbaum. *The Handbook of Artificial Intelligence*, volume II. 1982. Section describin MYCIN.

[15] Caroline Barrière. From Machine Readable Dictionaries to a Lexical Knowledge Base of Conceptual Graphs. In *Supplementary Proceedings of the International Conference on Conceptual Structures (ICCS'95)*, August 1995. Awarded Best Student Research Proposal.

[16] Caroline Barrière. Including certainty terms into a knowledge base of conceptual graphs. In *Colloque Etudiant de Linguistique Informatique de Montréal*, pages 184–191, Montreal, Canada, June 1996.

[17] Caroline Barrière and Fred Popowich. Building a Noun Taxonomy from a Children's Dictionary. In *Euralex'96: Seventh EURALEX International Congress on Lexicography*, pages 27–35, Gôteborg, Sweden, August 1996.

[18] Caroline Barrière and Fred Popowich. Concept Clustering and Knowledge Integration from a Children's Dictionary. In *Proc. of the 16 th COLING*, Copenhagen, Denmark, August 1996.

[19] Caroline Barrière and Fred Popowich. A Dynamic Lexical Knowledge Base for Natural Language Processing. In *AAAI-96 Fall Symposium: Knowledge Representation Systems Based on Natural Language*, pages 1–5, Cambridge, MA, November 1996.

[20] R. Basili, M.T. Pazienza, and P. Velardi. Integration of probabilistic and symbolic methods for semantic categorization. In *AAAI Symposium - Representation and Acquisition of Lexical Knowledge: Polysemy, Ambiguity and Generativity*, pages 8–20, Stanford, California, 1995. AAAI Press.

[21] Roberto Basili, Maria Teresa Pazienza, and Paola Velardi. Acquisition of Selectional Patterns in Sublanguages. *Machine Translation*, 8:147–173, 1993.

[22] Roberto Basili, Maria Teresa Pazienza, and Paola Velardi. What Can Be Learned from Raw Texts? *Machine Translation*, 8:147–173, 1993.

[23] R. Beckwith, C. Fellbaum, D. Gross, and G.A. Miller. WordNet: A Lexical Database Organized on Psycholinguistic Principles. In U. Zernik, editor, *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, chapter 9, pages 211–232. Lawrence Elbaum Associates, 1991.

[24] J.-L. Binot and K. Jensen. A Semantic Expert Using an Online Standard Dictionary. In *Proceedings of ICJAI-87*, pages 709–714, 1987.

[25] B. Boguraev. Building a Lexicon: The Contribution of Computers. *International Journal of Lexicography*, 4(3):228–260, 1991.

[26] B. Boguraev and J. Pustejovsky. Lexical ambiguity and the role of knowledge representation in lexicon design. In *Proc. of the 13 th COLING*, pages 36–41, Helsinki, Finland, 1990.

[27] Bran Boguraev and Ted Briscoe. *Computational Lexicography for Natural Language Processing.* Longman Group UK Limited, 1989.

[28] I. Bournaud and J-G. Ganascia. Conceptual Clustering of Complex Objects: A Generalization Space based Approach. In G. Ellis, R. Levinson, W. Rich, and J. Sowa, editors, *ICCS'95: Conceptual Structures: Applications, Implementation and Theory*, pages 173–187. Springer-Verlag, 1995.

[29] R.J. Brachman and J. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[30] Michael R. Brent. Automatic acquisition of subcategorization frames from untagged text. In *Proc. of the 29 th ACL*, pages 209–214, 1991.

[31] P. Brown, V.J. Della Pietra, P.V. deSouza, J.C. Lai, and R.L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–480, 1992.

[32] Rebecca Bruce and Louise Guthrie. Genus Disambiguation: A study in weighted preference. In *Proc. of the 14 th COLING*, pages 1187–1191, Nantes, France, 1992.

[33] R. Byrd. Discovering Relationships among Word Senses. In *Dictionaries in the Electronic Age: Proceedings of the 5th Annual Conference of the UW Centre for the New OED*, 1989.

