

Computer Understanding of Conventional Metaphoric Language

James H. Martin *
Computer Science Department and
Institute of Cognitive Science
University of Colorado, Boulder

CU-CS-473-90

*This article describes work done while the author was at the University of California, Berkeley. It was supported by the Defense Advanced Research Projects Agency under contract No. N00039-84-C-0089. I'd like to thank Robert Wilensky, George Lakoff, Peter Norvig and Richard Alterman for thoughtful discussions.

Abstract

Metaphor is a conventional and ordinary part of language. An approach to metaphor, based on the explicit representation of knowledge about metaphors, has been developed. This approach asserts that the interpretation of conventional metaphoric language should proceed through the direct application of specific knowledge about the metaphors in the language. MIDAS (Metaphor Interpretation, Denotation, and Acquisition System) is a computer program that has been developed based upon this approach. MIDAS can be used to represent knowledge about conventional metaphors, interpret metaphoric language by applying this knowledge, and dynamically learn new metaphors as they are encountered during normal processing.

1 Conventional Metaphor

Consider the problem of understanding the conventional metaphoric language in the following examples.

- (1) How can I *kill* a process?
- (2) How can I *get into* Lisp?
- (3) You can *enter* Emacs by typing “emacs” to the shell.
- (4) Nili *gave* Marc *her* cold.
- (5) Inflation is *eating up* our savings.

The italicized words in each of these examples are being used to metaphorically refer to concepts that are quite distinct from those that might be considered the normal meanings of the words. Consider the use of *enter* in (3). *Enter* is being used, in this example, to refer to the actions on a computer system that result in the activation of a program. This use is clearly different from what might be called the ordinary or basic meaning of the word that has to do with the actions that result in an object entering an enclosure.

While the word *enter* is used metaphorically in (3), this metaphor is neither novel nor poetic. Instead, the metaphorical use of *enter* results from a conventional systematic conceptual metaphor that allows computer processes to be viewed as enclosures. The various actions and states that have to do with the activation, deactivation, and use of computer processes are viewed as the standard actions and states that have to do with enclosures. This conceptual metaphor, structuring processes as enclosures, underlies the normal conventional way of speaking about these processes. Therefore, the uses of the words *get into* in (2) and *enter* in (3) are the ordinary conventional ways of expressing these concepts that nevertheless involve a systematic, productive metaphor.

2 The Metaphoric Knowledge Approach

The main thrust of the *Metaphoric Knowledge* approach to metaphor is that the interpretation of metaphoric language proceeds through the direct application of specific knowledge about the metaphors in the language. Moreover, it is asserted that the mechanisms that are used to apply this knowledge should be fundamentally the same as those used to interpret direct non-metaphorical language.

Under this view, 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 (Lakoff and Johnson, 1980) and the computational approaches to metaphor described in (Jacobs, 1985; Martin, 1986; Martin, 1987; Martin, 1988; Norvig, 1987).

It is useful here to consider an analogy between the study of metaphor and the study of syntax. Broadly speaking, the study of syntax is concerned with the representation, use and acquisition of sets of complex facts that may be said to represent the grammar of a language. The approach to metaphor, described here, proceeds in a similar fashion. In

particular, it addresses the *representation, use* and *acquisition* of explicit knowledge about the metaphors in the language.

This approach has been embodied in MIDAS (Metaphor Interpretation, Denotation, and Acquisition System)(Martin, 1988). 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.

Note that the term metaphor has historically been applied to a wide range of disparate phenomena. These have ranged from Kuhnian-shift type rethinkings of entire conceptual domains, to explicit formulaic simile statements like *Man is a Wolf*. It is important, therefore, to identify the particular kind of phenomena that the MIDAS approach addresses. MIDAS is solely concerned with modeling the everyday natural language task faced by readers of ordinary text. In particular, the quick and correct interpretation of ordinary text containing usually unnoticed commonplace metaphors.

In order to make the problem of understanding metaphors more concrete, consider the implications of (1) through (3) for a system like the UNIX Consultant (Wilensky et al., 1984; Wilensky et al., 1988). 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: Killing-Terminate
  You can kill a process by typing ^ C to the shell.

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

# Do you know how to enter lisp?
  Applying metaphor: Enter-Lisp
  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.

Consider the details of the Enter-Lisp metaphor in the last example. The knowledge that MIDAS has of this metaphor specifies that the action of invoking certain kinds of processes can be viewed as an entering action, where the process invoked plays the role of the enclosure, and that the user performing the action is viewed as entering the enclosure. MIDAS uses its knowledge of these conventional associations to infer that the use of *enter* by the user is most appropriately interpreted as a process invocation.

3 Previous Computational Approaches

The Metaphoric Knowledge approach, embodied in MIDAS, developed as a reaction to the strategies employed in previous computational approaches to metaphor. These approaches have adopted, what I call, a *knowledge-deficient* approach. By this, I mean an approach that makes no use of explicit knowledge about the metaphors in the language. These approaches do, of course, make use of other kinds of knowledge. They are deficient only with respect to explicit knowledge of the metaphorical conventions of the language. The knowledge-deficient approach has been manifested in two distinct processing strategies lying at opposite ends of a spectrum involving the representation and use of language conventions.

The *word-sense* strategy (Hirst, 1987; Wilensky and Arens, 1980; Riesbeck, 1975; Small and Rieger, 1982) recognizes that there are conventional uses of words that deviate from ordinary compositional, or literal, meaning. This strategy addresses the problem by listing each separate use as an isolated and unmotivated word-sense in the lexicon. Under this approach the uses of *enter lisp* and *get out of emacs* in the previous examples are handled by distinct isolated lexical entries.

While this approach adequately allows known conventional senses to be interpreted correctly, it nevertheless has a number of serious shortcomings. The listing of each separate use as an isolated fact in the lexicon simply fails to capture the generalizations among senses of different words or among the senses of a single word. Consider that a system that had knowledge about the use of *get out of* would not be able to handle related uses of *exit* or *leave* without listing them as separate facts. This failure to capture generalizations among word-senses provides the system with no basis for the prediction or classification of new uses as they are encountered.

At the opposite end of the spectrum, lie approaches that are based on analogy or similarity (Carbonell, 1981; DeJong and Waltz, 1983; Fass, 1988; Gentner et al., 1988; Indurkha, 1987; Wilks, 1978). These approaches assert that metaphors arise from an underlying conceptual similarity or analogy between the concepts representing the literal meaning of the words and the concepts underlying the ultimate meaning of the utterance. These approaches are at the opposite end of the spectrum because they use no knowledge about the conventions of the language. In particular, there is no use of any knowledge about metaphors that are a conventional part of the language. The task of interpreting metaphoric language is seen as a special purpose problem-solving task requiring access to knowledge and inference

techniques that are not otherwise a part of the normal language processing faculties.

Note that the computational costs of these analogy mechanisms are radically higher than those posed for direct non-metaphorical language. While the details of each approach differ, they are all fundamentally based on the stage model (Searle, 1979) where the literal meaning of the sentence is computed and judged to be ill-formed, and then an analogy system is employed to search for an appropriate target meaning.

The metaphoric knowledge approach taken in MIDAS is an attempt to fuse the advantages of each of these approaches. The conventionality of most metaphor is captured through the representation of explicit knowledge about known metaphors. Furthermore, the rich structure underlying the system of metaphors in the language is captured in this representation. This rich structure allows the flexibility to apply the known metaphors in novel situations. At the same time the use of direct knowledge in most cases allows MIDAS to avoid the high computational costs of the analogy approaches.

4 Constraints from Psycholinguistic Research

While it is difficult to apply results from psycholinguistics to computational models in a direct fashion, these results can nevertheless pose useful rough constraints. There are two basic results that are of interest here from psycholinguistics. Both stem from research on the relative difficulty of understanding what has been called literal language versus various kinds of metaphorical, idiomatic, and indirect language (Gerrig, 1989; Gibbs, 1984; Gibbs, 1989; Ortony et al., 1978; Glucksberg et al., 1982; Keysar, 1989).

The first result that will be used is that the time needed to process various kinds of non-literal language does not differ significantly from the time taken to interpret direct language in the appropriate context. Specifically, there is no experimental evidence to indicate that there is a radical difference between the time taken to interpret metaphorical language and that taken to interpret direct non-metaphorical language. This constraint has been referred to as the *total time constraint* (Gerrig, 1989).

This rough equivalence of time to process was taken as one of the basic constraints in the development of MIDAS. Specifically, it was decided that the mechanisms used by MIDAS to interpret conventional metaphors could not differ significantly, in terms of processing time, from the interpretation mechanisms assumed for direct non-metaphoric language.

The empirical result of equivalent time to process does not necessarily imply that similar mechanisms are at work. However, in the absence of more fine-grained empirical results indicating that fundamentally different processes are at work, it seems reasonable to assume that the mechanisms will be similar. Therefore, a further constraint was adopted that the mechanisms developed for interpreting metaphoric language should be fundamentally the same as those assumed for direct non-metaphoric language.

The second result of interest has to do with the non-optionality of metaphorical interpretations. The studies of Glucksberg (Glucksberg et al., 1982) and Keysar (Keysar, 1989) have shown that metaphorical interpretations may be generated even when there is a well-formed and pragmatically appropriate literal interpretation. More specifically they show

that metaphorical interpretations are produced when they are consistent with the context regardless of the well-formedness of a literal interpretation. This is in conflict with the stage model that predicts metaphorical interpretations should only be produced when the literal is ill-formed on some semantic or pragmatic grounds. This basic result was incorporated into MIDAS by simply requiring the system to return all literal and metaphorical interpretations that are consistent with the given constraints. The well-formedness of a literal interpretation is therefore not allowed to block the generation of metaphorical one. As we will see in Section 7.4 this may result in MIDAS returning multiple interpretations.

5 Overview of MIDAS

This section provides a brief overview of the three-part MIDAS approach to metaphor. 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 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.

