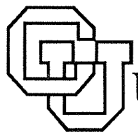
MetaBank: A Knowledge-Base of Metaphoric Language Conventions

James H. Martin

CU-CS-526-91 May 1991



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

MetaBank: A Knowledge-Base of Metaphoric Language Conventions

James H. Martin
Computer Science Department and
Institute of Cognitive Science
University of Colorado,
Boulder, CO
80309-0430
martin@cs.colorado.edu

CU-CS-526-91 May 1991

Abstract

The frequent and conventional use of non–literal language has been a major stumbling block for natural language processing systems since the early machine translation efforts. Metaphor, metonymy and indirect speech acts are among the most troublesome phenomena. Recent computational efforts addressing these problems have taken an approach that emphasizes the use of systematic knowledge about non–literal language conventions. This paper describes our current efforts to supply this knowledge in the case of conventional metaphor. We are constructing *MetaBank: an empirically derived and theoretically motivated knowledge–base of English metaphorical conventions.* More generally, this effort can be seen as an attempt to develop methodologies for empirically capturing language conventions in usable knowledge–base forms.

1 Introduction

The frequent and conventional use of non–literal language has been a major stumbling block for natural language processing systems since the early machine translation efforts. Metaphor, metonymy and indirect speech acts are among the most troublesome phenomena. This is both because they occur quite frequently and because they have resisted computational efforts to deal with them in a uniform and tractable manner.

Consider the following examples of conventional metaphor.

(1) It *came* to me that I had to prepare a talk for the conference.
(2) It *hit* me that I didn't have anything to say.
(3) It *struck* me that this wasn't a good situation.

The italicized words in these examples are being used in ways that clearly deviate from the physical or spatial meaning of the words. Nevertheless, there is little that is novel or unconventional in these examples. In order to understand and generate uses like these, we make use of a rich set of underlying metaphorical conventions. In these examples, these conventions structure beliefs and believers as objects with locations. Correspondingly, change of location indicates a change in belief, as in (1) where the new idea "comes to" the believer.

The main thrust of this *Metaphoric Knowledge* approach to metaphor is that the interpretation of metaphoric language should proceed through the direct application of specific knowledge about the metaphors in the language. This approach has been embodied in MIDAS (Metaphor Interpretation, Denotation, and Acquisition System)[15, 16]. MIDAS is a set of computer programs that can be used to perform the following tasks: explicitly represent knowledge about conventional metaphors, apply this knowledge to interpret metaphoric language, and learn new metaphors as they are encountered.

While the MIDAS project demonstrated some significant results, it nevertheless had a major shortcoming. The effectiveness of the approach for language interpretation, generation and acquisition was obviously dependent on the size and the correctness of the knowledge–base of non-literal conventions. Unfortunately, the knowledge–base used by MIDAS does not have any kind of real coverage, nor does it have an empirically verifiable basis.

This paper describes our current efforts to overcome these problems in the case of conventional metaphor. We are constructing of *MetaBank: an empirically derived and theoretically motivated knowledge–base of English metaphorical conventions.* More generally, this effort can be seen as an attempt to develop methodologies for empirically capturing a wider variety of language conventions in computationally usable knowledge–base forms.

2 Knowledge-Based Linguistic Approaches

MIDAS can be seen as a part of a recent trend in the area of semantic interpretation that uses systematic knowledge about non–literal language conventions. Among the research projects following this general approach are Zernik's RINA system (phrasal idioms) [25],

Fass's META5 and MET* systems (metaphor and metonymy) [4], and Hinkelman's system (indirect speech acts) [6]. These systems all attempt to leverage knowledge of systematic language conventions in an attempt to avoid resorting to more computationally expensive methods. At the same time, however, these approaches have all avoided a simplistic phrasal or word-sense approach that neither adequately captures generalizations nor allows for novel cases. Finally, like MIDAS they all use small knowledge–bases of conventions with no real coverage and little empirical validation. (Hinkleman's corpora-based research is a notable exception to this). The following discussion will present the details of how this general approach has been applied to the problem of conventional metaphor.

Under this knowledge–based approach, the proper way to approach the study of metaphor is to study the underlying details of individual metaphors, and systems of metaphors in the language. This approach follows on the metaphor work of Lakoff and Johnson [10] and builds directly on the computational approaches to metaphor begun in [9, 13, 14, 19]. The earlier efforts of Wilks [24], Hobbs [7], and Carbonell [2] are the most relevant computational predecessors to this work.

In order to make the problem of understanding metaphors more concrete, consider the implications of the following examples for a system like the UNIX Consultant [22, 23]. UC is a natural language consultant system that provides naive computer users with advice on how to use the Unix operating system. Metaphors like those shown above are ubiquitous in technical domains like UNIX. A system that is going to accept natural language input from users and provide appropriate natural language advice must be prepared to handle such metaphorical language. MIDAS has been integrated into UC in order to give it the ability to handle this kind of metaphoric language. Perhaps more importantly, Unix offered an ideal domain, rich in metaphors, in which MIDAS could be tested.

