A Computational Theory of Metaphor Comprehension and Analogical Reasoning

Bipin Indurkhya

COINS Technical Report 84-31

Department of Computer and Information Science
University of Massachusetts
Amherst, Massachusetts, 01003
December 1984

A COMPUTATIONAL THEORY OF METAPHOR COMPREHENSION AND ANALOGICAL REASONING

A Dissertation Presented

By

BIPIN INDURKHYA

Submitted to the Graduate School of the
University of Massachusetts in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 1985

Department of Computer and Information Science

A COMPUTATIONAL THEORY OF METAPHOR COMPREHENSION AND ANALOGICAL REASONING

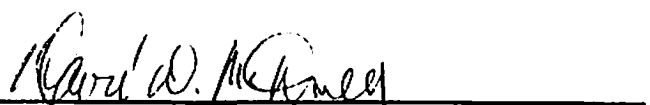
A Dissertation Presented

By

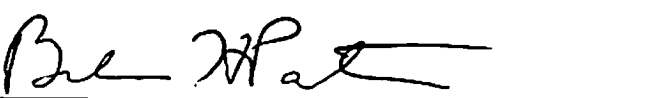
BIPIN INDURKHYA

Approved as to style and content by:

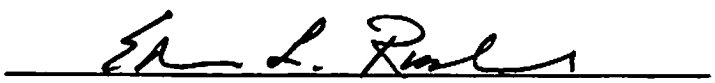
Dr. Michael A. Arbib, Chairperson of Committee



Dr. David D. McDonald, Member



Dr. Barbara H. Partee, Member



Dr. Edwina L. Rissland, Graduate Program Director
Department of Computer and Information Science

Dedicated to my aunt

Maya Gupta

for her love and understanding

and my uncle

Vishnu Prasad Gupta

for being more than an ideal father

ACKNOWLEDGEMENTS

It is with an overwhelming sense of gratitude and joy that I recall all those people without whose support and encouragement I could not have undertaken and finished this thesis. I would like to begin by thanking Jan Landsbergen and Remko Scha of Philips Research Labs, Eindhoven, The Netherlands, for introducing me to logic and linguistics, and for fostering my interest in metaphors before I came to Amherst. Remko, who is now at BBN, also made some very useful and insightful remarks on earlier drafts of Chapters 3 and 5 of this thesis.

Many thanks are due to Mary Hesse for providing impetus to this research while it was still in its embryonic stage. The pre-Gifford lectures given by her, and Michael Arbib, in the fall of 1982 were a source of great intellectual stimulus and helped me considerably in expanding my awareness of many inter-disciplinary issues related to metaphors.

I am very grateful to Ray Turner for carefully reading an earlier draft of Chapter 3 and making many valuable remarks and suggestions.

I will always be indebted to Barbara Partee for her unflagging support and moral encouragement, and for giving me a generous amount of her time in spite of a very busy schedule and numerous other commitments. She has an amazing capacity to bring out the best within you which I will always admire. Many of the ideas presented in this thesis—some explicitly acknowledged in the relevant sections—were evolved through many stimulating conversations I had with her. She also made many useful remarks to increase the readability of this thesis.

I am very grateful to David McDonald for his indispensable help in the programming phase of this research. He made many valuable remarks that added a great deal to the understandablity of this thesis. I am also indebted to him for allowing me the very generous use of his office.

My very special thanks go to Michael Arbib for suggesting metaphors as a dissertation topic, for his enthusiastic intellectual and moral support throughout, and for his relentless urging for details and precision. I marvel—almost to the point of envy—at his perceptiveness, and responsiveness to ideas, even when they are contrary to his own intuitions. I am especially thankful to him for the kindness and the patience he showed with me during the summer and fall of 1983. He also made the most meticulous remarks on earlier drafts of this thesis which were instrumental in bringing method and order to an anarchy of ideas.

The financial support for this thesis came from a number of sources. Many thanks are due to David McDonald and Michael Arbib for supporting the bulk of this research, from June 1982 to May 1984, through their NSF grant no. IST-8104984. I am also very thankful to Barbara Partee for providing very generous support—and a wonderful six weeks at Stanford—during the summer of 1984 and the first half of Fall 1984, and to Michael Arbib and Edward Riseman for supporting me through the departmental funds in the latter half of Fall 1984.

I would like to thank my aunt Maya and uncle Vishnu, to whom this thesis is dedicated, for their love, affection, and understanding, for providing long distance support and encouragement, and for taking care of me when I needed it most.

It is a great pleasure to acknowledge the overwhelming and enthusiastic support—both intellectual and emotional—that I received from my numerous friends and colleagues throughout my stay at Amherst. Just to mention a few, I am very thankful to Rukmini Vijaykumar, Gerry Pocock, Ann Goodman, Judy Franklin, and David Briggs for providing warmth and friendship, and for sharing emotional ups and downs. Special thanks are due to Ugo Buy for being an ideal friend and Tim Sheard for being the best housemate I ever had.

Finally, I would like to thank Art Gaylord and all the members of the RCF staff for providing a very friendly and efficient computing environment.

ABSTRACT

A COMPUTATIONAL THEORY OF METAPHOR COMPREHENSION AND ANALOGICAL REASONING

February 1985

Bipin Indurkhya, B.E., Bhopal University, Bhopal (India)

M.E.E., Philips International Institute, Eindhoven (The Netherlands)

Ph.D., University of Massachusetts, Amherst

Directed by: Professor Michael A. Arbib

In this thesis we propose a formal theory of metaphors and analogies. We start from the assumption that in a metaphor, or an analogy, some terms of one domain (source domain) are applied to terms of another domain (target domain). We address the problem of representing the information contained in a metaphor, or an analogy, and the means of computing it from the domains' knowledge.

We describe a formalism, called Schema-Language [SL], for representing domain knowledge which is based on the First Order Predicate Calculus. We then develop a theory of Constrained Semantic Transference [CST] which shows how the terms and structural relationships of the source domain can be coherently transferred to the target domain. The concept of a T-MAP, which is a partial coherent mapping from the terms of the source domain to the target domain, lies at the heart of CST.

We introduce two operators, called Augmentation and Positing Structure, that make it possible to create a new structure in the target domain constrained by the structure of the source domain. We show how to characterize metaphors and analogies by using T-MAPs which can explain many cognitive properties associated with them. We also present a characterization of metaphorical truth and metaphorical inference in CST. A major limitation of CST is that the notion of coherency is not computational.

We propose a theory of Approximate Semantic Transference [AST], which is derived from CST by replacing the coherency requirement on T-MAPs by approximate coherency. The partial approximate-coherent mappings of AST, called AT-MAPs, are computational and can be used as a basis for developing models of cognitive processes involved in comprehending metaphors and analogies. We propose two alternative formulations of approximate coherency. Based on one of these version, we present several algorithms, and priniciples that can be used in designing algorithms, for computing AT-MAPs from the knowledge of the source and target domains.

TABLE OF CONTENTS

"... You couldn't deny that, even if you tried with both hands."
"I don't deny things with my hands," Alice objected.
"Nobody said you did," said the Red Queen. "I said you couldn't if you tried."

— Lewis Carroll, "Alice Through the Looking Glass".

"Take some more tea," the March Hare said to Alice, very earnestly.

"I've had nothing yet," Alice replied in an offended tone: "so I can't take more."

"You mean you can't take less," said the Hatter: "it's very easy to take more than nothing."

— Lewis Carroll, "Alice's Adventures in Wonderland".

C H A P T E R I

INTRODUCTION

1.1 MOTIVATION AND BACKGROUND

In recent years, the omnipresence of metaphors in any language, as well as the crucial role played by metaphors in cognition, science, and religion, has increasingly been recognized. Any reader who still thinks that metaphors are mere deviant use of language that should be restricted to poetry and have no role in normal everyday discourse should look at Lakoff and Johnson [1980] who show by numerous examples how metaphors pervade our everyday life. They are all around us—in newspaper, TV, radio, what not—in such an abundance that often we do not notice them. They affect our lives, our works, our perceptions to a very remarkable extent. Still we do not have a clear understanding of what they are, and how they work. This thesis is a step towards providing such an understanding.

The reference to metaphors can be found as early as in the writing of Aristotle, who remarked in the *The Poetics* "the greatest thing by far is to have a command of metaphors". Until about 1936 metaphors were considered to be mere ornamental devices which impart pleasing effects to prose and poetry but should be

banned from serious discourse and science. Richards [1936] realized the omnipresence of metaphors in language and science, and even went as far as to claim that "thought is metaphoric, and proceeds by comparison, and the metaphors of language derive therefrom" [p. 51]. Since then it has been accepted by an increasingly large number of people that metaphors play a very crucial role in the development of any language. We mentioned earlier Lakoff and Johnson who have provided enough linguistic evidence to strengthen this hypothesis. Arbib and Hesse [1984] have also argued that all language is metaphorical. Even if one does not agree with this view it is hard to deny that metaphors are crucial to any language and any theory of language understanding should not ignore them. Some approaches towards understanding literary metaphors can be found in Guenther [1975], Levin [1979], and Van Dijk [1975].

The parallel between metaphors and scientific models has been noted for quite some time by philosophers of science. For instance, Black [1962] advanced several arguments to show how metaphors are related to models and then went as far as to claim, "[E]very implication-complex supported by a metaphor's secondary subject, I now think, is a *model* of the ascriptions imputed on the primary subject: Every metaphor is the tip of a submerged model" [1979, p. 31, emphasis Black's]. Boyd [1979], Hesse [1963, 1974, 1980], Kuhn [1979] and MacCormac [1976] have provide further insights into the role played by metaphors in the development of science and in theory change. Turbayne [1962] has presented the most impressive case to emphasize how metaphors can have a very significant effect on the growth of science by showing how Descartes' mechanistic metaphor dominated Newton and a

whole era of science after him. Gentner [1982] has provided psychological evidence to show the connection between scientific analogies and metaphors.

Often there is some underlying structural analogy or simile at the heart of most metaphors. Thus a study of metaphors is also relevant to the science of cognition. In Piaget's theory of intelligence [Piaget 1936; 1951] metaphors can be identified with the assimilatory function of intelligence. Miller [1979] and Ricoeur [1978] have also emphasized the role played by metaphors in cognition. Gentner and Gentner [1983] have provided psychological data to show how the notion of structural analogy dominates the process of analogical thinking and reasoning. Green [1979] and Petrie [1979] have also also noted the role of metaphors in learning.

Though the role of social metaphors in shaping culture has been emphasized in past [Berger & Luckmann 1966; Cassirer 1955; Foss 1949; Henle 1958; Ricoeur 1977; Schon 1979; Urban 1939; Whorf 1956], recently this view has been strengthened by the thesis that religion should also be looked upon as a social and cultural metaphor [Arbib & Hesse 1984; MacCormac 1976]. The most persuasive arguments for this thesis can perhaps be found in Frye [1981] where he analyzes the Bible from a literary point of view.

In face of all this evidence it seems that metaphors can be a very productive field of research since any insights obtained in this process are likely to have significant impact on many different disciplines. In particular we believe that the young field of Artificial Intelligence [AI henceforth] can benefit a great deal from such a study.

In spite of the fact that an enormous amount of work has been done on metaphors, we are still far from a computational theory that can be used to build models of the cognitive processes involved in comprehending or generating metaphors. On the one hand, very few of the philosophical theories of metaphors are spelled out with sufficient detail and rigor that they can be adapted for computational needs. Of course, there are a few, like MacCormac's theory [1982], that are precise enough but they fail to account for many cognitive features of metaphors, as we will discuss in the next chapter. On the other hand, AI has mainly concentrated on producing isolated models of analogical reasoning that operate on very small and restrictive domains. Further, most of these models, since their main concern is to produce an acceptable output within a reasonable time in the domain they are designed for, make use of many domain-dependent heuristics. Due to these heuristics it is very hard to distinguish those features of their programs that correspond to general domain-independent principles of analogy. To make things worse, AI has many knowledge representation schemes and often some parts of the program use some technicality or other of the representation that they use to increase their efficiency. For all these reasons one cannot always abstract a general theory of analogy embodied in a program and therefore it is not always possible to compare one model with another unless they happen to operate on the same domain and use the same knowledge representation paradigm.

In a way this is to be expected since AI is a very young though rapidly expanding field. All these early attempts at building models of analogical reasoning have provided us with much valuable data and initial experience. We now believe that the time is ripe to approach the problem of metaphors and analogy from the other end. Thus, we will first design a general domain independent theory of analogy and then build models of analogy in specific domains based on this theory. Of course, in implementing any model one can take advantage of the peculiarities of the domain—in a way this is inevitable because of efficiency considerations—but there will be a neat separation of principles that are domain dependent from those that are not. This will also make all the models transportable from one domain to another with little extra work.

This thesis marks the first step in this direction. Here we develop, rigorously, a theory of metaphors and analogy, with the purpose of designing and implementing computational models of cognitive processes involved in comprehending and generating metaphors, and analogical reasoning. Our theory is not tied to any one domain in particular. At this stage we do not consider efficiency as a criterion since we do not have any one particular program in mind and we would like to leave open many possibilities to be exploited in specific domains.

At a meta-level we hope that the whole field of AI will benefit from this paradigm since it can be applied to many other problems in AI as well.

1.2 POSING THE PROBLEM

Before posing the precise problem that we are addressing in this thesis we would like to list some assumptions, that we are starting from, concerning the nature of metaphors and analogies.

1. There is some structural analogy underlying all metaphors and scientific models. Thus, every time we use the word "metaphor", we mean metaphors, analogies and scientific models.

2. Metaphors are characterized as an application of some term(s) belonging to one domain, called the source domain, to some term(s) belonging to another domain, called the target domain. In literature the source and the target domains are also referred to as the secondary and the primary subjects respectively.

3. The notion of coherency is very crucial to metaphors. In other words, aptness of a metaphor is determined by its coherency.

These assumptions are not totally arbitrary and have often been argued in the literature in one form or another. In the next chapter we will provide the reader with some justification for these assumptions as well as references to places where more articulated arguments can be found.

Given these assumptions about the nature of metaphors, the paradigm that we mentioned in the last section, and that we would like to follow in this thesis, consists of the following stages:

1. Choose some appropriate formalism for representing the knowledge associated with a domain. This representation should be domain independent as as far as possible. In other words, we should not include some feature just because it happens to seem natural for one particular domain.

2. Formalize the notion of metaphor in the representation that was chosen in the previous stage. It should be possible to explain many of the cognitive features of metaphors in this formalization.

3. Generate computer models of the cognitive processes involved in the process of metaphor comprehension and analogical reasoning in specific domains that are based on the formalization of the previous stage. In designing these models keep a clean separation between the principles that are a consequence of the formalization and those that are domain dependent.

4. Use the experience gained from these models to revise the formalization, and/or the knowledge representation and go to the previous stage.

In this thesis we address the first two stages of this paradigm. The third and the fourth stages would typically involve a large number of people and require several years of research and experimentation.

1.3 PROPOSING A SOLUTION

In this thesis we propose a knowledge representation scheme, called Schema-Language [SL henceforth], that is derived from the first order predicate calculus, but differs from it in certain ways, and is based on a structural paradigm. The basic unit of representation in SL is a *domain* which consists of a set of tokens and a set of structural constraints. The tokens of a domain form its vocabulary and its structural constraints put constraints on the "meaning" of the tokens. Some of the structural constraints specify how the meanings of certain tokens are interrelated whereas the others show how the meaning of certain token can be derived from the meanings of certain other tokens. The latter kind of constraints are called derivations and a token shown to be derivable by any such constraint is called a derived token.

We claim that SL is very general in the sense that it allows representation of domains that admit of several possible descriptions and do not have a fixed set of primitives. Further, SL allows representation of partially specified domains.

Next we develop a theory of Constrained Semantic Transference [CST] that formalizes the notion of coherency across two domains. The concept of T-MAPs, which are partial coherent mappings from the source domain to the target domain, is a central part of CST. We define T-MAPs as follows. First we define mappings which are partial functions from the tokens of the source domain to the tokens of the target domain. Then, given a mapping and a partial structure of the source domain we can transfer this structure from the source domain to the target domain by replacing all the source domain tokens by their corresponding target domain tokens as given by the mapping. Thus, in the process of transformation the structural relationships among the source domain tokens are preserved whereas the tokens themselves get replaced by the target domain tokens. We now say that a mapping is coherent with respect to a partial structure of the source domain if the structure of the target domain is consistent with the transformed partial structure of the source domain. Finally, we define a T-MAP to be a pair of a mapping and a partial structure of the source domain such that the mapping is coherent with respect to the structural constraints of the target domain. Of course, we will be describing all these notions in detail, with many examples, in the rest of this thesis.

We also introduce two recursive operators, called Augmentation and Positing Structure, that allow adding structure to the target domain in a manner constrained by the structure of the source domain. Whereas Augmentation uses derivations of the derived tokens of the source domain to create new derived tokens in the target domain, Positing Structure is more general and allows new tokens and new structural constraints to be added to the target domain, in a manner constrained by the structural constraints of the source domain, even in the absence of derivations.

We claim that CST is a theory of metaphors and analogies superior to any other theory put forth so far. Apart from the fact that CST is spelled out in rigorous detail, many other theories like Gentner's theory and Hobbs' theory become special cases of CST. Also, we can explain several cognitive features of metaphors, like metaphorical highlighting and downplaying, and creating new vocabulary and structure in the target domain, in CST.

A major limitation of CST is that it is not computational, since coherency is not decidable, and therefore is not suitable for designing models of metaphor understanding and analogical reasoning. To overcome this difficulty we develop a computational version of CST, called Approximate Semantic Transference [AST]. In AST we require that the structural constraints of a domain be finite and the notion of coherency is replaced by approximate coherency, which is coherency within a specified complexity bound. T-MAPs that are approximately coherent are called AT-MAPs.

We claim that AST retains all the cognitive features of CST and, since it is also computational, can be used for designing models of processes involved in comprehending metaphors and analogies. We present several principles and algorithms that can be used in designing such programs. The principles and algorithms are based on our experiments with implementing programs that can generate AT-MAPs from the domain knowledge. We did not try to build a complete system that could understand metaphors or generate analogies since there are many other problems involved with these processes that are unrelated to metaphors, that we do not have a clear understanding of, and that are beyond the scope of this thesis.

1.4 ORGANIZATION OF THE THESIS

In Chapter 2 we will review some earlier work on metaphors and analogies by various scholars in different disciplines. Due to the vastness of the available literature and limitations of space here we will focus on only those theories and models that we consider to be more prominent and that are more relevant to this thesis. We will provide perspectives on metaphors from philosophical, AI, and psychological points of views. At the end of this chapter we will list a number of cognitive features of metaphors that we would expect any theory of metaphor to be able to handle.

In Chapter 3 we will develop our theory of Constrained Semantic Transference. We will first formally define the knowledge representation scheme SL and then introduce the notion of a T-MAP that allows part of the structure of the source domain to be transferred to the target domain coherently. The coherency

condition will be defined precisely and we will distinguish between weak and strong coherencies. We will also introduce two recursive operators, Augmentation and Positing Structure, on T-MAPs that allow one to add structure to the target domain constrained by the structure of the source domain. All this discussion will be supplemented with examples where necessary.

In Chapter 4 we will show how metaphors and analogies can be characterized in CST. We will begin with a discussion of "real-world" domains—as opposed to "self-contained" domains—and how they can be represented in SL. Then we will show how T-MAPs can characterize an interpretation of a metaphor or analogy. We will argue, by means of some simple examples, that this characterization can explain many cognitive features of metaphors and analogies. We will also compare CST with some of the other theories of metaphors that were discussed in Chapter 2. Finally, at the of this chapter we will discuss some of the limitations of CST as a theory of metaphor, especially with respect to computability and finite representability.

In Chapter 5 we will develop a theory of Approximate Semantic Transference that can best be seen as a computable version of CST. AST is developed so that it can be used as a basis for designing models of cognitive processes involved in comprehending metaphors and analogies. We also describe several principles and algorithms that can be used in designing such models in this chapter. We will conclude the discussion by showing an example of how the algorithms and principles presented in this chapter can be applied in analyzing a metaphor.

Finally, in Chapter 6 we will conclude the discussion of this thesis and point out several directions for further research.

"I proceed. 'Edwin and Morcar, the earls of Mercia and Northumbria, declared for him; and even Stigand, the patriotic archbishop of Canterbury, found it advisable—'"

"Found what?" said the Duck.

"Found it," the Mouse replied rather crossly: "of course you know what 'it' means."

"I know what 'it' means well enough when I find a thing," said the Duck: "it's generally a frog or a worm. The question is, what did the archbishop find?"

— Lewis Carroll, "Alice's Adventures in Wonderland".

CHAPTER II

PERSPECTIVES ON METAPHORS AND ANALOGIES

2.1 INTRODUCTION

In this chapter we will briefly review some of the earlier work done on metaphors and analogies in the fields of philosophy, psychology, and Artificial Intelligence. As we pointed out in Chapter 1, a vast amount of literature is available on metaphors. To do justice to all the available literature on metaphors is beyond the scope of this thesis and therefore we will be focusing only on what we consider to be the more prominent theories and those that are more relevant to our theory. We hope that the works discussed here will provide the reader with a glimpse into the multi-disciplinary nature of the problem and a proper background for our own theory, which will be developed in the subsequent chapters.

We will begin by presenting a philosophical perspective on metaphors in the next section. We will briefly discuss the substitution theory, the comparison theory, and the anomaly theory of metaphors, and then describe Black's interaction theory at some length. We will then present a critique of MacCormac's theory, which uses fuzzy set theoretic notions to give semantics of metaphors. Then we will briefly mention Hesse's observations concerning scientific models and then discuss the perspective on metaphors provided by Turbayne. Finally, at the end of this section we will describe Hesse's theory of analogy in short.

In Section 2.3 we will present an Artificial Intelligence perspective on metaphors and analogies. We will mainly discuss Carbonell's and Hobbs' works and point out their inadequacies. We will then briefly mention Kling's model of analogical reasoning that proves theorems by analogy and then discuss Winston's theory of transfer frames. Finally, at the end of Section 2.3, we will discuss some general shortcomings of most of the AI approaches to analogy.

Next, in Section 2.4 we will discuss theories of metaphors and analogies put forth by psychologists. Our two main candidates will be Gentner's theory of structure mapping, and Tourangeau and Sternberg's domains-interaction theory which is a refinement of Black's interaction theory. We will describe both these approaches briefly and then point out their shortcomings.

In Section 2.5 we will recapitulate the discussion of the previous three sections by presenting a list of the features of metaphors that we take as a starting point of our theory. This list also provides desiderata for any theory of metaphors, in our opinion. Later in Chapter 4 we will show how many of these features of

metaphors can be characterized in our theory of Constrained Semantic Transference.

2.2 PHILOSOPHICAL PERSPECTIVES

Though the references to metaphors in philosophy can be traced back to Aristotle, we start our survey with Richards' [1936] observation that metaphors play a broader in language than only to impart beauty to prose and poetry. In his writings one can find roots of what was later to be developed into *interaction theory* by Black [1962a]. Since then an increasingly large number of philosophers have studied metaphors and provided many valuable insights. Useful annotated bibliographies of important works can be found in Shibles [1971] and Johnson [1981].

There are several ways to classify theories of metaphors. The classification that we will be using here is the one presented in Mooij [1976]. [S]he classifies different theories of metaphors into two classes. Firstly, there are monistic theories which are characterized by a loss of reference to the literal extension in a metaphorical expression. Examples of monistic theories are anomaly or connotation theories, for instance the Verbal-opposition theory of Beardsley [1962], which characterizes a metaphor as a kind of anomaly or absurdity and derives the "meaning" of the metaphorical expression on the basis of its connotation or intension; substitution theory according to which a metaphorical expression stands for some equivalent literal expression; and theories which maintain that the meaning of a metaphorical expression is not explainable at all [Cf. Foss 1949]. Then there are dualistic theories in which the reference to the literal extension is maintained in a metaphorical expression. Examples of dualistic theories are comparison theory [Henle 1958], and

interaction theory [Black 1962a].