5.1 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 directed structured 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. These associations also serve to delimit the particular parts of the various source and target domains that are relevant to particular conventional metaphors.

Note that these source-target relations should not be thought of as weak associative links between disparate domains. Such associations would not capture the rich structure inherent in the system of metaphors. In particular, the relations used in MIDAS are directional in that they directly specify which concepts constitute the source and which the target. Moreover, they explicitly delimit what aspects of the source and target domains play a role in any given metaphor.

Note also 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 to 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 (Wilensky, 1986) representation language. KODIAK is an extended semantic network language in the tradition of KL-ONE (Brachman and Schmolze, 1985) and its variants. The details of KODIAK and the representation of metaphoric knowledge will be described in Section 6.1.

These sets of metaphoric associations representing conventional metaphors 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. The hierarchical organization of conventional metaphoric knowledge is the primary means used to capture the regularities exhibited by the system of metaphors in the language. Specifically, KODIAK is used to represent specialized domain specific metaphors, pervasive high-level metaphors, and the systems of relations among related metaphors.

5.2 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 interpretations may include direct non-metaphoric interpretations, as well as all the conventional metaphors that are invoked by the input.

Consider the details of the following shortened trace ¹ of a UNIX example which will be discussed more fully in Section 7. 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.

¹Output from MIDAS is shown in typewriter font. The following notations are used: concepts whose names end in a number represent instances of that category, uparrows indicate the immediate dominating category, while a rightharrow between two concepts points from a source concept to a target concept.

```
> (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))))
```

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

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

The case structure of this preliminary representation is checked against the semantic constraints of all the interpretations conventionally associated with the Entering concept. In this case, MIDAS finds that the direct interpretation and one of the other possible entering metaphors can be rejected before the appropriate Enter-Lisp metaphor is found.

It is important to realize that the order of the search performed here is arbitrary. MIDAS is exhaustively finding all conventional interpretations that are consistent with the input. The determination of consistency for any given interpretation is independent of the consistency of any of the other possible interpretations. In particular, the well-formedness of a direct, or literal, interpretation has no effect on whether or not a metaphorical interpretation will be found. It follows from this that the order of the search through the possible interpretations has no effect on which interpretations will ultimately be produced.

Applying conventional metaphor Enter-Lisp.

```
(A Enter-Lisp (↑ Container-Metaphor)
  (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 metaphor itself along with the attendant source and target concepts. Any further inferences that need to be performed by UC 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.

5.3 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. 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*.

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. Analogical reasoning is at the core of MIDAS's learning mechanism. However, unlike previous metaphor systems, MIDAS does not attempt to draw an analogy between source and target domains of a metaphor. Rather, MIDAS attempts to reason analogically from known metaphors.

In this case, the system finds and extends a closely related known metaphor that also uses *kill* to mean a kind of terminate. MIDAS finds that there is a known metaphor covering the use of *kill* in *kill a conversation* to mean to terminate. This known metaphor is applied

analogically to the current situation through the common notion of process meaning a series of related events happening over time.

```
=====
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))
```

```
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 systematic knowledge about existing known metaphors.

This approach, therefore, shifts the processing emphasis in the case of novel metaphors away from the notion of attempting to determine the right target concept by a direct

matching against the literal source. Rather, an attempt is made to determine the correct target through the use of an existing related metaphor. Therefore in this example, no attempt is made to find the intended target meaning by looking at the source details of literal slaying, rather the system examines the target concept of an already existing terminating as killing metaphor.

The focus of the remainder of this article is on the representation and use of metaphoric knowledge to interpret known conventional metaphors. Details of the MIDAS approach to the acquisition of new metaphors can be found in (Martin, 1990). Lakoff and Turner (Lakoff and Turner, 1988) address the general issue of the relationship between well-known conventional metaphors and novel poetic metaphor.

6 Knowledge Representation Details

This section first provides a brief description of the KODIAK representation language, it then reviews some of the systematic aspects of conventional metaphor that need to be captured, and shows how this is accomplished using KODIAK. While the emphasis in this section is on the use of KODIAK to represent metaphors, it should be noted that all the required background world knowledge of the source and target domains, as well as knowledge of UNIX, is represented in KODIAK

6.1 KODIAK

KODIAK is best seen as an extended semantic network language in the tradition of KL-ONE and its variants. The motivations for its development and its theoretical underpinnings are best described in (Wilensky, 1986). The actual implementation described here is a modified version of the one developed by Norvig (Norvig, 1987) for the FAUSTUS text inferencing system. The description of KODIAK provided here will be brief, introducing only those ideas and notations needed in order to follow the rest of the article. More details will be introduced along the way as necessary.²

Facts in KODIAK are represented as nodes connected together with primitive links. The language provides three types of nodes and eight primitive kinds of links. Figure 1 lists each node and link, gives an icon or label that will be used to denote it in diagrams, and provides a brief description of each type.

6.2 Representing Individual Metaphors

The first requirement for the representation is to be able to capture conventional metaphors as explicit concepts consisting of sets of associations between source and target concepts. Consider again Example (1), repeated here.

(1) How can I *kill* a process?

²KODIAK as an actual representation language and as a theory is in an almost constant state of flux. Therefore the details described here differ in detail but not in spirit from those described in (Jacobs, 1985; Wilensky, 1986; Norvig, 1987).

Absolutes (Rectangles) - concepts, e.g. person, action, idea
Relations (Implicit) - relations between concepts, e.g. actor-of-acting
Aspectuals (Circles) - arguments of relations, e.g. actor

Dominate (D) - a concept is a sub-class of another class
Instance (I) - a concept is an instance of a class
View (V) - a concept can be viewed as another concept
Constrain (C) - fillers of an aspectual must be of some class
Argument (a) - associates aspectuals with a relation
Fill (F) - an aspectual refers to some instance
Equate (=) - two concepts are co-referential
Differ (\neq) - two concepts may not be co-referential

Figure 1: Primitives of KODIAK

This example, from the UNIX domain, involves the conventional metaphor that to kill an ongoing process means to terminate it. The target concepts involve computer processes and the actions that terminate them. The source concept is that of the action of causing a living thing to die. The metaphor consists of the source concepts, target concepts, and the set of associations linking them.

Conventional metaphors like this one are captured in KODIAK through the use of a structured association called a *metaphor-sense*. A metaphor-sense is a concept that consists of a set of component relations that link a set of source concepts to a set of target concepts. The individual component associations are relations called *metaphor-maps*. These metaphor-maps are the associations used to connect source and target concepts. Moreover, these relations are given the status of full-fledged concepts, since relations in KODIAK are concepts. To reiterate, metaphor-senses, along with their component metaphor-maps, are represented explicitly as concepts along with the concepts that make up the various non-metaphorical source and target concepts.

Figure 2 shows the KODIAK representation of the source domain from (1). It states that a **kill**ing is a kind of **action** with a **result** that is a **death-event** which is in turn an **event**. The **kill-victim** of the **kill**ing is an inherited role from **action** indicating that the **kill-victim** is effected by the action. The **kill-victim** is constrained by the **C** link to be a **living-thing** and the **killer** must be an **animate-agent**. Finally the equate links require that the **kill-victim** must be the same as the **dier** participant of the **death-event**.

This figure also introduces an additional notational abbreviation. The links labelled **S** (for Slot) in this diagram are an abbreviation for more complex relations. It is frequently the case that when a relation between two absolutes is being shown, it is in terms of one of

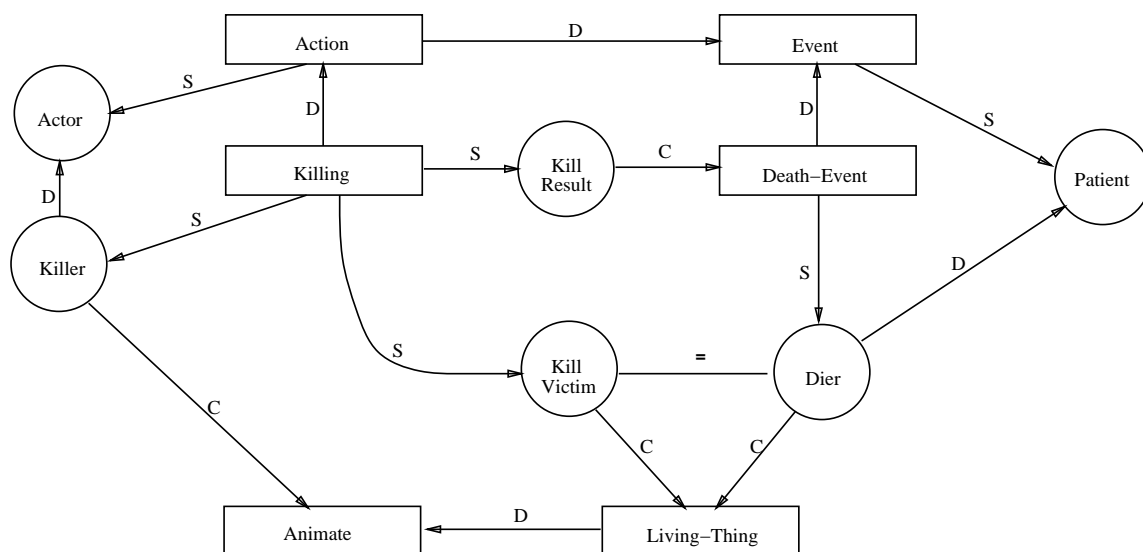


Figure 2: Killing

the absolutes relating to the aspectual farther away from it. In this situation we can replace the relation itself and the aspectual closest to the concept with the slot with an S link. The remaining aspectual and its constrainer remain the same. Remember that this is only a notational convenience to reduce the complexity of the diagrams. The actual underlying representation remains in terms of a relation, two aspectuals, and argument links.