Consider the following UC session illustrating the processing of a series of user queries.

```
#  How can I kill a process?
  Applying metaphor:  Terminating-As-Killing
You can kill a process by typing ^ C to the shell.

#  Tell me how to get out of emacs.
  Applying metaphor:  Uninvoking-As-Exiting
You can get out of emacs by typing ^ X^ C.

#  Do you know how to enter lisp?
  Applying metaphor:  Invoking-As-Entering
You can enter lisp by typing ``lisp'' to the shell.
```

In each of these examples, UC/MIDAS attempts to find the most coherent interpretation of the user's question, given its current knowledge of the conventions of the language. This involves checking the structure of the input against the constraints posed by all the possible conventional metaphorical and non-metaphorical interpretations. In each of these examples,

the only coherent interpretation is the one found through the application of a known Unix metaphor.

3 MIDAS

Since the remainder of this paper focuses on our proposed research for building a large realistic knowledge–base of metaphors, this section will first briefly illustrate how such a knowledge–base could be used. This is not intended as an exhaustive list since the knowledge–base certainly will have many unanticipated uses. Specifically, this section describes how the MIDAS system, as a part of UC, made practical computational use of the metaphors stored in its knowledge–base. In particular, it introduces the following three issues.

Representation: The explicit representation of the conventional metaphors in a language in the form of explicit associations between concepts.

Interpretation: The correct and efficient application of the this metaphoric knowledge to the interpretation of metaphoric language.

Learning: The dynamic acquisition of new knowledge about metaphors for which no known metaphor provides a coherent explanation.

3.1 Knowledge Representation

Consider the following simple example of a conventional UNIX metaphor. The metaphorical use of the word *in* reflects a systematic metaphorical structuring of Unix processes as enclosures.

(6) I am *in* Emacs.

Metaphors like this may be said to consist of the following component concepts: a *source* component, a *target* component, and a set of conventional associations from the source to target. The target consists of the concepts to which the words are actually referring. The source refers to the concepts in terms of which the intended target concepts are being viewed. In this example, the target concepts are those representing the state of currently using a computer process. The source concepts are those that involve the state of being contained within some enclosure.

The approach taken here is to explicitly represent conventional metaphors as sets of associations between source and target concepts. The metaphor specifies how the source concepts reflected in the surface language correspond to various target concepts. In this case, the metaphor consists of component associations that specify that the state of being enclosed represents the idea of currently using the editor, where the user plays the role of the enclosed thing, and the Emacs process plays the role of the enclosure. Note that these source-target associations are represented at the conceptual and not the lexical level. Any single lexical item or expression that can be construed as referring the source concept of a known metaphor may invoke that metaphor. In this example, the source component of the metaphor is attached to the concept of being enclosed, not to the lexical item *in*.

These sets of metaphoric associations, along with the concepts that comprise the source and target domains, are represented using the KODIAK [21] representation language. KODIAK is an extended semantic network language in the tradition of KL-ONE [1] and its variants. These sets of metaphoric associations are full-fledged KODIAK concepts. As such, they can be related to other concepts and arranged in abstraction hierarchies using the inheritance mechanisms provided by KODIAK. Specifically, KODIAK is used to represent specialized domain specific metaphors, pervasive high-level metaphors, and the systems of relations among related metaphors.

3.2 Interpretation

The interpretation process in MIDAS is basically one that views a given input sentence as providing a set of constraints on possible interpretations. MIDAS checks the input constraints against all the possible interpretations that can be conventionally associated with the input. Interpretations that are coherent with the constraints are returned. The possible conventional associations include direct non-metaphoric interpretations, as well as all the conventional metaphors that are invoked by the input.

Consider briefly the details of the following shortened trace of a Unix example. In this example, UC calls upon MIDAS to find a coherent interpretation for this use of *enter*. MIDAS finds, and attempts to apply, all the conventional metaphorical and non-metaphorical concepts associated directly with, or inherited by, this concept. In this case, it finds that the only conventional interpretation that is consistent with the input is the one that results from the application of the known Enter-Lisp metaphor.

```
> (do-sentence)
Interpreting sentence:

How can I enter lisp?

Interpreting concreted input.

(A Entering50 (↑ Entering)
   (enterer50 (↑ enterer) (A I203 (↑ I)))
   (entered50 (↑ entered) (A Lisp58 (↑ Lisp))))

Failed interpretation:  Entering50 as Entering.
Failed interpretation:  Entering50 as Enter-Association.
Valid known metaphorical interpretation:  Entering50 as Enter-Lisp.
```

The parser first produces a syntactic analysis and a preliminary semantic representation. At this point in the analysis, UC calls upon MIDAS to begin a deeper analysis of this initial representation.

The case structure of this preliminary representation is then checked against the semantic constraints of the interpretations conventionally associated with the Entering concept.