Let us consider Beardsley's revised Verbal-opposition theory. According to this theory, on encountering a metaphorical expression like "smiling sun" one looks at the connotation or intension of "smiling", which is the set of all properties of "smiling", and then sees how many of them can be *safely* applied to "sun". Further, the metaphor will fall in one of two classes depending on whether there are already other established metaphors like "smiling sea", "smiling sky", and "smiling garden" or not. In the former case the metaphor is less creative because the connotation of "smile" is already narrowed down to those properties that are common to all the other "smiling" metaphors. If none of the other metaphors are well-known then on encountering "smiling sun" for the first time one has to do more work in weeding out the unsafe properties from the connotation of "smiling" and therefore the metaphor is more creative.

Several objections can be raised against any such theory. Firstly, it does not specify any criterion to decide what constitutes a safe application of a property to an object. Further, it does not explain why certain metaphors are more apt than others[1] or how they can possibly fail to communicate. Finally, it does not provide any distinction between metaphors and myth or any criterion for the notion of metaphorical truth.

---

1. Beardsley uses this as an argument against the comparison theory.

According to the substitution theory, the metaphor replaces some corresponding literal expression. For example, "the sky is crying" replaces the literal utterance "it is raining". It has been demonstrated several times that it is not always possible to replace a metaphorical expression by a literal expression, especially when the metaphor is novel and creates a new perspective. Further, it does not specify any principle to constrain the substitution, nor does it specify any means to reverse the substitution to arrive at the intended meaning.

The comparison theory of metaphor in a way is a special case of the substitution theory in that it provides the principle under which a substitution can be made. Thus, this view looks upon a metaphor as an elliptical simile. At the first sight it seems to solve the problem with the substitution view by providing a criterion for deciding when a substitution is good but on deeper investigation we notice that this is begging the question since it does not provide any standard to decide what is a good comparison and what is a bad one. Further objections against comparison theory can be found in Black [1962a] and Mooij [1976].

Wheelright [1962] suggested a theory of metaphor that combines the comparison view and the anomaly view. Wheelright divides metaphors into two classes called *epiphors* and *diaphors*. Epiphors arrive at their meanings through a process of comparison and an extension of the terms involved in the metaphorical utterance. On the other hand diaphors derive their meanings by juxtaposition and synthesis, i.e. they correspond to the anomaly view of metaphors. All the objections that we raised against both these theories also apply to Wheelright.

MacCormac [1982] suggested a theory of metaphors that defines the notions of epiphors and diaphors formally and also attempts to provide a criterion for deciding the aptness of metaphors. His formalization is based on fuzzy set theory [Zadeh 1965]. He uses semantic markers for constructing patterns of association. These semantic markers are represented as fuzzy sets. Traditionally, fuzzy set theory uses a membership function from the domain of discourse to the interval (0, 1) to define a set. The value of this function for an entity defines the degree of membership of that entity. The relation *belongs to* is defined by establishing a three-valued logic in the bounded relationship $0 \leq b < a \leq 1$ such that an entity e *belongs to* a set A iff the value of the membership function of A, applied to e, is greater than or equal to a, i.e. is in the interval (a, 1). Similarly, e *does not belong to* A iff the value of the function is less than or equal to b, i.e. is in the interval (0, b). If e does not *belong to* A and does not *does not belong to* A, i.e. the membership function of A applied to e returns a value in the interval [b, a], then e is said to be *indeterminate* with respect to A. Thus every expression evaluates to *true*, *false* or *indeterminate*.

MacCormac extends this three-valued logic system to a four-valued one by using the bounded relationship $0 \leq b < c < a \leq 1$. The intervals (0, b) and (a, 1) are identified with truth and falsehood as before but the middle *indeterminate* is broken into two separate intervals. The interval [b, c] gives rise to diaphors and the interval [c, a] gives rise to epiphors. Thus, according to this theory if a predicate is attributed to an object, and the membership function of that predicate returns a value between b and c for that object then the predicate is said to be

metaphorically attributed to the object. In particular, it will be an instance of diaphor. Of course, this is a somewhat simplified version of his theory but it is sufficient for our discussion since our criticism of his theory is directed against its use of fuzzy set theory rather than its technical details.

Our main objection against this theory is that according to it a metaphoric statement is *less* true than a corresponding literal statement, the degree of truth decreasing from epiphors through diaphors to plain falsehood. We strongly disagree with this consequence. A metaphorical statement is no more or less true than a literal statement. Of course a metaphor can be vague in the sense that it can be subject to several interpretations but given any one interpretation, in order to determine the truth or falsity of the metaphor the criteria to be applied are the same as the ones applied to literal statements.

One can perhaps raise an objection that what then differentiates a metaphorical statement from a literal statement. The answer, in our opinion, is that the difference lies in the fact that in a literal statement the predicate is applied to an object which is in the same realm as the one in which the predicate is *normally* applied whereas in a metaphorical statement the predicate and the object belong to different realms. When an object in one realm is metaphorically referred to by using the predicates of another realm then whether the resulting expression is true or not is largely determined by the objective properties of the referent and the antecedent practices of the predicate. This, we claim, is not a matter of degree. Thus "the sky is crying" is by no means less true or false than "it is raining" in some specific situation, given an appropriate interpretation of the metaphor. Of course, a

metaphorical expression can be vague in the same way as a literal expression can be vague, but this is not the sense in which MacCormac associates fuzziness with metaphors.

Our intuitions that the same criteria be used to determine the status of metaphorical truth as the ones used for determining literal truth concurs with observations made by Goodman [1968; 1978] concerning the nature of metaphorical truth. None of the theories of metaphors presented so far provide any criterion for metaphorical truth and metaphorical inference that agrees with this intuition. This is one of the major motivations for developing our theory of Constrained Semantic Transference, which does provide such a criterion, as we will see later in Chapter 4.

Next we will consider the interaction theory of metaphors, proposed by Black [1962a; 1979], at some length since our own theory of Constrained Semantic Transference is based on it. According to the interaction theory a metaphorical statement has two distinct subjects, to be identified as the *primary* and the *secondary* one. Thus, for instance in the metaphorical utterance "the sky is crying" the sky is the primary subject and crying is the secondary subject. In the rest of this thesis we will use the term *source domain* to refer to the secondary subject and *target domain* to refer to the primary subject. Both the domains are regarded as a system rather than an individual thing.[1] With both the domains there is associated an "implicative complex" which is a set of inferences that can be drawn about that domain. The

---

1. Initially in *Metaphor* [1962a] Black took this position but later on [1979] he found it "needlessly paradoxical" and required that only the source domain be viewed as a system. For reasons that we will explain later, we are keeping with the earlier intuitions of Black.

metaphorical utterance works by projecting upon the target domain the implicative complex of the source domain. The metaphorical utterance selects, emphasizes, suppresses, and organizes features of the target domain by applying to it those features of the source domain that are approximately isomorphic with the implicative complex of the target domain. Thus, on hearing a metaphorical utterance the primary and the secondary domains *interact* with each other in the sense that the hearer selects some features of the source domain implicative complex and constructs a parallel implicative complex that can fit the target domain. This process reciprocally induces changes in the source domain.

Our reasons for keeping with the earlier intuition of Black, which is to regard both the domains as a system of relationships, is that viewing the target domain also as a system of relationships can provide us with much needed coherency criteria to decide what parts of the implicative complex of the source domain fits the target domain. This will also lead to uniformity in the sense that one need not regard a domain as a thing sometimes and as a system at others. Further, Hesse [1974, Chap. 2] has provided some very convincing arguments to to show that meanings are best viewed as a network of relationships rather than an indivisible entity. Her arguments are based on the observation that all descriptive predicates are introduced, learned and understood either by direct empirical association in some physical situation, or by means of sentences containing other predicates which have been already so understood, or by a combination of both. Further, no predicates can

function by means of empirical association alone.[1] Since, Hesse's arguments are not derived from specific properties of literal or metaphorical usage, but are based on the nature of "meaning", we believe that by representing meanings as systems we can use the same representation to process a literal as well as a metaphorical utterance.

We do accept the interaction theory in spirit but our chief complaint is the vagueness of several notions like "implicative complex", "interaction", "fit" that are used in the theory. Also, it does not specify any criterion for deciding how to determine what features of the implicative complex of the source domain fit the implicative complex of the target domain. We attempt to remove all these deficiencies in our theory of Constrained Semantic Transference as well as characterize many other features of metaphors that cannot be explained in the interaction theory.

Hesse has consolidated further the interaction theory of metaphors by showing how it can explain the nature of scientific metaphors, especially the theory ladenness of observations and the process of theory change [Cf. Hesse 1963; 1980]. The relationship between metaphors and scientific models has also been noted and emphasized by several other philosophers, as we pointed out in the previous chapter. Here we will only mention the perspective on metaphors and models provided by Turbayne.

---

1. These theses have also been argued for earlier by Quine [1953].

Turbayne [1962], in his very impressive and well articulated thesis, has argued that metaphors play a very central role in science and cognition. He supported this argument by showing how the mechanistic metaphor, namely that the world is like a machine, used by Descartes, dominated Newton and a whole era of science after him. His thesis is that though metaphors may initially be used to describe an ill-understood target domain in terms of the more familiar source domain, they can easily turn into myth if one forgets the presence of metaphors and takes them literally. He characterized a metaphor as a theory that only partially describes a model, namely the target domain. This characterization is spelled out precisely by Eberle [1970] in model theoretic terms. Though in a sense our theory can be seen as an evolution of Eberle's early attempt, his characterization falls far short of being a theory of metaphors since it leaves many issues open and does not explain many important features of metaphors like highlighting and downplaying aspects of the target domain, providing new perspective on the target domain etc. It does not even show how metaphors can give rise to partial models.

Lakoff and Johnson [1980] have demonstrated by means of many examples not only the pervasiveness of the use of metaphors by speakers of English but also that the notion of coherency is very crucial to metaphors. Their discussion have also convinced us that there is an underlying structural analogy at the heart of every metaphor. Again, what is lacking in their account is a precise formulation of coherency which we overcome in this thesis.

Finally at the end of this section we would like to mention Hesse's theory of analogy that uses a notion of *semantic distance* to characterize analogies [Hesse 1959]. In Hesse's theory every object is represented by the set of its properties. An inclusion relation is defined among objects on the basis of the inclusion of properties of one object in the set of properties of the other object. Thus, the object *person* is included in the object *animal*. The set of all properties that any object can have forms a kind of distributive lattice which is called a C-lattice. Joins and meets of two objects are defined in the usual way as the joins and meets between their associated set of properties. The dimension of an object is defined to be the position it occupies in the C-lattice. Thus, an object will have a higher dimension if its set of properties includes most properties. The notion of distance between two objects is introduced as the difference between their dimensions.

Given this framework, Hesse's formulation of analogy is as follows. An analogy of the kind A : B :: C : D holds if the joins of pairs <A, B>, <C, D>, <A, C>, and <B, D> exist. For example, in the analogy *person* : *fish* :: *nose* : *gills* the object *animal* provides the join of *person* and *fish*, and the object *breathing-organ* is the join of *nose* and *gills*. Since *nose* and *gills* are included in *person* and *fish* respectively, *person* provides the join of *person* and *nose*, and *fish* provides the join of *fish* and *gills*. Therefore the analogy holds.

Hesse realizes that this simple characterization includes a great many analogies since any two pairs of objects have a join in a C-lattice. To provide a criterion for apt analogy she uses the notion of distance to define *degree of analogy*. Intuitively an analogy is more apt if the distance between the objects in the analogy and their

pairwise joins, mentioned above, is small.

Hesse's theory does not suffer from any lack of precision and our criticism of it is directed at the meta-level. Though we believe that this characterization does capture a certain class of analogies it is certainly not broad enough to cover the analogies that often underlie a metaphor. Very often a metaphor can create an analogy between concepts that are *semantically distant*. Actually, studies made by Tourangeau and Sternberg [1982] show that the more distant are the domains the better is the metaphor. We will discuss their work in more detail in Section 2.4.

2.3 ARTIFICIAL INTELLIGENCE PERSPECTIVES

Artificial Intelligence has been relatively quiet over metaphor and very few attempts have been made to come up with models of metaphor comprehension [Carbonell 1982; Hobbs 1979; Russell 1982]. We will discuss the work of Carbonell and Hobbs in this section, both of which fall rather short of being a theory of metaphors.

Carbonell [1982] presents a method of interpreting and analyzing metaphors which uses a small number of generalized metaphor schemas. Every generalized metaphor has four components.

1. Recognition Network: This is responsible for deciding if an utterance is an instance of the metaphor schema.

2. Basic Mapping: This is responsible for establishing a partial mapping from the features of literal input onto a different meaning by the metaphor. The mapping is partial because not all the features can be mapped.

3. Implicit-Intention Component: This encodes the intention of the creator of the metaphor so that the metaphorical utterance, apart from suggesting the meaning expressed by the corresponding literal utterance, creates an extra shade of meaning.

4. Transfer Mapping: This is responsible for mapping additional features of the literal input onto the conceptual representation after transforming them appropriately.

Based on a handful of such generalized metaphors, he suggests the following control structure for analyzing a metaphorical utterance.

1. Determine whether an utterance is literal or metaphorical. In the latter case go to the next step.

2. Use Recognition Networks of all the generalized metaphors to find out which one is instantiated. If one is found then retrieve the remaining three components of that metaphor and go to the next step, otherwise abort.

3. Build the semantic frame of the utterance using the Basic Mapping of the metaphor.

4. Use the Transfer Mapping to fill the slots of the semantic frame constructed in the last step.

5. Use the Implicit-Intention Component to add extra information to the semantic frame, as long as it does not contradict the existing information.

6. Store this instantiation of the generalized metaphor in the short term memory for further reference in the current dialogue.

We would like to raise a number of objections against such an approach. Our main objection is that in his model all the information concerning the metaphor is to be *explicitly* represented in the Basic Mapping, Implicit-Intention Component and Transfer Mapping. This defeats the very purpose of a theory of metaphor, namely to explain our ability to understand and generate *new* metaphors. If this approach were to be used in any reasonably large system, a large team of very creative

people will be required, working for years perhaps, to list all extant metaphor schemes, but even this would not exhaust the range of possible metaphors. Anyone who reads any poetry or is interested in the history of languages must be aware of the fact that even the most mundane and commonplace objects and words contain a wide potential for novel metaphorical applications.

Among Carbonell's four components of generalized metaphors, the Implicit-Intention Component is the most controversial one. A metaphor *never* carries the extra shade of meaning this component is supposed to represent *explicitly*. Different people, even if they share a very similar cultural background, may see more or less meaning in a metaphor depending on their taste. This part of meaning conveyed by metaphor is very subjective and therefore should not be predetermined. What a good theory or model of metaphor must have is some means of *creating* all these components and not expect them to be given.

Further, Carbonell seems to have some problem with understanding the meaning of the word "metaphor" itself. He remarks "Analogical mappings are usually *new* mappings, not necessarily known to the understander . . . . If a mapping is often used as an analogy, it may become an accepted metaphor; the explanatory requirement is suppressed if the speaker believes the listener has become familiar with the mapping . . . . A mapping abstracted from the interpretation of several analogies can become packaged into a metaphor definition" [p. 421]. This seems to me more appropriate if we substitute the word "dead metaphor" instead of "metaphor". Perhaps not surprisingly, most of the examples discussed in his paper, including the generalized metaphors, are cases of dead metaphors. It is only in cases

of dead metaphors that one can design the four components mentioned above. The whole paper makes much more sense if we read "metaphor" as "dead metaphor" throughout.

Some further observations on metaphors can be found in Carbonell & Minton [1983]. Here they posit a LIKE relation and *mapping structure* to represent information conveyed by a metaphor. None of these notions is clearly defined and it appears from illustrations that their notion of mapping structure is very restrictive in the sense that they only allow identical structure to be transferred from the source domain to the target domain. This corresponds to Gentner's theory of structure mapping [1983] which we will discuss in the next section, and which is a special case of our theory as we will see in Chapter 4.

We now turn to Hobbs' model of metaphor comprehension and selective inferencing [1979]. He proposes a model which takes as input a predicate calculus expression, obtained by some pre-processor which translates English statements into predicate calculus, and "draws those inferences necessary to solve the discourse problems posed by the text" [p. 8]. He uses "discourse operations" to select appropriate inferences from the data base which contains a large collection of axioms representing world knowledge. With each possible inference he associates a salience measure which determines which inferences are more relevant to the context. Thus, in his model the "meaning" of a word or expression is identified with the inference processes that are triggered by that word.

How his model draws inferences can best be explained by using an example. The data base contains inference rules, or axioms, like

1. hog(x) → fat(x)
2. hog(x) → overconsumes(x,y), food(y)
3. hog(x) → sloppy(x)
4. hog(x) → has-four-legs(x)

Now when presented with an expression like hog(J) where J represents John, the context will support inferences like John is a man, which, in conjunction with some other axioms representing world knowledge about "man", will derive inferences like "has-two-legs(J)" etc. The word "hog" will trigger the axioms presented above as possible candidates for drawing inferences. The first two axioms can be used directly with a high degree of salience since they do not contradict any existing facts established so far. The fourth one is rejected since it contradicts the existing knowledge represented by "has-two-legs(J)". Finally the third one can be used to draw the inference "sloppy(J)" and the degree of salience associated with this inference will depend on the context and on how much this new inference is supported by the pre-existing facts.

In his model the difference between a metaphorical and a literal utterance is not that of kind but that of degree. An expression is literal if most of the inferences associated with the words in the expression can be drawn. In a metaphorical expression the inferences associated with the words in that expression can be divided into three classes. Firstly there are those that can be drawn with a

very high measure of salience. Then there are those that cannot be drawn since they contradict some already existing facts. Finally there are those that can be drawn with a low degree of salience. The last class is responsible for the suggestiveness of metaphor and is very important from a cognitive point of view.

Next he explains how his model handles metaphor schemas in which the structure of one domain is transferred to the other, often less structured, domain. He proposes to represent this transference by a "collection of axioms that are intricately woven together by their reference to a small set of common predicates." [p. 13]. For example to represent the metaphor "a variable is an entity at a location" he uses the axiom:

(1) variable(x) & value(w,y,x) → at(w,x,y)

That is, if x is a variable and w is the condition of y being its value, then w is also the condition of x being at y.

These kind of axioms are to be used backwards since the speaker of the utterance applies the axiom in the forward direction and so in order to derive the intended meaning the listener has to reason backwards. Thus on encountering the utterance "the variable x is at 5", the hearer searches through all the axioms in the data base to find those that have *at* on the right hand side of the implication. Any such axiom provides a possible interpretation for the utterance. In this manner, the above axiom specifies one of the many ways in which one object can be *at* another.

This is to be contrasted with traditional "production-rule" approach or "discrimination-net" approach. In this approach the meaning of the *at* metaphor in the above utterance will be encoded by an axiom like

(2) $at(w,x,y) \ \& \ variable(x) \rightarrow value(w,y,x)$

That is, if w is the condition of x being at y and x is a variable, then w is also the condition of x having the value y. In this approach on encountering the utterance "the variable x is at 5" we will search through all the axioms to find those that have *at* in their antecedent and then check to see if the other conjuncts in the antecedent are also satisfied, and if so, conclude the consequent of the axiom as a possible interpretation of the metaphor. If this approach is taken to represent metaphors then we will end up having a Carbonell type of model in which all possible interpretations have to anticipated in advance. Another advantage of the former representation, with axioms like (1) above, is that it makes possible to establish many other metaphors since the axiom (1) above, for example, in conjunction with other axioms featuring "variable", "at" etc. can be used to construct novel metaphors.

In the third part of Hobbs' paper he generalizes his theory. Without going into details of his example, which shows how the baseball schema can be metaphorically evoked to explain the President's vetoes of bills that Congress has passed, we can explain his strategy briefly as follows. Every metaphor involves two domains which he calls the new domain and the old domain. The old domain is more familiar or better understood and the new domain is the one we are trying to

understand. In each domain there are basic or primitive concepts and relationships, and complex concepts and relationships that are made from the basic concepts and relationships. The difference between the basic and the complex concepts is characterized by the rules of inferences that we have talked about before. Hobbs argues that the mapping between the concepts of the old domain and the new domain can be considered to be the identity mapping [p. 26]. The crucial question, he claims, is to determine which of the inference rules in the old domain can be carried over to the new domain. He does not exactly offer a solution but considers a few alternatives.

Though Hobbs' work provides an important first step towards providing a criterion of coherency, one major limitation of his approach, from the first step to the last, is that it only works if both the source and the target domains are expressed in terms of the same predicates. Thus, in the first stage he required that the inference processes associated with both "hog" and "John" are expressed from a common set of predicates. Otherwise his consistency check will always work and all the inferences of the source domain will be transferable to the target domain. In the second stage he had the axiom explicitly representing the relation among "variable", "at", and "value". Creating axioms like this is part of the process of understanding metaphors which he does not address at all. Finally, in the last stage, by requiring that the mapping from the concepts of the source domain to the concepts of the target domain be the identity he only captures a very small class of mappings, like Carbonell & Minton [1983] and Gentner [1983].

Though AI has failed to provide any satisfactory theory or model of metaphor comprehension, there has been an abundance of theories and models of analogical reasoning. Again, the vastness of the available literature prevents us from doing justice to all the interesting models that have been implemented and we will only briefly discuss the models of Kling and Winston here. We hope that this will provide the reader some feel for the kind of approach taken by AI researchers towards the problem of analogy.

Kling [1971] built a model of analogical reasoning, called ZORBA-I, that proves theorems by analogy. ZORBA-I is presented with a theorem T1 to prove, an analogous theorem T (chosen by the user), and a proof of T. The aim is to produce a mapping from the predicates and the axioms used in the proof of T to other predicates and axioms such that under the mapping the proof of T transforms into a proof of T1. ZORBA-I does this in two stages. First it builds an initial one-to-one mapping from the predicates that appear in T to predicates that appear in T1. Then it extends this mapping to form a complete analogy. Since the aim is to produce a complete mapping in a reasonable amount of time, several heuristics are used in building the initial mapping as well in extending it. There is no separation between principles and heuristics and it is not easy to abstract away from ZORBA-I its underlying theory of analogy.

Winston [1978] implemented a model that computes a transfer frame from the source domain to the target domain, each represented in a frame-like language [Minsky 1975]. Frames can be seen as sets of slot-value pairs. For instance, a frame for the concept fox may look something like:

Frame Name	Slot	Value
Fox	a-kind-of	mammal
	color	gray
	cleverness	very-high

The program is presented with a source frame and a target frame which correspond to the source and the target domains. The purpose of the program is to compute a transfer frame that determines which slots of the source frame can be transferred to the target frame.

Winston's model also performs this task in two stages. In the first stage it creates several potentially useful transfer frames by analyzing the information of the source domain and its relatives. He suggests several heuristics that are helpful in hypothesizing transfer frames. For instance, if some slot has very-high or very-low as its value then one may take it to be salient in some sense and create a potential transfer frame that transfers that slot.

In the second stage the transfer frames generated in the last step are pruned by studying the target domain and its relatives by using some heuristics. We will not go into heuristics here since again our criticism is directed at the meta-level. As with Kling's model, there is no way to distinguish principles from heuristics and one finds it hard to see what Winston's theory of analogy would look like after being abstracted from his program. Complications also arise since it is so heavily based on frames that if one chooses some other kind of representation then one will have to start all over again to build a model like Winston's.

Other AI models of analogy can be found in Mansour [1983] and Winston [1981, 1982]. What all these models share with ours is that they all are trying to compute analogical mapping from the domain knowledge, unlike Carbonell whose main concern is to use the information contained in a prescribed analogical mapping. The difference is, as we pointed out in the previous chapter, that their main concern is to present an acceptable analogy in their respective domains within certain time and space constraints, while our main goal at this stage is to lay down general principles for generating analogy that are domain independent. Of course, this is not to deny the important experience all these models have provided and which can be put to use in designing models based on our theory.

2.4 PSYCHOLOGICAL PERSPECTIVES

In this section we will discuss Gentner's theory of *structure mapping*, and Tourangeau & Sternberg's domains-interaction view. One important thing that distinguishes these two approaches from the ones presented earlier is that they both give a characterization of metaphorical aptness in their respective theories. We will explain these later in this section and also point out the limitations of their characterizations. Other psychological perspectives on metaphors can be found in Miller [1979] and Ortony [1979a].