[34] R.J. Byrd, N. Calzolari, M. Chodorow, J. Klavans, M. Neff, and O. Rizk. Tools and methods for computational lexicology. *Computational Linguistics*, 13(3-4):219–240, 1987.

[35] N. Calzolari. Detecting Patterns in a Lexical Data Base. In *Proc. of the 10 th COLING*, pages 170–173, Stanford, CA, 1984.

[36] N. Calzolari. The dictionary and the thesaurus can be combined. In M. Evens, editor, *Relational Models of the Lexicon*, Studies in Natural Language Processing, chapter 3, pages 75–96. Cambridge University Press, 1988.

[37] N. Calzolari. Acquiring and Representing Semantic Information in a Lexical Knowledge Base. In J. Pustejovsky and S. Bergler, editors, *Lexical Semantics and Knowledge Representation : First SIGLEX Workshop*, chapter 16, pages 235–244. Springer-Verlag, 1992.

[38] N. Calzolari and A. Zampolli. Methods and Tools for Lexical Acquisition. In M. Filgueiras, L. Damas, N. Moreira, and A. P. Tomas, editors, *Natural Language Processing: Proc. of the 2nd Advanced School in Artificial Intelligence EAIA '90*, pages 4–24. Springer, Berlin, Heidelberg, 1990.

[39] Anil S. Chakravarthy. Sense Disambiguation Using Relations and Adjacency Information. In *Proc. of the 33 th ACL*, pages 293–295, Cambridge,MA, 1995.

[40] E. Charniak. *Statistical Language Learning.* MIT Press, 1993.

[41] M. S. Chodorow, R. J. Byrd, and G. Heidorn. Extracting Semantic Hierarchies from a large on-line Dictionary. In *23rd Annual Meeting of the Association for Computational Linguistics*, pages 299–304, Chicago, Illinois, July 1985.

[42] K. Church, W. Gale, P. Hanks, and D. Hindle. Using statistics in lexical analysis. In U. Zernik, editor, *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, chapter 6, pages 115–164. Lawrence Elbaum Associates, 1991.

[43] K. Church and P. Hanks. Word association norms, mutual information and lexicography. In *Proceedings of the 27th Annual meeting of the Association for Computational Linguistics*, pages 76–83, Vancouver, BC, 1989.

[44] O. Cogis and O. Guinaldo. A linear descriptor for conceptual graphs and a class for polynomial isomorphism test. In G. Ellis, R. Levinson, W. Rich, and J. Sowa, editors, *ICCS'95: Conceptual Structures: Applications, Implementation and Theory*, pages 263–277. Springer-Verlag, 1995.

[45] A. Collins and M. Ross Quillian. How to make a language user. In E. Tulving and W. Donaldson, editors, *Organization of memory*, pages 310–354. Academic Press, NY, 1972.

[46] A. Copestake. The ACQUILEX LKB Representation Issues in Semi-Automatic Acquisition of Large Lexicons. In *Proceedings of 3rd ANLP*, pages 88–95, 1992.

[47] A. Copestake. Representing Lexical Polysemy. In *AAAI Symposium - Representation and Acquisition of Lexical Knowledge: Polysemy, Ambiguity and Generativity*, pages 21–26, Stanford, California, 1995. AAAI Press.

[48] A.A. Copestake. An approach to building the hierarchical element of a lexical knowledge base from a machine readable dictionary. In *Proceedings of the Workshop on Inheritance in Natural Language Processing, Tilburg*, 1990.

[49] Jim Cowie, Joe Guthrie, and Louise Guthrie. Lexical Disambiguation using Simulated Annealing. In *Proc. of the 14 th COLING*, pages 359–365, Nantes, France, 1992.

[50] Jim Cowie, T. Wakao, L. Guthrie, W. Jin, J. Pustejovsky, and S. Waterman. The *diderot* Information Extraction System. In *The First Conference of the Pacific Association for Computational Linguistics*, pages 5–14, Harbour Center, Campus of SFU, Vancouver, April 1993.

[51] D.A. Cruse. *Lexical Semantics.* Cambridge University Press, 1986.