Figure 3 shows the corresponding concepts from the target domain. It states that a `terminate-process-action` is a `terminate-action` which is a kind of `action`. The `terminated-process` role is an inherited role specifying the patient of the action. The result of the action is a `terminate-process-effect` which is a kind of `terminate-event`. Finally, the `terminated-process` is equated to the `terminated-process-event` of the `terminate-process-effect`. This is analogous to the relationship between the `kill-victim` and the `dier` shown in Figure 2.

Metaphor maps are needed to link all the core source concepts in Figure 2 to their counterparts in the target domain. In particular, the `killing` maps to the `terminate-action`, the `kill-victim` maps to the `terminated-process`, the `killer` maps to the `actor` of the `terminate-action`, and the `result` of the `killing` maps to the `result` of the `terminating`.

As noted above, these associations are realized in KODIAK by using directed relations called *metaphor-maps*. A metaphor-map is a kind of VIEW relation whose aspectuals specify corresponding source and target concepts. Figure 4 shows the details of one of the metaphor-maps, the `killed-process` map that underlies (1). The `killed-process` map

constrains the source aspectual to be a **kill-victim** and the target aspectual to be a **terminated-process**. The bottom part of Figure 4 shows a shorthand notation for illustrating metaphor-maps.

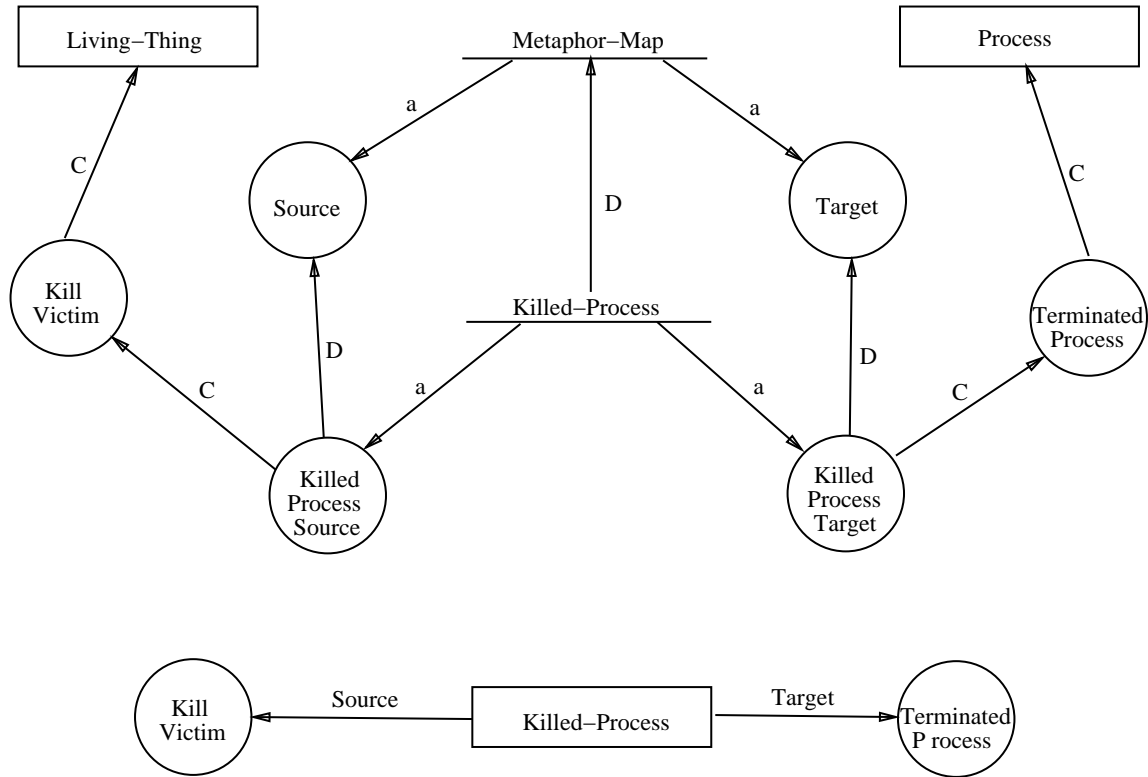


Figure 4: Metaphor Maps

The complete set of maps underlying (1) is shown in Figure 5. The co-occurrence of all these maps, which constitutes the conventional metaphor that terminating something can be viewed as a killing, is shown in this diagram as the **kill-terminate-sense** metaphor-sense. Figure 6 shows the abbreviated notation for illustrating metaphor-senses. The sense itself is represented as the box enclosing the individual maps. This abbreviated representation will be used for the remainder of this article.

To a significant extent, metaphor-senses are the minimal meaning-bearing unit of conventional metaphors. Metaphor-maps represent the building blocks out of which meaningful metaphor-senses are constructed. The metaphor-sense represents the level at which one would say that there is a conventional metaphor that to terminate something is to kill it. This level of representation will frequently correspond to a single metaphorical word sense.

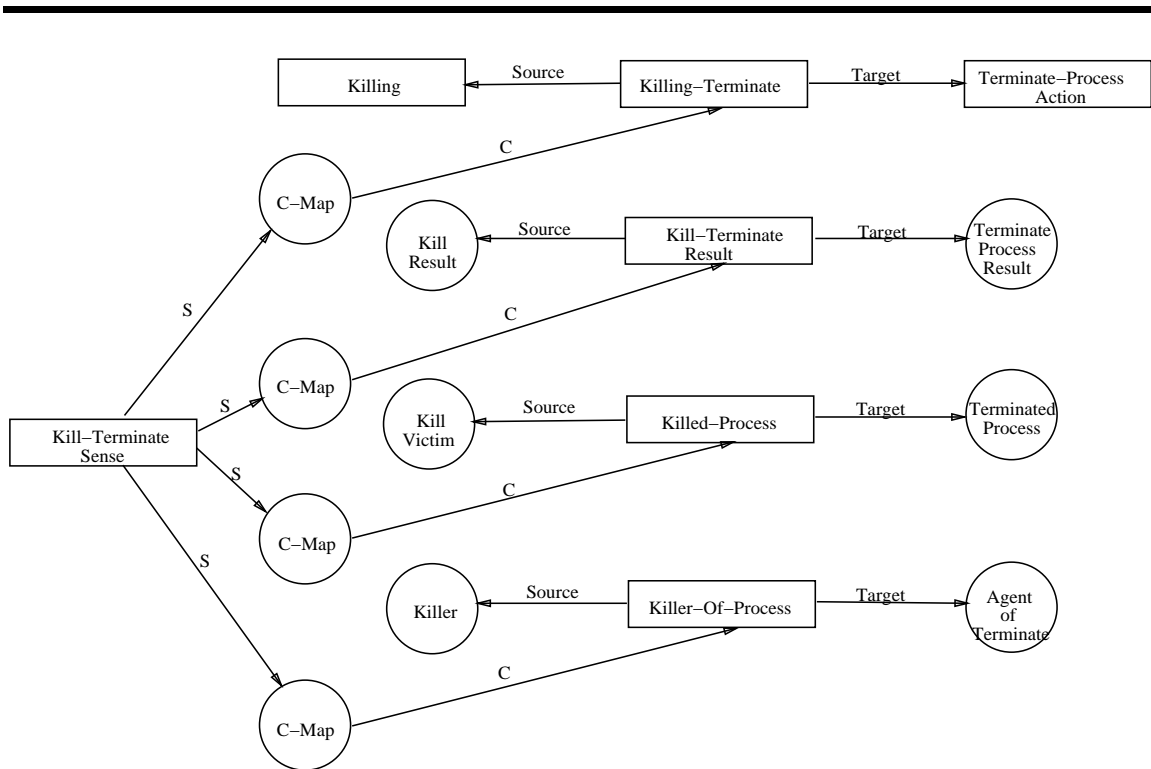


Figure 5: Kill-terminate-sense

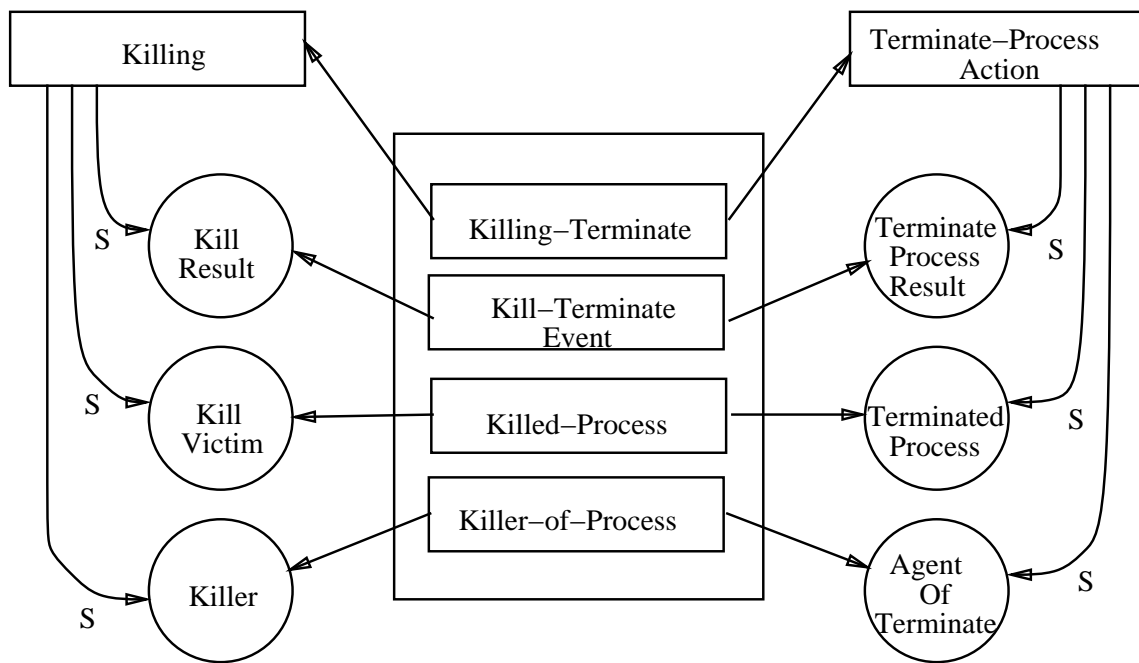


Figure 6: Kill-Terminate-Sense Abbreviated

```
(A Enclosing-Participating (↑ Metaphor-Sense)
  (enclosing-active-participation-map Enclosing → active-participation)
  (enclosure-activity enclosure → activity)
  (enclosed-participant enclosed → participant))
```

Figure 7: Participation As Enclosing

6.3 Representing Systematicities among Metaphors

One of the principle goals in MIDAS for the representation of metaphoric knowledge was to be able to capture the systematic relationships among metaphors. These systematic relationships provide the basis for MIDAS's ability to learn new metaphors. This section will provide some brief details about how two such systematicities are captured using KODIAK.