In this case, MIDAS finds that the direct interpretation and one of the other known Entering metaphors can be rejected before the appropriate Enter-Lisp metaphor is found.

```
Applying conventional metaphor Enter-Lisp.

(A Enter-Lisp  (↑ Container-Metaphor Metaphor-Schema)
   (enter-lisp-res enter-res → lisp-invoke-result)
   (lisp-enterer enterer → lisp-invoker)
   (entered-lisp entered → lisp-invoked)
   (enter-lisp-map Entering → Invoke-Lisp))

Mapping input concept Entering50 to concept Invoke-Lisp30
  Mapping input role enterer50 with filler I203 to
               target role lisp-invoker30
  Mapping input role entered50 with filler Lisp58 to
               target role lisp-invoked30

Yielding interpretation:

(A Invoke-Lisp30 (↑ Invoke-Lisp)
   (lisp-invoked30 (↑ lisp-invoked)
    (A Lisp58 (↑ Lisp)))
   (lisp-invoker30 (↑ lisp-invoker)
    (A I203 (↑ I))))
```

MIDAS then begins the process of mapping from the given source concepts to the appropriate target concepts based on the constraints imposed by the metaphor. The mapping process, called *metaphoric unviewing*, creates a new instance of the target concept for use in further processing. Any further inferences that need to be performed are based on this newly created target concept. In this example, the source concept of **Entering** is mapped to the target concept **Invoke-Lisp** as specified by the metaphor.

```
Calling UC on input:

(A How-Q207 (↑ How-Q)
   (topic206 (↑ topic)
    (A Invoke-Lisp30 (↑ Invoke-Lisp)
       (lisp-invoked30 (↑ lisp-invoked) (A Lisp58 (↑ Lisp)))
       (lisp-invoker30 (↑ lisp-invoker) (A I203 (↑ I))))))

UC: You can enter lisp by typing 'lisp' to the shell.
```

Finally, UC uses this new target concept as the basis for answering the user's question by using its long-term knowledge about how to initiate the Lisp system. Note that UC makes use of the metaphor in expressing its answer to the user.

3.3 Learning

MIDAS will inevitably face the situation where a metaphor is encountered for which none of its known metaphors provides an adequate explanation. This situation may result from the existence of a gap in the system's knowledge-base of conventional metaphors, or from an encounter with a novel metaphor. In either case, the system must be prepared to handle the situation. Consider the following example.

In this example, the user has employed the conventional UNIX metaphor that the termination of an ongoing process can be viewed as a killing. However, unlike the previous example, MIDAS finds that it is initially unable to interpret this example because it has no knowledge of this conventional metaphor. More precisely, it determines that the given input can not adequately satisfy the constraints associated with any of the concepts conventionally associated with the word *kill*.

```
> (do-sentence)
Interpreting sentence:

How can I kill a process?

Interpreting concreted input.

(A Killing16 (↑ Killing)
   (killer16 (↑ killer) (A I46 (↑ I)))
   (kill-victim16 (↑ kill-victim)
    (A Computer-Process10 (↑ Computer-Process))))

Failed interpretation:  Killing16 as Killing.
Failed interpretation:  Killing16 as Kill-Delete-Line.
Failed interpretation:  Killing16 as Kill-Sports-Defeat.
Failed interpretation:  Killing16 as Kill-Conversation.

No valid interpretations.
```

At this point, MIDAS has exhausted all the possible conventional interpretations of the primal representation. In particular, the direct non-metaphoric interpretation and three known metaphorical interpretations are rejected because their restrictions of the role of the `kill-victim` fail to match the semantics of the concept filling that role in the input, a `computer-process`.

```
============================================================
Entering Metaphor Extension System
============================================================

Attempting to extend existing metaphor.

Selecting metaphor Kill-Conversation to extend.

Attempting a similarity extension inference.

Creating new metaphor:  Killing-Terminate-Computer-Process

(A Killing-Terminate-Computer-Process (↑ Kill-Metaphor)
   (kill-victim-c-proc-termed-map kill-victim → c-proc-termed)
   (killer-c-proc-termer-map killer → c-proc-termer)
   (killing-terminate-computer-process-map Killing
    → Terminate-Computer-Process))
```

This example illustrates the operation of the learning component of MIDAS, the Metaphor Extension System (MES). This system is invoked by MIDAS when it discovers a metaphor for which it has no adequate knowledge. The task of the MES is to attempt to extend its knowledge of some existing metaphor in a way that will yield a coherent interpretation for the new use and provide a basis for directly understanding similar uses in future. In this case, the system finds and extends a known closely related metaphor that also uses *kill* to mean terminate.

```
Final interpretation of input:

(A How-Q46 (↑ How-Q)
   (topic46 (↑ topic)
    (A Terminate-Computer-Process10
       (↑ Terminate-Computer-Process)
       (c-proc-termer10 (↑ c-proc-termer)
(A I46 (↑ I)))
       (c-proc-termed10 (↑ c-proc-termed)
(A Computer-Process10
   (↑ Computer-Process))))))

UC: You can  kill a computer process by typing ^c to the shell.
```

Finally, the target concept determined by the MES is used to provide an answer to the user.

The approach taken in MIDAS to the understanding of new or unknown metaphors is called the Metaphor Extension Approach. The basic thrust of this approach is that a new metaphor can best be understood by extending an existing well-understood metaphor or combining several known metaphors in a systematic fashion. Under this approach, the ability to understand and learn new metaphors depends critically on the availability of systematic knowledge about existing known metaphors.

4 Limitations to the Knowledge–Based Paradigm

As mentioned in section 1 the primary current limitation to systems following a knowledge–based approach to non–literal phenomena is the size and correctness of the knowledge–base itself. Consider the ways that this general problem manifests itself in MIDAS.