In proposing her theory of structure mapping, Gentner [1982, 1983] starts from the following assumptions:

1. Domains and situations are psychologically viewed as systems of objects, object-attributes and relations between objects.

2. Knowledge is represented as propositional networks of nodes and predicates. The nodes represent concepts treated as wholes, called objects; the predicates applied to the nodes express propositions about the concepts or objects.

3. Two syntactic distinctions are made among predicate types. Firstly, the object attributes are distinguished from the relationships. Attributes are predicates that take one argument, for example LARGE, whereas relations are predicates that take two or more arguments, for instance GREATER-THAN. Secondly, the first-order predicates are distinguished from the higher-order predicates. The first-order predicates take only objects as arguments whereas the higher-order predicates also take propositions as arguments.

4. These representations, including the distinction between different kinds of predicates, are intended to reflect the way people construe a situation rather than what is logically possible.

Starting from these assumptions, she proposes her theory of structure mapping which provides rules for interpreting analogies. In her theory an analogy is characterized by a mapping of objects from the source domain to the objects of the target domain. Thus, to every object S_i in the source domain there is associated an object T_i of the target domain. The rules for constructing such a mapping from an analogy are as follows:

1. Discard attributes of the objects. Thus if S_i has some attribute A then T_i need not have the same attribute.

2. Try to preserve the relations between objects. Thus if some relation R holds between S_i and S_j then R should also hold between T_i and T_j.

3. In deciding which relations are preserved, choose those relations that appear as arguments to higher order relations. This is called the *systematicity principle*.

For instance, in the analogy "the hydrogen atom is like a solar system", the objects of the source domain, namely *sun* and *planets* are mapped onto *nucleus* and *electrons* of the target domain respectively. Here the relation *attracts* between *sun* and *planets* is preserved in the process of mapping whereas attributes of *sun* namely *heavy*, *big* are not preserved. Further, according to the systematicity principle, the relation *more-massive-than* is more likely to be preserved than say *hotter-than* since *more-massive-than* appears as an argument to some other higher-order relations that control the relationships among *sun* and *planets* according to Newton's laws. Thus, the systematicity principle formally captures the intuitive notion that since a structure mapping is supposed to transfer the structure of the source domain to the target domain the relations that are richly interconnected are more likely to be transferred. The systematicity principle also provides a criterion for deciding aptness of metaphors and analogies.

Though the object attributes are not required to be preserved, Gentner does use them to distinguish literal similarities, analogies, abstractions, and anomalies from each other. The criteria for distinction are as follows:

1. In a literal similarity many of the attributes of the source domain objects, as well as many of the relations among them, are preserved in the process of mapping.

2. In an analogy few of the attributes are preserved but many of the relations are preserved by the mapping.

3. Abstractions are the same as analogies except that they are characterized by the presence of very few object attributes in both the source and the target domains.

4. In anomaly few of the attributes and few of the relations are preserved.

Gentner also provides empirical and psychological evidence in support of her structure mapping theory and systematicity principle.

Though we agree in spirit with Gentner's theory we believe that by requiring that the relations be *preserved* only a certain class of analogies are captured. Our theory, as we will explain in the next chapter, does not distinguish between objects, attributes or relations and all of them can participate in the mapping process. This generalization has a consequence that we can no longer use the preservation of relations as a criterion for defining structure mapping across domains. We will develop coherency conditions in our theory to overcome this problem and show in Chapter 4 that preservation of relations is a special case of coherency.

Tourangeau and Sternberg [1982] proposed a domains-interaction theory of metaphors, which is a development of Black's interaction theory described earlier. According to the domains-interaction theory,

1. a metaphor involves seeing something in one domain (called tenor) in terms of something in another domain (called vehicle);

2. because features are often specific to a domain, they must be transformed, i.e. seen in a new way, if we are to find correspondences across domains;

3. since the features that structure the domains are reinterpreted or transformed by the metaphor, the whole domain, and not just a particular term in it, partakes in this conceptual "interaction";

4. either the context, or in default of this, the domains themselves, can provide the structure that makes salient the features or dimensions that figure in the interpretation of the metaphor; and

5. the domains involved place limitations on the manner by which features or dimensions applying within the domain of the vehicle can be altered so as to fit the tenor [p. 217].

Their theory also provides the following mechanisms for determining the correspondence between features of the source and the target domains.

1. There may be some common feature or dimension crosscutting the two domains;
2. we may abstract the features themselves;
3. the features or dimensions may be naturally correlated;
4. there may be punning connection or a common label linking them;
5. the dimensions may map onto a common absolute scale;
6. they both may relate to a mediating dimension in a third domain; and
7. the concepts may have similar network structures [p. 221].

In order to characterize the aptness of metaphors in their theory, Tourangeau and Sternberg distinguish between *within-domains* similarity and *between-domains* similarity among the source and the target domains. Within-domains similarity relates to the degree to which a correspondence can be established between features of the source domain and the features of the target domain in accordance with the mechanisms just listed. Between-domains similarity refers to the degree to which the two domains themselves resemble each other. Based on this distinction, their hypothesis is that aptness of a metaphor increases with within-domains similarity between the source and the target domains and decreases with the between-domains similarity. They also provide empirical and psychological evidence in support of this hypothesis.

Though Tourangeau and Sternberg's theory take a first step towards removing the vagueness and fuzziness of Black's interaction theory, it is still not rigorous enough to provide a basis for computational models of metaphors and analogies. Also, we believe that all seven above mechanisms for establishing a correspondence between the features of the source and the target domains are special cases of a more general mechanism of coherency which lies at the heart of our theory. We will discuss the relationship of their theory to our theory in Chapter 4.

2.5 DESIDERATA FOR A THEORY OF METAPHORS

We conclude this chapter by listing a number of features of metaphors that form the starting point for our theory.

1. There is some structural analogy underlying all metaphors and scientific models. Thus, every time we use the word "metaphor", we mean metaphors, analogies and scientific models.

2. Metaphors are characterized as an application of some term(s) belonging to one domain, called the source domain, to some term(s) belonging to another domain, called the target domain. In literature the source and the target domains are also referred to as the secondary and the primary subjects respectively.

3. The source and the target domain need not be different or disjoint. In other words, they may be the same or may partially overlap.

4. There is no basic unit of metaphors in discourse. Thus, a word, a phrase, a sentence, a passage or a whole book can be metaphorical.

5. The notion of coherency is very crucial to metaphors. In other words, aptness of a metaphor is determined by its coherency.

6. A metaphor can be subject to several interpretations. Thus a metaphor can miscommunicate.

7. Metaphors highlight certain parts of the target domain whereas they hide certain others.

8. The process of metaphorical inference and the notion of metaphorical truth are no different from their literal counterparts. Notice that we are not making any claim about what the nature of literal truth or literal inference is.

9. Sometimes one metaphor can express what would normally require several literal statements.

10. Metaphors redescribe the target domain in terms of the source domain. Thus different metaphors will yield different descriptions of the same target domain.

11. Metaphors play a very crucial role in the process of meaning change. Thus, a new metaphor through repeated use can fade into literal meaning or polysemy.

12. Both source and target domains participate in generating an interpretation of metaphors.

13. Metaphors often can create new insights and perspectives.

14. Since people can generate and comprehend metaphors, the cognitive processes involved are inherently computational.

These features are not arbitrary and we hope that the brief discussion of this chapter provided the reader with some justification or references to places where more articulated arguments can be found. We expect any theory of metaphors to explain all, or at least as many as possible, of these features.

In the next chapter we will develop our theory of Constrained Semantic Transference and in Chapter 4 we will discuss how several of the features of metaphors presented above can be explained in our theory.

"... Impenetrability! That's what I say!"

"Would you tell me, please," said Alice, "what that means?"

"Now you talk like a reasonable child," said Humpty Dumpty, looking very much pleased. "I meant by `impenetrability' that we've had enough of that subject, and it would be just as well if you'd mention what you mean to do next, as I suppose you don't mean to stop here all the rest of your life."

"That's a great deal to make one word mean," Alice said in a thoughtful tone.

"When I make a word do a lot of work like that," said Humpty Dumpty, "I always pay it extra."

— Lewis Carroll, "Alice Through the Looking Glass".

C H A P T E R III

A THEORY OF CONSTRAINED SEMANTIC TRANSFERENCE

3.1 INTRODUCTION

In the last chapter we stressed the pervasiveness of metaphors in natural languages and the lack of a systematic account of the process of understanding and creating a novel metaphor, or of a representation of the information that is conveyed by such a metaphor. As a first step towards a theory of metaphor we propose, in this chapter and the next, a theory of Constrained Semantic Transference [CST henceforth] that addresses the latter issue. Though in CST we do not directly concern ourselves with the problem of modeling the processes by which people comprehend or generate a novel metaphor, we do use it as a basis for developing a theory of Approximate Semantic Transference [AST henceforth] in Chapter 5 that addresses the issues of computability and finite representability. In Chapter 5 we will also show how AST can be used as a formal basis for developing models of

cognitive processes involved in understanding metaphors and analogies.

This chapter contains most of the formal development of CST, with some very simple examples, and in the next chapter we will show how CST can be used to characterize metaphors, analogies, and scientific models.

In the previous chapters we characterized metaphors as the application of one or more terms from the source domain to one or more terms of the target domain. We also pointed out that there is some underlying structural analogy between the source and the target domain at the core of most metaphors. In CST we will make all these notions more formal.

Before developing a theory of semantic transference we must define the notion of *domain* precisely and choose some formalism for representing the knowledge associated with a domain. In the next section we develop a formalism, called Schema Language, for representing domain knowledge. The formalism is based on the first order predicate calculus but differs from it in certain ways.

Sections 3.3 and 3.4 contain all the formal details of CST. In Section 3.3 we formally define the notion of an admissible coherent partial mapping, called T-MAP, from terms of the source domain to terms of the target domain that lies at the heart of CST. A T-MAP makes it possible to transfer part of the structure of the source domain to the target domain coherently. We will also discuss some formal properties of T-MAPs in this section.

The basic concept of a T-MAP, as defined in Section 3.3, is analogous to that of a homomorphism between algebras and will not be of so much interest as such. What makes it more interesting are the two recursive operators introduced in Section 3.4 that extend the notion of a T-MAP. These operators are called Augmentation and Positing Structure and they make it possible to create new structure in the target domain, constrained by the structure of the source domain.

In Section 3.5 we will make some remarks on extending SL to allow higher order predicates and on relaxing admissibility requirement on T-MAPs. Finally in Section 3.6 we will summarize the main points of this chapter.

3.2 REPRESENTING DOMAIN KNOWLEDGE: SL

In this section we develop a formalism, called Schema-Language [SL henceforth], for representing domain knowledge. SL is based on the language of the first order predicate calculus. In Section 3.2.1 we will describe SL informally followed by the formal definitions in Section 3.2.2. In Section 3.2.3 we will give examples of two domains, DAG and FAMILY, in SL. These domains will also serve as a basis for examples in subsequent sections.

3.2.1 AN INFORMAL INTRODUCTION TO SL

A domain in SL consists of a set of *tokens* and a set of *structural constraints*. A token can be an individual constant, an n-place operation, or an n-place predicate. With every token we associate a *type*. The type of a token contains the information as to whether the token is an individual constant, an operation or a predicate. If it is an operation or a predicate its type also contains its arity.

Whereas the set of tokens associated with a domain forms the vocabulary of the domain, its structural constraints show how the various tokens are interrelated. In other words the structural constraints put constraints on the meaning of different tokens. Constraints are expressed by sentences in the predicate calculus. For instance in the domain DAG [Cf. Section 3.2.3] the first structural constraint says that every *root* is a *node*, thereby relating the meaning of *node* to that of *root*.

So far our description of a domain is analogous to the model theoretic notion of a *theory*. There are two important features of structural constraints that make them somewhat different. Firstly, we do not require that every token appearing in any structural constraint must belong to the set of tokens associated with the domain. The reason for this is that it is very rare outside the realm of pure mathematics that one can find isolated and rather self contained domains. Often our knowledge of a domain is richly interconnected with our knowledge of several other domains. The structural constraints of a domain also contain information about how the tokens of a domain interrelate with the tokens of the other domains. Therefore, we allow the tokens belonging to the other domains to appear in the structural constraints of a domain. Such tokens are called *external* and in SL we require that

all external tokens be declared in the definition of the domain along with their types. The domain of an external token need not be known.

The second distinguishing feature of SL is a special subclass of structural constraints that play a very crucial role in our theory of CST. These constraints show how it is possible to derive some tokens by using other tokens. For instance, in the domain DAG [Cf. Section 3.2.3] the eighth constraint shows how the token *leaf* can be derived from the tokens *node* and *arc*. The token that is shown to be derivable by any such constraint is called *derived*.

There are two ways in which derived tokens are different from *definitions* of predicate calculus. Firstly, a derived token of a domain in SL is not "non-primitive" in any absolute sense. A token Y can be derivable by using token X and at the same time the token X may also be derivable by using Y. If the token X is acting as "primitive" then the derivation of Y can be evoked and if the token Y is acting as primitive then the derivation of X can be used. Of course, both X and Y can be acting as primitive at the same time and none of their derivations need be used but they cannot both be non-primitive at the same time unless they have other possible derivations. In the domain FAMILY [Cf. Section 3.2.3] the tokens *parent* and *child* are examples of two such tokens that are derivable by using each other. This is a very crucial feature of SL because this makes it possible to represent knowledge associated with a domain without deciding in advance what its primitives are. A domain may admit of several possible descriptions and several choices of primitives and an SL representation of that domain contains all this information without any bias. For instance, the domain FAMILY can be completely described by using tokens

male, *female*, and *child* as primitives, i.e. all other tokens can be derived from them, or by using tokens *mother*, *father*, *son*, and *daughter* as primitives. This richness of representation is necessary for any theory of metaphor because, as we pointed out in the last chapter, a metaphor often results in a redescription of the target domain in terms of the source domain, and therefore the representation of the target domain must be describable, at least partially, in different possible ways.

A second related feature is that we allow more than one possible derivation of a derived token in a domain. For instance, in the domain FAMILY [Cf. Section 3.2.3] the token *parent* can be derived in several different ways. It can be derived from the tokens *mother* and *father* and also from the token *child*. The reason for including all the derivations in the definition of the domain is that we do not know in advance what the primitives of the domain will be and which derivations will be useful. This flexibility is required for a theory of metaphor because different metaphors structure the same target domain in different ways by choosing different primitives and using different derivations.

Finally, we require that the set of structural constraints of a domain be consistent and closed under logical entailment though it need not be complete. Thus, if we consider the set of structural constraints as a theory of predicate calculus—with the external tokens added to the vocabulary of the theory—then the consistency requirements makes sure that the theory has a model. Closure assumption amounts to saying that if a statement X logically follows from the structural constraints of a domain then it must be included in the set of structural constraints of that domain. This is a very strong assumption and will be used to derive some

formal properties of CST in Section 3.3. In the next chapter we will discuss the limitations of CST due to the closure requirement and discuss why most of the "real-world" domains fail to meet the closure condition. In Chapter 5 we will remove this restriction and study mappings between domains that are not closed but are finite.

3.2.2 FORMAL SPECIFICATIONS OF SL

In this section we make the discussion of the previous section more formal. The presentation of this section is complete in the sense that no formal background is assumed and all the concepts used are clearly defined but a reader unfamiliar with logic may find it unsatisfactory due to lack of discussion. Any reader seeking a thorough treatment, with many examples, of some concepts used in this section is referred to some standard text book on logic [Cf. Carnap 1958; Chang & Keisler 1973; Mendelson 1979; Smullyan 1968; Suppes 1957]. On the other hand, readers familiar with logic can skim the next few pages lightly until we present the definition of domain.

We begin by repeating some familiar definitions from model theory and predicate calculus. T is a denumerable set of types $\{e,t,o_1,o_2,...,p_1,p_2,...\}$. Here e is the type corresponding to individuals, t is the boolean type, o_n is the type of n-place operations, and p_n is the type of n-place predicates.[1] We let Var be a denumerable set of variables of type e. A vocabulary is defined to be a pair

---

1. In Section 3.5 we will describe some different sets of types that also allow higher order predicates and operations.

<V, f>, where V is a set of tokens and f is a function that assigns a type to every token, i.e. f: V → T. Often we will implicitly assume the existence of a type assignment function associated with every set of tokens and will not mention it explicitly. We now recursively define the set of *terms* over a vocabulary V as follows:

1. If X ∈ Var then X is a term.
2. If X ∈ V and f(X) = e then X is a term.
3. If X ∈ V and f(X) = o_n for any n, and X1,...,Xn are terms, then X(X1,...,Xn) is a term.
4. Nothing else is a term.

Next we give the usual recursive definition of the set of *formulas* over V as follows:

1. If X ∈ V and f(X) = t then X is a formula.
2. If X and Y are terms then X=Y is a formula.
3. If X ∈ V and f(X) = p_n for any n, and X1,...,Xn are terms, then X(X1,...,Xn) is a formula.
4. If X is a formula then ~X is a formula.
5. If X and Y are formulas then X&Y, XvY, X→Y, and X↔Y are all formulas.
6. If X is any formula and x is a variable then (x)X and (Ex)X are formulas.
7. Nothing else is a formula.

We call X an *expression* if it is either a term or a formula. The set of expressions is called Exp. Now we associate a set of *free variables* with every expression by defining a function FV: Exp → 2^{Var}. If X is a term then FV(X) is determined as follows:

1. If X ∈ Var then FV(X) = {X}.
2. If X ∈ V then FV(X) = Ø.
3. If X is of the form Y(Y1,...,Yn) then FV(X) = FV(Y1) ∪...∪ FV(Yn).

If X is a formula then FV(X) is determined in the following manner:

1. If X ∈ V then FV(X) = Ø.
2. If X is of the form Y=Z then FV(X) = FV(Y) ∪ FV(Z).
3. If X is of the form Y(Y1,...,Yn) then FV(X) = FV(Y1) ∪...∪ FV(Yn).
4. If X is of the form ~Y then FV(X) = FV(Y).
5. If X is of the form Y&Z, YvZ, Y→Z, or Y↔Z then FV(X) = FV(Y) ∪ FV(Z).
6. If X is of the form (x)Y or (Ex)Y then FV(X) = FV(Y) - {x}.

We call X a *sentence* if X is a formula and FV(X) = Ø. Also a variable x ∈ V is said to be *free* in a formula Y if x ∈ FV(Y).

We are now ready to introduce some semantic notions. Given a vocabulary <V, f>, a *model* of V is a pair <A, F> where A is some nonempty set and F is a function which maps every member X of V as follows:

1. If $f(X) = e$ then $F(X) \in A$.
2. If $f(X) = t$ then $F(X) \in \{0, 1\}$.
3. If $f(X) = p_n$ then $F(X) \subseteq A^n$.
4. If $f(X) = o_n$ then $F(X) \in \{A^n \rightarrow A\}$.

Here by A^n we mean the cartesian product of A with itself n times, and by $\{A \rightarrow B\}$ we mean the set of all functions from A to B.

We will now define the notion of *truth* in a model. Let $M = \langle A, F \rangle$ be a model of vocabulary $\langle V, f \rangle$, and let g be a variable assignment function, i.e. g assigns an element of A to every variable in Var. We first recursively define a denotation function d, with respect to M and g, that assigns an element of A to every term over V.

1. If $X \in Var$ then $d(X) = g(X)$.
2. If $X \in V$ and $f(X) = e$ then $d(X) = F(X)$.
3. If X is of the form $Y(Y1,...,Yn)$ then $d(X) = F(Y)(d(Y1),...d(Yn))$. We recall that if X is of the form $Y(Y1,...Yn)$ then $f(Y) = o_n$ and therefore $F(Y)$ will be a function from A^n to A.

Thus given the denotation of every term over V in a model M, we can now recursively define the truth function m, with respect to M and g, that assigns an element of the boolean set {0, 1} to every formula over V in the following fashion:

1. If $X \in V$ and $f(X) = t$ then $m(X) = F(X)$.
2. If X is of the form $Y = Z$ then $m(X) = 1$ iff $m(Y) = m(Z)$.

3. If X is of the form ~Y then m(X) = 1 iff m(Y) = 0.

4. If X is of the form Y&Z then m(X) = 1 iff m(Y) = 1 and m(Z) = 1.

5. If X is of the form YvZ then m(X) = 1 iff m(Y) = 1 or m(Z) = 1.

6. If X is of the form Y→Z then m(X) = 1 iff m(Y) = 0 or m(Z) = 1.

7. If X is of the form Y↔Z then m(X) = 1 iff m(Y) = m(Z).

8. If X is of the form (Ex)Y then m(X) = 1 iff there is some a ∈ A such that m(Y) = 1 with respect to M and g′ where g′ is a variable assignment which is exactly like g except that g′(x) = a.

9. If X is of the form (x)Y then m(X) = 1 iff for every a ∈ A m(Y) = 1 with respect to M and g′ where g′ is a variable assignment which is exactly like g except that g′(x) = a.

If X is a formula of V then we say that X is *true* in M iff the truth function m, with respect to M and g, yields m(X) = 1 for every variable assignment g.

Given the notion of truth in a model, we can now define some other related concepts. A sentence X over V is a *tautology* if X is true in all models of V. X is *contradictory* if X is not true in any model of V. If S is a set of sentences over V then M is said to be a model of S iff M is a model of V and every sentence in S is true in M. S is said to be *consistent* if it has a model and *inconsistent* if it is not consistent, i.e. if it has no models. A sentence X is a *consequence* of a set of sentence S iff X is true in every model of S.

At this point we introduce the notation (E!x)X, where x is a variable and X is a formula, to mean that "there is exactly one x such that X" as follows. If X is any formula, and Y and Z are variables such that Z does not occur in X, then by X(Y/Z) we denote the formula obtained from X by replacing all free occurrences of Y, if any, in X by Z. Obviously if X is a sentence, i.e. it does not contain any

introduced earlier. The definitions given here are adapted from Suppes [1957]. Given a set of tokens V, a type assignment function f: V → T, and a set S of sentences over V, we say that a sentence X (X ∈ S) introduces a token Y (Y ∈ V) iff,

1. If f(Y) = p_n then X is of the form Y(x1,...,xn) ↔ Z such that x1,...,xn are distinct variables; FV(Z) ⊆ {x1,...,xn}; and Y does not occur in Z.

2. If f(Y) = o_n then X is of the form Y(x1,...,xn) = x ↔ Z such that x,x1,...,xn are distinct variables; FV(Z) ⊆ {x,x1,...,xn}; Y does not occur in Z; and there is a subset of S, say S_1, such that none of the sentences in S_1 contain Y and the sentence (x1)...(xn)(E!x)Z is a consequence of the sentences of S_1. This last condition is necessary to ensure that the definition introduces the token uniquely.

Now we are ready to formally define our concept of domain. A *domain* is a quintuple $<V_I, V_E, S_C, S_D, f>$ such that,

1. V_I and V_E are sets of tokens such that none of them are variables and moreover, $V_I \cap V_E = \emptyset$.
2. $f: V_I \cup V_E \rightarrow T$, is a type assignment function.
3. S_C is a set of sentences over the tokens $V_I \cup V_E$.
4. $S_D \subset S_C$ such that every sentence in S_D is a derivation and the token introduced by any such derivation is in V_I.

Intuitively, V_I is the internal vocabulary of the domain, V_E is the external vocabulary of the domain, S_C is the set of structural constraints, S_D is the set of derivations, and f is the type assignment function.

Now we introduce some axioms that one may require the domains to satisfy.

Axiom of Consistency: The set of sentences S_C is consistent.

Axiom of Relevance: Every sentence in S_C contains at least one token from V_I.

Axiom of Inclusion: Every token in $V_I \cup V_E$ appears in at least one sentence of S_C.

Axiom of Significance: None of the sentences in S_C are tautologies.

Axiom of Closure: If X is a consequence of S_C then X must be included in S_C.