It is clearly the case that many of the specialized metaphors presented thus far should actually be represented at a higher level of abstraction. Consider for example the Enter-Lisp metaphor presented above. Within the UNIX domain this is simply one instance of a more general metaphor that permits interactive programs to be viewed as enclosures or environments. At an even higher level of abstraction, this metaphor is an instance of the pervasive Container metaphor (Lakoff and Johnson, 1980) found throughout English. Consider the following examples.

- (7) How can I *get into* emacs?
- (8) John *got into* Columbia.
- (9) Jackson *entered* the presidential race today.

In each of these examples, the source concept of entering is used to structure a change of state of some person with respect to some event or activity. The target concepts in each differ widely but the overall structure of the metaphor remains the same in each case: a state-change is viewed as an entering, the participant is doing the entering, and the abstract activity or event is viewed as the enclosure being entered.

KODIAK captures these abstract metaphors by making use of the fact that both metaphor-maps and metaphor-senses are full-fledged concepts, and can therefore be arranged in abstraction hierarchies. An abstract metaphor is merely a metaphor-sense whose component metaphor-maps relate more abstract source and target concepts. Consider the following KODIAK representation of an abstract Enclosure metaphor.

This metaphor-sense captures the abstract metaphor that the state of taking part in an activity can be viewed as an Enclosing via the Enclosing-Participating metaphor. The activity itself is linked to the source role of the `enclosure`, via the `enclosure-activity map`, while the `participant` is linked to the `enclosed` via the `enclosed-participant map`.

In the case where a more specific instantiation of an abstract metaphor-sense exists, it is captured as a child of the higher level metaphor-sense. Correspondingly, the com-

ponent maps of the more specific metaphor are each dominated by the appropriate abstract maps in the parent metaphor-sense. Consider the following representation of the **Enclosing-Using-Computer-Process** metaphor.

```
(A Enclosed-Using-Computer-Process ( $\uparrow$  Enclosing-Participating)
  (enclosing-active-process-map Enclosing  $\rightarrow$  Active-Process)
  (enclosed-process-user enclosed  $\rightarrow$  process-user)
  (enclosure-process-in-use enclosure  $\rightarrow$  process-in-use))
```

Figure 8: Using Computer Process As Enclosing

This metaphor-sense represents the idea that actively using a program can be viewed as an enclosing relationship. The program in use is be viewed as an **Enclosure**, via the **enclosing-active-process-map**, while the **user** plays the role of the **enclosed**. At the same time, it relates this sense and its metaphor-maps to their appropriate abstract counter-parts via dominate links to the **Participation-As-Enclosing** metaphor-sense.

A second kind of important systematicity is demonstrated by the following examples.

- (10) Nili *has* a cold.
- (11) Nili *gave* Marc her cold.
- (12) Marc *got* his cold from Nili.

It is clear that the metaphors underlying these uses are systematically related in a way not directly accounted for by an inheritance relation. At the core of each of these examples is the notion that being in an infected state with respect to a cold can be viewed as possession of the infection. This notion is directly manifested in (10). Examples (11) and (12) extend this core notion of possession to the idea of transfer. Namely that causing someone else to become infected can be viewed as a transfer of possession to the effected person. A representation that merely captured these metaphors as three separate metaphor-senses would clearly be failing to capture the significant fact that they all share a common core concept.

This sharing of important component parts is facilitated by the representation of metaphor-senses in KODIAK. The metaphor-maps that serve as component parts of metaphor-senses are in some sense independent concepts that can play roles in many conceptual structures. In particular, individual metaphor-maps may be shared by many related metaphor-senses. Thus the metaphor-maps that represent the core notion of **Infection-As-Possession** may be systematically shared by metaphors that extend this core. The following examples will demonstrate the exact nature of this sharing.

Consider the details of the **Have-Cold** metaphor underlying (10) as shown in Figure 9, and the **Give-Cold** metaphor underlying (11) as shown in Figure 10. The **Give-Cold** metaphor-sense represents the conventional metaphor that the event of infecting can be

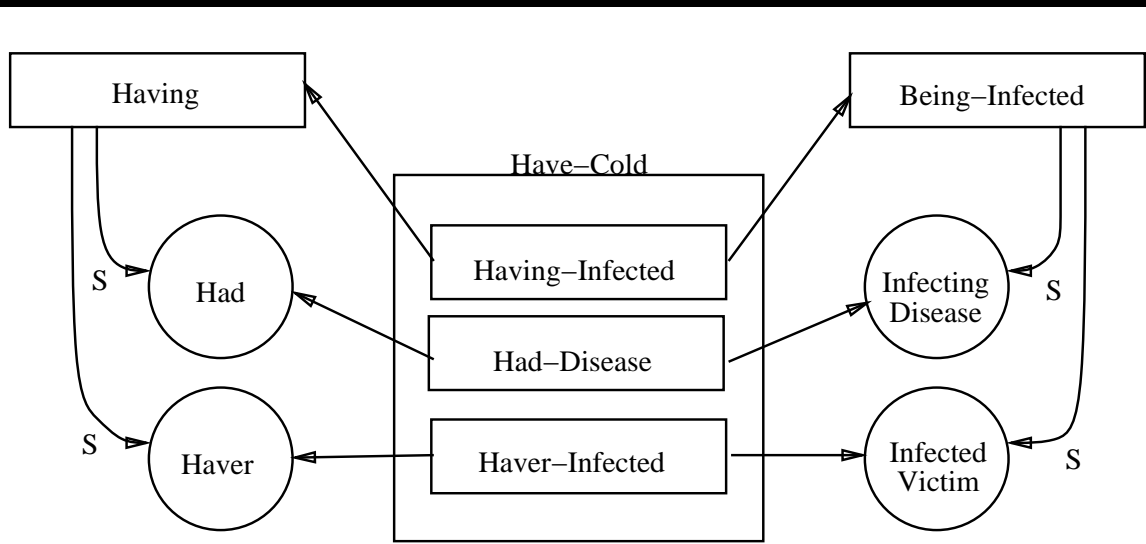


Figure 9: Have-Cold metaphor

viewed as a transfer. The relevant maps to this discussion are the **givee-infected**, **given-cold**, and the **give-res-inf-result** mappings. These maps represent the ideas that the recipient of the giving is viewed as the newly infected person, the given thing is viewed as the cold, and the result of the giving is the result of the infecting action.

Figure 11 shows the extension relationship between the core metaphor **Have-Cold** and the extended sense **Give-Cold**. In this diagram, we see that all the component metaphor-maps from the core **Have-Cold** metaphor are shared and specialized by the extended sense. The extended **Give-Cold** metaphor is distinguished from the core metaphor by the metaphor-maps it adds to the core. In this case, the new metaphor-maps relate to the notion of transfer and causing someone to become infected.

6.4 Limitations of the Representation

The ability of MIDAS to process conventional metaphor is clearly limited by its ability to represent these conventions. There are two distinct issues that need to be addressed when considering MIDAS's ability to capture these conventions: capturing the convention itself, and capturing the underlying motivation for the convention.

It is well-established from both linguistic and psychological evidence that concrete physical and spatial concepts tend to serve as the source of conventional metaphors. Target domains tend to be more abstract concepts that require explication in terms of more concrete concepts. Moreover, image-based similarity and the importance, or centrality, of certain physical domains, such as the body itself, clearly tend to motivate certain metaphors cross-