- There is no way of telling if the known metaphors that are included are represented correctly.
- There is no way of telling if a given metaphor is of high enough frequency and range to warrant significant effort.
- MIDAS lacks any empirical basis for both the abstract domain independent metaphors, and the domain specific UNIX metaphors, that are included in its knowledge-base.
- There is no way of determining the coverage of the current knowledge–base. How many conventional metaphors are really out there?
- There is no empirical basis for the particular hierarchical relationships embodied in the current knowledge–base.

Of course, these problems of knowledge–base size, correctness and organization do not differentiate these natural language processing efforts from many other knowledge–based AI efforts. However, within the field of natural language processing, significant progress has recently been made in the construction of large lexicons and larger more realistic grammars. The research we are engaged in continues this trend.

5 MetaBank

The MetaBank project is an attempt to solve some of the above problems for the case of conventional metaphor and metonymy. To be specific, we are identifying resources and developing methodologies that will allow us to construct a robust knowledge–base of English metaphoric and metonymic conventions. This knowledge–base will be represented in a form that will make it generally useful to natural language processing applications. We will also show that it has direct relevance to various projects involved in the construction of what have been called large common-sense knowledge–bases.

Considered more generally, this effort is an attempt to develop a set of tools and methodologies that will enable us to derive useful generalizations about the non–literal and non–compositional conventions of a language from on–line resources. Seen in this light, this effort can be seen as a parallel to current efforts to build large lexicons using similar tools and resources.

The following sections describe the three–part approach we are following for the construction of MetaBank. The first part involves the collection of on–line textual resources and databases of linguistic generalizations. The second part is the development of a methodology for analyzing these resources and deriving generalizations from them. The final part is the actual construction and use of a knowledge–base based on the preceding analyses.

5.1 Resources

We are planning to make use of the following on–line resources as a part of this effort.

1. Berkeley Metaphor List: An on–line list of known metaphors with analyses.
2. Domain Specific Corpora: A variety of UNIX specific texts.
3. General Text Corpora: Selected texts from the Association for Computational Linguistics Data Collection Initiative.

The following sections describe our efforts to make use of these resources.

5.1.1 Berkeley Metaphor List

The Metaphor Project, at the University of California at Berkeley, under the direction of George Lakoff, has been collecting an on-line database of conventional English metaphors. The database entries for each metaphor include descriptions of the source and target domains, a set of example sentences using the metaphor, a short analysis, and where appropriate, pointers to a longer analysis from which the entry was derived. The entries are being compiled by hand from the published metaphor literature, and from an on-going series of graduate seminars at Berkeley. Currently, a total of approximately 50 metaphors of varying levels of specificity have been entered into the database.

This Berkeley collection, merged with the metaphors already represented and analyzed as a part of the MIDAS project, will be used as the starting point for MetaBank. As will be discussed, it will also be used to guide the analysis of the various corpora.

5.1.2 Corpora

We are making use of two kinds of text corpora in this project: text having to do with the UNIX domain, and more general text selected from the ACL/DCI. The focus in our initial efforts will be on the UNIX text. More generic text will mainly be used to verify and extend results gained from the domain specific text.

The primary practical reason for the use of specialized text from UNIX domain is that the main testbed for MetaBank will be natural language application programs that operate in this domain. It is generally agreed that NLP systems will for some time to come be most useful and robust in limited, specialized domains. This has in the past entailed the construction by hand of specialized lexicons. What we are seeking is a means by which generic lexicons can be productively applied to new domains via knowledge of generic English conventional metaphors. Therefore the particular UNIX and computer system metaphors discovered, are not the primary accomplishment. We are primarily interested in the mechanisms by which generic English metaphors are specialized into particular domains.

Within the UNIX domain, we are using two kinds of text that can be characterized as expert prose and user prose. We currently are using two sources of expert prose. The first source is the standard UNIX documentation known as the MAN pages. These are descriptions of the individual commands, subsystems, and internal system calls of UNIX. For our purposes, they are interesting in that they provide access to the language of a large number of authors who have had significant impacts on the domain, and they are read by almost all users of the system at some point. The second source of expert prose that we are using is the full text of a textbook called *UNIX System Administrators Handbook*. This text is intended for readers who will be primarily responsible for administering and maintaining UNIX system.

The first set of data containing users language about UNIX was obtained from the Computing Research Laboratory at New Mexico State University. A series of Wizard of OZ studies [17, 18] were conducted with users as a part of an effort to build a natural language consultant system for UNIX. [5] Users were told that their natural language queries were being handled by a program, while the queries were in fact actually handled by human experts. It, therefore, provides us with text containing both user and expert utterances.

The next, and largest, source of text for user's language are several archives of user mail to consultants for help in using UNIX. The first is an archive from the Computer Science Department at Berkeley that was used in the construction of the UNIX Consultant system. The second is from a similar ongoing effort being collected at the Computer Science Department at Boulder. In both cases, users having difficulty with the system are asked to mail to a trouble alias. Mail to this alias is monitored by systems personnel and dealt with. A total of approximately 150,000 words of text (both user queries and consultants answers) have been collected.

