# A Computational Model of Metaphor Interpretation

**James H. Martin**

*Department of Computer Science*
*University of Colorado at Boulder*
*Boulder, Colorado*

# Preface

Metaphor, and other forms of non-literal language, have long been relegated to the back of the bus in the field of Natural Language Processing. With few exceptions, researchers in NLP, have considered metaphor to be either too difficult to face, or too peripheral to worry about. It is worth taking a moment to consider these two issues before going on to the approach taken here.

The history of the field of metaphor is shrouded in mysticism and awe. The literature regales us with the novelty, creativity, and poetry of metaphor. Metaphors lead us to fundamentally new ways of viewing the world. Under the influence of a metaphor, we are allowed to somehow go beyond our current knowledge to reorganize our conceptual systems. Given the inherent difficulties with such characterizations, it is hardly surprising that AI researchers have avoided the topic like the plague.

The second issue, the peripheral nature of metaphor, is somewhat paradoxical. Once you begin to study metaphor, you tend to see them everywhere. Ordinary everyday language becomes both a source of wonder and of infinite data. In light of this, the view that metaphor is somehow a peripheral language phenomena becomes somewhat hard to understand. The problem, of course, is in recognizing a metaphor when you see one.

If we accept the above characterizations of the problem, then metaphors must be rare indeed. Participants in ordinary discourse are not often led to a startling new view of the world. Rather, they see and interpret the world quite well in terms of their current conceptual system. Therefore, the paradox is that a cursory examination of the data tells us that language is filled with metaphor, yet these metaphors usually pose no problem for the understander, and seldom lead to any new view of the world.

The answer of course is that the vast majority of metaphors do not cause us to view the world in a whole new light, but rather enable us to see the world as we do. These frequent, systematic, and conventional metaphors are the focus of this book. Perhaps they are not the most earth-shaking metaphors you have ever run across, but they certainly aren't dead or frozen

either. They provide the conceptual structure that enables us to make sense of the vast majority of the language we encounter.

The fundamental successes in AI, and Cognitive Science, came about when researchers were able to characterize, represent, and exploit *knowledge* and *constraints* about the phenomena in question. The difficulty that most NLP systems have had with metaphor is that they lacked this fundamental knowledge. The work presented here provides this knowledge. Metaphor is taken to be a conventional and ordinary part of language. It is assumed that the interpretation of metaphoric language proceeds through the same kind of process that all other language undergoes: the direct application of specific knowledge about the language being used. More precisely, the interpretation of metaphoric language proceeds through the direct application of explicit knowledge about the systematic conventional metaphors in the language. Correspondingly the creativity, or generativity, of novel metaphors is accomplished through the systematic elaboration and combination of already well-understood metaphors. This study of metaphor, therefore, is concerned with the systematic *representation, use,* and *acquisition* of knowledge about the metaphors in the language.

MIDAS (Metaphor Interpretation, Denotation, and Acquisition System) is a computer program that embodies this approach. MIDAS can be used to perform the following tasks: represent knowledge about conventional metaphors, efficiently interpret metaphoric language by applying this knowledge, and dynamically learn new metaphors as they are encountered during normal processing. MIDAS has been successfully integrated as a part of the UNIX Consultant system. UC is a natural language consultant system that provides naive computer users with advice on how to use the UNIX operating system. By calling upon MIDAS, UC can successfully interpret and learn conventional UNIX domain metaphors, as they are encountered during the course of UC's normal processing.

The following descriptions can be used as a roadmap for the reader who does not intend to sit down and read through the entire book. Chapter 1 gives a quick overview of the main ideas and some brief examples of MIDAS in operation on some interesting examples. Chapter 2 provides a review of the relevant past computational approaches to metaphor. Chapters 3 and 4 provide a thorough, and somewhat dense, analysis and description of MIDAS's knowledge and its representation. Chapter 5 demonstrates how this knowledge can be used to interpret conventional metaphoric language.

The next three chapters are concerned with techniques for learning new metaphors. Chapter 6 provides a concise overview of these techniques, while Chapters 8 and 9 go into the gory details.