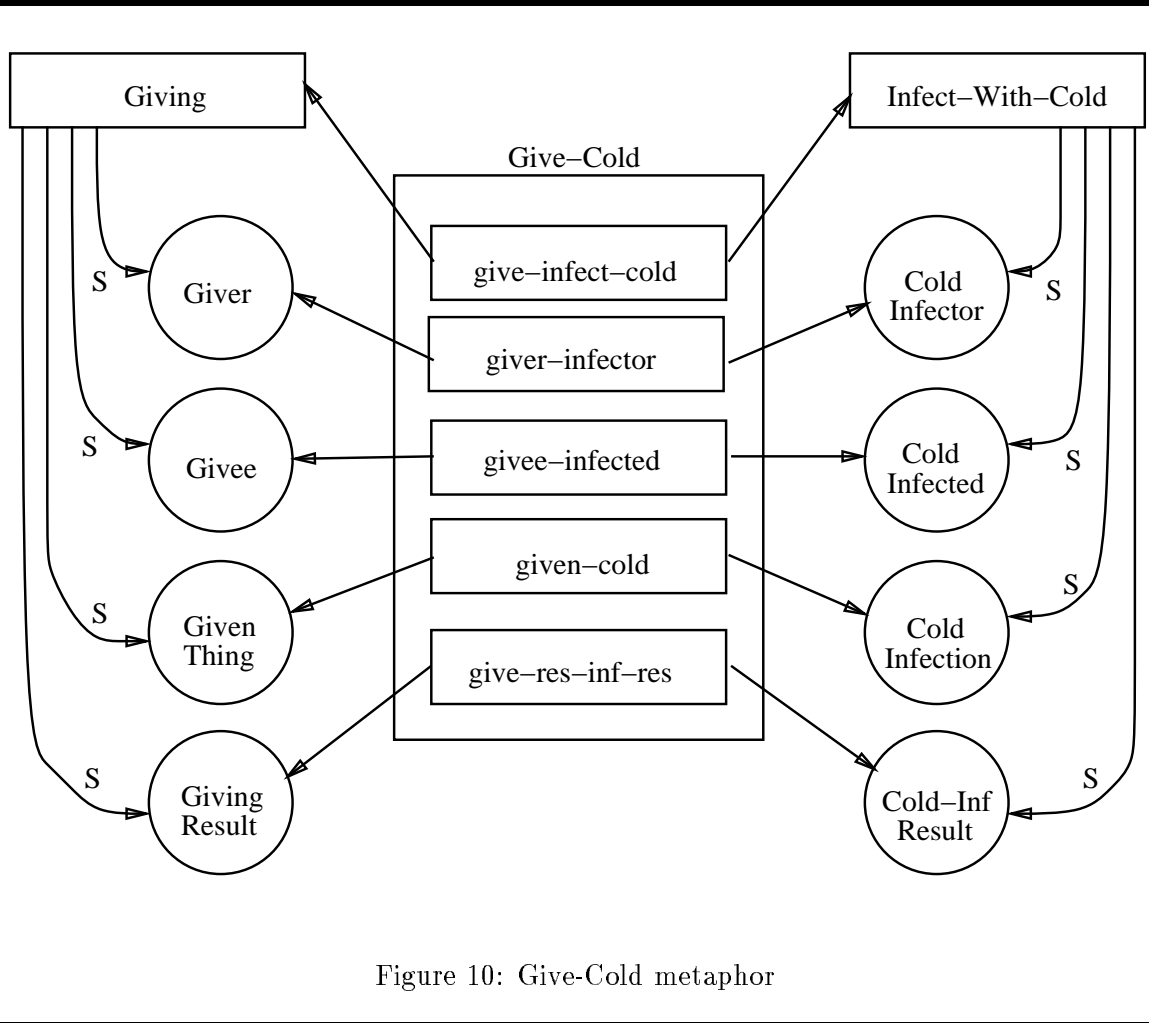


Figure 10: Give-Cold metaphor

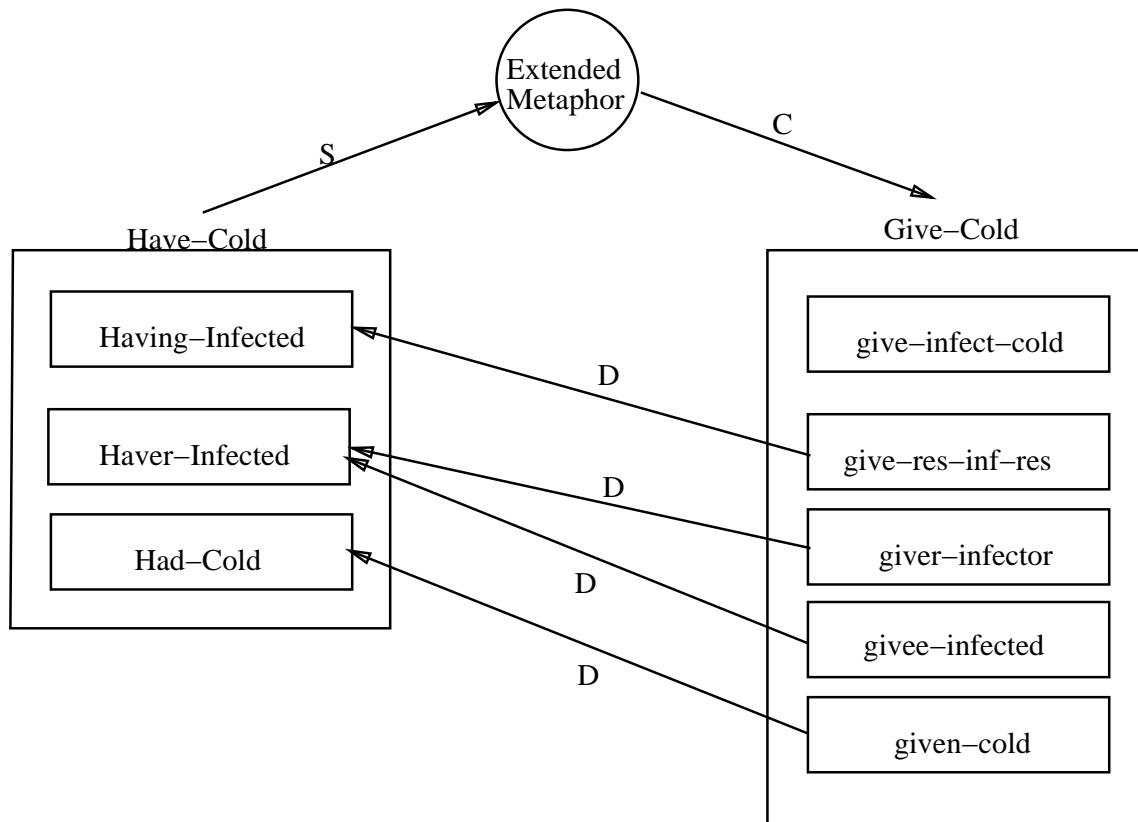


Figure 11: Having Extending to Giving

linguistically (Lakoff and Johnson, 1980; Lakoff, 1987).

The representation language used here clearly does not completely account for this kind of motivation. There is nothing in terms of the representation language itself that would judge the representation of a conventional metaphor as ill-formed if it was not in accord with these findings. Therefore, while the representation can facilitate the application of metaphors for interpretation and constrain plausible extensions of metaphors during learning, it can not ensure that any single metaphor is well-formed based on constraints that it does not have access to.

How then is the representation constrained? Our approach to this problem has been to use the available empirical linguistic evidence as a guide in representing these metaphors. The initial knowledge-base is constructed so that it is in accord with the linguistic analyses of known metaphors. In addition, the constrained nature of the metaphor learning mechanisms ensures that newly learned metaphors will be coherent with existing ones.

7 Interpretation of Conventional Metaphors

This section describes how the metaphoric knowledge just described can be applied to interpret conventional metaphoric language. The main thrust of the approach is that normal processing of metaphor proceeds through the application of specific knowledge about the metaphors in the language. Metaphor is treated as a normal and conventional part of the language. This is directly reflected in MIDAS in the way that metaphors are processed. In particular, the interpretation of metaphor is not viewed as an exception to normal processing. As discussed in Section 3, previous approaches have treated metaphors as anomalous inputs that are only dealt with when literal interpretations are found to be ill-formed. The approach taken here is that metaphoric and literal interpretations have equal status and are evaluated *using interpretation mechanisms that are fundamentally the same*.

7.1 Overview of Metaphor Interpretation

The interpretation of sentences containing metaphoric language is a two step process. In the first step, a syntactic analysis and a preliminary semantic representation are produced. In the second step, a final conceptual representation is produced. This final form is based on the constraints posed in the preliminary representation and the known metaphorical and literal conventions.

The following sections give overviews of the basic tasks performed in each step. Section 7.2 gives the detailed interpretation algorithm, illustrated with examples intended to highlight important issues raised by this approach.

7.1.1 Initial Parse

The first step in the interpretation of an input sentence is the production of a syntactic analysis and a preliminary semantic representation known as a primal representation (Wilensky, 1989). The primal representation produced by the parser represents concepts

```

(A Giving1 (↑ Giving)
  (agent1 (↑ agent) (A John1 (↑ John)))
  (patient1 (↑ patient) (A Mary1 (↑ Mary)))
  (object1 (↑ object) (A Gift1 (↑ Gift))))

(A Giving2 (↑ Giving)
  (agent2 (↑ agent) (A John2 (↑ John)))
  (patient2 (↑ patient) (A Mary2 (↑ Mary)))
  (object2 (↑ object) (A Cold1 (↑ Cold))))

```

Figure 12: Primal Representations

derivable from knowledge in the grammar and lexicon available to the parser.³ This primal representation should be simply considered as an intermediate stage in the interpretation process where only syntactic and surface lexical information has been utilized. In no real sense does it correspond to an interpretation, it is merely the product of an intermediate stage of processing prior to the application of conceptual information. In particular, it can not be construed as a literal meaning since the meanings attached to the lexical items have not yet been accessed.

Consider the following examples.

- (13) John gave Mary a gift.
 (14) John gave Mary a cold.

The primal representations for these examples are given in Figure 12. The primal representation in these examples has simply constructed a skeletal structure where new instances of the absolutes mentioned in the sentence have been associated with case roles. These input roles are left at the level of uninterpreted case roles and no constraint checking is done on their fillers. In other words, any selection restrictions conventionally associated with these roles have not yet been checked against the specified fillers.

The similarity between the primal representations of these two sentences reflects the similarity in the surface forms. The following interpretation process will take these similar primal forms and produce two very different conceptual interpretations.

³This use of primal content differs from Wilensky's formulation in several ways. Wilensky does not envision these as separate processing stages at all, but rather as aspects of the analysis of an utterance. However, the process model developed in this thesis gives rise naturally to an intermediate stage of representation that does correspond to part of Wilensky's primal content. A more complete discussion of the role of the primal representation and its relation to Wilensky's is given in (Martin, 1988)

7.1.2 Interpretation

The interpretation process consists of deciding which of the concepts that are conventionally associated with the primal concept can most coherently account for the constraints imposed by the primal input. Possible candidate interpretations arise from two main sources. Non-metaphorical candidates include the directly mapped literal interpretation of the primal representation and any more specific descendents of that concept. Candidate metaphorical interpretations include any metaphor-senses that are directly attached to, or inherited by, the non-metaphorical candidate concepts.