[52] Walter Daelemans. Memory-based Lexical Acquisition and Processing. In *Lecture Notes in Artificial Intelligence: Machine Translation and the Lexicon.* 1994.

[53] K. Dahlgren. The Autonomy of Shallow Lexical Knowlege. In J. Pustejovsky and S. Bergler, editors, *Lexical Semantics and Knowledge Representation : First SIGLEX Workshop*, chapter 18, pages 255–268. Springer-Verlag, 1992.

[54] H. S. Delugach. An Exploration Into Semantic Distance. In H. D. Pfeiffer and T. E. Nagle, editors, *Conceptual Structures: Theory and Implementation*, pages 119–124. Springer, Berlin, Heidelberg, 1993.

[55] René Dirven. Dividing up physical and mental space into conceptual categories by means of English prepositions. In C. Zelinsky-Wibbelt, editor, *The Semantics*

of Prepositions: From Mental Processing to Natural Language Processing, pages 73–97. Mouton de Gruyter, 1993.

[56] W. Dolan. Word sense ambiguation: clustering related senses. In *Proc. of the 15 th COLING*, pages 712–716, Kyoto, Japan, 1994.

[57] W. Dolan and S. Richardson. Not for its own sake: knowledge as a byproduct of natural language processing. In *AAAI-96 Fall Symposium: Knowledge Representation Systems based on Natural Language*, pages 11–19, 1996.

[58] W. Dolan, L. Vanderwende, and S. D. Richardson. Automatically deriving structured knowledge bases from on-line dictionaries. In *The First Conference of the Pacific Association for Computational Linguistics*, pages 5–14, Harbour Center, Campus of SFU, Vancouver, April 1993.

[59] M. Evens, B. Litowitz, J. Markowitz, R. Smith, and O. Werner. *Lexical-semantic relations: a comparative survey.* Linguistic Research, Edmonton, Alberta, 1980.

[60] Andrew Fall. Spanning Tree Representations of Graphs and Orders in Conceptual Graphs. In *Proceedings of the ICCS'95 Conference; Conceptual Structures: Applications, Implementation and Theory*, pages 232–246. Springer-Verlag, 1995.

[61] J. Fargues, M-C. Landau, A. Dugourd, and L. Catach. Conceptual Graphs for Semantics and Knowledge Processing. *IBM Journal of Research and Development*, 30(1):70–79, 1986.

[62] Dan Fass. Lexical Semantic Constraints. In J. Pustejovsky, editor, *Semantics and the lexicon*, chapter 13, pages 263–289. Kluwer Academic Publishers, 1993.

[63] C.J. Fillmore. The case for case. In *Universals in Linguistic Theory.* Holt, Rinehart and Winston, Chicago, 1968.

[64] N. Foo, B.J. Garner, A. Rao, and E. Tsui. Semantic Distance in Conceptual Graphs. In T.E. Nagle, J.A. Nagle, L.L. Gerholz, and P.W.Eklund, editors,

Conceptual Structures: Current Research and Practice, chapter 7, pages 149–154. Ellis Horwood, 1992.

[65] E. Fox, T. Nutter, T. Ahlswede, M. Evens, and J. Markowitz. Building a Large Thesaurus for Information Retrieval. In *Proceedings of the 2nd ACL Conference on Applied Natural Language Processing*, 1987.

[66] G. Gazdar and C. Mellish. *Natural Language Processing in PROLOG*, chapter 6. Addison-Wesley Publishing Company, 1989.

[67] H.P. Grice. Logic and conversation. In P. Cole and J.L. Morgan, editors, *Syntax and Semantics 3*, pages 41–58. 1975.

[68] R. Grishman and R. Kittredge. *Analyzing Language in Restricted Domains: Sublanguage Description and Processing.* Lawrence Elbaum Associates, Hillsdale, NJ, 1986.

[69] R. Grishman and J. Sterling. The Acquisition of Selectional Patterns. In *Proc. of the 14 th COLING*, pages 658–665, Nantes, France, 1992.