To illustrate the kind of text being collected, consider the following example from the Boulder corpus.

```
>From rieman@tigger  Wed Apr 11 17:27:13 1990
Received:  by anchor.colorado.edu (cu.generic.890828)
Received:  by eclipse.colorado.edu (cu.generic.890828)
Received:  by tigger.colorado.edu (cu.generic.890828)
Date:  Wed, 11 Apr 90 17:27:08 MDT
From:  John Rieman <rieman@tigger>
Message-Id:  <9004112327.AA17545@tigger.colorado.edu>
```

```
To:  trouble@eclipse
Subject:  locked terminal
Cc:  rieman@tigger

My terminal hanging off of eclipse tty06 is locked again.    I can see
the processes (login to tigger) on eclipse, but I can't kill 'em --
eclipse says they aren't mine.

For your info here's how this happened (it has happened before):

I typed ''remote'' at the eclipse login prompt and logged into tigger.    My
default terminal setting is vt100 -- that's what I use most of
the time.    I only use the zenith when I'm not doing anything
important or time-consuming.

I forgot to tell tigger I was using a zenith and I more'd a file.

The zenith locked.

I hit shift-reset to knock some sense into the zenith, then pressed
ctrl-z to get out of more.    I killed the more process.

I typed ''setenv TERM z89'' to tell tigger I was using a zenith.

Then I tried again to use more.    It locked again, and that's where I
am now.    Resetting the terminal has no effect, and killing my tigger
process from another machine has no effect.

-john
```

Consider some of the metaphorical conventions illustrated by this message. (Eclipse and tigger are the names of computers in the following).

I can't *kill* 'em -- eclipse *says* they aren't *mine*

The use of *kill* here reflects a metaphor that views processes as being alive, where termination corresponds to killing The use of *say* reflects the metaphor that views computers as communicative agents capable of carrying on dialogs with users. In this case, program output is viewed as dialog by the computer. Finally, the use of the possessive *mine* illustrates the process as possession of the user metaphor, that underlies many of the security mechanisms in computer systems.

...ctrl-z to *get out of* more. I *killed* the more process.

This sentence illustrates a use of the process as enclosure metaphor. It manifests itself here in the notion of exiting or *get out of* as suspension of an ongoing process. Note that in the immediately following sentence the user switches metaphors to refer to a different aspect of the same process.

...to *tell* tigger I was using a zenith...

This sentence is another instance of the communicative agent metaphor. Here, however, we see simple input to a program as a speaking to the computer.

...and thats *where I am* now...

This final example is an illustration of a general metaphor structuring states as locations. Note that this final metaphor illustrates an important point. These domain specific corpora will also contain uses of non-domain specific metaphors. Therefore, domain independent abstract metaphors will be derived from all the textual sources we collect, including the technical ones.

Preliminary study of this data indicates that unconstrained user language is quite metaphorical in nature. Moreover, users are much freer in their use of metaphor than are the authors of corresponding manuals and text books.

5.2 Metaphor Analysis

The text analysis methodology we are developing uses the Berkeley metaphor list as a generator of probes into the various text corpora. The following steps illustrate our current methodology.

Step 1: Choose a metaphor of interest from the current list of metaphors. This is the target metaphor.

Step 2: Generate a set of linguistic probes for this metaphor. This involves selecting a set of words associated with the source domain of the chosen metaphor. The selection of these words will be guided by the analysis of the metaphor provided by the Berkeley list to ensure that the words are likely to occur with the metaphor. These words will typically be chosen from a well–defined and well–studied semantic field from a spatial or physical domain.

Step 3: Choose a corpus of interest appropriate to the metaphor being studied. This will typically involve an alternation between the domain specific corpora and the more general corpora. The purpose of this alternation is to determine that the behavior of the metaphor in the domain specific corpora is consistent with its behavior in other domains and with its description in the Berkeley list.

Step 4: Probe the selected corpus for uses of these verbs. This step simply involves searching the corpus for instances of the probes.

Step 5: Classify the resulting sentences according to their meaning. This typically involves classifying the use according the the semantic properties of the arguments to the probe verb. This classification step will initially be dependent on the subjective ability of the human classifier to identify various uses. This step will result in one the following possibilities.

1. A literal use of the probe word.
2. An instance of the metaphor in question.
3. Another metaphor in the known metaphor list.

4. Another conventional metaphor not yet in the list.

5. An isolated homonymous word sense with no metaphoric basis.

6. A novel use of some kind.

The result of each probe is, therefore, a combination of information about the metaphor in question, and information about other uses of the probe words. Some of the information that can be gleaned about the target metaphor is:

- How frequently does this metaphor occur?
- How frequently are the probe words used with the target metaphor?
- How many of the probe words from the source semantic field actually occur with this metaphor.
- What is the correct level of abstraction for the source and target domains of the metaphor.

Some of the information that can be gotten from negative probe sentences (sentences not relevant to the target metaphor) is:

- Identification of conventional metaphors not in the current database.
- Frequency information about other metaphors.
- Frequency information about the probe word with other meanings.
- Identification of novel metaphors. (Metaphors that would not be judged as conventional).
- Frequency of occurrence of novel uses.