Two basic inference processes are used, either separately or in tandem, to accomplish this interpretation task; these are *concretion* (Norvig, 1987; Wilensky, 1986; Wilensky et al., 1988) and *metaphoric unviewing* (Martin, 1988; Wilensky, 1989). Briefly, concretion is a kind of specialization inference that replaces an abstract concept by a more well-specified concept. Metaphoric unviewing replaces a given concept that plays the role of the source concept in a metaphor with the corresponding target concept.

Constraint checking is the key operation underlying both concretion and metaphoric unviewing. The input concepts specified in the primal representation are compared against the constraints of the candidate interpretations. Checking the constraints on an interpretation consists of insuring that the specified input filler of each role is coherent with the semantic constraints on that role. For a filler to coherently fill a role it must either be an instance of the concept that constrains that role or an instance of a descendent of the constraining concept.

Constraint checking in a concretion inference consists of insuring that the fillers of the roles in the current concept can satisfy the constraints on all the more specific roles in the more specific category. Metaphoric interpretations are evaluated in a similar fashion. The filler of the source role must be coherent with the semantics specified for the corresponding target role in the particular metaphor being applied.

To illustrate these inferences, consider Example (13) again. The interpretation process finds that the specified role fillers in the primal representation satisfy the constraints on the literal interpretation of *give* as the concept **Giving**. This initial concretion results in the creation of the **Giving** concept shown as **Giving1** in the top half of Figure 13.

The concretion inference should, however, find the most specific concept that can accommodate the input. In this case, we find that **Giving1** can be replaced by the more specific **Giving** concept, **Gift-Giving**. A concretion inference is, therefore, a recursive procedure proceeding down the hierarchy to the most specific category possible. In this example, the concept **Gift-Giving** is known to be a kind of **Giving** where the role of the **given** must be a gift of some kind. In the current example, this is known directly from the use of the word *gift*. The final concreted representation of this example is shown as **Gift-Giving1** in the bottom of Figure 13.

In the case of Example (14), the only interpretation that can account for the input results from a metaphoric unviewing inference. The metaphor **Give-Infection** is used to produce the representation shown as **Infect-With-Disease1** in the top part in Figure 14. Figure 15 shows the **Give-Infection** metaphor with all of its metaphorical mappings. This diagram

```

(A Giving1 (↑ Giving)
  (giver1 (↑ giver) (A John1 (↑ John)))
  (givee1 (↑ givee) (A Mary1 (↑ Mary)))
  (given1 (↑ given) (A Gift1 (↑ Gift))))

(A Gift-Giving1 (↑ Gift-Giving)
  (gift-giver1 (↑ gift-giver) (A John1 (↑ John)))
  (gift-givee1 (↑ gift-givee) (A Mary1 (↑ Mary)))
  (gift-given1 (↑ gift-given) (A Gift1 (↑ Gift))))

```

Figure 13: Final Concretion Step

shows that the role of the `giver` corresponds to the `infector`, the `givee` corresponds to the `infected`, and the `given` corresponds to the `infection`. All the specified fillers of the source roles in the input can coherently fill the target roles. In particular, note that `Cold1`, the filler of the source role `given1`, satisfies the `Disease` constraint on the target role `infection`, corresponding to role `given1`.

```

(A Infect-With-Disease1 (↑ Infect-With-Disease)
  (infector1 (↑ infector) (A John2 (↑ John)))
  (infected1 (↑ infected) (A Mary2 (↑ Mary)))
  (infection1 (↑ infection) (A Cold2 (↑ Cold))))

(A Cold-Infect1 (↑ Cold-Infect)
  (cold-infector1 (↑ cold-infector) (A John2 (↑ John)))
  (cold-victim1 (↑ cold-victim) (A Mary2 (↑ Mary)))
  (infected-cold1 (↑ infected-cold) (A Cold2 (↑ Cold))))

```

Figure 14: Infection Concretion

Once it has been determined that the target roles of the metaphor can be coherently filled by the input fillers, a new instantiation of the target concept is produced to replace the primal representation. This newly created concept is now subject to possible further interpretation via concretion. This reflects the fact that the metaphor applied in the unviewing inference may have its target concept represented at a level that is more abstract than the level desired for the given example.

In this example, the system has knowledge of a concept that is more specific than `Infect-With-Disease`. The concept `Cold-Infect` represents specific information about infecting someone with the common cold. The abstract target concept produced via the

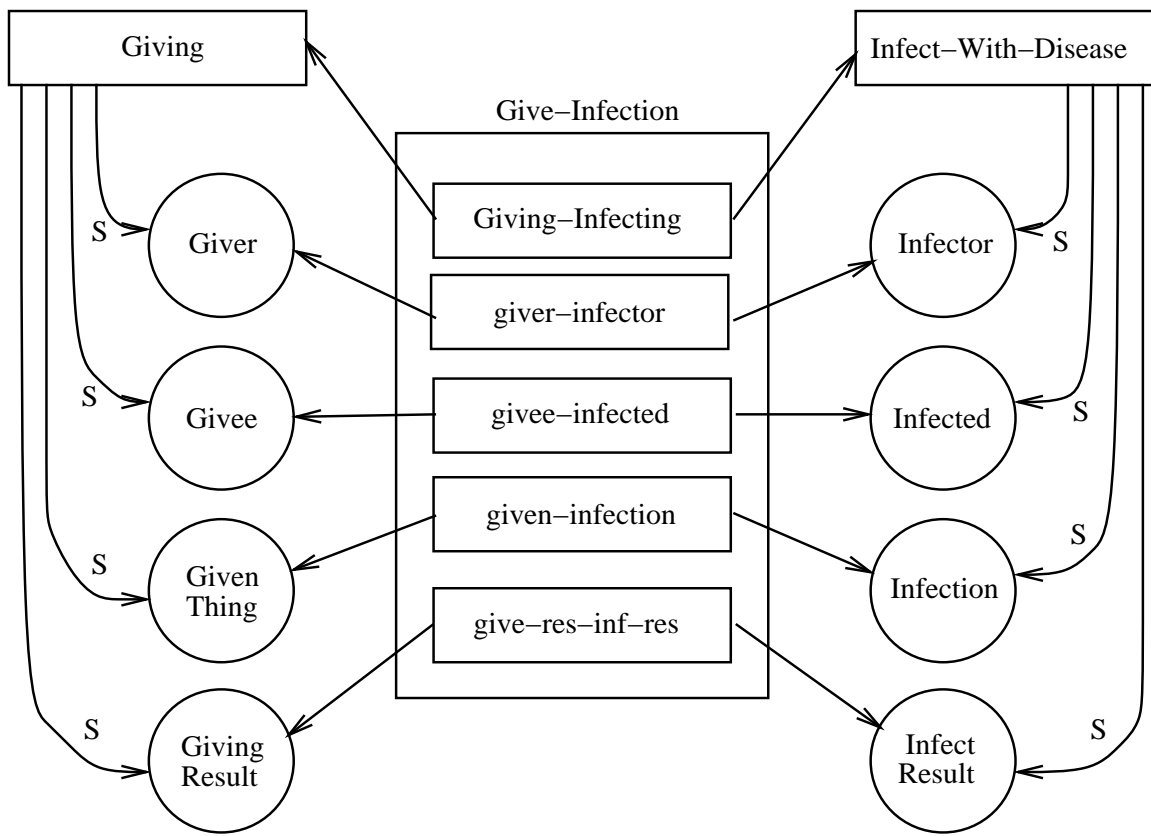


Figure 15: Giving a Disease

metaphoric unviewing inference is, therefore, concreted to this more specific concept. The final representation of the input is shown in Figure 14 as the concept `Cold-Infect1`.

Section 4 introduced two constraints derived from some empirical psycholinguistic results: the *total time constraint*, which imposes a processing time limit, and the *similar mechanisms constraint*, which constrains the allowable mechanisms. It can now be seen that the mechanism of metaphoric unviewing underlying metaphoric interpretation in MIDAS satisfies these constraints. The basic process of checking the filler of a particular source role against the constraints imposed on the corresponding target concept is the same as the process used in concretion. The sole difference lies in the kind of structured association that is traversed to find the candidate interpretation to be checked. In a concretion inference, a `dominate` association is traversed, while in unviewing a `metaphor-map` link is traversed. This sameness in processing, operating over similar structures, results in similar processing times for literal and metaphoric language, thus satisfying the total time constraint.

Note that, in general, having similar mechanisms for literal and metaphoric language is not sufficient to meet the total time constraint. In order to meet the constraint, the necessary differences in the representation of literal and metaphorical examples should not have any major effect upon the running time of the algorithm. For example, Gentner's structure mapping theory, and its computational realizations, entail similar mechanisms for computing literal similarity matches and true analogies. However, as with any analogical matcher, the time required by the algorithm to process literal similarity versus metaphorical analogies varies considerably.

7.2 Interpretation Algorithm

This section presents the details of the interpretation algorithm. After each of the steps of the algorithm has been introduced, a series of detailed processing examples will be presented to illustrate the algorithm and to present various theoretical issues raised by the approach.

Step 1: Parser produces a primal representation for the input sentence.

Step 2: Case roles are concreted to the appropriate semantic relations associated with the primal concept.

Step 3: Collect all possible interpretations, both metaphorical and literal, that are conventionally associated with the primal concept.

Step 4: Validate each of the possible interpretations. This consists of insuring that the concepts specified in the input satisfy the semantic constraints imposed by each of the possible interpretations.

Step 5: Apply all the consistent interpretations. This consists of the instantiation of the concepts underlying each of the possible coherent interpretations. This application may result in the replacement of the primal concept by either a direct non-metaphorical interpretation or a conventional metaphorical one.

Step 6: Return all the interpretations that are consistent with the input concepts.

The most important point to realize about the strategy embodied in this algorithm is that the literal meaning of the input does not have a privileged status. Previous systems that have attempted to deal with metaphor have treated them as ill-formed exceptions to

normal processing. As indicated in Section 3, these systems will only attempt a metaphorical interpretation when a violation of a selection restriction prevents a coherent reading for the literal meaning. The ill-formedness of the literal meaning in these systems therefore drives both the detection and interpretation of metaphors.