Most often one would require the domains to satisfy the first four axioms stated above. We refer to such domains as *canonical domains*. The axiom of closure, as stated above, is never consistent with the axioms of relevance and significance. But one can modify it to require *relevant closure*, i.e. if X is deducible from S_C and X is relevant then X must be in S_C; *significant closure*, i.e. if X is deducible from S_C and X is significant then X must be included in S_C; or *relevant significant*

closure, i.e. if X is deducible from S_C and X is both relevant and significant then X must be included in S_C. In the next chapter we will discuss the effects of the closure axiom on the representability of domains.

At the end of this section we would like to remark that the use of logic in SL for representing knowledge is motivated by Hayes' [1975] work on axiomatizing commonsense knowledge associated with certain domains. We have also been influenced by Hesse's network theory of meaning [Hesse 1974] in designing SL. Another significant feature of SL, worth mentioning here, is that we have developed it independently of the existing AI knowledge representation schemes like Frame [Minsky 1975], KL-One [Brachman 1978], or Krypton [Brachman *et al.* 1983]. This was done deliberately so that we can focus our attention on those aspects of the representation that are crucial to a theory of metaphors. We believe that SL is compatible with several of these AI knowledge representation formalisms in that the features of SL can be built in Frames, KL-One, or Krypton. To carry out this task is a topic of future research.

This completes our formal specifications of SL. We now give examples of two domains that are represented in SL. The following chapters contain more examples of SL domains.

3.2.3 EXAMPLES OF TWO DOMAINS IN SL

In this section we present two domains that are represented in SL. The domains chosen are small but rich enough to bring out most of the advantages of SL as well as its limitations. These two domains will also serve as examples in the rest of this chapter. Examples of some other domains in SL are included in the following chapters.

In the following discussion we take {u,v,w,x,y,z} $\subseteq$ Var and X,Y,Z are used as the variables of the meta-language. Also, in giving example of a domain we will follow the practice of first giving the formal representation followed by comments.

DOMAIN 1: DIRECTED ACYCLIC GRAPHS (DAG)

This domain is specified as <I1, E1, S1, d1, f1> where

1. I1 = {node, root, leaf, arc}
2. E1 = $\varnothing$
3. f1(X) = p_1 if X = root, node or leaf; f1(arc) = p_2
4. S1 is the significant closure of the following set of sentences:
    1. (x) [root(x) → node(x)]
    2. (x) [leaf(x) → node(x)]
    3. (x)(y) [arc(x,y) → [node(x) & node(y)]]
    4. (x) [node(x) → ~arc(x,x)]
    5. (x)(y) [arc(x,y) → ~arc(y,x)]

6. (x)(y)(z) [[arc(x,y) & arc(y,z)] → ~arc(x,z)]
7. (x)(y) [arc(x,y) → (z) [arc(z,y) ↔ z=x]]
8. (x) [leaf(x) ↔ (y) [node(y) → ~arc(x,y)]]
9. (x) [root(x) ↔ (y) [node(y) → ~arc(y,x)]]

5. d1 contains the following two sentences, both of which are derivations:
   1. (x) [leaf(x) ↔ (y) [node(y) → ~arc(x,y)]]
   2. (x) [root(x) ↔ (y) [node(y) → ~arc(y,x)]]

Comments: DAG is an example of a self-contained domain i.e. a domain that has no external tokens. The second example in this section is also a self-contained domain. The following chapters contain examples of domains that are not self-contained and have many external tokens. We notice that DAG satisfies all the axioms mentioned in the previous section except the closure axiom (we only required significant closure) and is therefore canonical. Another thing worth noting about this axiomatization is that by requiring atransitivity of *arc* (structural constraint 6 above) we allow only single-rooted trees, forest of trees, and rootless trees. If we want to restrict ourselves further by allowing only single-rooted trees, we will have to add the following sentence to S1:

(Ex) [root(x) & (y) [root(y) ↔ x=y]]

We will refer to the domain that is obtained by adding the above sentence to the structural constraints of DAG as TREE.

DOMAIN 2: FAMILY

This domain is specified as <I2, E2, S2, d2, f> where

1. I2 = {person, male, female, mother, father, parent, child, son, daughter, sibling, brother, sister}
2. E2 = $\emptyset$
3. f2(X) = p_1 if X = person, male or female; f2(X) = p_2 otherwise
4. S2 is the significant closure of the following sentences:
    1. (x) [person(x) ⟷ [male(x) v female(x)]]
    2. (x) [male(x) → ~female(x)]
    3. (x)(y) [parent(x,y) ⟷ [mother(x,y) v father(x,y)]]
    4. (x)(y) [mother(x,y) ⟷ [female(x) & parent(x,y)]]
    5. (x)(y) [father(x,y) ⟷ [male(x) & parent(x,y)]
    6. (x)(y) [mother(x,y) → (z) [mother(z,y) ⟷ z=x]]
    7. (x)(y) [father(x,y) → (z) [father(z,y) ⟷ z=x]]
    8. (x)(y) [parent(x,y) ⟷ child(y,x)]
    9. (x)(y) [child(x,y) ⟷ [daughter(x,y) v son(x,y)]]
    10. (x)(y) [daughter(x,y) ⟷ [female(x) & child(x,y)]]
    11. (x)(y) [son(x,y) ⟷ [male(x) & child(x,y)]]
    12. (x)(y) [sibling(x,y) ⟷ [x≠y & (Ez) [parent(z,x) & parent(z,y)]]]
    13. (x)(y) [sibling(x,y) ⟷ [sister(x,y) v brother(x,y)]]
    14. (x)(y) [sister(x,y) ⟷ [female(x) & sibling(x,y)]]
    15. (x)(y) [brother(x,y) ⟷ [male(x) & sibling(x,y)]]

16. (x) [person(x) → ~X(x,x)] where X is mother, father, parent, child, son, daughter, sibling, brother, or sister.

17. (x)(y) [X(x,y) → ~X(y,x)] where X is mother, father, parent, child, son, or daughter.

18. (x)(y)(z) [[X(x,y) & X(y,z)] → ~X(x,z)] where X is mother, father, parent, child, son, or daughter.

5. d2 contains the following sentences, all of which are derivations:

    1. (x) [person(x) ↔ [male(x) v female(x)]]

    2. (x) [female(x) ↔ [person(x) & ~male(x)]]

    3. (x)(y) [parent(x,y) ↔ [mother(x,y) v father(x,y)]]

    4. (x)(y) [parent(x,y) ↔ child(y,x)]

    5. (x)(y) [mother(x,y) ↔ [female(x) & parent(x,y)]]

    6. (x)(y) [father(x,y) ↔ [male(x) & parent(x,y)]

    7. (x)(y) [child(x,y) ↔ [daughter(x,y) v son(x,y)]]

    8. (x)(y) [child(x,y) ↔ parent(y,x)]

    9. (x)(y) [daughter(x,y) ↔ [female(x) & child(x,y)]]

    10. (x)(y) [son(x,y) ↔ [male(x) & child(x,y)]]

    11. (x)(y) [sibling(x,y) ↔ [x≠y & (Ez) [parent(z,x) & parent(z,y)]]]

    12. (x)(y) [sibling(x,y) ↔ [sister(x,y) v brother(x,y)]]

    13. (x)(y) [sister(x,y) ↔ [female(x) & sibling(x,y)]]

    14. (x)(y) [brother(x,y) ↔ [male(x) & sibling(x,y)]]

Comments: As we remarked earlier, FAMILY is also a self-contained domain and therefore has no external tokens. Also, like DAG it is a canonical domain. One of

the significant thing about this domain is that it admits of several choices of "primitives". Thus FAMILY can be completely described by several sets of primitives. Some of the choices are as follows:

1. {male, female, parent}
2. {male, female, parent, sibling}
3. {male, female, child, sibling}
4. {person, male, mother, father, sibling}

We would like to draw the reader's attention to the fact that the "primitives" need neither be mutually independent nor be minimal in any sense. Another thing to note about the domain FAMILY is the multiple derivations of derived tokens. Which derivations are to be used will depend on the primitives that are being used in describing the domain. Finally, we remark that in our axiomatization we do not require that every person has a mother and a father, though we do require that they be unique if they exist.

3.3 INTRODUCING COHERENT PARTIAL MAPPINGS IN SL: T-MAPS

In this section we formally introduce T-MAPs which are partial coherent mappings from the tokens of a source domain to the tokens of a target domain. We first define *mapping* followed by *admissible mapping* and finally *coherent mapping* or T-MAP. All the definitions are followed by very simple examples and discussion, if necessary. More examples of T-MAPs will be presented in the following chapters as and when necessary. Also, at the end of this section we will describe some of the

formal properties of T-MAPs.

Definition: Given a source domain D1 = <I1, E1, S1, d1, f1> and a target domain D2 = <I2, E2, S2, d2, f> a *mapping* from D1 to D2 is a partial function from I1 to I2.

Thus a mapping maps some of the internal tokens of the source domain to internal tokens of the target domain. We will represent mappings as a set of pairs, a pair being (i1 → i2) where i1 ∈ I1 and i2 ∈ I2. If F is a mapping and a pair (i1 → i2) ∈ F then we will often use the notation i1 ∈ F and i2 ∈ F if it is clear from the context whether we are talking about a token from the source domain or from the target domain.

If we take the source domain to be FAMILY and the target domain to be DAG then the followings are some examples of mappings from FAMILY to DAG. As we will see later, not all of them are admissible and coherent.

- FD1: {(brother → node); (mother → arc)}
- FD2: {(male → node); (mother → arc)}
- FD3: {(male → node); (father → arc)}
- FD4: {(person → node); (father → arc)}

Definition: A mapping F from the source domain D1 = <I1, E1, S1, d1, f1> to the target domain D2 = <I2, E2, S2, d2, f> is called *admissible* iff for every i ∈ I1, if i is included in F then f1(i) = f(F(i)).

Thus, an admissible mapping preserves types[1] of the tokens that are mapped. In the examples above, FD1 is not admissible since f1(brother) = p2 and f2(node) = p1. All the remaining mappings are admissible.

An admissible mapping allows part of the structure of the source domain to be transformed to a form that can be a *possible* structure of the target domain. We capture this notion in the following definitions:

Definition: Given an admissible mapping F from the source domain D1 = <I1, E1, S1, d1, f1> to the target domain D2 = <I2, E2, S2, d2, f>, we say that a structural constraint S ∈ S1 is *transformable* by F iff for every token X that appears in S the following conditions hold,

1. if X ∈ I1 then X ∈ F, and
2. if X ∈ E1 then X ∈ E2.

Definition: Given an admissible mapping F from the source domain D1 = <I1, E1, S1, d1, f1> to the target domain D2 = <I2, E2, S2, d2, f>, we say that a structural constraint s1 ∈ S1 is *transformed* to s2 by F iff s2 is obtained from s1 by replacing every such token X in s1 that is in I1 by F(X). We will often use the notation F(s1) to refer to s2.

Thus, a structural constraint of the source domain can be transformed by an

---

1. Preservation of types is a sufficient but not necessary condition for a mapping to be admissible. We will discuss this in more detail in Section 3.5, when we present another definition of admissibility that does not require that types be preserved but requires that the type structure be preserved.

admissible mapping by replacing every internal token in that structural constraint by the corresponding token as given by the mapping. For instance, the following is a structural constraint of the domain FAMILY that is transformable by FD2:

(1) $(x)\ [mother(x,y) \rightarrow \sim male(x)]$

This is transformed by FD2 as:

(2) $(x)\ [arc(x,y) \rightarrow \sim node(x)]$

As we will see later, this contradicts the structural constraints of DAG and therefore FD2 is not coherent with respect to this structural constraint.

We can easily extend these definitions to sets of structural constraints. Thus, if F is an admissible mapping from the source domain D1 = <I1, E1, S1, d1, f1> to the target domain D2 = <I2, E2, S2, d2, f> and $S \subseteq S1$ then we say that S is transformable by F if every structural constraint in S is transformable by F. We use the notation F(S) to mean the set of structural constraints obtained by transforming every structural constraint in S by F. Further, we say that S is *maximal* with respect to F iff every structural constraint of S1 that is transformable by F is in S.

The admissibility requirement on a mapping ensures that the transformed structural constraint is "meaningful" in the target domain but it does not mean that it actually holds in it. This brings us to our definition of coherency.

In order to define *coherency* of an admissible mapping F with respect to a transformable set of structural constraint S, we examine how F(S) interacts with S2, the structural constraints of the target domain D2. There are three possible outcomes of the interaction of F(S) with S2. The first possibility is that there are one or more transformed structural constraints in F(S) that contradict one or more structural constraints in S2. In this case we say that F is *incoherent* with respect to S. For instance, the structural constraint of FAMILY, mentioned above in (1), that is transformed by FD2 contradicts explicitly the following structural constraint of DAG:

(3) $(x)(y)\ [arc(x,y) \rightarrow [node(x)\ \&\ node(y)]]$.

Thus, FD2 is not coherent with respect to any set of structural constraints of FAMILY that includes the structural constraint (1) above.

The second possible outcome is that every transformed structural constraint in F(S) is a consequence of the structural constraints in S2. In this case we say that F is *strongly coherent* with respect to S. If S is maximal with respect to F then we say that F is *maximal strongly coherent*. As reader can verify, both FD3 and FD4 are maximal strongly coherent. In the following chapters we will present some more examples of strongly coherent mappings.

The third possibility, a rather interesting one, is that F(S) is consistent with the structural constraints in S2. In this case F is said to be *weakly coherent* with respect to S. Here again, if S is maximal with respect to F then we say that F is *maximal weakly coherent*. An example of a weakly coherent mapping, that is not strongly coherent, is the following admissible mapping from DAG to FAMILY:

- DF5: {(arc → parent); (node → person)}

which is maximal weakly coherent. To see why this mapping is not strongly coherent, consider the following structural constraint from DAG:

(4) $(x)(y)\ [arc(x,y) \rightarrow (z)\ [arc(z,y) \leftrightarrow z=x]]$

This is transformed by DF5 to be:

(5) $(x)(y)\ [parent(x,y) \rightarrow (z)\ [parent(z,y) \leftrightarrow z=x]]$

This transformed structural constraint is not a consequence of the structural constraints of FAMILY but is consistent with them. Another example is the following admissible mapping from TREE to FAMILY:

- TF1: {(arc → mother); (node → female)}

which is maximal weakly coherent. To see why this mapping is only weakly coherent, consider the following structural constraint from TREE:

(6) $(Ex)\ [root(x)\ \&\ (y)\ [root(y) \leftrightarrow x=y]]$

Since TREE is significantly closed, the following sentence, that is obtained by rewriting the above structural constraint using the derivation for *root*, is also a structural constraint of TREE.

(7) $(Ex)\ [(y)[node(y) \rightarrow \sim arc(y,x)]\ \&\ (y)[(z)[node(z) \rightarrow \sim arc(z,y)] \leftrightarrow x=y]]$

Transforming this with TF1 results in:

(8) $(Ex)\ [(y)[female(y) \rightarrow \sim mother(y,x)]\ \&$
$(y)[(z)[female(z) \rightarrow \sim mother(z,y)] \leftrightarrow x=y]]$

One can verify that this transformed structural constraint is not a consequence of the structural constraints of FAMILY but it is not contradicted either. In other words it is consistent with the structural constraints of FAMILY. Therefore, if TF1 is strongly coherent with respect to any set S, which is a subset of the set of structural constraints of the domain TREE, then by adding structural constraint (7) to S we get a weakly coherent mapping. In particular TF1 is maximal weakly coherent. We now state all these definitions formally.

Definition: Given an admissible mapping F from the source domain D1 = <I1, E1, S1, d1, f1> to the target domain D2 = <I2, E2, S2, d2, f>, and a set of structural constraints S of D1 that is transformable by F ($S \subseteq S1$), we say that F is *weakly coherent* with respect to S iff $F(S) \cup S2$ is consistent; and F is *strongly coherent* with respect to S iff every sentence in F(S) is a consequence of S2. Moreover, if S is maximal with respect to F then we say that F is *maximal weakly coherent* or *maximal strongly coherent*, as the case may be.

If a mapping is either strongly coherent or weakly coherent then we say that it is *coherent*. Now we define a T-MAP from the source domain to the target domain as follows:

Definition: A T-MAP from the source domain D1 = <I1, E1, S1, d1, f1> to the target domain D2 = <I2, E2, S2, d2, f> is a pair <F, S> where F is an admissible mapping from D1 to D2, $S \subseteq S1$ is a set of structural constraints of D1 that is

transformable with respect to F, and F is coherent with respect to S.

Often when referring to a maximal coherent T-MAP we will omit the mention of the set of structural constraints S.

Having introduced the definitions we now describe some formal properties of T-MAPs. In this section our main concern is formal exposition and only in the next chapter will we discuss how they help or hinder in capturing some cognitive features of metaphors and analogies.

The first, rather obvious, property of T-MAPs directly follows from the definition and from the fact that if every sentence in a set of sentences is a consequence of another set of sentences then their union yields a consistent set of sentences. Thus, we have,

Fact 1: Every strongly coherent T-MAP is also weakly coherent.

In order to mention a second property of T-MAPs we introduce the notion of an *extension* and a *restriction* of a T-MAP. If T1 = <F1, S1> and T2 = <F2, S2> are T-MAPs from the source domain D1 to the target domain D2, then we say that T1 is a *restriction* of T2 iff F1 $\subseteq$ F2 and S1 $\subseteq$ S2. There are two ways by which a T-MAP <F, S> can be restricted. Firstly, we can keep F the same but throw out some of the structural constraints from S thereby reducing the structure that is being transferred. Secondly, we can reduce F by removing some of its elements. This will result in some of the tokens of D1, that were originally included in F, being left out. In the second case the transferred structure S may have to be reduced too since any structural constraint in S that contains any of the tokens that have been

left out will now not be transformable and will have to be removed.

If T1 is a restriction of T2 then we say that T2 *extends* T1. A straightforward implication of this definition is,

Fact 2: All restrictions of a T-MAP are also T-MAPs.

In other words, an incoherent mapping cannot be extended to a T-MAP. Though this property is obvious it can be useful in reducing the computational complexity of any algorithm that computes T-MAPs from one domain to another, as we will show in Chapter 5.

The last thing we would like to discuss in this section is the *symmetry* property of T-MAPs. If a T-MAP T = <F, S> from the source domain D1 to the target domain D2 is such that F is injective (one-to-one), i.e. no two tokens of D1 are assigned the same token of D2 by F, then we can define F^{-1} which will be an admissible mapping from D2 to D1. In such a case we will say that T is *invertible* since F^{-1} will allow parts of the structure of D2 to be transferred to D1 and thereby generate T-MAPs from D2 to D1. If T is maximal then we define its *inverse* T^{-1} to be the pair $<F^{-1}, S2>$ where S2 is the set of structural constraints of D2 that is maximal with respect to F^{-1}.

For instance, consider the following admissible mappings from FAMILY to DAG:

- FD3: {(Male → Node); (Father → Arc)}
- FD4: {(Person → Node); (Father → Arc)}

Both of them are maximally coherent and are therefore T-MAPs. Their inverses

yield the following mappings from DAG to FAMILY:

- o DF3: {(Node → Male); (Arc → Father)}
- o DF4: {(Node → Person); (Arc → Father)}

The symmetry property of T-MAPs is stated in the following theorem:

Theorem 1: If T is an invertible maximal weakly coherent T-MAP from the source domain D1 to the target domain D2 and both D1 and D2 satisfy the axiom of closure then T^{-1} is a maximal weakly coherent T-MAP from D2 to D1.

In other words, this theorem states that the inverse operation preserves weak coherency of admissible mappings. Thus in our example above, both DF3 and DF4 are maximal weakly coherent.

The proof of this theorem is based on Craig's Interpolation Lemma [Chang & Keisler 1973, pp. 84-85] and is presented in the Appendix.

Finally, we remark that the symmetry property does not hold for strongly coherent T-MAPs. In other words, a T-MAP can be strongly coherent but its inverse may not be strongly coherent, though, of course, it will be weakly coherent. For example, consider the following admissible mapping from FAMILY to TREE:

- o FT1: {(mother → arc); (female → node)}

It can be verified that this is a maximal strongly coherent mapping. The inverse of this T-MAP is TF1 and we showed earlier in this section that TF1 is not maximal strongly coherent. Similarly, we note that the mapping FD5,

- FD5: {(parent → arc); (person → node)}

is maximal strongly coherent though its inverse DF5 is not strongly coherent.

3.4 TWO MECHANISMS FOR EXTENDING T-MAPS

So far our notion of a T-MAP is analogous to that of a homomorphism between two algebras. Though it has several interesting features as it is that—as we will see in the next chapter—makes it an attractive theory of metaphors, what makes it even more powerful are the two recursive operators introduced in this section. The operators are called *Augmentation* and *Positing Structure* and the reason they make CST a very powerful theory is that they allow one to add structure to the target domain constrained by the structure of the source domain. Thus both these operators accept as input a source domain, a target domain, and a T-MAP from the source domain to the target domain; and result in a T-MAP that is an extension of the input T-MAP and a new target domain which is a superset of the input target domain.

In this section we will formally define these operators with simple examples to show how they work. We will also discuss how these operators affect weak and strong coherency of the input T-MAP. We will only mention briefly the motivation behind introducing these operators and most of the discussion on how they lend usefulness to CST as a theory of metaphors will be reserved for the next chapter.

3.4.1 AUGMENTATION

Augmentation makes use of derived tokens in extending a T-MAP. It works something like this. Suppose T = <F, S> is a T-MAP from the source Domain D1 to the target domain D2. Further, suppose that R1 is a derived token of D1 that is not already included in F and that can be derived by using the tokens that are included in F. Then we create a new token, say R1′, in D2 and extend T by including (R1 → R1′) in F and the derivation of R1 in S. Thus R1′ serves as a new derived token of D2 whose derivation is obtained by transforming the derivation of R1 by F. This transformed derivation of R1′ is added to the set of structural constraints of D2. An example will make this clearer. Consider the following admissible mapping from FAMILY to DAG, which is maximal strongly coherent,

- FD5: {(parent → arc); (person → node)}

Now a derivation of *sibling* in terms of *parent* is,

- (x)(y) [sibling(x,y) ⟷ [x≠y & (Ez) [parent(z,x) & parent(z,y)]]]

An application of Augmentation to FD5 will create a new token, say *sib*, in DAG. The new T-MAP will be:

- AFD5: {(person → node); (parent → arc); (sibling → sib)}

The above derivation of *sibling* will be added to the set of structural constraints of FAMILY that are being transferred by AFD5. Also, the newly created token *sib* will get a corresponding derivation in DAG as follows:

o (x)(y) [sib(x,y) ↔ [x≠y & (Ez) [arc(z,x) & arc(z,y)]]]

Now we will formally define Augmentation.

Definition: Given a source domain D1 = <I1, E1, S1, d1, f1>, a target domain D2 = <I2, E2, S2, d2, f>, a T-MAP T = <F, S> from D1 to D2, a derived token X of D1 (X ∈ I1), a derivation D of X (D ∈ d1), and a token Z; Aug(D1,D2,T,X,D,Z) is an *Augmentation* if all of the following conditions are satisfied:

1. X is not mapped by F.
2. If f1(X) = p_n then D is of the form X(x1,...,xn) ↔ Y where Y is transformable by F.
3. If f1(X) = o_n then D is of the form X(x1,...,xn) = x ↔ Y such that Y is transformable by F and the sentence (E!x)F(Y) is a consequence of S2.
4. If f1(X) = e then D is of the form X = x ↔ Y such that Y is transformable by F and the sentence (E!x)F(Y) is a consequence of S2.
5. Z does not occur in either I2 or in E2.

In this case, the result of applying Augmentation is a pair <T′, D2′> where D2′ is a new target domain <I2′, E2, S2′, d2′, f′> and T′ = <F′, S′> is a new T-MAP from D1 to D2′ which are obtained as follows:

1. I2′ = I2 ∪ {Z}
2. f2′ is exactly like f2 except that f2′(Z) = f1(X)
3. F′ = F + {(X → Z)}. Here by + we denote the disjoint union.