Note that this method of classification of probe words based on the text accompanying probe words, is analogous to the methods used in Choueka [3], Lesk [12] and Zernik [25] for word-sense tagging. In these efforts, local context (the words immediately surrounding the target word) was successfully used to classify a given use of a word in a text as an instance of a pre-determined dictionary sense. In this work, the role of the dictionary sense is replaced by a conventional metaphor. Also relevant is the corpus work done by Hinkelman in studying speech acts. In that work, words such as *please* and *hereby*, which can be used as performatives, were used as probes into a large corpus. The resulting text was then analyzed to determine the intended meaning of each occurrence.

This methodology can be seen as a combination of top-down and bottom-up techniques. On the top–down side it makes full use of the analytical linguistic work that has been done (as it is represented in the Berkeley database) to guide the search for metaphors in a text. The bottom-up side is the actual examination of large amounts of text. Note that methodology we are using allows us to probe the corpus in a focussed manner by using metaphors from the Berkeley list, while at the same time it allows us to discover metaphors not contained in the original list.

5.3 An Example: The Container Metaphor

In order to make this methodology more concrete, consider the following example analysis. In this example, the Boulder `trouble` mail archive is probed for instances of container metaphors. This is one of the more productive and well-studied kinds of metaphor in English. The Berkeley Metaphor list contains thirteen container metaphors (thirteen distinct target domains structured with the source concept container). In addition, significant effort was spent on computer system container metaphors as a part of the MIDAS project.

For the purposes of the this example, the probes *enter, get into, exit* and *get out of* from the source domain of containers or enclosures were used to probe the selected corpus. The following table shows the results.

choose "search" from the menu	enter "X", hit "apropos", and when the
BTW, when I try to login, it let's me	enter my password, then prints daveheib
Randomly in these two windows when I	entered the vi editor
I	enter my password when 'rlogin'g in from
prompt I have to be right there to	enter my passwd
Kermit tells me to	enter a Receive command and I do so.
I have a file called calendar that I	enter appointments
in. It seems that whenever I	enter an item in this file,
the appointment that I just	entered. It's not a huge
As soon as I	enter windows it fails (most of the time).
they	get into the queue, but they don't print
Oh, good grief! I can't	get into my home directory
log me into my home directory and I cannot	get into it by any other means.
a program that attempts to	get into computers around the Internet.
an intruder making persistent attempts to	get into Internet
I can then do a successfull :vi to	get into vi
hung up when I tried	exiting out of X windows. Keyboard, mouse, etc.
I tried to	exit, at which point I received the prompt
I was	exiting suntools when cashew choked in a
now hangs when I want to	exit and save using the VI editor
window, log into the machine. Now,	exit the login
i was on the console	exiting suntools with crtl-d crtl-q
pressed ctrl-z to	get out of more.
the only way I can	get out of it is to completely shut
would like to get my degree and	get out of here sometime

This probe yielded the following results from this probe:

1. All the probes occurred with the interactive–system as enclosure metaphor that was extensively studied with MIDAS. This was also the most frequent use.

2. A use of the metaphor with the MORE command which neither UC nor MIDAS had classified as an interactive–system.

3. A conventional use, structuring the home directory as an enclosure for the user, that was not known to MIDAS. It was used only with *get into.*

4. A conventional use of *enter* as in *to enter your passwd.* Again not known to MIDAS, and not discussed in the Berkeley list.

5. A relatively new use structuring programs as entering computers from the outside. As in *a program that attempts to get into computers around the Internet.* Mentioned in the Berkeley list, not known to MIDAS.

6. One generic use contained in the Berkeley list, *get my degree and get out of here*, that could be considered literally as leave, or metaphorically as graduate.

The net result of this was a strong confirmation of the Interactive-Process-As-Enclosure metaphor as currently known to MIDAS. It occurred quite frequently, with the characteristics predicted by the current representation. It also resulted in two conventional uses that have to be added to the metaphor list and added to MIDAS, and one new computer use that also needs to be added.

5.4 Knowledge–Base Construction

The first phase of construction involves integrating all of the metaphors from the Berkeley list into the current MIDAS knowledge–base. This involves representing the following kinds of knowledge.

- Knowledge about the source concepts of the metaphor. Particular care will be taken with the representation of the source concepts since it is assumed that they will occur across domains.

- Knowledge of the target concepts.

- Knowledge about the metaphors themselves. This information will largely come from the analyses of the metaphors from the Berkeley list.

The representation of this knowledge will entail significant effort since the Berkeley list does not give detailed information about the source and target domains. In this phase we intend to make extensive use of existing formalizations of physical and spatial domains. [8, 11].

This knowledge will be represented using the KODIAK [21] representation language. KODIAK is an extended semantic network language in the tradition of KL-ONE [1] and its

variants. Briefly, MIDAS currently uses a hierarchical knowledge–base of structured associations linking source and target concepts that are conventionally related metaphorically. The details of KODIAK and the representation of metaphoric knowledge can be found in [16]. It should be noted that the KODIAK representation is sufficiently similar to other extended semantic network systems to ensure the transportability of MetaBank to other systems.