There are two main problems with this approach. The first is the fact that it gives an importance, or centrality, to the literal meaning over other conventional meanings that does not seem warranted. From a processing point of view there seems to be no empirically justifiable reason for an interpreter to expect the literal meaning over any other interpretation. The second problem is more immediate for these systems. There are conventional metaphors that do not exhibit surface selection restriction violations in their literal reading. For example, the use of *kill* in (15) and negation in (16) allow the literal reading of each to be well-formed.

(15) McEnroe killed Connors.

(16) No man is an island.

What is needed, therefore, is a strategy that permits metaphoric interpretation that it is not dependent upon the well-formedness of the literal interpretation.

The strategy adopted here considers the literal meaning and all conventional metaphorical readings equally. The only requirement is that the interpretation ultimately chosen must satisfy the constraints imposed by the input. Steps 3 and 4 of the algorithm collect and validate all the known conventional uses, metaphorical and literal. The recognition of the use of a conventional metaphor is in no way dependent upon the success or failure of the literal interpretation. In particular, note that although the literal meaning is evaluated and may be constructed during Steps 3 and 4, it does not play a role in the construction of the possible alternative metaphorical meanings.

Each of the steps of the interpretation algorithm, and some of the issues raised by the strategy it embodies, will now be illustrated in terms of the processing of the following UC example. The details of the initial syntactic processing and semantic interpretation not related to metaphor interpretation are omitted from the following trace. Full details of these phases are presented in (Martin, 1988).

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

How can I enter lisp?

Interpreting primal input.

(A Entering50 (↑ Entering)
  (agent597 (↑ agent) (A I203 (↑ I)))
  (patient562 (↑ patient) (A Lisp58 (↑ Lisp))))

Concreting input relations.
```

Concreting patient to entered.
Concreting agent to enterer.

The patient and agent roles, with their respective filler concepts I203 and Lisp58, were derived solely from the syntax of the sentence, and the verb class that *enter* belongs to. In the next step of processing, these generic roles are replaced by the more specific roles that are actually attached to the primal Entering concept.

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.

At this point, a deeper conceptual analysis of the primal representation has begun. In particular, the structure of the primal representation is checked against the semantic constraints of the interpretations conventionally associated with this concept. In this case, the literal interpretation and one of the other known Entering metaphors are rejected before the correct metaphor is found and applied. These interpretations are rejected because the input concepts filling the roles of *enterer* and *entered* do not match the requirements for these roles in these interpretations. In particular, the interpretation as a physical Entering requires that the *entered* concept must be a kind of enclosure. The filler of the *entered* role in the input, Lisp58, fails this requirement, therefore this interpretation is rejected. Similarly the Enter-Association metaphor specifies that the *entered* concept must be a kind of Association. Again, Lisp58 fails to satisfy this constraint and causes the rejection of the metaphoric interpretation posing this constraint.

Valid known metaphorical interpretation.

Applying conventional metaphor Enter-Lisp.

```
(A Enter-Lisp (↑ Container-Metaphor)
  (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))))
```

The known `Enter-Lisp` metaphor has been found and applied to the given input concepts. The main source concept is interpreted as an instance of the `Invoke-Lisp` concept according to the `enter-lisp-map`. The input roles `enterer` and `entered` are interpreted as the target concepts `lisp-invoker` and `lisp-invoked` respectively.

Final interpretation of 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))))))
```

This interpretation of the `Entering` concept is then used to fill the role of the `topic` role of the `How-Question` that constitutes the representation of the rest of the sentence. This `How-Question`, with the reinterpreted `topic` concept, is then passed along to the next stage of UC processing. UC then prints the answer as follows.

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.

7.3 Interpretation Using Abstract Metaphors

In the previous example, the system found and applied an existing metaphor that was represented at a fairly specific level of detail. As discussed in Section 6.3 MIDAS can represent

and apply highly abstract metaphor-senses representing the abstract metaphors that exist in English. These more abstract metaphors typically have a fairly specific physical source domain coupled with an abstract target. Examples of abstract metaphors known to MIDAS are container, possession and location metaphors for representing states (Lakoff and Johnson, 1980).

In the following example, the system has knowledge of the **Have-State** metaphor. This metaphor represents the widespread metaphor that the concept of being in some state can be expressed as a possession. In this example, the state of being infected with a cold is expressed as a possession. The infected person is viewed as being in possession of the infection metaphorically. However, MIDAS does not have a specific metaphor representing this Infection-As-Possession use. Rather, it is forced to apply the more abstract metaphor **Have-State**. A metaphoric unviewing inference first maps the possession concept to a stative, a concretion inference then maps this abstract concept to the intended concept of **Infected-State**.

> (do-sentence)

Interpreting sentence:

John has a cold.

Interpreting primal input.

(A Having7 (↑ Having)
 (agent52 (↑ agent) (A John49 (↑ John)))
 (patient52 (↑ patient) (A Cold24 (↑ Cold))))

Concreting input relations.

Concreting patient to had.
Concreting agent to haver.

Interpreting concreted input.

(A Having7 (↑ Having)
 (haver7 (↑ haver) (A John49 (↑ John)))
 (had7 (↑ had) (A Cold24 (↑ Cold))))

Failed interpretation: Having7 as Having.

Failed interpretation: Having7 as Have-Idea.

Failed interpretation: Having7 as Have-Permission.

Valid known metaphorical interpretation.

Applying conventional metaphor Have-State.

```
(A Have-State (↑ Metaphor-Sense)
  (have-state-map Having → State)
  (had-state-value had → state-value)
  (haver-state-holder haver → state-object))
```

MIDAS, at this point, has determined that the literal interpretation and two metaphorical interpretations are not appropriate. The system finds that the abstract **Have-State** metaphor is applicable to the primal concepts. In particular, the concepts filling the roles of the **haver** and **had** can satisfy the constraints on the abstract target concepts specified by the metaphor.

Yielding interpretation:

```
(A State4 (↑ State)
  (state-value4 (↑ state-value) (A Cold24 (↑ Cold)))
  (state-object4 (↑ state-object) (A John49 (↑ John))))
```

MIDAS applies the metaphor to yield an instance of the **State** concept. This concept indicates that the target concept is some state that holds between **John** and **Cold**.

Concretion yields:

```
(A Cold-Inf-State4 (↑ Cold-Inf-State)
  (cold-inf-person4 (↑ cold-inf-person) (A John49 (↑ John)))
  (cold-inf-of4 (↑ cold-inf-of) (A Cold24 (↑ Cold))))
```

The concretion process then replaces this abstract state concept with the most specific known kind of state that can accommodate the input values. In this case, the resulting concept is **Cold-Inf-State** representing the state of a person being infected with a common cold.

Final interpretation of input:

```
(A Cold-Inf-State4 (↑ Cold-Inf-State)
  (cold-inf-person4 (↑ cold-inf-person) (A John49 (↑ John)))
  (cold-inf-of4 (↑ cold-inf-of) (A Cold24 (↑ Cold))))
```

7.4 Ambiguous Interpretations and Literal Meaning

It is possible that MIDAS may find multiple valid interpretations of a given input sentence. One such situation arises when there are coherent literal and metaphorical interpretations of a sentence. Consider Example (15) again.

(15) McEnroe killed Connors.

This is a straightforward use of the conventional metaphor that to kill someone in a competition means to defeat them. This metaphor is particularly evident in the sports pages of any newspaper. The obvious problem for a system that relies upon selection restriction violations is that this example does not involve any violation of the selection restriction on the literal meaning. The MIDAS strategy avoids this problem by attempting to find all conventional interpretations of the input. In the following example, the system discovers that there are two legitimate interpretations to this example.

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

McEnroe killed Connors.

Interpreting primal input.

(A Killing144 (↑ Killing)
  (agent596 (↑ agent) (A Mcenroe46 (↑ Mcenroe)))
  (patient561 (↑ patient) (A Connors45 (↑ Connors))))

Concreting input relations.
  Concreting patient to kill-victim.
  Concreting agent to killer.

Interpreting concreted input.

(A Killing144 (↑ Killing)
  (killer89 (↑ killer) (A Mcenroe46 (↑ Mcenroe)))
  (kill-victim89 (↑ kill-victim) (A Connors45 (↑ Connors))))

Valid literal interpretation.

(A Killing144 (↑ Killing)
  (killer89 (↑ killer) (A Mcenroe46 (↑ Mcenroe)))
  (kill-victim89 (↑ kill-victim) (A Connors45 (↑ Connors))))
```

The literal interpretation of `killing144` as an instance of `kill` is found to be a valid reading. This follows from the fact that the input roles `McEnroe` and `Connors` completely satisfy the constraints on the `killer` and `kill-victim` roles. In particular, they are both known to be `animates`.

Valid known metaphorical interpretation.

Applying conventional metaphor Kill-Sports-Defeat.

The system goes on to find that the known Kill-Sports-Defeat metaphor can also adequately accommodate the input concepts. In particular, the knowledge-base contains the information that these participants are known to be competitors and this satisfies the constraints on the target roles of metaphor. This metaphor is, therefore, applied with new target concepts instantiated and filled in to represent the new interpretation of the primal representation.

```
(A Kill-Sports-Defeat (↑ Kill-Metaphor)
  (killed-defeated kill-victim → defeated)
  (killer-defeator killer → defeator)
  (kill-defeat Killing → Sports-Defeat))
```

Mapping input concept Killing144

to concept Sports-Defeat60

Mapping input role kill-victim89 with filler Connors45

to target role defeated60

Mapping input role killer89 with filler Mcenroe46

to target role defeator62

Yielding interpretation:

```
(A Sports-Defeat60 (↑ Sports-Defeat)
  (defeated60 (↑ defeated) (A Connors45 (↑ Connors)))
  (defeator62 (↑ defeator) (A Mcenroe46 (↑ Mcenroe))))
```

MIDAS goes on to find that the remaining killing metaphors are not applicable. In each of the remaining cases, the filler of the kill-victim role fails to meet the requirements of the target role in the metaphor.

Failed interpretation: Killing144 as Kill-Conversation.

Failed interpretation: Killing144 as Kill-Delete-Line.

Choosing among ambiguous interpretations.