Chapter 10 returns to a review of the literature. It presents MIDAS handling a comprehensive set of examples culled from the computational

metaphor literature, along with analyses of the examples from a knowledge-based perspective. The set of examples gathered in Chapter 10 can serve as a test suite for future approaches. Finally, Chapter 11 points out some limitations in MIDAS and suggests some future directions.

# Chapter 1

# Introduction

## 1.1   Conventional Metaphor

Consider some of the problems associated with understanding the conventional metaphoric language in the following sentences.

**(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 agent 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 containment are used to refer to actions and states that have to do with the activation, deactivation, and use of these computer processes. This conceptual metaphor, structuring processes as enclosures, underlies the normal conventional way of speaking about these processes.

Therefore, the uses of the words *enter* in (3) and *get into* in (2) are ordinary conventional ways of expressing these concepts that nevertheless involve a metaphor.

## 1.2   The Metaphoric Knowledge Approach

The main thrust of the approach to metaphor presented here is that the interpretation of conventional metaphoric language proceeds through the direct application of specific knowledge about the metaphors in the language. Correspondingly, the interpretation of novel metaphors is accomplished through the systematic extension, elaboration, and combination of already well-understood metaphors. The proper way to approach the topic of metaphor, therefore, is to study the details of both individual metaphors and the system of metaphors in the language.

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 syntactic creativity, or generativity, of language arises from the ability to manipulate these facts about the grammar of the 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. It is this explicit knowledge of metaphor that accounts for both the conventional and novel metaphors that we encounter.

This approach has been embodied in MIDAS (Metaphor Interpretation, Denotation, and Acquisition System). MIDAS is a set of computer programs that can be used to perform the following tasks: explicitly represent knowledge about conventional metaphors, use this knowledge to interpret metaphoric language, and learn new metaphors as they are encountered.

It is important to note here 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 fast and correct interpretation of text containing many usually unseen 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 [38, 39]. 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. Consider the following UC session illustrating the processing of (2).

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

How can I get into lisp?

  Applying conventional metaphor Enter-Process.

UC: You can get into lisp by typing lisp to the shell.
```

In this example, the user has employed the conventional metaphor, described above, that entails that programs can be viewed as enclosures or environments. The action of entering such an enclosure, underlying the phrase *get into*, corresponds to the action that begins the use of the program. In order to appropriately handle this example, UC must be able to access and apply specific knowledge about this conventional metaphor.

UC handles this kind of metaphoric language by calling upon MIDAS. In this example, UC calls upon MIDAS to find a coherent interpretation for this use of *get into*. MIDAS finds and applies the conventional metaphor that allows the invocation of a program to be viewed as an entering. Chapter 5 gives the full details of how MIDAS interprets conventional metaphors for which it has adequate knowledge.

The previous example illustrates the situation where a user employs a conventional metaphor of which MIDAS has explicit knowledge. However, a system like MIDAS will inevitably encounter metaphors for which it has no adequate conventional knowledge. Consider the following UC session illustrating (1).

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

How can I kill a process?

  No valid interpretations of killing.
```

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

Attempting to extend existing metaphor.
Extending similar metaphor Kill-Conversation.
Creating new metaphor:  Killing-Terminate-Computer-Process

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

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 in the previous example, MIDAS finds that it is initially unable to interpret this example because it has no knowledge of this conventional metaphor. This example illustrates the operation of the learning component of MIDAS, the Metaphor Extension System (MES). This system is invoked by MIDAS when it determines that is has encountered 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 metaphor that also uses *kill* to mean terminate. The full details of the MES component of MIDAS will be given in Chapters 6 through 10.

## 1.3   Previous Approaches

The approach to handling metaphor embodied in MIDAS developed in response to the strategies employed in previous computational approaches to metaphor. Chapter 2 will provide a detailed analysis of these approaches. This section will serve to highlight the relationships between the approach taken here and these previous approaches.

Previous computational approaches to metaphor have adopted an approach that makes no explicit use of knowledge about conventional metaphors. The basic assumption of this approach is that metaphors can be understood without direct knowledge of the conventional metaphors in a language. Clearly such a strategy must resort to alternate sources of knowledge when attempting to interpret metaphoric language. Two kinds of knowledge have been used, yielding two distinct strategies.