4. $S' = S \cup \{D\}$
5. $S2' = S2 \cup \{F'(D)\}$
6. $d2' = d2 \cup \{F'(D)\}$

Given two T-MAPs T1 from domain D1 to domain D2 and T2 from domain D1 to D2′ we say that T2 is an augmentation of T1 if there are X, D, and Z such that Aug(D1,D2,T1,X,D,Z) is defined and is equal to <T2, D2′>.

It is easy to see from the definition that Augmentation preserves coherency of the input T-MAP. If the input T-MAP is strongly (weakly) coherent then the output T-MAP is also strongly (weakly) coherent. The reason behind this is that the only structural constraint that is added to the input T-MAP is the derivation of the token for which a new token is created in the target domain. Since a corresponding derivation is added to the target domain, the additional structural constraint transferred by the T-MAP is trivially a consequence of the set of structural constraints of the new target domain and therefore the coherency of the output T-MAP is the same as that of the input T-MAP.

Augmentation lends much usefulness to T-MAPs since it makes it possible to name a structure of relationships in the target domain for which none existed before. Thus, once the base T-MAP is established, any number of tokens from the source domain that can be derived using the tokens included in the base T-MAP can be added to the T-MAP without affecting its coherency. As we will explain in the next chapter, when we discuss how T-MAPs characterize metaphors, this mechanism captures an important cognitive feature of metaphors which is that often several literal statements can be replaced by a single metaphorical statement

conveying the same idea.

3.4.2 POSITING STRUCTURE

In a way Positing Structure generalizes Augmentation. Whereas Augmentation introduces new tokens in the target domain based on the derivations in the source domain, this operator, namely Positing Structure, allows one to introduce new tokens in the target domain and map the source domain tokens to these newly created tokens even when there are no derivations of the source domain tokens. Also, whereas Augmentation adds to the structure of the target domain only the transformed derivation of the source domain token, Positing Structure allows one to decide what structural constraints one wants to add to the target domain involving newly created tokens as long as the newly added constraints do not make the mapping incoherent.

We will introduce this operator by means of an example. Consider the weakly coherent mapping FD5 that was introduced earlier.

- FD5: {(person → node); (parent → arc)}

Now in order to apply Positing Structure to this, we first create new tokens in the target domain DAG. Let the newly created tokens be called *dmale* and *dfemale*. Now the original T-MAP FD5 can be extended by mapping some of the source domain tokens, that were not included in the T-MAP, to these newly created tokens in the target domain. Thus, we get another T-MAP PFD5:

- PFD5: {(person → node); (parent → arc);
  (male → dmale); (female → dfemale)}

So far, since we have not added any structural constraints in the target domain to relate the newly created tokens with its other existing tokens, the coherency of PFD5 with respect to the structure that is maximal to FD5, is not affected. If FD5 were a strongly coherent mapping, then PFD5 would also be strongly coherent with respect to all the structural constraints that relate tokens *person* and *parent*. If we add all those structural constraints that relate newly mapped tokens of the source domain, namely *male* and *female*, among themselves and with other tokens that were in the original mapping FD5 to the structure to be transferred by PFD5, then the resulting T-MAP will be weakly coherent. Thus, PFD5 is maximally weakly coherent.

Positing Structure allows one to add structural constraints in the target domain relating newly introduced tokens among themselves as well as with other existing tokens, the only restriction being that the added constraints must be consistent with the transformed structural constraints of the source domain. In the above example, for instance, the tokens that are added to the input T-MAP FD5 from the source domain FAMILY as a result of applying this operator are *male* and *female*. Some of the structural constraints of FAMILY that relate these tokens among themselves and with the other tokens in the input T-MAP FD5 are:

- (x) [male(x) → ~female(x)]
- (x)(y) [father(x,y) → [male(y) v female(y)]]

On transforming them with PFD5, they become:

- (x) [dmale(x) → ~dfemale(x)]
- (x)(y) [arc(x,y) → [dmale(y) v dfemale(y)]]

Now we can add any structural constraints relating *dmale*, *dfemale* among themselves, as well as with the other tokens of DAG as long as they are consistent with all the above transformed constraints.

We will now formally define Positing Structure.

Definition: Given a source domain D1 = <I1, E1, S1, d1, f1>, a target domain D2 = <I2, E2, S2, d2, f>, a T-MAP T = <F, S> from D1 to D2, a set of tokens I1′, a set of sentences S1′, a set of tokens Z, a set of sentences S3, and a bijection f from I1′ to Z; PS(D1,D2,T,I1′,S1′,Z,S3,f) is a *Positing Structure* if the following conditions are satisfied:

1. I1′ ⊆ I1, and every token X ∈ I1′ is such that X is not mapped by F.
2. S1′ ⊆ S1 and every sentence X in S1′ is such that every token in X is either in I1′ ∪ E2 or is mapped by F.
3. Z ∩ (I2 ∪ E2) = ∅.
4. S3 is a set of sentences over the vocabulary <I2 ∪ E2 ∪ Z, f2 + (f1 o f^{-1})>. Here by + we denote disjoint union of two functions, by o we denote function composition, and f^{-1} is the inverse of f.
5. (F + f)(S1′ ∪ S) ∪ S3 ∪ S2 is consistent.

In such a case, the result of applying Positing Structure is a pair <T′, D2′> where D2′ is a new target domain <I2′, E2, S2′, d2, f′> and T′ is a T-MAP <F′, S′> from D1 to D2′ which are obtained as follows:

1. $I2' = I2 \cup Z$
2. $S2' = S2 \cup S3$
3. $f2' = f2 + (f1 \circ f^{-1})$
4. $F' = F + f$
5. $S' = S \cup S1'$

It is easy to see from the step 5 above that Positing Structure always results in a weakly coherent T-MAP though it may destroy strong coherency of the input T-MAP.

It can also be verified from the definitions that Augmentation is a special case of Positing Structure. Actually Positing Structure is a very general operator which allows almost any structure to be added to the target domain as long as it is consistent with the structure of the source domain.

The motivation behind introducing this operator is twofold. Firstly, in order for Augmentation to be applicable to a T-MAP it may be necessary at first to extend the T-MAP by Positing Structure. For instance, in the examples we presented earlier, we cannot include the token *sister* in the T-MAP FD5 by applying Augmentation to it since *sister* cannot be defined without using the token *female*. In order to add the token *female* to FD5 we will have to resort to Positing Structure.

Secondly this mechanism captures an important cognitive property of metaphors and scientific models. Metaphors are known to provide extra richness of meaning that is missing from their literal counterpart, if one exists. For instance, in "the sky is crying", the metaphor conveys more information than the mere fact that it is raining in the sense that one may attribute many humanlike properties to the sky in

understanding the metaphor. This richness is often very subjective even among people that share the same cultural background. Positing Structure captures this property of metaphors since it allows one to add structure to the target domain, and thereby extend a T-MAP, in several possible ways that can be very subjective.

Similarly, often scientific models are not merely useful for explanations but they create new insights into an ill-understood target domain and enrich our knowledge. Positing Structure captures this property of models since by using this operator we can create several hypotheses about the target domain. An effort to confirm or refute them results in an increased knowledge of the target domain. Since the hypotheses were based on the structure of the source domain, the model will have quite a remarkable effect in establishing an epistemological perspective or bias on the knowledge of the target domain. We will discuss these two aspects of T-MAPs in more detail in the next chapter.

3.5 SOME REMARKS ON EXTENDING CST

In this section we will remark on how SL can be extended to allow higher order predicates and how the admissibility requirement on T-MAPs can be relaxed so that the types of tokens need not be preserved, but only the type structure must be preserved.

There are several ways in which higher order predicates can be allowed in SL. We will present one such system here which is based on Montague's use of types in his formal semantics of natural language [Montague 1975]. In Montague's system the set of types T2 is recursively defined as follows:

1. e, t $\in$ T2. Here e and t are two distinct objects.
2. If a, b $\in$ T2 then <a,b> $\in$ T2.

Intuitively, e is the type corresponding to individual constants, t is the boolean type, and <a,b> is the type corresponding to functions from type a to type b. Thus, from our earlier definition of types, the type p_1 of T corresponds to <e,t> of T2, o_1 corresponds to <e,e>, p2 corresponds to <e,<e,t>> etc. Of course, in this system we will also have types like <<e,t>,<e,t>>; <e,<e,<e,t>>> etc. which have no corresponding types in T.

Given this set of types, we associate a denumerable set of variables Var_a with every type a. Given a vocabulary <V, f> where V is a set of tokens and f is a function from V to T2, we recursively define the set of terms T_a associated with every type a as follows:

1. If $X \in Var_a$ then $X \in T_a$.
2. If $X \in V$ and $f(X) = a$ then $X \in T_a$.
3. If $X \in T_{<a,b>}$ and $Y \in T_a$ then $X(Y) \in T_b$.
4. If $X \in T_b$ and $y \in Var_a$ then $\lambda yX \in T_{<a,b>}$.
5. If $X, Y \in T_t$ then $\sim X$, $X\&Y \in T_t$.
6. If $X \in T_t$ and $y \in Var_a$ for any a then $(y)X \in T_t$.

We associate a set of free variables with every term in a way similar to the one presented in Section 3.2.2. The only significant difference is the addition of the following rule:

1. If X is of the form λyZ then FV(X) = FV(Z) - {y}.

The terms of type t are called formulas and a formula that has no free variable is called a sentence.

Given a nonempty set A we now associate a set of denotations D_a, with respect to A, with every type a as follows:

1. $D_e = A$.
2. $D_t = \{0,1\}$.
3. $D_{<a,b>} = \{D_a \rightarrow D_b\}$. We recall that $\{A \rightarrow B\}$ is the set of all functions from A to B.

The notion of model is also very similar to the one presented in Section 3.2.2. A model for vocabulary <V, f> is a pair <A, F> where A is some nonempty set and F assigns to every member of V, that is of type a, an element from the set of denotations D_a corresponding to that type.

Given a model M = <A, F> of the vocabulary <V, f>, and a variable assignment g that assigns to every variable of type a an element of D_a, we recursively define the meaning function m, with respect to M and g, that assigns an element of D_a to every term of type a.

1. If $X \in Var_a$ then m(X) = g(X).
2. If $X \in V$ then m(X) = F(X).
3. If X is of the form Y(Z) then m(X) = m(Y)(m(Z)). We recall that if X is of the form Y(Z) then Y and Z will have types <a,b> and a respectively for some a and b. Thus m(Y) will be a function from D_a to D_b and m(Z) will be an element of D_a.

4. If X is of the form λyZ with $y \in Var_a$ and $Z \in T_b$, then m(X) is the function h with the domain D_a such that for every element k of D_a h(k) is m(Z) with respect to M and g′, where g′ is exactly like g except that g′(y) = k.

5. If X is of the form ~Y then m(X) = 1 iff m(Y) = 0.

6. If X is of the form Y&Z then m(X) = 1 iff m(Y) = 1 and m(Z) = 1.

7. If X is of the form (y)Z with y being a variable of type a, then m(X) = 1 iff for every element k of D_a, m(Z) = 1, with respect to M and g′, where g′ is exactly like g except that g′(y) = k.

All the remaining concepts, like truth in a model, consistency etc. can be directly carried over and so we will not repeat them here. The definition of domain also remains unchanged except that the types of tokens are assigned from T2 and the structural constraints consist of sentences as defined above.

One is not restricted to the type system just described and it is possible to design one's own set of types by specifying the basic types along with the operations for building higher types. For instance, in the context of programming languages one often starts out with the basic types as {*int*, *bool*, *char*} and use binary operators like × (Cartesian product), + (disjoint union), → (function type), and unary operators like *list*, *stack* to build higher types [Cf. Milner 1978]. Some other examples of different type systems used in natural language semantics can be found in Bronnenberg *et al.* [1980] and Scha [1981]. Of course, using a different system of types requires that the semantics be fully worked out.

Finally, we would like to remark on the admissibility requirement of a mapping. The admissibility condition is imposed on the mappings so that the transformed structural constraints are meaningful in the target domain. We recall that in order for a mapping to be admissible we required that the types of the tokens that are mapped be preserved. We will now show that this is a sufficient but not a necessary condition for admissibility. Suppose D1 = <I1,E1,S1,d1,f1> and D2 = <I2,E2,S2,d2,f2> are two SL domains that are based on the system of types T2 just described. Let F be a mapping from D1 to D2 and I be the set of internal tokens of D1 that are mapped by F. Further, suppose F does not preserve types of the tokens mapped, but instead it satisfies the following condition. For every $X \in I$,

$$\tau(f1(X)) = f2(F(X))$$

where τ is function from T2 to T2 defined as follows:

1. $\tau(e) = \langle e,t \rangle$.
2. $\tau(t) = t$.
3. $\tau(\langle a,b \rangle) = \langle \tau(a),\tau(b) \rangle$ for any a, b.

The reader can easily verify that F is admissible in the sense that any structural constraint of D1 that can be transformed by F and that does not contain any external tokens will be meaningful in the target domain D2, provided that every variable X of type a is replaced by a variable of type $\tau(a)$ in the process of transformation.

The reason for this is that the function τ, as defined above preserves the type structure. In general a type system can be looked upon as an algebra with a set of basic types Σ, and a set of recursive operators that allow one to construct more types from Σ. Any function from Σ to Σ can be extended to an endomorphism, which is a structure preserving type function, and can give rise to admissible mappings. The identity mapping is just a special case of this. Further, it may not even be necessary that the source and the target domain use the same system of types, all we need is a homomorphism from the algebra of the source domain types to the algebra of the target domain types to have admissible mappings. The notion of semiotic morphisms defined by Goguen and Linde [1984] uses this general definition of admissibility. Of course, whether such mappings are found in structural metaphors and analogies is a question that needs looking into.

3.6 SUMMARY

In this chapter we began by formally developing a knowledge representation scheme, called Schema-Language, which is derived from the first order predicate calculus but differs from it in some ways, and is based on a structural paradigm. SL allows representation of partially specified domains and domains that admit of several choices of primitives. We also presented examples of two SL domains.

Next we defined the concept of a T-MAP, which is a partial coherent mapping from one SL domain to another. T-MAPs are analogous to homomorphisms between two algebras. T-MAPs formalize the notion of coherency that lies at the heart of metaphors. We also distinguished between weak and strong coherency and

derived some properties of T-MAPs.

Though T-MAPs are interesting for a theory of metaphors as such, what makes them even more interesting are two recursive operators, Augmentation and Positing Structure, that allow new structure to be posited in the target domain. We formally defined these operators and showed how they affect the coherency of T-MAPs.

We made some remarks on extending SL to allow higher order predicates to occur and relaxing the admissibility requirement on T-MAPs so that they need not be type preserving.

In the next chapter we will show how our theory of CST can be used to characterize metaphors and analogies.

"I don't rejoice in insects at all," Alice explained, "... But I can tell you the names of some of them."

"Of course they answer to their names?" the Gnat remarked carelessly.

"I never knew them do it."

"What's the use of their having names," the Gnat said, "if they won't answer to them?"

— Lewis Carroll, "Alice Through the Looking Glass".

C H A P T E R IV

CST AS A THEORY OF METAPHORS

4.1 INTRODUCTION

In the previous chapter we formally developed our theory of CST and defined the concept of a T-MAP that allows a set of structural relationships among the tokens of the source domain to be transferred to the target domain in a coherent way. We also introduced two operators that make it possible to posit new structure in the target domain constrained by the structure of the source domain. In this chapter our main concern is to show how CST can be used to characterize metaphors, analogies and scientific models and what are its limitations. We will also relate CST to some other theories of metaphors in this chapter.

We will discuss the issues involved in representing "real-world" domains in SL in the next section. As opposed to "self-contained" domains, such as the ones presented in the previous chapter, there is no unique way to represent real-world domains and wide fluctuations in the internal and external tokens as well as in the structural constraints can be expected among different representations of the same

domain. We will argue that this fluctuation and subjectiveness of representation is a consequence of the nature of the real-world domain and not an artifact of SL. Further, we will claim that since SL allows a domain to be represented in different ways, it is a suitable representation scheme for a formal theory of metaphors. Finally, we will present an SL representation of two real-world domains, namely PLOWING and SAILING, at the end of this section.

In Section 4.3 we will relate metaphors and analogies to T-MAPs and demonstrate, by using some examples, how T-MAPs can be used to characterize metaphors and analogies. Further, we will show how several cognitive properties of metaphors can be explained in our characterization. We will also develop the notions of metaphorical truth, metaphorical inference and analogical deduction in our formalization in this section.

In Section 4.4 we will compare CST with some of the more prominent theories of metaphor that were discussed in Chapter 2. In Section 4.5 will discuss some limitations of CST. One of the shortcoming concerns the computability and finite representability of CST. This will be overcome in Chapter 5 by defining the concept of an *approximate* T-MAP, which is then used to to develop a theory of Approximate Semantic Transference that can be used for designing models of cognitive processes involved in creating and understanding metaphors and analogies. Finally, in Section 4.6 we will summarize the main points of this chapter.

The discussion of this chapter takes a rather static view of metaphors. That is, it does not provide any explanation of the dynamic process of arriving at the meaning of a metaphor. This issue will be addressed in the next chapter where we will provide several algorithms to model the process of arriving at an interpretation of a metaphor.

4.2 REPRESENTING REAL-WORLD DOMAINS IN SL

In this section we will discuss issues involved in representing real-world domains in SL. Real-world domains are to be contrasted with self-contained domains, such as the ones presented in the previous chapter, that are often found in the realm of pure mathematics. Self-contained domains often satisfy the axiom of closure and they are so called because of the absence of any external tokens. On the other hand the main characteristics of real-world domains are that they have many external tokens and they do not satisfy the axiom of closure. Examples of such domains are CHAIR, LOVE, PLOWING, SHIP, INFLATION etc. We call them "real-world" because the concept or the object that is represented in the domain is taken from everyday life.

The real-world domains differ from self-contained domains in four major respects which make it harder to represent them in any formalism, let alone SL. Firstly, the real-world domains do not have a clearly defined boundary that separates them from the other domains. For instance, the boundary between the domains SHIP and SEA is very vague because it is not clear how much of the structure of SEA is part of SHIP and vice versa. Of course, the domains do not have to be

mutually disjoint, and a structural constraint like "if the weight of X is less than the weight of water it displaces then X floats in water" can be part of the structure of SHIP as well as SEA. But still this does not provide any criteria for deciding the boundary of any of these domains. On one extreme one will end up including *everything* in any domain since with some imagination any concept or object can be linked with any other concept or object. On the other extreme almost everything will be dubbed as external and there will not be much of interest in any domain. In practice, most often one would want a representation somewhere between these two extremes.

The second aspect of real-world domains concerns the fact that even if some rather ad hoc boundary of a domain has been decided, there are many ways in which the set of internal and external tokens of the domain can be chosen. Thus, for instance in representing the domain CHAIR one may choose four different tokens LEG-1, LEG-2, LEG-3, and LEG-4, or be content with only one token LEG. Similarly in another example, a representation of SHIP may have a separate token corresponding to every individual metal plate, rivet etc. or may only have tokens that correspond to bigger units like HULL, BOW etc. The reason for this fluctuation is that an adequate representation is very much dependent on the function it has to perform. An engineer who is working with a blueprint of the ship may prefer the former representation of SHIP whereas a sailor might choose the latter representation scheme. Normally, if one knows in advance the function of the representation then one can choose a set of tokens that is adequate for that function but in metaphors since one does not know a priori what kind of functions

will be found useful in interpreting a metaphor the choices cannot be so easily made. Looking at it from the other direction, given a certain choice of tokens it will constrain which metaphors will be interpretable and which will not make any sense. In an actual system one may either resort to psychological study of some human subjects to choose an appropriate representation of the domain or may make some ad hoc choices.

The third major difference between real-world and self-contained domains concerns structural constraints. Obviously, if two representations of the same domain have different sets of tokens then they will also have different sets of structural constraints. But even if the set of tokens do not differ in any appreciable way, the set of structural constraints can still be significantly different. For instance, in representing the domain SAILING, two representations that both include the tokens CREW and CAPTAIN may differ from each other in whether the captain's role in controlling the ship, or the crew's role, is included in its structural constraints. Again, how a metaphor is interpreted will depend very much on how the structural constraints of the source and the target domain were chosen. In a "rich" domain one would expect many structural constraints and lots of redundancies. This issue too can be decided psychologically when making choices among several possibilities.

The final distinguishing feature of real-world domains also concerns structural constraints. The set of structural constraints of a real-world domain is not closed. This is because people normally do not carry all the implications of all their knowledge in their head. Sometimes the knowledge of an object that a person has may even be inconsistent without the person being aware of it. We will discuss this

issue in more detail in Section 4.5.

There is thus no unique way to represent real-world domains in SL. This is not an artifact of SL since the points we made can be made about any knowledge representation scheme. Fortunately, SL makes it possible to represent a domain in several ways. This flexibility is necessary for a theory of metaphors since otherwise only certain metaphors, namely those that favor the representation scheme, would be recognizable.

We now give an SL representation of two real-world domains called PLOWING and SAILING. The choices of the internal and external tokens, as well as the structural constraints, have been made on a totally ad hoc basis. The point of representing these domains is not to show that this is the best way of representing them but to give the reader some sense of what a real-world domain might look like in SL. Also many of the examples of the T-MAPs used in this chapter will use these domains as the source and target domains. In presenting a domain we will continue the practice of giving the formal representation followed by comments. Also, for easier understandability, while presenting the structural constraints we will give the SL sentence stating the constraint followed by a sentence of English describing its intended meaning. We would also like to mention that both these domains use the system of types T2 explained in Section 3.5 and therefore allow tokens of higher order types.

DOMAIN 3: PLOWING

This domain is specified as <I3, E3, S3, d3, f3> where

1. I3 = {earth, farmer, field, germination, human, loose-and-broken, plow-a, plow-edge, plow-o, seed, weeds}

2. E3 = {agent-of, assist, contains, controls, harder-than, instrument-of, moves-through, partially-submerged, sharp}

3. S3 is the following set of sentences:

    1. controls(farmer, plow-o): The farmer controls the plow.

    2. agent-of(plow-a) = farmer: The farmer is the agent of plowing.

    3. instrument-of(plow-a) = plow-o: Plow is the instrument of plowing.

    4. plow-a(field, plow-o): The plow plows the field.

    5. moves-through(field, plow-o): The plow moves through the field.

    6. contains(field, earth): The field contains earth.

    7. partially-submerged(plow-o, earth): The plow is partly submerged in the earth.

    8. contains(plow-o, plow-edge): The plow has an edge.

    9. sharp(plow-edge): The edge is sharp.

    10. harder-than(plow-edge, earth): The edge is harder than the earth.

    11. splits(earth, plow-o): The plow splits the earth.

    12. result-of(plow-a, loose-and-broken(earth)): The result of plowing is a field with loose and broken earth.

    13. human(farmer): The farmer is human.

    14. uproots(plow-o, weeds): The plow uproots the weed.

15. contains(field, weeds) → ~loose-and-broken(earth): The weeds in the field cause the earth in it to be not loose and broken.

16. assist(loose-and-broken(earth), germination(seed)): Loose and broken earth assists the process of seed germination.

4. d3 = ∅

5. f3 is specified as follows:

    1. f3(X) = e if X is earth, farmer, field, plow-edge, plow-o, seed, or weeds.

    2. f3(X) = <e, t> if X is human, germination, loose-and-broken, or sharp.

    3. f3(X) = <e, <e, t>> if X is contains, controls, harder-than, moves-through, partially-submerged, or plow-a.

    4. f3(X) = <<e, <e, t>>, e> if X is agent-of or instrument-of.

    5. f3(assist) = <t, <t, t>>

Comments: In the representation of plowing, the token *plow-a* corresponds to the action of plowing whereas the token *plow-o* corresponds to the object plow, as the reader must have guessed. Also, in giving the structural constraints we have unfolded, or uncurried, the higher order tokens merely for the ease of writing. Thus, instead of writing "controls(farmer)(plow-o)" we have written the first structural constraint above as "controls(farmer, plow-o)".

As we mentioned before, this representation is very ad hoc and a lot of structure of plowing is left out. We have only included enough structure so that it can serve as an example in the discussion to follow in the next section.