```
(A Killing144 (↑ Killing)
  (killer89 (↑ killer) (A Mcenroe46 (↑ Mcenroe)))
  (kill-victim89 (↑ kill-victim) (A Connors45 (↑ Connors))))
```

```
(A Sports-Defeat60 (↑ Sports-Defeat)
  (defeated60 (↑ defeated) (A Connors45 (↑ Connors)))
  (defeator62 (↑ defeator) (A Mcenroe46 (↑ Mcenroe))))
```

There are now two competing coherent interpretations. The process of disambiguating this kind of example must be performed by an interpreter that has full access to the discourse context. A context-sensitive text inference system like FAUSTUS (Norvig, 1987; Norvig, 1989) or DMAP (Riesbeck and Martin, 1985) could accept the ambiguous meanings from the interpreter and choose one based on how well it fits into the current context. There is insufficient information in the single sentence given above to allow MIDAS to choose one of the meanings. The important point to note here is that making a metaphorical interpretation versus a literal one should be based on how well that interpretation fits the known context. It should not be dependent on whether or not the literal meaning is coherent.

The main thrust of this approach to metaphor interpretation is the application of specific knowledge about the conventional metaphors in the language. The initial parse of a sentence produces a primal representation that is essentially a set of constraints on the final representation derived from the grammar and lexicon. The main task of the interpreter is to find any interpretation of the input, metaphorical or literal, that is coherent with the constraints posed by this primal representation. Ultimately, however, the choice among otherwise coherent interpretations must be left up to a mechanism that has access to the wider discourse context.

8 Psycholinguistic Constraints Reconsidered

Section 4 reviewed some of the major results from psycholinguistic research and posed some constraints for a computational theory of metaphor. The major constraint is the *total time* constraint that the time needed to process metaphoric language should be equivalent to the time needed to process direct non-metaphoric language. This result has posed a major problem for the standard existing computational proposals for handling metaphor. The standard approach is a two stage model where the literal meaning of the sentence is first judged to be ill-formed, the intended target meaning is then determined through the use of an analogy algorithm. This general approach is far more expensive computationally than the algorithms that have been proposed for handling direct non-metaphorical language. This disparity between the complexity of proposed algorithms for metaphorical and non-metaphorical language is in direct conflict with the total-time result.

MIDAS offers a way out of this paradox. The basic interpretation mechanisms used by MIDAS for metaphorical and non-metaphorical language, unviewing and concretion, are fundamentally the same with the same computational complexity. MIDAS is thus in accord with the total time constraint. This is achieved, of course, through the use of direct prior knowledge about a range of abstract and specific conventional metaphors in the language. Therefore, the relevance of the MIDAS timing results to the empirical timing data depends on the degree of conventionality of the metaphors used in the experiments.

For the most part, the metaphor examples that have been cited in the psychological literature correspond closely to general abstract metaphors that can be assumed to be known to any competent native speaker of the language. While they vary in minor elaborative details, the primary underlying structure is conventional. The MIDAS model predicts that the speed with which subjects handle these examples results from the application of widespread conventional knowledge of the language. The uniformity of the experimental results is an indication of the degree to which these conventional metaphors are a part of the knowledge held by competent speakers of the language.

What is clearly called for is a more subtle experimental design that takes knowledge of abstract conventional metaphors into account. In particular, such an experimental design should examine the behavior of subjects on the following kinds of data: lexicalized conventional metaphors, abstract conventional conceptual metaphors, unconventional metaphors that are direct and consistent elaborations or combinations of known ones, and finally completely novel metaphors. The prediction made by MIDAS is that the time needed to determine the appropriate meaning of metaphors in context should be relative to the degree of novelty of the metaphor with respect to known conventional metaphors.

9 Conclusions

The MIDAS approach to metaphor developed as a reaction to previous computational approaches. Two factors characterized these approaches: powerful special purpose analogy programs, and little or no explicit knowledge about the metaphors in the language. In contrast, MIDAS uses large amounts of hierarchically organized specific knowledge about the metaphors in the language, and relies upon processes that are fundamentally the same as those already needed to interpret non-metaphorical knowledge. This approach arises from the simple belief that metaphor is a normal conventional part of language.

MIDAS has demonstrated the effectiveness of this knowledge-based approach. In particular, it has demonstrated that it is possible to capture systematic knowledge about metaphor using straightforward knowledge representation techniques, and that this knowledge can be efficiently applied to interpret conventional metaphoric language.

References

- Brachman, R. J. and Schmolze, J. (1985). An overview of the kl-one knowledge representation system. *Cognitive Science*, 9:346–370.
- Carbonell, J. (1981). Invariance hierarchies in metaphor interpretation. In *Proceedings of the Third Meeting of the Cognitive Science Society.*, pages 292–295. Cognitive Science Society.
- DeJong, G. F. and Waltz, D. L. (1983). Understanding novel language. *Computers and Mathematics with Applications*, 9:131–147.

- Fass, D. (1988). *Collative Semantics: A Semantics for Natural Language*. PhD thesis, New Mexico State University, Las Cruces, New Mexico. CRL Report No. MCCC-88-118.
- Gentner, D., Falkenhainer, B., and Skorstad, J. (1988). Viewing metaphor as analogy. In Helman, D., editor, *Analogical Reasoning*. Kluwer Academic Publishers.
- Gerrig, R. J. (1989). Empirical constraints on computational theories of metaphor: Comments on indurkha. *Cognitive Science*, 13(2):235–241.
- Gibbs, R. W. (1984). Literal meaning and psychological theory. *Cognitive Science*, 8:275–304.
- Gibbs, R. W. (1989). Understanding and literal meaning. *Cognitive Science*, 13(2):243–251.
- Glucksberg, S., Gildea, P., and Bookin, H. (1982). On understanding nonliteral speech: Can people ignore metaphors. *Journal of Verbal Learning and Verbal Behavior*, 21:85–98.
- Hirst, G. (1987). *Semantic Interpretation and the Resolution of Ambiguity*. Cambridge University Press.
- Indurkha, B. (1987). Approximate semantic transference: A computational theory of metaphors and analogy. *Cognitive Science*, 11:445–480.
- Jacobs, P. S. (1985). *A Knowledge-Based Approach to Language Production*. PhD thesis, University of California, Berkeley, Computer Science Department, Berkeley, CA. Report No. UCB/CSD 86/254.
- Keysar, B. (1989). On the functional equivalence of literal and metaphorical interpretations in discourse. *Journal of Language and Memory*, 28:375–385.
- Lakoff, G. (1987). *Women, Fire and Dangerous Things*. University of Chicago Press, Chicago, Illinois.
- Lakoff, G. and Johnson, M. (1980). *Metaphors We Live By*. University of Chicago Press, Chicago, Illinois.
- Lakoff, G. and Turner, M. (1988). *More Than Cool Reason: A Field Guide to Poetic Metaphor*. University of Chicago Press, Chicago, Illinois.
- Martin, J. H. (1986). The acquisition of polysemy. In *The Proceedings of the Fourth International Conference on Machine Learning*, Irvine, CA.
- Martin, J. H. (1987). Understanding new metaphors. In *The Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy.
- Martin, J. H. (1988). *A Computational Theory of Metaphor*. PhD thesis, University of California, Berkeley, Computer Science Department, Berkeley, CA. Report No. UCB/CSD 88-465.

- Martin, J. H. (1990). *A Computational Model of Metaphor Interpretation*. Academic Press, San Diego, CA.
- Norvig, P. (1987). *A Unified Theory of Inference for Text Understanding*. PhD thesis, University of California, Berkeley, Computer Science Department, Berkeley, CA. Report No. UCB/CSD 87-339.
- Norvig, P. (1989). Marker passing as a weak method for text inferencing. *Cognitive Science*, 13(4):569–620.
- Ortony, A., Schallert, D., Reynolds, R., and Antos, S. (1978). Interpreting metaphors and idioms: Some effects of context on comprehension. *Journal of Verbal Learning and Verbal Behavior*, 17:465–477.
- Riesbeck, C. (1975). Conceptual analysis. In Schank, R. C., editor, *Conceptual Information Processing*. North Holland, Amsterdam.
- Riesbeck, C. and Martin, C. (1985). Direct memory access parsing. Technical Report YALEU/DCS/RR 354, Yale University.
- Searle, J. (1979). Metaphor. In Ortony, A., editor, *Metaphor and Thought*, pages 92–123. Cambridge University Press.
- Small, S. and Rieger, C. (1982). Parsing and comprehending with word experts. In Lehnert, W. and Ringle, M., editors, *Strategies for Natural Language Processing*. Lawrence Erlbaum Associates, Hillsdale, NJ.
- Wilensky, R. (1986). Some problems and proposals for knowledge representation. Technical Report UCB/CSD 86/294, University of California, Berkeley, Computer Science Division.
- Wilensky, R. (1989). Primal content and actual content: An antidote to literal meaning. *Journal of Pragmatics*, 13:163–186.
- Wilensky, R. and Arens, Y. (1980). Phran - a knowledge-based approach to natural language analysis. Technical Report UCB/ERL/M80/34, University of California, Berkeley.
- Wilensky, R., Arens, Y., and Chin, D. (1984). Taking to unix in english: An overview of uc. *Communications of the ACM*, 27:574–593.
- Wilensky, R., Chin, D., Luria, M., Martin, J., Mayfield, J., and Wu, D. (1988). The berkeley unix consultant project. *Computational Linguistics*, 14(4):35–84.
- Wilks, Y. (1978). Making preferences more active. *Artificial Intelligence*, 11:197–223.