The *word-sense*, or *phrasal*, strategy [18, 30, 34, 37, 43] recognizes that there are conventional uses of words that deviate from ordinary composi-

tional, or literal, meaning. The word-sense strategy addresses this problem by listing each separate use as an isolated and unmotivated word-sense in the lexicon. While this approach adequately allows known conventional senses to be interpreted correctly, it has a number of shortcomings.

The first shortcoming involves a representational issue. The listing of each separate use as an individual fact in the lexicon fails to capture the systematicities among senses of different words or among the senses of a single word. This enumeration of each use as an isolated and unmotivated fact about the language leads to the second more critical shortcoming. The lack of structure in the knowledge-base makes it difficult to predict or classify the meaning of new uses when they are encountered. The MIDAS approach presented here is an attempt to list these uses as conventional parts of our language knowledge while still capturing the rich conceptual structure that gives rise to them.

The other major paradigm [4, 5, 6, 31, 40, 41] has held that metaphorical language can be understood without the use of specific word-senses by making use of more general knowledge about the concepts involved in the metaphor. The basic approach to interpreting these metaphors has been to first recognize their presence by virtue of the fact that they deviate from the known semantics of a literal interpretation. The task of interpreting this input is then seen as a general cognitive task requiring access to context, world knowledge, and analogical inference techniques.

The approach taken here is an attempt to fuse these approaches. The metaphors in the language are viewed as conventional parts of the language. They are as conventional a part of the language as the rules of the grammar or words in the lexicon. The proper approach, therefore, is to study them from the perspective of normal language processing through the correct application of specific knowledge. Unlike the word-sense approach, this knowledge must reflect the rich structure underlying the conventional metaphors in the language and attempt to account for how these metaphors can be acquired. Like the more general knowledge-based approaches, representing the knowledge of conventional metaphors requires the use of world knowledge.

## 1.4 Constraints from Psycholinguistics

While it is difficult to apply results from psycholinguistics to computational models in a direct fashion, these results can nevertheless pose useful rough constraints. The results that are of interest here stem from research on the relative difficulty of understanding what has been called literal language

versus various kinds of metaphorical, idiomatic, and indirect language [11, 12, 13, 27].

The basic 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* [11].

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.

# 1.5   Details of the Approach

This section will provide an overview of the three-part MIDAS approach to metaphor. In particular, it will give detailed examples of the following topics:

- **Representation:** The explicit representation in a know-ledge-base of the conventional metaphors in the language in the form of explicit associations between concepts.

- **Interpretation:** The correct and efficient application of the above metaphoric knowledge to the interpretation of metaphoric language.

- **Learning:** The acquisition of new metaphors when examples are encountered for which no known metaphor provides a coherent explanation.

## 1.5.1 Representation

Consider the following example of a conventional UNIX metaphor.

**(6)** I am *in* Emacs.

The metaphorical use of the word *in* reflects a systematic metaphorical structuring of processes as enclosures. Metaphors like this may be said to consist of two sets of component concepts, a *source* component and a *target* component. 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, or relations, 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 sets of metaphoric associations, along with the concepts that comprise the source and target domains, are represented using the KODIAK [35] representation language. KODIAK is an extended semantic network language in the tradition of KL-ONE [1] and its variants. The details of KODIAK and the representation of metaphoric knowledge will be fully described in Chapter 4.

These sets of metaphoric associations representing conventional metaphors in the language 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. This hierarchical organization of conventional metaphoric knowledge is the primary means used to capture the regularities exhibited by the system of metaphors in the language. Chapter 3 provides an analysis of some of these regularities and poses some representational requirements. Chapter 4 will demonstrate how these requirements are met in KODIAK.

## 1.5.2 Interpretation

Metaphor is a normal and conventional part of language. The interpretation of utterances containing metaphors should reflect this fact in the way that

the metaphors are processed. In particular, the interpretation of metaphor should not be viewed as an exception to normal processing. The approach taken here is that metaphoric and literal interpretations have equal status, and are evaluated using interpretation mechanisms that are fundamentally the same.