As noted above the Berkeley list currently contains approximately 50 high level metaphors. The current MIDAS knowledge-base contains 22 domain independent metaphors (11 of which are contained in the Berkeley list) and an additional 18 UNIX specific specializations. We believe that a stable MetaBank for English will contain no more than 200 high level generic metaphors.

The second phase of construction consists of feeding the results of the text analysis work back into the knowledge–base. This involves refining, or altering completely, the representation of known metaphors to reflect the results of the analysis. Preliminary results indicate that three kinds of alterations to the representation are usually required.

- Semantic refinement of the source or target representations.
- Change in level of abstraction of the metaphor in either the source or target concepts.
- Inclusion of newly uncovered metaphors.
- Change in scope or range of the metaphor. More concepts from the target domain may be expressed by concepts from the source domain than had originally been predicted. A more subtle, and difficult to verify, change in scope involves narrowing the metaphor. It may be that the metaphor as represented is too broad, predicting uses that are not appearing in the data. The obvious problem, of course, is that the failure of a use to appear in a given corpus does not in any way indicate that it will never appear.

6 Knowledge–Base Use

This paper has focused on the construction of MetaBank, without saying much about how it will be used. We currently envision three plans for using, and thereby evaluating, the coverage and correctness of MetaBank. The first involves its use in continuing development of natural consultant systems in the operating system domain.

The second involves a new project in the area of natural language generation of instructional texts. The generation of appropriate metaphors is an area in which the knowledge–base paradigm has only been briefly studied. [9] It promises to provide a rigorous test for the approach. The representation in MetaBank must not only allow correct interpretation, but rule out the generation of incorrect or unconventional metaphors.

The final plan for evaluating the usefulness of MetaBank is to make it freely available for use in other natural language processing projects. As noted above, the representation chosen for the knowledge-base will be sufficiently general to allow ease of porting.

7 Related Areas

There are two on–going areas of research that are of significant relevance to the work we are engaged in here. These are the construction of large machine usable lexicons, and the construction of large common–sense knowledge-bases.

7.1 Large Lexicons

In recent years, the lexicon has taken on a more central role, both in linguistic theories, and in computational efforts. Paralleling this trend, there has been increasing interest in the construction of large, robust, on–line lexicons. The advent of machine readable dictionaries and large text corpora has made much of this work possible. The notion of a word-sense, however, remains problematic [20] in much of this work. Many of the fine grained sense-distinctions made by lexicographers often seem unmotivated and arbitrary. On the other hand, many of the senses needed in the specialized domains in which these lexicons will be used are missing altogether.

In order to make this problem more concrete, consider again the belief metaphors discussed at the beginning of this report.

(4) It *came* to me that I had to prepare a talk for the conference.
(5) It *hit* me that I didn't have anything to say.
(6) It *struck* me that this wasn't a good situation.

(7) I *arrived at* the conclusion...
(8) I *reached* the conclusion...
(9) I *stumbled* across this idea when...

In each of these examples, a physical motion or locative word is being used to refer to either a belief state or a change in belief. Current lexical approaches are forced to simply attempt to list a meaning having to do with belief as a separate sense for each of the italicized words in these examples. The problem with this approach becomes obvious when one considers the productivity of the conceptualization underlying these senses. Consider the following examples.

(10) I was *led* to the conclusion that...
(11) I was *dragged kicking and screaming* to the conclusion...

(12) John was being *pulled* to the center of the debate.
(13) John's *position* on the debate was unchanged.
(14) John would not *budge* from his *position.*
(15) Mike's *position* on this issue has not *shifted.*

In these examples, a relatively straightforward conventional metaphor representing four core mappings could account for all the surface lexical items: beliefs are objects with locations, believers are objects with locations, shared location entails active belief by the believer, and finally changes in location indicate changes in belief.

It is clear that a static lexical approach to this kind of phenomena is not appropriate. It places the impossible burden on the lexicon builder of capturing in a finite list a phenomena that is inherently generative. The existence of a robust MetaBank would largely relieve this burden from the lexicon builder. If a word sense can safely be predicted from the spatial or physical meaning of the word combined with a known metaphor then the sense need not be listed in the lexicon.

7.2 Common–Sense Knowledge–Bases

CYC [11] and Tacitus [8] are two major recent efforts to construct large common–sense knowledge–bases. The MetaBank project relates to these projects in a number of ways. It obviously requires a common–sense conceptual representation for the various source and target domains that play roles in conventional metaphors. The first point of contact, therefore, is that we would like to avoid having to produce detailed representations of these domains ourselves. We plan to take advantage of the detailed analyses of various common–sense domains that have been done already.

The second point of contact with these efforts is a more subtle one. Builders of common–sense knowledge–bases often find themselves in one of two problematic situations when representing various domains. The first situation occurs when the designer has produced a logically consistent ontology for a domain based on non-language criteria, that bears no obvious relationship to the way that the domain is actually expressed in natural language.

Consider, as a concrete example, the various temporal logics that have been developed to reason about time. It is quite clear that none of these logics can predict the following conventional language uses.

(16) I had a lot of time to *kill.*
(17) I can't *waste* my time on things like this.
(18) Y ou shouldn't *spend* any time on that.