[70] R.V. Guha and D.B. Lenat. Enabling Agents to Work Together. *Communications of the ACM*, 37(7):127–142, 1994.

[71] J. Guthrie, L. Guthrie, Y. Wilks, and H. Aidinejad. Subject-Dependent Co-Occurrence and Word Sense Disambiguation. In *Proc. of the 29 th ACL*, pages 146–152, 1991.

[72] L. Guthrie, B.M. Slator, Y. Wilks, and R. Bruce. Is there content in empty heads? In *Proc. of the 13 th COLING*, volume 3, pages 138–143, Helsinki, Finland, 1990.

[73] Ollivier Haemmerlé. *CoGITo: une plate-forme de développement de logiciels sur les graphes conceptuels.* PhD thesis, Université Montpellier II, France, janvier 1995.

[74] Ollivier Haemmerlé. Implementation of Multi-Agent Systems using Conceptual Graphs for Knowledge and Message Representation: the CoGITo Platform. In G. Ellis, R. Levinson, W. Rich, and J. Sowa, editors, *Proceedings Supplement, ICCS'95: Conceptual Structures: Applications, Implementation and Theory*, pages 13–24. Springer-Verlag, 1995.

[75] Marti A. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proc. of the 14 th COLING*, pages 539–545, Nantes, France, 1992.

[76] D. Hindle. Noun Classification from Predicate Argument Structures. In *Proc. of the 28 th ACL*, pages 268–275, 1990.

[77] D. Hindle and M. Rooth. Structural Ambiguity and Lexical Relations. In *Proc. of the 29 th ACL*, pages 229–236, 1991.

[78] H. Hirakawa, Z. Xu, and K. Haase. Inherited Feature-based Similarity Measure Based on Large Semantic Hierarchy and Large Text Corpus. In *Proc. of the 16 th COLING*, pages 508–513, Copenhagen, Denmark, 1996.

[79] Graeme Hirst. Resolving Lexical Ambiguity Computationally with Spreading Activation and Polaroid Words. In S.I. Small, G.W. Cottrell, and M.K. Tanenhaus, editors, *Lexical Ambiguity Resolution*, chapter 3, pages 73–107. Morgan Kaufmann, 1988. COPIE, A LIRE.

[80] J. Hobbs, W. Croft, T. Davies, D. Edwards, and K. Laws. Commonsense Metaphysics and Lexical Semantics. *Computational Linguistics*, 13(3-4):241–250, 1987.

[81] J.R. Hobbs and J. Bear. Two Principles of Parse Preference. In *Proc. of the 13 th COLING*, volume 3, pages 162–167, Helsinki, Finland, 1990.

[82] N. Ide and . Veronis. Machine Readable Dictionaries: What have we learned, where do we go? In *Proceedings of the Post-Coling94 International Workshop on directions of lexical research*, pages 137–146, Bejing, China, 1994.

[83] K. Jensen and J.-L. Binot. Disambiguating Prepositional Phrase Attachments by Using On-Line Dictionary Definitions. *Computational Linguistics*, 13:251–260, 1987.

[84] Judith Klavans, R. Byrd, N. Wacholder, and M. Chodorow. Taxonomy and Polysemy. Technical Report RC 16443 (#73060), IBM T.J. Watson Research Center, 1991.

[85] Judith Klavans, M. S. Chodorow, and N. Wacholder. From Dictionary to Knowledge Base via Taxonomy. In *Proceedings of the 6th Annual Conference of the UW Centre for the New OED: Electronic Text Research*, pages 110–132, 1990.

[86] Hideki Kozima. Text segmentation based on similarity between words. In *Proc. of the 31 th ACL*, pages 286–288, 1993.

[87] Hideki Kozima and Teiji Furugori. Similarity between Words Computed by Spreading Activation on an English Dictionary. In *Proc. of the 6 th EACL*, pages 232–239, 1993.

[88] Marie-Claude Landau. Solving Ambiguities in the Semantic Representation of Texts. In *Tenth International Workshop on Expert Systems and Their Applications*, pages 119–130, 1990.