The main thrust of this approach is that normal processing of metaphoric language proceeds through the direct application of specific knowledge about the metaphors in the language.

The interpretation of sentences containing metaphoric language is a two-step process. The first step in the interpretation of an input sentence is the production of a syntactic parse and a preliminary semantic representation. In the second step, this preliminary representation is replaced by the most specific conventional interpretation that can coherently account for the input. This final interpretation may correspond to a literal interpretation or to one of a number of conventional metaphorical interpretations.

This general interpretation process has been implemented in the Metaphor Interpretation System (MIS) component of MIDAS. The MIS examines the initial primal representation in an attempt to detect and resolve uses of conventional UNIX metaphors. In the following UC example, a user has posed a question involving the conventional metaphor structuring processes as enclosures. The MIS component finds and resolves this metaphor. The resolution produces an instantiation of a coherent target concept representing the correct conventional meaning of the utterance.

In the following example, trace output is interspersed with a running commentary shown in normal Roman font. New examples are introduced with bold horizontal lines, while running commentary is delimited with narrower lines.

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

How can I get into lisp?

Interpreting primal input.

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

The input phrase *get into* is treated as a phrasal unit with a conventional association to preliminary representation named **Entering**. The preliminary semantic representation produced in this step is called the *primal*

*representation* [36] The primal representation produced by the parser repre-
sents concepts derivable from knowledge of the grammar and lexicon avail-
able to the parser. In particular, the primary task accomplished in this
phase is the appropriate assignment of filled case roles to the preliminary
concept underlying the head of a phrase. This primal representation repre-
sents a level of interpretation that is explicitly in need of further semantic
processing.

It is extremely important to note here that this primal representation
should not be confused with what has traditionally been called a literal
meaning. The primal representation should be simply considered as an
intermediate stage in the interpretation process where only syntactic and
lexical information has been brought to bear. In particular, the Entering
concept shown above does not represent a commitment to, or an activation
of, the physical or literal interpretation of *enter*. [1]

```
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 verb class that *enter* belongs to,
and the syntax of the sentence. In this next step of processing, these generic
roles are replaced by the more specific roles that are actually attached to
the Entering concept. An inference where a given concept is replaced by a
more specific concept, based on partial information, is called a *concretion
inference* [26, 39] The details of concretion are given in Chapter 5. In this
example, it is inferred that the agent is an enterer and the patient of an
Entering is an entered.

```
Interpreting concreted input.

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

---

[1]This 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 underlying MIDAS
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 Chapter 5.

---

```
Failed interpretation:   Entering50 as Entering.
Failed interpretation:   Entering50 as Enter-Association.
```

---

The literal interpretation and one of the other known Entering meta-phors 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 an actual **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 meta-phoric interpretation posing this constraint.

Note that the fact that the system considers the literal interpretation first is an artifact of the search procedure. It does not indicate any reliance on attempting the literal meaning first as was the case in previous approaches. All the conventional metaphorical uses have equal status with the known literal concept.

---

```
Valid known metaphorical interpretation.

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)))))
```

---

The `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.

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.

---

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 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 get into lisp by typing lisp to the shell.

---

Note that when a conventional metaphor has been employed by the user in asking a question, UC's natural language generator uses the same

metaphor in producing the answer. In this example, the system uses the same enclosure metaphor employed by the user to express the plan.

A complete description of the MIS component of MIDAS is given in Chapter 5. In particular, it describes how the MIS can perform the following tasks: find relevant candidate metaphors to apply, choose a correct one to apply based upon the constraints of the utterance, and instantiate a representation of the appropriate target concepts.

## 1.5.3    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.

The approach taken here 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 metaphor in a systematic fashion. The basis for this approach is the belief that the known set of conventional metaphors constitutes the best source of information to use in understanding new metaphors. In other words, MIDAS attempts to first extend or combine known conventions to account for a novel use, before attempting to resort to more general and expensive methods based on context and world knowledge.