DOMAIN 4: SAILING

This domain is specified as <I4, E4, S4, d4, f4> where

1. I4 = {captain, crew, sail, sea, ship, ship-bow, water}
2. E4 = {agent-of, contains, controls, floats-in, harder-than, human, instrument-of, moves-through, object, partially-submerged, people, place, salty, separated-by-water, sharp, transport, used-for}
3. S4 is the following set of axioms:
    1. floats-in(ship, water): The ship floats in water.
    2. contains(sea, water) & salty(water): The sea contains salty water.
    3. sail(sea, ship): The ship sails in the sea.
    4. agent-of(sail) = captain: The captain is the agent of sailing.
    5. agent-of(sail) = crew: The crew is the agent of sailing.
    6. instrument-of(sail) = ship: The ship is the instrument of sailing.
    7. used-for(ship, transport(x,y,z)) → ((people(x) v object(x)) & place(y) & place(z) & separated-by-water(y,z)): The ship is used for transporting people or objects across water.
    8. controls(captain, ship): The captain controls the ship.
    9. controls(crew, ship): The crew controls the ship.
    10. controls(captain, crew): The captain controls the crew.
    11. contains(ship, ship-bow): The ship has a bow.
    12. sharp(ship-bow): The bow is sharp.
    13. partially-submerged(water, ship): The ship is partly submerged in water.
    14. harder-than(ship-bow, water): The bow is harder than water.

15. human(captain): The captain is human.

16. moves-through(sea, ship): The ship moves through the sea.

17. (x)[contains(crew, x) → human(x)]: All the crew members are human.

4. d4 = ∅

5. f4 is defined as follows:

    1. f4(X) = e if X is captain, crew, sea, ship, ship-bow, or water.

    2. f4(X) = <e, t> if X is human, object, people, place, salty, or sharp.

    3. f4(X) = <e, <e, t>> if X is contains, controls, floats-in, harder-than, moves-through, partially-submerged, sail, or separated-by-water.

    4. f4(X) = <<e, <e, t>>, e> if X is agent-of or instrument-of.

    5. f4(transport) = <e, <e, <e, t>>>

    6. f4(used-for) = <e, <t, t>>

Comments: The tokens of SAILING have been chosen arbitrarily and we are not making any psychological claims about our choice. All the names of the tokens are self explanatory. In writing down the structural constraints we did not try to include all the aspects of sailing but have only given those that are relevant to the example in the next section. Also, as with the structural constraints of PLOWING, we have unfolded all the higher order tokens for convenience of writing.

4.3 ANALOGIES AND METAPHORS IN CST

In this section we describe how metaphors and analogies can be characterized in CST. We demonstrate, by some examples, how several cognitive features of metaphors are captured in our characterization. We also develop the notions of metaphorical truth and inference in our theory.

4.3.1 T-MAPS AS METAPHORICAL INTERPRETATIONS

In our characterization of metaphors in CST we associate a T-MAP with every *interpretation* of a metaphor. We emphasize the word interpretation because in general a metaphor may have several possible interpretations and thus in our theory several T-MAPs may be associated with a given metaphor.

In our earlier discussion in Chapters 1 and 2, we described metaphors roughly as the application of one or more terms belonging to the source domain to one or more terms belonging to the target domain. Thus in SL a metaphorical utterance will correspond to a sentence in which some tokens are taken from the source domain and others from the target domain. Given a metaphorical utterance, a T-MAP is a *possible interpretation* for it iff all those tokens of the source domain that occur in the SL sentence corresponding to the utterance are included in the T-MAP. For example, any T-MAP from FAMILY to DAG [Cf. Chapter 3] that includes the token *sibling* of FAMILY can be a possible interpretation for the utterance "node A is sibling of node B". Of course, many of these T-MAPs may not correspond to an appropriate interpretation intended by the speaker. But this precisely captures the feature of metaphor that they can miscommunicate.

Given a metaphor and a T-MAP that is a possible interpretation for it, the *meaning* of the metaphorical utterance under that interpretation is constructed by taking the SL sentence that corresponds to the metaphor and replacing all occurrences of all the source domain tokens in it by their corresponding target domain tokens as given by the T-MAP. Thus in interpreting the above sentence in the context of T-MAP AFD5 of Chapter 3, repeated below for convenience, the occurrence of *sibling* is replaced by token *sib* that was created in the target domain DAG by augmentation.

- AFD5: {(person → node); (parent → arc); (sibling → sib)}

Since Augmentation also added the definition of *sib* to the domain DAG the interpretation given to this sentence will be that there is some node C such that there are arcs from C to B and from C to A.

This is formally stated as follows. In the context of a T-MAP T = <F, S> from the source domain D1 to the target domain D2, a sentence X is said to be *interpretable* under T iff all the tokens in X are either internal or external tokens of D2, or are internal tokens of D1 and are included in F. The meaning and truth, with respect to T, of any sentence X interpretable under T is the same as the meaning and the truth of the sentence obtained from X by replacing every such token Y in X that is an internal token of D1 by F(Y).

We note here that we are not specifying how the meaning of the resulting literal expression is to be interpreted but that is another problem. We would like to remark though that this characterization is in keeping with our earlier intuitions that the means of arriving at a literal truth and a metaphorical truth are similar and a metaphorical statement is no *less* true than a literal statement. Also, this concept of metaphorical truth is independent of whether one is using boolean logic, three-valued logic, fuzzy logic or some other kind of truth maintenance system to represent meaning and truth.

Finally, we would like to remark that the process of arriving at some interpretation of a metaphor is very dynamic and T-MAPs can best be seen as snap-shots of this process. In this chapter we will largely ignore this dynamism and therefore will not be concerned with the process of arriving at a T-MAP. This issue will be discussed in Chapter 5.

4.3.2 METAPHORICAL INFERENCE AND ANALOGICAL REASONING

The notion of metaphorical inference or analogical deduction is slightly more complicated. Given a T-MAP $T = \langle F, S \rangle$ from the source domain to the target domain D2 that is an interpretation of some metaphor, the first thing that suggests itself is that if S2 is the set of structural constraints of the target domain D2 then any inference that can be derived from the set $S2 \cup F(S)$ can be dubbed a metaphorical inference under the given interpretation. The problem with this characterization is that many of the inferences can be derived from S2 alone and one would not want to label them as metaphorical deductions since otherwise we

will not be able to explain why metaphor highlights only certain aspects of the target domain.

To overcome this difficulty we can modify the definition in the following way. Any inference that can be derived from S2 ∪ F(S) but cannot be derived from S2 alone is a metaphorical inference. This definition fails for strongly coherent T-MAPs since in that case S2 implies F(S) and therefore none of the inferences will be metaphorical.

The final definition that we offer is as follows. Any inference that can be derived either from F(S) alone, or from S2 ∪ F(S) but not from S2 alone is called a metaphorical inference under T, or simply an inference under T. As we will see later this definition does not have any of the above shortcomings and can explain how a metaphor can highlight certain aspects of the target domain and suppress certain others.

In our theory we characterize analogies as strongly coherent T-MAPs. We notice that for strongly coherent T-MAPs the above definition reduces to the effect that only those inferences that can be derived from F(S) alone qualify as metaphorical inferences or analogical deductions. This concurs with our intuitions since in analogies one carries out a chain of reasoning in the source domain, which is more familiar and cognitively more accessible to the subject, and transforms the result to the target domain. In representing them with strongly coherent T-MAPs any inference derived from S will also hold in the target domain after being transformed.

4.3.3 METAPHORICAL HIGHLIGHTING AND DOWNPLAYING

One salient feature of metaphors is that they highlight certain aspects of the source and target domains and downplay or hide some other aspects. For instance, consider the utterance "the ship plowed through the sea". Some of the aspects of the ship's motion through the sea that are emphasized by the metaphor are:

- The ship has a sharp bow.
- The ship is partly submerged in water.
- As the ship moves through the sea, its bow pushes the water aside.

Similarly, the aspects of plowing that are emphasized are:

- The plow has a sharp edge.
- The plow is partly submerged in earth.
- As the plow moves through the field, its edge pushes the earth aside.

The aspects of ship's motion that are downplayed by the metaphor include:

- The water around the ship is salty.
- There are people inside the ship.

Similarly, some of the aspects of plowing that are downplayed are:

- Plowing loosens the earth and thereby makes it easier for the newly germinated seed to come out.
- Plowing removes old roots and weeds.

T-MAPs capture this property of metaphors because of their partial nature. Given a T-MAP T = <F, S> that represents an interpretation of some metaphor the tokens that are included in F, and the structure S of the source domain is highlighted by the metaphor. The structure of the the target domain that is emphasized by the metaphor is the set of tokens that form the image of F and all the metaphorical inferences that can be drawn in the target domain under the interpretation given by T. For instance, in the example just given we first show how this interpretation of the metaphor can be characterized with a T-MAP by using the domains presented in Section 4.2. The partial mapping that maps tokens of PLOWING to tokens of SAILING is given by:

PS1: {(farmer → captain); (plow-o → ship);
(plow-a → sail); (field → sea);
(earth → water); (plow-edge → ship-bow)}

The structure of PLOWING that is transferred by this T-MAP is:

1. controls(farmer, plow-o)
2. partially-submerged(plow-o, earth)
3. moves-through(field, plow-o)
4. plow-a(field, plow-o)
5. contains(field, earth)
6. contains(plow-o, plow-edge)
7. sharp(plow-edge)
8. harder-than(plow-edge, earth)

It can be easily verified from the structural constraints of SAILING given in Section

4.2 that this T-MAP is strongly coherent.

In characterizing the metaphor with this T-MAP we notice that the tokens of the source and target domains that are included in PS1 are highlighted. Further, the structure of PLOWING that is represented by the structural constraints given above is emphasized. The structure of the target domain SAILING that is highlighted by the metaphor consists of all those sentences of the target domain that can be called metaphorical inference under PS1 according to the definition given above. They include the following structural constraints of SAILING.

1. controls(captain, ship)
2. partially-submerged(ship, water)
3. moves-through(sea, ship)
4. sails(sea, ship)
5. contains(sea, water)
6. contains(ship, ship-bow)
7. sharp(ship-bow)
8. harder-than(ship-bow, water)

Any token of the target domain that is not included in the mapping and any structural constraint that is not highlighted is downplayed. Thus, in the above example, the structural constraint of SAILING that corresponds to "the ship is used to transport objects or people across water" is downplayed. Similarly, for the source domain, for any token that is not included in the mapping or any structural constraint that is not in S we say that it does not participate in the metaphor.

We emphasize again that the T-MAP given above characterizes an interpretation of the metaphor. The metaphor "the ship plowed through the sea" is subject to several interpretations. For instance, the representations of SAILING and PLOWING used in Section 4.2 are fairly arbitrary. In general different people may have very different representations of the same domain based on their experiences and knowledge. Even with the same representation, there may be several T-MAPs characterizing different interpretations of the same metaphor. In the above example, for instance, we can also interpret the metaphor by the following mapping:

PS2: {(farmer → crew); (plow-o → ship);
(plow-a → sail); (field → sea);
(earth → water); (plow-edge → ship-bow)}

In this case, the structure of PLOWING that is transferred by the T-MAP remains the same. But in this case the structural constraint corresponding to "the captain controls the ship" is no longer a metaphorical inference and is therefore not highlighted. Instead the constraint that corresponds to "the crew controls the ship" is highlighted by PS2. We note here that the representation of SAILING includes both these facts without any bias. The emphasis is created solely by the T-MAP characterizing the metaphor.

A related property of this characterization that we would like to mention here is that both the source and the target domain participate in the highlighting and downplaying of the target domain. To adapt a metaphor from Goodman [1968, p. 69], the target domain yields to the structure of the source domain while protesting. Thus, changing the source domain will result in different features of the target domain being highlighted, as in "the ship sought its way through the sea". In this

metaphor, the captain's or the crew's role in controlling the ship is downplayed. Similarly, applying the same source domain to different target domains will result in different structural constraints of the source domain participating in the metaphor. Thus, the "plow" metaphor used in "the chairman plowed through the discussion" will cause different constraints of PLOWING to be used in highlighting the target domain MEETING (not presented here).

4.3.4 ENRICHING THE TARGET DOMAIN

Another important property of metaphors is that often they derive their usefulness from the fact they allow naming a system of relationships in the target domain for which no corresponding literal term exists. For instance, in the example "node A is sibling of node B", there is no token in the domain DAG which corresponds to the token *sibling*. In another example, in "the chairperson plowed through the discussion", there is no exact term in the vocabulary related to meetings, discussions etc. that means the same as the metaphor. Because of this property a metaphorical statement can replace several literal statements that convey approximately the same concept.

The operator Augmentation, that we described in the previous chapter, captures this property of metaphors. In the context of a T-MAP that maps some tokens of the source domain to tokens of the target domain, any token of the source domain that can be defined by using tokens included in the T-MAP can be transferred to the target domain even though no corresponding token may exist there. Thus, Augmentation enriches the vocabulary of the target domain by creating

a name for a structure of relationships in the target domain.

There is another sense in which metaphors enrich the target domain which can best be explained by means of an example. Consider "the sky is crying".[1] In interpreting this metaphor, apart from the intended meaning to the effect that it is raining, one sees an extra shade of meaning emphasizing—or rather creating—a human perspective on the sky. Thus, the metaphor suggests a notion of sadness and grief that is attributed to the sky and that is absent from the corresponding literal utterance "it is raining". Similarly, in understanding "the ship plowed through the sea" one may attribute farmerlike properties to the captain. This extra shade of meaning is created solely by the presence of the metaphor.

This feature of metaphors is captured in our theory by the operator Positing Structure, defined in the previous chapter, that allows extra structure to be posited in the target domain. Thus, for instance one can attribute human characteristics to the sky, or farmer's attributes to the captain as the case may be. The important thing to note here is that the positing of structure is constrained by the structure of the source domain. Thus different metaphors can potentially enrich the same target domain in different ways. Whereas in "the ship plowed through the sea" the metaphor suggests that some attributes of the farmer be applied to the captain, in "the ship sought its way through the sea" the metaphor allows one to attribute human characteristics to the ship. We also note that this process is very subjective and one may see more or less meaning in a metaphor depending on one's tastes.

---

1. Cf. Chapter 5 for an analysis of this metaphor in our theory.

Correspondingly our operator Positing Structure is also very general and allows new structure to be posited to the target domain in several possible ways.

4.3.5 SCIENTIFIC MODELS IN CST

Scientific models and metaphors often have the characteristic that the target domain is less structured—or at least less of its structure is known—and cognitively remote from the subject compared to the source domain. Correspondingly they derive their usefulness in two ways. Firstly, the metaphors in science bring the target domain within the cognitive grasp of the subject by talking about it in terms of more familiar objects as in "gas molecules are like billiard balls". Secondly, and more importantly, as observed by Hesse [1974, 1980] and Turbayne [1962], they can determine the direction of scientific growth by controlling the way in which questions about the target domain are posed.

In our theory we divide the growth of scientific models into two stages. In the initial stage, when the model is first suggested, the target domain is usually less understood and therefore its representation will have much less structure to it. Thus, there will be several possible weakly coherent T-MAPs from the richly structured source domain. The process of verifying coherency in this case amounts to active experimentation in the target domain. In other words, after an initial T-MAP is established, one can make several hypotheses about the target domain that are based on one's knowledge of the source domain. An attempt to verify these hypotheses leads to an increased knowledge of the target domain. In making the hypotheses one either attempts to extend the initial T-MAP by transferring more structural

constraints of the source domain, or resort to Positing Structure, or a combination of the two. Thus, the source domain will influence which questions are asked about the target domain and thereby affect scientific growth.

In the second stage, the model becomes a tool of analogical reasoning. At this time the T-MAP corresponding to the model becomes a strongly coherent one and it derives its usefulness from the fact that the subject is more familiar with the source domain and therefore finds it easier to talk about the target domain in terms of the source domain. Further, if one knows what structure of the source domain is transferred by the T-MAP then one will know which inferences of the source domain can be safely applied to the target domain after being appropriately transformed.

4.3.6 ORIENTATIONAL METAPHORS IN CST

So far the examples that we have considered are instances of what Lakoff & Johnson [1981] have called structural metaphors. In this and the next few sections we will show how some other classes of metaphors can also be characterized in our theory of CST. In order to show this we will first present some examples of each class of metaphors and then argue that the notion of coherency lies at the heart of these other metaphors too. We will not be presenting SL representations of the domains involved or explicit T-MAPs characterizing an interpretation of the metaphors involved in each case but hope that the examples presented earlier coupled with the reader's imagination will lend sufficient force to our arguments.

We will begin by considering orientational metaphors in this section. Orientational metaphors are those that impose a spatial orientation on a concept. Some examples, taken from Lakoff & Johnson, are as follows:

1. *Happy is up; Sad is down:* I'm feeling up. My spirits sank. That boosted my spirits. I fell into a depression.
2. *More is up; Less is down:* My income rose last year. The numbers of errors she made is incredibly low.
3. *Good is up; Bad is down:* Things are looking up. We hit a peak last year. She does a high-quality work. Things were at an all time low.

Lakoff & Johnson observed that any orientational metaphor, for instance "good is up; bad is down", defines a coherent system rather than a number of random and isolated cases. We claim that all these metaphors can be characterized as T-MAPs in CST since they all transfer part of the structure of the spatial domain to their respective target domains. Thus in the case of "good is up; bad is down" metaphor the opposition of up and down in the source domain is preserved in the target domain by the opposition of good and bad. Of course, in our theory the T-MAP corresponding to "good is down; bad is up" will also preserve this opposition and therefore can give rise to metaphors. The reason that the former metaphor is preferred is that the corresponding T-MAP also transfers the asymmetry of up and down—which is that the up posture is normal and more desirable whereas the down condition usually results from tiredness, sickness, or defeat—across to the target domain.

This multiplicity of coherent mappings is more noticeable in the motion metaphors of time. It has been observed that in the description of time in terms of motion there are two commonly used metaphors in English. One of them looks at time as a moving object and assumes that we are stationary. This metaphor is used in sentences like, "in the following weeks . . .", "in the preceding weeks . . .". The other metaphor looks at time as stationary and looks upon us as though we were moving through it. This metaphor is apparent in sentences like, "in the weeks ahead . . .", "forget what is behind you and look ahead to the days to come". Both these metaphors correspond to different T-MAPs that carry the structure of motion across to the domain of time. Further, both these metaphors transfer the asymmetry of motion which is preserved by the asymmetry of time.

4.3.7 ONTOLOGICAL METAPHORS IN CST

Lakoff & Johnson define ontological metaphors to be those that view events, activities, emotions, ideas etc. as entities, substance or persons. Some examples are:

1. *Mind is a brittle object:* Her ego is very fragile. He broke down under pressure. The experience shattered her.
2. *Mind is a machine:* I'm a little rusty today. Her wheels are turning now! He is still grinding out the solution to this problem.
3. *Visual fields are containers:* The ship is coming into view. I have her in sight. I can't get everyone in sight at once.
4. *Personification:* His theory explained to me the behavior of chickens. Her religion tells her that she cannot drink wines. This fact argues against your theory.

We claim that all these cases are examples of structural coherency. For instance in

the "mind is a machine" metaphor the structure of the source domain MACHINE is transferred across to the target domain MIND. Similarly in "mind is a brittle object" the source domain is BRITTLE-OBJECT though the target domain remains the same. As we discussed before, different source domains cause different aspects of the target domain to highlighted or emphasized. In other words the set of metaphorical inferences that can be derived about the target domain will be different in each case. This explains why, as Lakoff & Johnson observe, "he cracked up" carries the implication that the cracking up may be harmful to other people around him whereas "he broke down" does not carry any such implication.

Similarly, in the "visual fields are containers" metaphor the structural relationships among the tokens of CONTAINERS are preserved in the domain of VISUAL-FIELD.

Personifications differ from other ontological metaphors in that their interpretations are characterized by T-MAPs that have been extended by a liberal use of Positing Structure. For instance in "his theory explained to me..." the humanlike property is ascribed to theories though there is no corresponding property in its domain. This creation of properties in the target domain is constrained by the structure of the source domain, or in other words by the normal usage of the tokens of the source domain.[1]

---

1. We notice here that in all the examples of personification the target domain is much less structured and does not contain much interesting vocabulary. Perhaps this is the reason that metaphors are so often used in describing them.

4.3.8 METONYMIES IN CST[1]

In a metonymy an entity is used to refer to another entity that is related to it. A detailed analysis of metonymic reference can be found in Nunberg [1978]. We give below some examples taken from Lakoff & Johnson.

1. *Part for the whole:* We don't hire longhairs. The Giants need a stronger arm in the right field.
2. *Producer for the product:* I will have a Heineken. She bought a Picasso. I love reading Wittengenstein.
3. *Object used for the user:* The BLT is a lousy tipper. The gun she hired wanted fifty grand. The sax is sick with flu today.
4. *Controller for the controlled:* Reagan invaded Grenada. Karajan gave an excellent concert.

In our theory of CST, metonymies correspond to one-to-one T-MAPs that have no structure associated with them but are characterized by the fact that there is a uniform way to build the mapping. Consider "producer for the product" metonymies for instance. In this case the source domain consists of all producers and the target domain consists of products. Now we can associate with every producer their product in a one-to-one fashion. The associated T-MAP does not transfer any structure but it has the characteristic property that given an element of the source domain (or target domain) there is a systematic way to arrive at its counterpart in the target domain (or source domain).

---

1. This section is based on an interesting discussion I had with Remko Scha.

4.3.9 APTNESS OF METAPHORS

The appropriateness of a metaphor or analogy is a very complex issue and several mechanisms for characterizing the aptness of metaphors have been proposed in the literature. For instance, we recall from Chapter 2 that Gentner's systematicity principle uses the existence of the higher order relations to describe aptness. Earlier on, the aptness was characterized in terms of the similarity between the source and the target domains. The anomaly theory suggests that more dissimilar are the domains the better is the metaphor. On the other hand the comparison theory suggests that the more similar the domains, the more apt the metaphor. There is another hypothesis that characterizes aptness as an inverted-U function of similarity, i.e. the aptness increases as the domains get more and more similar up to a point and then it starts decreasing as the metaphor collapses into literal similarity.

Tourangeau & Sternberg [1982] offered an aptness criterion that we described earlier in Chapter 2 and will repeat here. According to their hypothesis the aptness of metaphors increases with the degree of within-domains similarity and decreases with the degree of between-domains similarity. They also offered psychological evidence in support of their hypothesis.

A major problems with all these hypotheses, except Gentner's, is that since they try to explain the aptness of metaphors in terms of the source and target domains alone, they fail when there are several possible metaphors between the same pair of domains. Thus, for instance, they cannot explain why "good is up; bad is down" is a better metaphor than "bad is up; good is down". Gentner's systematicity principle does not have this problem because it tries to focus on the characteristics

of a mapping rather than on the domains themselves. But the systematicity principle cannot be applied to CST because we do not require that any relations be preserved.

We are not offering any simplistic criteria for determining the aptness of metaphors because we think that the issue is a complex one and there are several factors that affect aptness. There can be several possible T-MAPs between any pair of domains and which are more apt than others depends on the speaker's intentions, the aspects of both the domains that the speaker thinks are important, the assumptions that the speaker makes about the hearer's knowledge of the domains etc. Due to the complexity of this issue we will only paraphrase Gentner's and Tourangeau & Sternberg's hypotheses in our theory to provide aptness criteria for certain metaphors and leave the issue open.

As we explained in Chapter 2, Gentner's systematicity principle states that the relations that participate in forming higher-order relations are more likely to be preserved in a mapping characterizing an analogy. This principle is based on the assumption that the relations between objects are preserved from the source to the target domain. Since CST also allows relations to be mapped to other relations we offer a *generalized systematicity principle* to replace the systematicity principle. Intuitively, the generalized systematicity principle states that the T-MAPs that transfer a set of richly interconnected tokens are more interesting, or apt, than the ones that transfer isolated structural constraints. By a richly interconnected set of tokens we mean that there are many structural constraints relating the meaning of the tokens with each other. In quantitative terms, given a T-MAP <F, S>, this can

be expressed as a high ratio of the number of structural constraints in S to the tokens included in F [Cf. Hayes 1975].