[89] Adrienne Lehrer. *Semantic fields and Lexical structure.* North-Holland Publishing, 1974.

[90] Michael Lesk. Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone? In *Proceedings of the 1986 ACM SIGDOC Conference*, pages 24–27, 1987.

[91] R. Levinson and G. Ellis. Multilevel hierarchical retrieval. *Knowledge-Based Systems*, 5(3):233–244, 1992.

[92] J. Markowitz, T. Ahlswede, and M. Evens. Semantically Significant Patterns in Dictionary Definitions. In *Proceedings of ACL-24*, pages 112–119, 1986.

[93] Judith Markowitz. An exploration into graded set membership. In M. Evens, editor, *Relational Models of the Lexicons*, chapter 11, pages 239–260. Cambridge University Press, 1988.

[94] Robert Mercer. Presuppositions and Default Reasoning: A Study of Lexical Pragmatics. In J. Pustejovsky and S. Bergler, editors, *Lexical Semantics and Knowledge Representation : First SIGLEX Workshop*, chapter 22, pages 321–339. Springer-Verlag, 1992.

[95] Andrei Mikheev. Learning Part-of-Speech Guessing Rules from Lexicon: Extension to Non-Concatenative Operations. In *Proc. of the 16 th COLING*, pages 770–775, Copenhagen, Denmark, 1996.

[96] George A. Miller. WordNet: An On-line Lexical Database. *International Journal of Lexicography*, 3(4), 1990.

[97] George A. Miller. Nouns in WordNet: A Lexical Inheritance System, 1993. Article from WordNet repository, clarity.princeton.edu.

[98] Guy W. Mineau. Induction on Conceptual Graphs: Finding Common Generalizations and Compatible Projections. In T.E. Nagle, J.A. Nagle, L.L. Gerholz, and P.W.Eklund, editors, *Conceptual Structures: Current Research and Practice*, chapter 14, pages 295–310. Ellis Horwood, 1992.

[99] Guy W. Mineau and Mourad Allouche. The Establishment of a Semantical Basis: Toward the Integration of Vocabularies. In *Proceedings of KAW'95*, Banff, Canada, 1995.

[100] H. Miyoshi, K. Sugiyama, M. Kobayashi, and T. Ogino. An Overview of the EDR Electronic Dictionary and the Current Status of its Utilization. In *Proc. of the 16 th COLING*, pages 1090–1093, Copenhagen, Denmark, August 1996.

[101] S. Montemagni and L. Vanderwende. Structural Patterns vs. String Patterns for Extracting Semantic Information from Dictionaries. In *Proc. of the 14 th COLING*, pages 546–552, Nantes, France, 1992.

[102] Robert C. Moore. The role of logic in knowledge representation and commonsense reasoning. In *Readings in Knowledge Representation*, pages 336–341. Morgan Kaufmann Publishers, 1985.

[103] J. Nakamura and M. Nagao. Extraction of Semantic Information from an Ordinary English Dictionary and its Evaluation. In *Proc. of the 12 th COLING*, pages 459–464, Budapest, Hungary, 1988.

[104] N. Nicolov, C. Mellish, and G. Ritchie. Sentence Generation from Conceptual Graphs. In G. Ellis, R. Levinson, W. Rich, and J. Sowa, editors, *ICCS'95: Conceptual Structures: Applications, Implementation and Theory*, pages 74–88. Springer-Verlag, 1995.

[105] Jean-Francois Nogier and Michael Zock. Lexical Choice as Pattern Matching. In T.E. Nagle, J.A. Nagle, L.L. Gerholz, and P.W.Eklund, editors, *Conceptual Structures: Current Research and Practice*, chapter 21, pages 414–435. Ellis Horwood, 1992.

[106] J.T. Nutter, E. Fox, and M. Evens. Building a lexicon from machine-readable dictionaries for improved information retrieval. *Literary and Linguistic Computing*, 5(2):129–137, 1990.