The basic strategy is to first find a known metaphor that is systematically related to the new example. This candidate metaphor is then applied to the new example in an attempt to produce an appropriate target meaning. The process of applying the candidate metaphor to the new example is dependent upon the kind of semantic connection between the candidate and the new example. Three kinds of connections are recognized, yielding three kinds of extension inferences: *similarity extension*, *core-extension* and *combined-extension*. The details of these inferences are given in Chapters 6 through 9.

Once the intended target meaning of the new example has been determined, a new metaphor is created and stored away for future use. When metaphors of this type are encountered again the system can interpret them directly.

This strategy is realized in the Metaphor Extension System (MES) component of MIDAS. When no coherent explanation can be found for a given primal input it is passed along to the MES. The basic steps of MES algorithm are given in the following sections. These steps will then be made more concrete in terms of a detailed trace from the MES.

**Step 1: Characterize** the new input. Partial source and target components of a new metaphor are extracted from the primal representation accepted as input. The terms *current source* and *current target* will be used to refer to the source and target concepts derived from the input example.

**Step 2: Search** for related metaphors. This step searches for any known metaphors that are potentially related to this new use. The search consists of an attempt to find a path or paths through the network from the current source to the current target concepts that contains a known metaphor. A metaphor contained in such a path is judged to be relevant.

**Step 3: Evaluate** the set of candidate metaphors found in Step 2. The purpose of this step is to select a metaphor from the set found in Step 2 for further processing. This choice is based on a set of criteria to determine the metaphor that is closest conceptually to the current example.

**Step 4: Apply** this previously understood metaphor to the current example. The candidate metaphor is applied to the current target based on the relationship between the candidate mapping and the current example. Depending on this relationship, either a similarity, core, or combined extension inference is performed.

**Step 5: Store** the new metaphor. Create and store a new metaphor consisting of the source and target concepts identified in the above steps along with appropriate associations between them. This new metaphor will be used directly when future instances of this metaphor are encountered.

Consider the processing of the following example from the MES.

---

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

How can I kill a process?

Interpreting primal input.

(A Killing16 (↑ Killing)
   (agent87 (↑ agent) (A I46 (↑ I)))
   (patient76 (↑ patient)
    (A Computer-Process10 (↑ Computer-Process))))
```

---

The parser accepts the input sentence, as specified by the user, and produces a primal representation of the input in the form of KODIAK concepts. This primal representation contains only information derivable from the grammar and lexicon. This primal representation is taken to be one that is in need of further elaboration and interpretation. In this example, the concepts associated with the verb *kill* are interpreted first.

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

Interpreting concreted input.

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

The case relations specified in the initial primal representation are replaced by more specific relations that are directly attached to the concept **Killing**. This is called the concreted primal representation. Note that the constraints on the more specific relations are not checked against the fillers of the roles during this concretion. In this example, this concretion results in the **agent** role being concreted to the **killer** and the **patient** role being concreted to the **kill-victim**.

```
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.
Attempting to extend existing metaphor.
```

Once the concreted representation has been created, it must undergo further processing to find a valid interpretation. The system attempts to determine if the given input is consistent with any of the known conventional interpretations, literal or metaphorical. In this case, the input is not consistent with either the literal **Killing** concept or any of the three known metaphorical uses of *kill*.

As in the previous example, the fact that the system considers the literal interpretation first is an artifact of the search procedure. The conventional metaphorical uses have equal status with the known literal concept, **killing**.

At this point, all the possible conventional interpretations of the primal input have been eliminated as potential readings. The input is now passed to the Metaphor Extension System in an attempt to extend an existing metaphor to cover this new use and determine the intended meaning.

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

Searching for related known metaphors.

Metaphors found:  Kill-Conversation Kill-Delete-Line
                  Kill-Sports-Defeat
```

The first step in the extension step is to collect all the relevant known metaphors that might be related to this new use. This initial search scans through all the metaphors directly attached to the input concept, and also at all the metaphors attached to concepts that are core-related to the input concept. In this case, the system has knowledge of three metaphors that share the same source concept with the current use.

```
Selecting metaphor Kill-Conversation to extend from.

(A Kill-Conversation (↑ Kill-Metaphor Metaphor-Schema)
    (kill-c-res kill-result → conv-t-result)
    (killed-conv kill-victim → conv-termed)
    (killer-terminator killer → conv-termer)
    (kill-term Killing → Terminate-Conversation))
```

The candidate metaphors are ranked according to a *conceptual distance* metric. This is a measure of how close the candidate metaphors are to the new example. The primary factor contributing to this metric is a measure of similarity between the target concepts of the candidate metaphor and the input filler concepts. This conceptual distance metric is fully explained in Chapter 8. The candidate metaphor that is judged to be closest to the input example according to this metric is chosen as the candidate metaphor for further processing.