The ontology of time in any system based on these logics clearly must be augmented if they are to be able to deal with how we talk about time. In the case of English, the metaphor structuring time as a resource is necessary.

The second situation occurs when the knowledge–base designer is forced to rely heavily on linguistic evidence from a single language for the representation of a particular common–sense domain. In this case, the resulting representation may simply be a representation of a conventional metaphor for the domain for a given language. (Indeed for many applications this may be sufficient). This problem is most apparent when one considers the problem of translation.

The MetaBank project provides a partial solution to both of these problems. In the first situation, MetaBank provides the connection between the representations of domains and

how the domains are expressed in language by augmenting domain representations with representations of conventional metaphors. In the second situation, MetaBank provides a motivated role for linguistic data that does not force an entirely linguistic representation.

8 Results

Following are the major results we expect of the MetaBank project:

- A knowledge–base of English metaphorical conventions that will be plausibly useful to many natural language processing applications.
- Corpora-based empirical information about the nature and distribution of metaphor usage in English.
- A methodology that will extend to the study of other forms of non-literal language conventions.

References

[1] Ronald J. Brachman and James Schmolze. An overview of the kl-one knowledge representation system. *Cognitive Science*, 9:346–370, 1985.

[2] Jaime Carbonell. Invariance hierarchies in metaphor interpretation. In *Proceedings of the Third Meeting of the Cognitive Science Society.*, pages 292–295. Cognitive Science Society, August 1981.

[3] Y. Choueka. Looking for needles in a haystack. In *Proceedings of the RIAO*, March 1988.

[4] Dan Fass. *Collative Semantics: A Semantics for Natural Language.* PhD thesis, New Mexico State University, Las Cruces, New Mexico, 1988. CRL Report No. MCCS-88-118.

[5] Louise Guthrie, Paul McKevitt, and Yorick Wilks. Oscon: An operating systems consultant. In *Proceedings of the Fourth Annual Rocky Mountain Conference on Artificial Intelligence*, 1989.

[6] Elizabeth Hinkelman and James Allen. Two constraints on speech act ambiguity. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, 1989.

[7] Jerry Hobbs. Metaphor, metaphor schemata, and selective inferencing. Technical Report Technical Note 204, SRI, San Mateo, CA, December 1979.

[8] Jerry Hobbs, William Croft, Todd Davies, Douglas Edwards, and Kenneth Laws. Commonsense metaphysics and lexical semantics. *Computational Linguistics*, 13(3), 1987.

[9] Paul S. Jacobs. *A Knowledge-Based Approach to Language Production.* PhD thesis, University of California, Berkeley, Computer Science Department, Berkeley, CA, 1985. Report No. UCB/CSD 86/254.

[10] George Lakoff and Mark Johnson. *Metaphors We Live By.* University of Chicago Press, Chicago, Illinois, 1980.

[11] Douglas B. Lenat and R.V. Guha. *Building Large Knowledge Bases.* Addison-Wesley, 1990.

[12] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of SIGDOC*, 1986.

[13] James H. Martin. The acquisition of polysemy. In *The Proceedings of the Fourth International Conference on Machine Learning*, Irvine, CA, 1986.

[14] James H. Martin. Understanding new metaphors. In *The Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, 1987.

[15] James H. Martin. *A Computational Theory of Metaphor.* PhD thesis, University of California, Berkeley, Computer Science Department, Berkeley, CA, 1988. Report No. UCB/CSD 88-465.

[16] James H. Martin. *A Compututational Model of Metaphor Interpretation.* Academic Press, Cambridge, MA, 1990.

[17] Paul McKevitt and William Ogden. Wizard-of-oz dialogues in the computer operating systems domain. Technical Report MCCS-89-167, Computing Research Laboratory, New Mexico State University, Las Cruces, NM, 1989.

[18] Paul McKevitt and William Ogden. Ozwiz ii: Wizard-of-oz dialogues in the computer operating systems domain. Technical Report MCCS-90-181, Computing Research Laboratory, New Mexico State University, Las Cruces, NM, 1990.

[19] Peter Norvig. *A Unified Theory of Inference for Text Understanding.* PhD thesis, University of California, Berkeley, Computer Science Department, Berkeley, CA, 1987. Report No. UCB/CSD 87-339.

[20] James Pustejovsky. Towards a generative lexicon. *Computational Linguistics*, 17(1), 1991.

[21] Robert Wilensky. Some problems and proposals for knowledge representation. Technical Report UCB/CSD 86/294, University of California, Berkeley, Computer Science Division, May 1986.

[22] Robert Wilensky, Yigal Arens, and David Chin. Taking to unix in english: An overview of uc. *Communications of the ACM*, 27, 1984.

[23] Robert Wilensky, David Chin, Marc Luria, James Martin, James Mayfield, and Dekai Wu. The berkeley unix consultant project. *Computational Linguistics*, 14(4), 1988.

[24] Yorick Wilks. Making preferences more active. *Artificial Intelligence*, 11, 1978.

[25] Uri Zernik. *Strategies in Language Acquisition: Learning Phrases from Examples in Context.* PhD thesis, University of California, Los Angeles, Computer Science Department, Los Angeles, CA, 1987.