[107] N. Ostler and B.T.S. Atkins. Predictable Meaning Shift: Some Linguistic Properties of Lexical Implication Rules. In J. Pustejovsky and S. Bergler, editors, *Lexical Semantics and Knowledge Representation : First SIGLEX Workshop*, chapter 7, pages 87–100. Springer-Verlag, 1992.

[108] F. Pereira, N. Tishby, and L. Lee. Distributional Clustering of English Words. In *Proc. of the 33 th ACL*, Cambridge,MA, 1995.

[109] John G. Proakis. *Digital communications.* McGraw-Hill Book Company, 1989.

[110] J. Pustejovsky. The generative lexicon. *Computational Linguistics*, 17(4):409–441, 1991.

[111] J. Pustejovsky and S. Bergler. *Lexical Semantics and Knowledge Representation : First SIGLEX Workshop.* Springer-Verlag, 1992.

[112] J. Pustejovsky and B. Boguraev. Lexical Knowledge Representation and Natural Language Processing. *Artificial Intelligence*, 63:193–223, 1993.

[113] M. Quillian. Semantic memory. In M. Minsky, editor, *Semantic Information Processing.* MIT Press, Cambridge, MA, 1968.

[114] Han Reichgelt. *Knowledge Representation: an AI perspective.* Ablex Plublishing Corporation, 1991.

[115] Philip Resnik. Disambiguating Noun Groupings with Respect to WordNet Senses. In *Third Workshop on Very Large Corpora*, 1995.

[116] Philip Resnik. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *Proc. of the 14 th IJCAI*, volume 1, pages 448–453, Montreal, Canada, 1995.

[117] Francesc Ribas. On Learning more Appropriate Selectional Restrictions. In *Proc. of the 7 th EACL*, pages 112–118, Dublin, Ireland, 1995.

[118] S. Richardson. *Determining Similarity and Inferring Relations in a LKB.* PhD thesis, The City University of NY, 1997.

[119] Gerard Sabah and Anne Vilnat. A Hierarchy of Relational Types in Conceptual Graphs to Handle Natural Language Parsing. In *Proceedings of ICCS'93*, pages 198–215, 1993.

[120] G. Salton, J. Allan, and C. Buckley. Automatic structuring and retrieval of large text files. *Communications of the ACM*, 37(2):97–108, 1994.

[121] R. Schank and R. Abelson. Scripts, plans and knowledge. In *Advance papers 4th Intl. Joint Conf. Artificial Intelligence*, 1975.

[122] Roger C. Schank. Language and Memory. In B.J. Grosz, K.S. Jones, and B.L. Webber, editors, *Readings in Natural Language Processing*, pages 171–191. Morgan Kaufmann, 1986.

[123] M. Schoelles and H. Hamburger. The NLP Role in Animated Conversation for CALL. In *Proc. of the 5 th Conference on Applied Natural Language Processing*, pages 127–134, Washington, DC, USA, March 1997.

[124] Stuart C. Shapiro. The CASSIE Projects: An Approach to Natural Language Competence. In J.P. Martins and E.M. Morgado, editors, *EPIA '89 : 4th Portuguese Conference on Artificial Intelligence*, pages 362–380. Springer-Verlag, 1989.

[125] Frank Smadja. Macrocoding the Lexicon with Co-occurence Knowledge. In U. Zernik, editor, *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, chapter 7, pages 165–189. Lawrence Elbaum Associates, 1991.

[126] J. Sowa. *Conceptual Structures in Mind and Machines.* Addison-Wesley, 1984.

[127] J.F. Sowa. *Principles of Semantic Networks: Exploration in the Representation of Knowledge.* Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[128] John F. Sowa. Conceptual Graphs Summary. In T.E. Nagle, J.A. Nagle, L.L. Gerholz, and P.W.Eklund, editors, *Conceptual Structures: Current Research and Practice*, chapter 1, pages 3–51. Ellis Horwood, 1992.

[129] John F. Sowa. Semantic Networks. In S.C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 1493–1511. Wiley, New York, 1992.