The selected metaphor is classified for further processing according to its relationship to the input example. In this case, the candidate metaphor is in a similarity relationship to the input metaphor.

```
Attempting a similarity extension inference.

Extending similar metaphor Kill-Conversation with
target concept Terminate-Conversation.
```

---

```
   Abstracting Terminate-Conversation to ancestor concept
   Terminating producing abstract target meaning:

(A Terminating3 (↑ Terminating)
   (terminated3 (↑ terminated)
    (A Computer-Process10 (↑ Computer-Process)))
   (terminator3 (↑ terminator) (A I46 (↑ I)))))
```

---

The first step in the processing of a similarity extension inference is to identify the concepts specified in the input example with their corresponding target concepts. In this example, the concept Computer-Process10 is identified with the target role of terminated-conversation, and the role of I46 is identified with the target role of conversation-terminator. The constrainers of these concepts, however, are too specific to accept these input concepts. In this example, there is a mismatch between the input concept Computer-Process and the candidate target concept Conversation.

The next step, therefore, is to abstract the target concept of the candidate to the first concept that can accept the concepts specified in the input. In this case, the concept Terminate-Conversation is abstracted to its ancestor concept Terminating. The ancestor of the terminated-conversation role has as a constrainer the abstract concept Process, which can constrain the more specific concept Computer-Process.

---

```
   Concreting target concept Terminating
      to Terminate-Computer-Process

   producing concreted meaning:

(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)))))
```

---

The next step in the similarity extension inference is to look down the hierarchy from Terminating to see if there are any more specific concepts beneath this one that can adequately accommodate the input concepts. The specific existing concept Terminate-Computer-Process10 is found. The concept c-proc-termed10 is a more specific concept than terminated and can still accept the input concept Computer-Process10 as a filler, since the constraining concept on the concept c-proc-termed is a Computer-Process.

---

```
Creating new metaphor:

  Mapping main source concept Killing
         to main target concept Terminate-Computer-Process.
    Mapping source role killer
         to target role c-proc-termer.
    Mapping source role kill-victim
         to target role c-proc-termed.

 (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))
```

---

The next stage of processing creates a new metaphor that represents this newly learned use. The role correspondences from the input example to the target concepts of the candidate metaphor form the basis for a new set of metaphoric associations that make up the new metaphor-sense. In this case, the main source concept, Killing, is mapped to the intended target concept Terminate-Computer-Process. The source aspectuals killer and kill-victim are mapped to the concepts c-proc-termer and c-proc-termed, respectively. In each case, a new metaphor-map is created to connect the source and target concept in the knowledge base. The map is then classified properly in the hierarchy of existing maps and connected to the newly created metaphor-sense representing this new metaphor.

In the case of a similarity extension inference, the newly created metaphor-maps are made siblings (children of the same parent) of the corresponding metaphor-maps from the candidate metaphor used. The newly created metaphor-sense is also made a sibling of candidate metaphor. In the current example, the newly created metaphor-sense, Kill-Terminate-Computer-Process, is made a sibling of the candidate metaphor Kill-Conversation. The names given to the new metaphor-maps and metaphor-sense are created by concatenating the names of the component source and target concepts (although the names themselves are not relevant to the functioning of the program).

```
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))))))))
```

The final representation of the input sentence now contains the intended target concept, **Terminate-Computer-Process**, as the topic of user's original how-question.

```
Calling UC on 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 process by typing ^ C to the shell.
```

In the case where the system is processing a UC question, the final representation of the input concepts are then passed to the UC system for further processing. In this case, the system proceeds to answer the user's question on how to terminate a process. In the event that the main concept that UC is addressing has been derived metaphorically, UC's generator uses the user's source language when expressing its answer. In this case, the generator adopts the user's use of *kill* to describe the termination.