To paraphrase Tourangeau & Sternberg's notions of *within-domains* similarity and *between-domains* similarity in our theory we proceed roughly as follows. Within-domains similarity, with respect to a T-MAP <F, S>, is characterized by the number of the structural constraints of the source domain that are included in S. Thus, it characterizes how much of the structure of the source domain is transferred to the target domain by the metaphor. The second form of similarity refers to the degree to which the source and the target domains are similar to each other. One way to characterize this between-domains similarity in our theory could be by looking at how many internal tokens are common to both the source and the target domains. A large number of common internal tokens will mean a greater degree of between-domains similarity and vice versa. There can, of course, be other ways to characterize this notion, for instance by using some notion of semantic distance among the tokens of the source and the target domain.

Given these new definitions of within-domains and between-domains similarity we can now apply Tourangeau & Sternberg's criteria to decide aptness of a metaphor. In this characterization "good is up; bad is down" turns out to be a better metaphor than "good is down; bad is up" for the following reason. In both these metaphors the opposition of up and down in the source domain is preserved by the opposition of good and bad in the target domain. The difference lies in the fact that in the "good is up; bad is down" metaphor the asymmetry of up and down, namely that *up* is a normal and more desirable position and *down*—a position

that often results from sickness, tiredness, or defeat—is a less favorable state, is also preserved by the asymmetry of good and bad. Thus, this metaphor transfers more structural constraints than the "good is down; bad is up" metaphor, thereby providing a greater degree of within-domains similarity.

4.4 COMPARING CST WITH OTHER THEORIES OF METAPHOR

In this section we will compare our theory of CST with some of the more prominent theories of metaphor that were discussed in Chapter 2. We will begin by pointing out that though from the discussion of the previous section it may seem that our theory of CST characterizes the *substitution* view of metaphor in fact our theory goes beyond it for two main reasons. The first is that in our theory we allow a term from the source domain to stand for a system of relationships in the target domain for which there is no literal name. Any T-MAP that has been augmented at least once will have this property. This property of metaphor cannot be explained in the substitution view of metaphor.

The second difference between the substitution view of metaphor and CST is somewhat deeper. Even in the case that a literal term, corresponding to the metaphor, does exist in the target domain, CST differs from the substitution theory by providing coherency as a criterion for deciding under what conditions the substitution can be made. Further, in CST we can explain why, even though only one term of the source domain was substituted, several related terms and their relationships also play a significant role in comprehending a metaphor. Similar observations can be made concerning the comparison theory.

Our theory is more in the spirit of the interaction theory of metaphors [Black 1962a, 1979], especially with the domains-interaction view [Tourangeau & Sternberg 1982]. CST is a step ahead of any of these theories in spelling out the coherency condition precisely, as well as providing operators like Augmentation and Positing Structure that make it more powerful. We note that the T-MAPs are produced as a result of interaction among the structural constraints of the source and the target domains.

Unlike MacCormac's theory, that relies on fuzzy-set theoretic notions, CST does not make metaphorical utterances *less* true then literal utterances. Rather, once an interpretation of a metaphor is fixed and it is determined which relationships from the source domain are to be transferred to the target domain, the same criteria that would be applied in determining the truth of literal statements about the target domain will also be applied to the metaphor.

Comparing CST with Hobbs' theory we first notice that whereas Hobbs' required that the set of inferences associated with the source and the target domain to be expressed by using a common vocabulary, CST does not require this though identity mappings are by no means ruled out. Thus where Hobbs only allowed identity mappings for horizontal links, CST also allows several other mappings besides identity. Hobbs provided criteria for dividing the expectations, or structural constraints in our terminology, of the source domain in three classes by a metaphor. Firstly, there are those that definitely satisfy the expectations of the target domain. Then, there are those that definitely contradict the expectations of the target domain. Finally, the most interesting class, which makes a metaphor richer in

meaning than the corresponding literal statement, contains those expectations of the source domain that do not satisfy the expectations of the target domain but do not contradict them either. In CST this distinction is captured by the concepts of weak and strong coherency. Given a mapping F, the first class corresponds to those structural constraints of the source domain that are implied by the structural constraints of the target domain (after being transformed) or in other words those constraints of the source domain that F is strongly coherent with. The second class corresponds to those constraints of the source domain that are contradictory with the structural constraints of the target domain. These are the constraints that F is not coherent with. Finally, the third class will contain those structural constraints of the source domain that are consistent with the constraints of the target domain but are not implied by them. That is, these constraints can be added to the strongly coherent T-MAP by making it weakly coherent.

In comparing CST with Gentner's theory we note that her notion of structure mapping is a special case of a T-MAP. In CST we do not distinguish between objects, attributes, or relations and also allow mappings, not necessary the identity, among attributes and relations. Further, her systematicity principle amounts to saying that those T-MAPs that transfer a significant amount of structure are more interesting from a cognitive point of view. Since Gentner's theory is a special case of CST, the psychological evidence cited by her in favor of her theory also strengthens CST.

4.5 LIMITATIONS OF CST AS A THEORY OF METAPHORS

So far we have shown how CST can provide a first step towards a theory of metaphor. We demonstrated, by using some simple examples, how several cognitive features of structural metaphors and analogies can be explained in our theory of CST. In this section we will point out some of the major limitations of our formalization and indicate possible ways to overcome these limitations. Some, but not all, of these limitations will be removed in the next chapter when we develop a theory of Approximate Semantic Transference.

One of the major limitations of this theory is that it does not explain the process by which some metaphors that seem to be contradictory or incoherent at first sight can *create* very novel perspectives on the target domain. Many of the poetic metaphors share this spirit and some of them derive their usefulness merely because of this property. A related feature of metaphors that we do not address in our theory is how a metaphor can *create* a new perspective on the target domain. Our theory shows how certain features of the target domain can be highlighted and certain others downplayed but not how new features are created in it.[1]

The explanation we offer for this phenomenon is as follows. In general in comprehending a metaphor there is a three-way tension going on rather than a two-way tension as we explained in our theory. The interaction taking place in understanding a metaphorical sentence is not merely between the source domain and

---

1. Of course, the operator Positing Structure allows one to posit the existence of some rather arbitrary structure in the target domain as we discussed in Section 4.3 but we are referring to a different sense of creativity here.

the target domain but among the source domain, the target domain and the object that is represented in the target domain. Thus in interpreting the metaphor "the ship plowed through the sea" the domains PLOWING and SAILING interact with themselves and with the actual process of sailing to produce an interpretation of the metaphor. The reason for this distinction is that our representation of an object or concept is often only an approximation of the real nature of that object or concept. For reasons that we discussed in Section 4.2 it is almost impossible to represent an object *completely* or even to know it completely. Thus our knowledge, and our representation, of an object is only an approximation and normally we perceive that object based on our biased or approximate representation. A metaphor can, by mere juxtaposition or other techniques, forces us to look beyond our representation of the object in order to make sense of the metaphor. This process can give rise to a new perspective on the object that was missing from our representation. Sometimes this three-way process can turn into a four-way process and the object represented by the source domain may be involved too.

Thus we are taking a position here that every object has its own properties and structure which may not be directly accessible or knowable by us. Our knowledge of the object consists of the experience that we, and others in the society—past or present—have with it. This knowledge is an approximation of the structure of the object but is not complete. Any *new* perspective that a metaphor can create is *new* only with respect to our representation of the object and is not new to the object. Coherency conditions will still be required but not between two domains or representations of two objects but between the objects themselves. The

coherency condition thus provides a criterion to decide between a failed metaphor and a successful metaphor. In the absence of any such condition any metaphor will be as good as any other and everything will be chaos.

A related aspect of metaphors that we did not address in our theory is the process of meaning change that is associated with a metaphor. It has been acknowledged that a metaphor normally goes through phases of being a novel metaphor, then a used metaphor, to finally becoming a dead metaphor. The meaning of the metaphor, as well as many of the associated words and phrases, also gradually changes through this process. Our theory does not provide any mechanisms for addressing the problem of meaning change.

One way to incorporate these features in our theory is to use credibility factors, or weights, associated with the structural axioms. Thus the domain representation of an object will contain the actual nature of the object, and not just our knowledge of the object. Our knowledge and biases will be reflected in the weights of the structural axioms. In this representation we will say that a new perspective is created by the metaphor if the corresponding T-MAP highlights the structural axioms of the target domain that have very low weights. The process of meaning change will be reflected in modified weights.

A positive aspect of this solution is that the weights can be used as useful heuristics in building computational models of metaphor comprehension as we will explain in the next chapter.

Some other limitations of our theory concern computability and finite representability.[1] Firstly, the domains in our theory are not finitely representable if they satisfy the axiom of closure. This is not surprising because if we want to use our theory to model processes of metaphor comprehension and generation then have to consider the fact that people do not carry all the implications of their knowledge in their head. Sometimes they may even be inconsistent and hold two contradictory pieces of information at the same time and not be bothered by it. The reason for this is that people in general always operate under time bounded or otherwise constrained situations and have to generate their inferences or decisions under these conditions. Thus, if the contradictory pieces of information are not active at the same time, the person holding them will not encounter any immediate problem and may not even notice it.

Another related shortcoming of our theory concerns the fact that the concept of coherency, as we defined it, is not computable. This is because it cannot be decided if a set of axioms is consistent or if a set of axioms implies another set of axioms.

We will show in the next chapter how these last two limitations of our theory can be removed by not requiring closure of the domains and replacing the coherency requirement of a T-MAP to coherency within a specified complexity bound. These will be referred to as Approximate T-MAPs, and the resulting theory called the theory of Approximate Semantic Transference.

---

1. Cf. Partee [1979a; 1979b] for a discussion of issues concerning computability and finite representability of semantic theories.

4.6 SUMMARY

In this chapter we demonstrated how CST can be used as a theory of metaphors. We began with a discussion of the issues involving the representation of "real-world" domains and why there is no unique way to represent them. We also argued that this variety of representation is necessary for any theory of metaphors.

Then we showed how T-MAPs can be used to characterize interpretations of metaphors. In our characterization T-MAPs provide the context in which it is possible to use some terms of the source domain along with some terms of the target domain in a meaningful fashion. We also developed the concepts of metaphorical truth and metaphorical inference in our theory. We also pointed out how several cognitive features of metaphors can be explained in our theory. We also provided a characterization for analogies, orientational metaphors, ontological metaphors, and metonymies.

We briefly compared our theory of CST with some other theories of metaphor and then discussed some limitations of CST. A major limitation of CST is that it is not computational and therefore cannot model the cognitive processes involved in understanding and comprehending metaphors and analogies. This limitation will be overcome in the next chapter.

"There's nothing like eating hay when you're faint," he remarked to her, as he munched away.

"I should think throwing cold water over you would be better," Alice suggested: "—or some salt-volatile."

"I didn't say there was nothing better," the King replied. "I said there was nothing like it." Which Alice did not venture to deny.

— Lewis Carroll, "Alice Through the Looking Glass".

"Well, just then I was inventing a new way of getting over a gate—would you like to hear it?"

"Very much indeed," Alice said politely.

"I'll tell you how I came to think of it," said the Knight. "You see, I said to myself `the only difficulty is with the feet: the head is high enough already.' Now, first I put my head on the top of the gate—then the head's high enough—then I stand on my head—then the feet are high enough, you see—then I'm over, you see."

"Yes, I suppose you'd be over when that was done," Alice said thoughtfully: "but don't you think it would be rather hard?"

"I haven't tried it yet," the Knight said, gravely; "so I can't tell for certain—but I'm afraid it would be a little hard."

— Lewis Carroll, "Alice Through the Looking Glass".

CHAPTER V

A THEORY OF APPROXIMATE SEMANTIC TRANSFERENCE

5.1 INTRODUCTION

In the previous two chapters we outlined a theory of Constrained Semantic Transference [CST] which allows a part of the structure of the source domain to be transferred to the target domain coherently. We also demonstrated, by means of some examples, how several properties of structural metaphors and analogies can be explained in our theory of CST. At the end of the previous chapter we pointed out some of the limitations of CST as a theory of metaphors. One of the limitations of CST discussed was that it is not computational and therefore it does not constitute a model of the cognitive processes involved in understanding and generating metaphors and analogies as it is.

In this chapter we develop another theory, which we call a theory of Approximate Semantic Transference [AST henceforth], which is based on CST but is computational. AST is different from CST in two major aspects. Firstly, in AST we do not require that the set of structural constraints of a domain be closed under entailment but instead require that it be finite. Secondly, we relax the coherency requirement on a T-MAP to *approximate* coherency, also called τ-coherency, which is coherency within some specified complexity bound τ. Approximate coherent mappings will be called AT-MAPs. Of course, we will explain these notions in greater detail in the next section.

AT-MAPs are very similar to T-MAPs and most of the discussion of Chapter 4, on how they can characterize structural metaphors and analogies, also applies to them and therefore we will not repeat it in this chapter. In subsequent sections we will be addressing the problem of generating AT-MAPs from the domain knowledge.

We are not presenting a complete system that understands or generates metaphors, or makes deductions based on analogies, since each of these processes involve several other problems—unrelated to metaphors—that have not been fully understood yet and are beyond the scope of this thesis. AT-MAPs are proposed as a means for characterizing metaphors and analogies. In a Natural Language Understanding System, they provide a context in which the incoming utterance is to be interpreted. In a Natural Language Generation System, they provide a context for generating the next utterance. In a reasoning system they provide a context, or a framework based on analogy, in which reasoning about the target domain, based on

knowledge of the source domain, can be carried out. Any module that computes AT-MAPs has to be embedded in a higher-level system that *understands* the incoming utterance, *generates* a new utterance, or *reasons* about a domain. Therefore the best strategy for computing AT-MAPs in an actual system will depend on the control structure of the higher-level system and the particular way in which it uses the information contained in an AT-MAP. We will suggest several principles that can be useful in designing algorithms for computing AT-MAPs in a variety of situations. We will also suggest possible ways these algorithms can be embedded in a higher-level system.

We will begin by presenting, in Section 5.3, some algorithms for determining τ-coherency of a given mapping. We will also present an algorithm that accepts an admissible mapping as input and returns a set of structural constraints of the source domain such that the given mapping is approximately coherent with respect to this set. These algorithms will be used by algorithms in Section 5.4 in computing AT-MAPs from the domain knowledge.

In Section 5.4 we will address the problem of computing AT-MAPs from the domain knowledge. We will present some algorithms for computing AT-MAPs in different situations. We will suggest two heuristics, that are derived from the domain knowledge alone, that can help our algorithms in their search for AT-MAPs. We will also specify means by which the higher-level system can also guide our algorithms in their search. This facility is provided so that our algorithms can make use of the information, if any, contained in the higher-level system.

In Section 5.5 we will present some examples to illustrate how the algorithms described in Section 5.4 can be configured to produce a model of the process of interaction involved in understanding a metaphor. These examples will also illustrate the complexity of this task and several other problems involved with it.

In Section 5.6 we will present an alternative formulation of approximate coherency called ν-coherency. This formulation is mathematically more elegant and removes some of the deficiencies of the earlier version. In the absence of any heuristic the algorithms based on this formulation will exhibit an exponential growth in their complexity. We will present one such non-deterministic algorithm for determining weak ν-coherency of a mapping. Finally, in Section 5.7 we will summarize the main points of this chapter.

5.2 APPROXIMATE SEMANTIC TRANSFERENCE AND AT-MAPS

In this section we will develop a theory of Approximate Semantic Transference, which is a computational version of CST, and define AT-MAPs, which are computational counterparts of T-MAPs. We begin by adding two more features to the notion of *domain* defined in Chapter 3. Firstly, in AST we do not require that the domains satisfy the axiom of closure which means that it is is not necessary that if X is a consequence of the structural constraints of a domain then X be included in the structural constraints of that domain. Instead we require that the set of structural constraints of a domain satisfy the following axiom of finiteness.

Axiom of Finiteness: The set of structural constraints of a domain is finite.

Apart from this axiom of finiteness we require that the structural constraints of a domain satisfy the axiom of inclusion, axiom of relevance, and the axiom of significance mentioned in Chapter 3. Later in this section we will define τ-coherency and replace the axiom of consistency—presented in Chapter 3—with an axiom requiring only pairwise τ-consistency.

Secondly, we require that a measure of salience be defined over the internal tokens of a domain and another measure of salience be defined over its structural constraints. In other words, we require that there be two functions, g and h, associated with every domain where g is a function from the set of internal tokens to integers and h is a function from the set of structural constraints to integers.

One way to compute these salience measures from the information contained in the domain alone is as follows. Given a domain $D = \langle I, E, S, d, f \rangle$, we first compute the salience measure g on the internal vocabulary I of D by letting, for every internal token i in I, g(i) be the number of structural constraints in which i appears. Thus a token that appears in more structural constraints will have a higher salience than one that does not appear as often. Next we compute the salience of any structural constraint in S by taking the sum of the saliences of all the internal tokens that appear in that structural constraint. In other words, if X is any structural constraint of D then h(X) is defined to be the summation $\Sigma g(j)$ over all the tokens j that appear in X. We remark here that if a token appears more than once in X then its salience is added that many times in computing the salience of

X.[1]

The salience measures, if computed in the fashion described above, are nonmonotonic with respect to the addition of new information. That is, if new structural constraints are added to a domain then this will result in a new ordering among the internal tokens and the structural constraints of that domain that may be very different from the previous ordering.

It is also possible to assign salience to internal tokens and structural constraints empirically.

The salience measures that are assigned to a domain at the time it is defined, such as the ones computed in the above manner, are called static since they do not take into account the shift of focus as a result of changing context. Of course, if the higher-level system has enough information to redefine the salience in a domain then it is possible to either let it override the static salience, or to compute the new salience as some weighted mean of the static salience and the salience given by the higher-level system. This update can be carried out periodically to incorporate the change of context. We will call this updated salience dynamic salience. In the rest of this chapter whenever we use the term "salience" we will normally mean "dynamic salience" unless explicitly stated otherwise.

---

1. This heuristic for computing salience was suggested to me by Barbara Partee.

The salience of a token reflects the bias of an individual concerning which parts of the domain represented she believes to be more significant. Similarly, the salience of a structural constraint reflects what aspects of the parts involved are considered more important. The salience of a structural constraint should not be confused with the credibility factor that might also be associated with a structural constraint, as proposed at the end of the previous chapter. The difference is that the salience of a structural constraint only reflects the significance that a person attaches to some aspects of the domain whereas a credibility factor would represent her confidence in asserting the constraint. For instance, in the domain WEATHER, the structural constraint representing the fact "if it is raining then the sky is cloudy" may be given a low salience but will have a credibility factor of 1. On the other hand a structural constraint representing the fact "if the sky is cloudy then it is raining" may have a high salience but will have a credibility factor less then 1 since it is not always true. In AST we do not have credibility factors and therefore all the structural constraints have the same status with respect to their truth value, namely that they are all true.

Salience as an empirical heuristic has been exploited in the past by some Artificial Intelligence programs [Cf. Conklin 1983]. We exploit it in two different ways to reduce the computational complexity of our algorithms. Firstly the salience of the structural constraints is used to determine the order in which they are checked for coherency. Secondly, we use the salience of internal tokens to determine the priority in which the extensions of an AT-MAP are tested for coherency. This will, of course, become clearer when we present our algorithms.

Finally, we do not require the uniqueness condition for derivations in AST. We recall from Section 3.2 that the derivation of a token X, which denotes a function, is required to have the form $X = x \leftarrow\!\!\!\rightarrow Z$, where x is variable of the same type as X, such that the sentence $(\exists x)Z$ is a consequence of the structural constraints of the domain. This condition was required to ensure that the derivation introduces a new token uniquely. We do not require this condition in AST and therefore the uniqueness of derived tokens is not guaranteed.

Having described how domains of AST differ from domains of CST we now come to the notion of coherency which is another major aspect in which AST and CST differ from each other. We recall from Chapter 3 that in CST an admissible mapping[1] F from the source domain to the target domain is said to be weakly coherent with respect to a structure S, where S is a subset of the structural constraints of the source domain that is transformable by F, iff F(S) is consistent with the structural constraints of the target domain. Further, we say that F is strongly coherent with respect to S iff F(S) can be deduced from the structural constraints of the target domain. We notice that strong and weak coherency are both not decidable and since T-MAPs are defined as pairs <F, S>, where F is coherent (weakly or strongly) with respect to S, they are not computable.

---

1. An admissible mapping is a mapping that preserves the types of the tokens mapped.

To overcome the undecidability of coherency we introduce the notion of *approximate coherency* or τ-coherency as follows. Let P be an inconsistency-checker that is sound and memoryless. It accepts as input two sentences X and Y and returns FALSE if it can derive a contradiction from X and Y and TRUE otherwise. By soundness of P we mean that if P returns FALSE then X and Y are indeed contradictory in the semantic sense. In other words, P never returns FALSE if X and Y are consistent. Note that this does not exclude the possibility that there may be inconsistent pairs of sentences for which P cannot derive a contradiction, and for which it returns TRUE. By memoryless we mean that to derive a contradiction from X and Y, P will always take the same amount of time no matter how often it is called with X and Y as input and no matter what other contradictions were derived by P before. Given such an inconsistency-checker, an amount of time τ, and two structural constraints X and Y, we say that X is τ-provable from Y if P can derive a contradiction between ~X and Y in time less than or equal to τ. Further, we say that X is τ-consistent with Y if P cannot derive a contradiction between X and Y in time less than or equal to τ. We remark here that, as is to be expected, the notion of τ-consistency as defined above is symmetric though τ-provability is not symmetric. All the following definitions are in the context of P and we will not mention it explicitly.

Instead of requiring that the set of structural constraints of a domain be consistent, in AST we only require that they be pairwise τ-consistent. In other words, given any two structural constraints of an AST domain, P should not be able to derive a contradiction from them in time less than or equal to τ. Since the

possibility of pairing a structural constraint with itself is not excluded, this ensures that the structural constraints are themselves τ-consistent.

We now define τ-coherency as follows. Given an admissible mapping F from the source domain to the target domain and a set S, which is a subset of the structural constraints of the source domain that is transformable by F, we will say that F is *weakly τ-coherent* with respect to S iff for every structural constraint X in S and for every structural constraint Y of the target domain F(X) is τ-consistent with Y. Further, we will say that F is *strongly τ-coherent* with respect to S iff for every structural constraint X in S there is some structural constraint Y of the target domain such that F(X) is τ-provable from Y. If F is weakly τ-coherent or strongly τ-coherent with respect to S then we say that F is τ-coherent with respect to S. Further, if F is coherent with respect to all the transformable structural constraints of the source domain then we will say that F is maximally τ-coherent.

In the rest of the discussion in this chapter we will implicitly assume the existence of some time complexity measure τ and will not mention it unless it is necessary. Thus, we will use the terms strong coherency for strong τ-coherency and weak coherency for weak τ-coherency. We will also often use the term approximate coherency in place of τ-coherency.

Finally, we define the notion of AT-MAP as follows. An AT-MAP from the source domain to the target domain is a pair <F, S> where F is an admissible mapping from the source domain to the target domain, S is subset of the structural constraints of the source domain that is transformable by F, and F is approximately coherent with respect to S.

We notice that a consequence of this definition is that even those conditions of the target domain that can be trivially derived from its structural constraints do not play any role in determining approximate coherency of a mapping. Further, the transformability of the structural constraints of the source domain, with respect to any mapping, is very much dependent on their explicit form. For instance, conjoining a transformable structural constraint with a non-transformable one will result in a non-transformable constraint and the former constraint will no longer affect the coherency of the mapping. We will discuss these issues in more detail in Section 5.6 and propose an alternative formulation of approximate coherency that does not have this problem.

Having defined AT-MAPs, we will now briefly mention how the properties of T-MAPs that were discussed in Chapter 3 are affected by relaxing the coherency condition to approximate coherency. We first remark that the concepts of *extension*, *restriction* and *inverse* of a T-MAP, defined in Chapter 3, can be directly carried over to AT-MAPs. Thus, for instance, an AT-MAP T1 = <F1, S1> from the source domain D1 to the target domain D2 is said to be a restriction of an AT-MAP T2 = <F2, S2> iff F1 $\subseteq$ F2 and S1 $\subseteq$ S2. Similarly, the other definitions follow.