[130] John F. Sowa. Relating Diagrams to Logic. In G. W. Mineau, B. Moulin, and J. F. Sowa, editors, *Conceptual Graphs for Knowledge Representation: Proc. of the First International Conference on Conceptual Structures ICCS'93*, pages 1–35. Springer, Berlin, Heidelberg, 1993.

[131] Lucy Vanderwende. Ambiguity in the Acquisition of Lexical Information. In *AAAI Symposium - Representation and Acquisition of Lexical Knowledge: Polysemy, Ambiguity and Generativity*, pages 174–179. AAAI Press, 1995.

[132] P. Velardi and M.T. Pazienza. Computer aided interpretation of lexical cooccurrences. In *Proc. of the 27 th ACL*, pages 185–192, Vancouver, Canada, 1989.

[133] P. Velardi, M.T. Pazienza, and M. De Giovanetti. Conceptual Graphs for the Analysis and Generation of Sentences. *IBM Journal of Research and Development, special issue on language processing*, 32(2):251–267, 1988.

[134] Paola Velardi. Acquiring a Semantic Lexicon for Natural Language Processing. In U. Zernik, editor, *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, chapter 13, pages 341–368. Lawrence Elbaum Associates, 1991.

[135] J. Veronis and N.M. Ide. Word Sense Disambiguation with Very Large Neural Networks Extracted from Machine Readable Dictionaries. In *Proc. of the 13 th COLING*, volume 2, pages 389–394, Helsinki, Finland, 1990.

[136] P. Vossen. The automatic construction of a knowledge base from dictionaries: a combination of techniques. In *EURALEX'92: International Congress on Lexicography*, pages 311–326, Finland, 1992.

[137] P. Vossen. Right or Wrong: Combining Lexical Resources in the EuroWordNet Project. In *EURALEX'96: International Congress on Lexicography*, pages 715–728, Goteborg, Sweden, 1996.

[138] P. Vossen and A. Copestake. Untangling Definition Structure into Knowledge Representation. In *Inheritance, Defaults and the Lexicon*, chapter 13, pages 246–274. Cambridge University Press, 1993.

[139] P. Vossen, W. Meijs, and M. den Broeder. Meaning and structure in dictionary definitions. In B. Boguraev and T. Briscoe, editors, *Computational Lexicography for Natural Language Processing*, chapter 8, pages 171–192. Longman Group UK Limited, 1989.

[140] M. Webster and M. Marcus. Automatic Acquisition of Lexical Semantics of Verbs from Sentence Frames. In *Proc. of the 27 th ACL*, pages 177–184, Vancouver, Canada, 1989.

[141] Y. Wilks, D. Fass, C-M Guo, J.E. McDonald, T. Plate, and B.M. Slator. Providing Machine Tractable dictionary tools. In J. Pustejovsky, editor, *Semantics and the lexicon*, chapter 16, pages 341–401. Kluwer Academic Publishers, 1993.

[142] Y. Wilks, D. Fass, G-M Guo, J. McDonald, T. Plate, and B. Slator. A tractable machine dictionary as a resource for computational semantics. In Bran Boguraev and Ted Briscoe, editors, *Computational Lexicography for Natural Language Processing*, chapter 9, pages 193–231. Longman Group UK Limited, 1989.

[143] R. Wille. Concept lattices and conceptual knowledge systems. In F. Lehmann, editor, *Semantic networks in artificial intelligence*, pages 493–516. 1992.

[144] W. A. Woods. What's in a link: Foundations for semantic networks. In D. Bobrow and A. Collins, editors, *Representation and Understanding.* Academic Press, New York, 1975.

[145] David Yarowsky. Unsupervised Word Sense Disambiguation rivaling Supervised Methods. In *Proc. of the 33 th ACL*, pages 189–196, Cambridge,MA, 1995.

[146] U. Zernik. *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon.* Lawrence Elbaum Associates, 1991.

[147] Uri Zernik. Train1 vs. Train2: Tagging Word Senses in Corpus. In U. Zernik, editor, *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, chapter 5, pages 91–112. Lawrence Elbaum Associates, 1991.