The following session demonstrates the altered processing by the system now that the **Killing-Terminate-Computer-Process** has been acquired. The same question is again posed to the system.

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

How can I kill a process?

Interpreting primal input.

(A Killing17 (↑ Killing)
   (agent88 (↑ agent) (A I47 (↑ I)))
   (patient77 (↑ patient)
   (A Computer-Process11 (↑ Computer-Process))))

Concreting input relations.

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

Interpreting concreted input.

(A Killing17 (↑ Killing)
   (killer17 (↑ killer) (A I47 (↑ I)))
   (kill-victim17 (↑ kill-victim)
   (A Computer-Process11 (↑ Computer-Process))))

Failed interpretation:  Killing17 as Killing.

Valid known metaphorical interpretation.

Applying conventional 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))
```

The application of this known metaphor immediately yields the intended interpretation.

---

Yielding interpretation:

```
(A Terminate-Computer-Process11
   (↑ Terminate-Computer-Process)
   (c-proc-termed11 (↑ c-proc-termed)
    (A Computer-Process11 (↑ Computer-Process)))
   (c-proc-termer11 (↑ c-proc-termer) (A I47 (↑ I)))))
```

Concretion yields:

```
(A Terminate-Computer-Process11
   (↑ Terminate-Computer-Process)
   (c-proc-termed11 (↑ c-proc-termed)
    (A Computer-Process11 (↑ Computer-Process)))
   (c-proc-termer11 (↑ c-proc-termer) (A I47 (↑ I)))))
```

---

As in the previous example, all the conventional meanings are attempted before an interpretation is settled upon.

---

```
Failed interpretation:  Killing17 as Kill-Delete-Line.
Failed interpretation:  Killing17 as Kill-Sports-Defeat.
Failed interpretation:  Killing17 as Kill-Conversation.
```

Final interpretation:

```
(A Terminate-Computer-Process11
   (↑ Terminate-Computer-Process)
   (c-proc-termed11 (↑ c-proc-termed)
    (A Computer-Process11 (↑ Computer-Process)))
   (c-proc-termer11 .(↑ c-proc-termer) (A I47 (↑ I)))))
```

Calling UC on input:

```
(A How-Q47 (↑ How-Q)
   (topic47 (↑ topic)
    (A Terminate-Computer-Process11
       (↑ Terminate-Computer-Process)
       (c-proc-termed11 (↑ c-proc-termed)
        (A Computer-Process11 (↑ Computer-Process)))
       (c-proc-termer11 (↑ c-proc-termer) (A I47 (↑ I))))))))
```

UC: You can kill a process by typing ˆ C to the shell.

---

# 1.6   Summary

The MIDAS approach to metaphor is fundamentally a knowledge-based one. The metaphors that are a conventional part of the language are represented directly as associations between source and target concepts. The normal interpretation of metaphoric language proceeds through the application of this metaphoric knowledge. The approach also shows that new metaphors can be learned by incrementally extending these known metaphors.

The rest of this book provides motivations for and descriptions of the various components of MIDAS. Chapter 2 presents an analysis of the relevant previous research that led to MIDAS. An analysis of the structure of both individual conventional metaphors and systems of metaphors is given in Chapter 3. This serves as a motivation for the representation described in Chapter 4. Chapter 4 also serves to introduce the general features of the KODIAK representation language along with the specific details of KODIAK that are used to represent metaphoric language. A description of the metaphor interpretation component of MIDAS is given in Chapter 5.

The remainder of the book is devoted to the topic of learning new metaphors. Chapter 6 gives an overview of the Metaphor Extension System (MES), which can acquire new metaphors when new unknown metaphors are encountered during normal processing. Chapters 7 through 9 provide the details for the basic extension inferences used by the MES. Chapter 10 provides a set of extended examples illustrating how the MES can learn metaphors that were handled by previous computational approaches. Finally, some conclusions are given in Chapter 11.