It is easy to verify from the definitions and the memorylessness property of the inconsistency-checker that all the restrictions of an AT-MAP are also AT-MAPs. In other words an incoherent mapping can never be extended to an AT-MAP. We will use this property in Section 5.4 to provide a pruning principle that will be useful in reducing unnecessary calls to the inconsistency-checker.

Further, we have the fact that a strongly coherent AT-MAP is weakly coherent too. It is not as obvious as in the case of T-MAPs but follows easily from the soundness of the inconsistency-checker, that the structural constraints of the domains are pairwise τ-consistent, and the definitions given above. On the other hand Theorem 1 does not hold for AT-MAPs, or in other words the inversion may not even preserve weak coherency and therefore the inverse of an invertible AT-MAP need not be an AT-MAP. Finally, since approximate coherency is decidable, AT-MAPs can be computed.

Before closing this section we would like to remark that the operators Augmentation and Positing Structure, defined in Section 3.4, can also be applied to AT-MAPs without any change to their definitions. Since both these operators are computational, including them does not affect computability of AST.

We will now present some algorithms for deciding approximate coherency and then will address the problem of computing AT-MAPs from the domain knowledge.

5.3 ON DECIDING APPROXIMATE COHERENCY

In this section we will present some algorithms that are useful in establishing coherency of a mapping, or determining the set of structural constraints of the source domain that the mapping is approximately coherent with. All the algorithms presented in this chapter will assume the existence of a sound and memoryless inconsistency-checker P. As mentioned in the previous section, P accepts as input three arguments X, Y and τ and returns a boolean value, TRUE or FALSE, in a time less than or equal to τ. If P can derive a contradiction between X and Y in

time less than τ then it returns FALSE. If it fails to derive a contradiction between X and Y at the end of time τ then it terminates and returns TRUE. For brevity the third argument τ will often be implicit.

In experimenting with these algorithms we used an inconsistency-checker that was based on the resolution principle [Robinson 1965]. But it is also possible to use a different kind of theorem prover, such as one based on the natural deduction principle [Smullyan 1968]. It is also possible to use a constructive theorem prover, such as the Boyer-Moore theorem prover [Boyer & Moore 1979], as an inconsistency-checker. In order to determine consistency of X and Y from any constructive theorem prover we will attempt to prove ~X from Y. If the proof can be obtained in time less than or equal to τ then it will return FALSE, meaning that X and Y are not consistent, and TRUE otherwise. We notice that in such a case the symmetry of τ-consistency is destroyed since a constructive theorem prover may be able to prove ~X from Y in time less than τ, thereby making X τ-inconsistent with Y, but may not be able to prove ~Y from X in time less than τ, thereby making Y τ-consistent with X.

All the algorithms presented in this chapter are independent of the particular principle or strategy used in P but their result may be affected. We already pointed out how using a constructive theorem prover as an inconsistency-checker may destroy the symmetry of τ-consistency. Among inconsistency-checkers that are based on the refutation principle replacing P with a faster inconsistency-checker may result in some mappings that were originally accepted as coherent, getting rejected as incoherent.

We first present an algorithm that determines whether a mapping is weakly τ-coherent with respect to a given structure.

Algorithm 5.3.1: Checking Approximate Weak Coherency

Input: An admissible mapping F from the source domain D1 = <I1, E1, S1, d1, f1> to the target domain D2 = <I2, E2, S2, d2, f2>; and a set $S \subseteq S1$ such that every structural constraint in S is transformable by F.

Output: TRUE if <F, S> is a weakly coherent AT-MAP and FALSE otherwise.

Algorithm:

1. If S or S2 is empty then return TRUE and terminate. Otherwise go to the next step.
2. Set X to be the most salient structural constraint of S.
3. Set S2′ = S2
4. Set Y to be the most salient structural constraint of S2′.
5. Call P with F(X) and Y as inputs.
6. If P returns FALSE then return FALSE and terminate. Otherwise go to the next step.
7. Set S2′ = S2′ - {Y}. If S2′ is empty then go to the next step. Otherwise go to step 4.
8. Set S = S - {X}. If S is empty then return TRUE and terminate. Otherwise go to step 2.

Next we present a similar algorithm that determines whether a mapping is strongly τ-coherent with respect to a given structure.

Algorithm 5.3.2: Checking Approximate Strong Coherency

Input: An admissible mapping F from the source domain D1 = <I1, E1, S1, d1, f1> to the target domain D2 = <I2, E2, S2, d2, f2>; and a set S $\subseteq$ S1 such that every structural constraint in S is transformable by F.

Output: TRUE if <F, S> is a strongly coherent AT-MAP and FALSE otherwise.

Algorithm:

1. If S is empty then return TRUE and terminate. If S2 is empty then return FALSE and terminate. If both S and S2 are non-empty then go to the next step.
2. Set X to be the most salient structural constraint of S.
3. Set S2′ = S2
4. Set Y to be the most salient structural constraint of S2′.
5. Call P with ~F(X) and Y as inputs.
6. If P returns FALSE then go to step 8. Otherwise go to the next step.
7. Set S2′ = S2′ - {Y}. If S2′ is empty then return FALSE and terminate. Otherwise go to step 4.
8. Set S = S - {X}. If S is empty then return TRUE and terminate. Otherwise go to step 2.

Both these algorithms are very simple, given P, and it is easy to see that they indeed do what they set out to do. If S, the structure of the source domain that is being transferred, contains m structural constraints, and there are n structural constraints in the target domain D2 then the following complexity bounds on the algorithms can easily be derived. The Algorithm 5.3.1 will *always* take $mn\tau$ time to

return TRUE if its input happens to be a weakly coherent AT-MAP. In case of Algorithm 5.3.2, the worst case complexity is $mn\tau$ if its input is a strongly coherent AT-MAP. If we assume that on average for every structural constraint X of S we have to scan S2, the set of structural constraints of the target domain, half way through before the right one, that is the structural constraint that implies F(X), is found, and that P takes $\frac{\tau}{2}$ to find every proof, then the average case complexity for Algorithm 5.3.2 will be $(\frac{mn\tau}{2} + \frac{m\tau}{2})$.

Since every call to the theorem prover P may cost as much as τ time, it should be avoided as much as possible. One principle that can be used to reduce calls to P is as follows. If X and Y are two structural constraints such that the none of the tokens of X occur in Y then we know right away that X cannot be proved from Y and X and Y are consistent. This is assuming, of course, that X is not a tautology and neither X nor Y are themselves τ-inconsistent. Since the domains in AST satisfy the axioms of significance and their structural constraints are pairwise τ-consistent, both these assumptions will always be satisfied. Therefore, we can replace Step 5 in Algorithm 5.3.1 with Step 5a given below.

> **Step 5a (Algorithm 5.3.1):** If none of the tokens that occur in F(X) occur in Y then go to step 7. Otherwise call P with F(X) and Y as inputs and then go to step 6.

Similarly, Step 5 in Algorithm 5.3.2 is replaced with Step 5b as follows.

> **Step 5b (Algorithm 5.3.2):** If none of the tokens that occur in F(X) occur in Y then go to step 7. Otherwise call P with ~F(X) and Y as inputs, and then go to the next step.

In the next section we will mention another principle that can be used to avoid

unnecessary calls to the theorem prover.

To finish this section we present another algorithm that performs a somewhat different function. Whereas Algorithms 5.3.1 and 5.3.2 checked to see whether an input mapping is approximately coherent or not, the algorithm to be presented accepts as input an admissible mapping and returns a set of structural constraints of the source domain that can be transferred across to the target domain coherently by the input mapping. This algorithm uses Algorithm 5.3.1 or 5.3.2 depending on whether the mapping must be weakly coherent or strongly coherent with respect to the set of structural constraints that are returned. Some of the algorithms presented in the next section will be making use of this function.

Algorithm 5.3.3: Collecting Approximately Coherent Structural Constraints

Input: An admissible mapping F from the source domain D1 = <I1, E1, S1, d1, f1> to the target domain D2 = <I2, E2, S2, d2, f2>.

Output: A set S ($S \subseteq S1$) of structural constraints of D1 such that <F, S> is an AT-MAP from D1 to D2. Whether the AT-MAP is weakly coherent or strongly coherent will depend on whether Algorithm 5.3.1 or 5.3.2 is used in step 5.

Algorithm:

1. Set S = ∅. Set S1′ = S1.
2. If S1′ is empty then return S and terminate. Otherwise go to the next step.

3. Set X to be the most salient structural constraint of S1′.
4. If X is transformable by F then go to the next step. Otherwise go to step 7.
5. Call Algorithm 5.3.1 or 5.3.2 to check if F is approximately coherent with respect to {X}.
6. If the previous step returns TRUE then set S = S ∪ {X} and go to the next step. Otherwise go to the next step.
7. Set S1′ = S1′ - {X} and go to step 2.

Having described how the coherency of a mapping can be checked, we now turn to the problem of computing AT-MAPs from domain knowledge.

5.4 ON COMPUTING AT-MAPS

In this section we will describe some algorithms for computing AT-MAPs in different situations and suggest ways in which they can be embedded in a higher-level system. We will begin by describing some principles and heuristics that can be used in designing algorithms for computing AT-MAPs. The first principle is to use an incremental approach in generating AT-MAPs. By an incremental approach we mean that we always start with a given AT-MAP, possibly empty, extend it by mapping another token from the source domain to an appropriate token of the target domain and then check the resulting mapping for coherency.

The second principle, which we call the *pruning principle*, is based on the property of AT-MAPs mentioned in Section 5.2, namely that all restrictions of an AT-MAP are also AT-MAPs. If by the term k-element AT-MAP we mean an AT-MAP that maps k tokens from the source domain, then the pruning principle is

stated as follows:

Pruning Principle: If T is a k-element AT-MAP then all its (k-1)-element restrictions must be (k-1)-element AT-MAPs.

These two principle can be used in several ways to reduce the expensive calls to the inconsistency-checker since, as we mentioned earlier, every call to the inconsistency-checker may result in as much as τ time. For instance, if one is searching for AT-MAPs in a breadth-first manner[1] then one can keep the list of (k-1)-element AT-MAPs and before checking the coherency of a k-element mapping we can generate its (k-1)-element restrictions and verify by looking in the list of (k-1)-element AT-MAPs that they do occur there. Similarly, if a depth-first control structure is being used, as will generally be the case, one can use the pruning principle combined with the incremental approach to reduce the number of structural constraints that must be checked for coherency. All this will be made clearer later in this section.

Next we define the notion of structural connectivity that will be used as a heuristic in forming the set of plausible extensions of an AT-MAP. A token X is said to be *structurally connected* with a token Y if they appear together in at least one structural constraint of the domain. Further, we define the measure of structural connectivity of X with Y to be the number of structural constraints in which they appear together. Obviously, structural connectivity is a symmetric notion i.e. the

---

1. As we will point out later in this section, breadth-first search should only be used for off-line computing of AT-MAPs with domains that have a small number of tokens.

structural connectivity of X with Y is the same as the structural connectivity of Y with X.

From this we can now define structural connectivity of a token X with a set of tokens Y. We first define *pseudo structural connectivity* of a token X with a set of tokens Y to be the number of structural constraints in which all the tokens in Y appear along with X. The *structural connectivity* of X with Y is then the summation, over every subset Z of Y, $C(X,Y) = \Sigma C_p(X,Z)\ W(n(Z))$, where n(Z) is the number of tokens in Z, $C_p(X, Z)$ is the pseudo structural connectivity of X with Z, and W is some weighing function that increases monotonically with n such that $W(0) = 0$ and $W(1) = 1$. A simple example of W could be the identity function. The intuition behind weighing the summation terms by the number of tokens in the set is that total connectivity with a large set of tokens is more significant than connectivity with individual tokens in the set. In the rest of this section we will use the function C to represent structural connectivity, i.e. if X is some token and Y is a set of tokens then C(X, Y) will denote the connectivity of X with Y.

Having presented some notions that will be useful in designing algorithms for computing AT-MAPs we now turn to the algorithms themselves. As we discussed before, since we are not presenting a complete system that *understands* metaphor, our algorithms should be taken as suggestive and actual algorithms should be designed after studying the needs of the higher-level system. To give a general algorithm that generates AT-MAPs efficiently in all situations is a formidable task and beyond the scope of this thesis, though in specific situations one can take advantage of the peculiarities of the domains and design heuristics that can guide

the search for AT-MAPs.

Before an algorithm for generating AT-MAPs can be used we will first have to identify the source and the target domains. Most often this will be done by the higher-level system since it may know from the context what these domains are. For instance on encountering the utterance "the sky is crying" the higher-level system will recognize that the token *sky* is from the domain WEATHER and the token *cry* is from the domain EMOTIONAL-STATE. It will also know from the context which of the domains is the source domain and which is the target depending on whether a person is being talked about and "sky" is being attributed to her eyes or weather is being talked about and "crying" is being attributed to sky.

In case that the higher-level system does not have clearly defined domains we suggest the following algorithm, based on the notion of structural connectivity defined earlier, to determine the source and target domains. We assume that the higher-level system has all the structural constraints, belonging to all the domains, stored in some data base that is accessible to our algorithm. The structural constraints are assumed to be stored in such a fashion that it is possible to retrieve all structural constraints in which a token, or a set of tokens, occurs. We also assume that all the tokens and their types are stored in the data base and are accessible to our algorithms. Further, we assume that the higher-level system can sense the presence of the metaphor and isolate the tokens. For instance, on encountering "the ship plowed through the sea" we assume that the higher-level system can isolate the tokens *ship*, *sea*, and *plow*. The Algorithm 5.4.1 is now used to separate these tokens into two sets corresponding to two domains.

Algorithm 5.4.1: Separating Tokens Corresponding to Two Domains

Input: A set of tokens X.

Output: A two-set partition of X. In other words, two sets of tokens Y and Z such that $Y \cup Z = X$ and $Y \cap Z = \emptyset$. Further, structural connectivity of any member of Y with Z should be much lower than with Y and similarly for Z, i.e. if C is the connectivity function mentioned earlier then for every $z \in Z$ we have $C(z,Y) << C(z,Z)$ and for every $y \in Y$ we have $C(y,Z) << C(y,Y)$.

Algorithm:

1. If X is empty then return Y and Z to be empty and terminate. If X has only one element then return Y to be X and Z to be empty and terminate. Otherwise go to the next step.
2. Compute pairwise structural connectivity for all pairs of tokens in X. Choose a pair that has the minimum connectivity, let this be <x, y>.
3. Set $Y = \{x\}$; $Z = \{y\}$; and $X = X - \{x, y\}$.
4. If X is empty then terminate.
5. For every token x in X compute $f(x) = C(x, Y) - C(x, Z)$; where C is the structural connectivity function.
6. If there is some x in X such that f(x) is maximum and is positive or zero then set $Y = Y \cup \{x\}$ and $X = X - \{x\}$.
7. If there is some x in X such that f(x) is minimum and is negative then set $Z = Z \cup \{x\}$ and $X = X - \{x\}$. Then go to step 4.

Thus, Algorithm 5.4.1 results in a separation of the tokens corresponding to two domains based on structural connectivity. Of course, if the higher-level system can provide this separation based on some other context-dependent information then this

algorithm can be dispensed with.

Next we show how, given this separation of tokens, each set of tokens can be used to identify their respective domains. This algorithm is also based on structural connectivity and can be used if the higher-level system cannot provide this information.

Algorithm 5.4.2: Isolating the Domain Corresponding to a Set of Tokens[1]

Input: A set of tokens X; an integer n, which limits the number of iterations.

Output: A domain <I, E, S, d, f> such that $X \subseteq I$.

Algorithm:

1. Set I to be X and S to be all the structural constraints in which any member of X appears.
2. If n = 0 then go to step 6.
3. Let Y be the set of tokens that appear in some structural constraint in S but are not included in I. If Y is empty then go to step 6, otherwise go to the next step.
4. Set $I = I \cup Y$. Collect all the structural constraints that are not in S and such that some token in Y occurs in them. Add them to S.
5. Set n = n - 1 and go to step 2.
6. Set E to be the set of tokens that appear in some structural constraint in S but are not included in I.

---

1. This algorithm is based on an idea suggested to me by Barbara Partee.

7. Set d to be the set of those structural constraints of S that are derivations.
8. Set f to be the restriction of the global type function to I ∪ E.

In case the set of tokens given to this algorithm is from a closed and self-contained domain then this algorithm will return the whole domain, provided n is large enough. Otherwise, depending on the value of n, this will return an approximate domain definition. The larger the n, the larger will be the size of the approximation.

Thus, we can identify domains using the above algorithms in case the higher-level system is unable to do so. Still the higher-level system will have to identify which is the source domain and which is the target domain since this distinction is dependent on the context and cannot be inferred from the information contained in the domains alone.

Once the source and the target domains are identified the next problem is to determine an appropriate AT-MAP that can provide an interpretation of the metaphor or analogy. We propose to break this task in two stages. In the first stage one or more strongly coherent AT-MAPs are established, which we shall call *basis* AT-MAPs. In the second stage a basis is extended by either including more tokens, applying Augmentation or Positing Structure, or a combination of these. The intuition behind generating AT-MAPs in two steps is that often the implications derived from a metaphorical utterance can be divided in two classes, as we discussed in Chapters 2 and 4. The first class of implications corresponds to what the speaker intended by the metaphor and the second class corresponds to the additional implications that the hearer derives from the metaphor. This second class of

implications may or may not have been intended by the speaker but is consistent with the first class of implications that the speaker did intend. Thus, if the utterance happens to be "the sky is crying", the first class of implications will contain "it is raining" and the second class may contain implications to the effect that "the sky is sad". Our two stages of generating AT-MAPs are supposed to correspond to these two sets of implications. Thus the first stage generates an AT-MAP that provides a basic interpretation of the metaphor, one that the speaker most likely intended, and the second stage provides extra richness of meaning that hearer associates with the metaphor. These two proposed stages of metaphor comprehension correspond to the two stages of of ZORBA-I [Kling 1971].

The algorithm for the first stage uses a notion of *structural template* to make guesses about which mappings are more likely to be AT-MAPs. The structural template corresponding to a structural constraint is derived by replacing all occurrences of every internal token in the structural constraint by a new variable of appropriate type. By *new* variable we mean that the variable should not already be present in the structural constraint. An example will make this clearer. Suppose the following is a structural constraint of the domain WEATHER,

- raining(sky) → fall-down(raindrops, sky) (1)

If all the tokens occurring in the above structural constraint, except *fall-down*, are internal to WEATHER, and X and Y are distinct variables of type e and P is a variable of type <e,t> then the structural template corresponding to the above structural constraint is given as follows:

- P(X) → fall-down(Y, X) (2)

Notice that all the occurrences of the same token are replaced by the same variable. With every structural template we associate a list of pairs of the kind {<v, t>} where v is a variable and t is the token that was replaced by v. Thus the list associated with the above structural template (2) will be {<P, raining>; <X, sky>; <Y, raindrops>}.

Now given a structural template St, its associated list of variable-token pairs VT, and a structural constraint S we say that St *matches* S if it is possible to associate with every variable in VT a token of S in such a way that on replacing all the occurrences of every variable, that is in VT, in St by the corresponding token of S both S and St become identical. An example will make this clear. Consider the following structural constraint:

- crying(eyes) → fall-down(teardrops, eyes) (3)

Now suppose we replace all occurrences of P by *crying*, X by *eyes*, and Y by *teardrops* in (2) then it becomes identical with (3). Therefore, we will say that the structural template (2) matches the structural constraint (3). Readers familiar with unification [Robinson 1965] will recognize this as an instance of unification. Thus, in other words, a structural template matches a structural constraint if they can be unified.

The algorithm that we are going to present next uses the notions of structural template and matching in forming a plausible mapping and then calls Algorithm 5.3.3 to collect all the structural constraints that can be transferred by that mapping. All the AT-MAPs that are generated in this fashion are passed to the higher-level system so that it can decide which ones to throw away and which ones to keep and pass to the next stage.

Apart from the source and the target domains, Algorithm 5.4.3 also takes as input a set of source domain tokens. This should be provided by the higher-level system in order to focus the direction of search by Algorithm 5.4.3. In the simplest case, the higher-level system can pass the tokens of the source domain that occur in the metaphorical utterance. Thus, if the utterance being interpreted is "the ship plowed through the sea" then this set will contain the token *plow*, and if the utterance is "the sky is crying" then it will contain the token *cry*. In a more complex case, the higher-level system may include some other tokens that it thinks may be relevant in interpreting the metaphor. For instance, on encountering the utterance "node A is a sister of node B", the higher-level system may choose some derivation of *sister* and include all the tokens that occur in that derivation in the set of tokens passed to Algorithm 5.4.3. Of course, in the absence of any information, the whole set of internal tokens of the source domain can be passed in which case the search for appropriate basis AT-MAPs will take a much longer time.

We are now ready to present the algorithm to generate basis AT-MAPs.

Algorithm 5.4.3: Generating Basis AT-MAPs

Input: Source domain D1 = <I1,E1,S1,d1,f1>; target domain D2 = <I2,E2,S2,d2,f2>; a set I of internal tokens of D1 ($I \subseteq I1$).

Output: A set B_t of AT-MAPs from D1 to D2.

Algorithm:

1. Set $B_t = \emptyset$.
2. Set S = {X | X ∈ S1 and there is some token i ∈ I such that i occurs in X}. In other words, S is the set of those structural constraints of the source domain D1 which contain at least one token from I.
3. If S is empty then terminate. Otherwise go to the next step.
4. Set X to be the most salient structural constraint in S. Set Xt to be the structural template corresponding to X, S = S - {X}, and S2′ = S2.
5. If S2′ is empty then go to step 3. Otherwise go to the next step.
6. Set Y to be the most salient structural constraint of S2′. If Xt matches Y then go to the next step. Otherwise go to step 10.
7. From Xt and Y compute a mapping F as follows. In the variable-token pairs associated with Xt, for every pair <v, t>, where v is a variable and t is a token of D1, map t to that token of Y that was assigned to v in the process of matching. F will be an admissible mapping from D1 to D2.
8. Call Algorithm 5.3.3 with F, D1, and D2 as input. Let S′ be the set of structural constraints returned by Algorithm 5.3.3. We recall that F is coherent with respect to S′.
9. If S′ is not empty then set $B_t = B_t \cup \{<F, S'>\}$.
10. Set S2′ = S2′ - {Y} and go to step 5.

Having discussed one possible way to generate basis AT-MAPs we now come to the

problem of extending AT-MAPs. We will first present an algorithm that searches for an appropriate AT-MAP by using a depth-first control strategy with the possibility of backtracking. This algorithm accepts as input a k-element AT-MAP, with k greater than 0, and returns a (k+1)-element AT-MAP that is a one-element extension of the input AT-MAP. The decision of whether to backtrack or to continue the search will have to be made by the higher-level system since it will be the one to be using the AT-MAPs that are generated by our algorithm. Also, the higher-level system will have to supply the salience measure for the tokens of the source and target domains. This measure will be used, in conjunction with the static salience of the tokens, to determine their dynamic salience as we explained earlier. In this algorithm we will assume the dynamic salience of the tokens to be given.

The algorithm will use two global stacks, called S-STACK and SL-STACK, that are initialized to be empty. Unless they are empty, the top of the S-STACK always contains the most recently added element to the AT-MAP and the top of the SL-STACK always contains a list of the singleton mappings that are yet to be considered. Every element of any list in SL-STACK has a measure of a-salience attached to it. This a-salience is different from static and dynamic salience introduced earlier and we will specify in the algorithm how it is computed. All this information is kept in order to facilitate the process of backtracking.

We first describe the algorithm that produces a one-element extension of the input AT-MAP and then will present another related algorithm that will perform the backtracking. We assume that S-STACK and SL-STACK were initialized to be empty by the higher-level system before attempting to analyze a metaphor.

Algorithm 5.4.4: Depth-First Search for AT-MAPs

Input: An AT-MAP T1 = <F1, S> from the source domain D1 = <I1, E1, S1, d1, f1> to the target domain D2 = <I2, E2, S2, d2, f2> such that F1 has at least one element.

Output: An AT-MAP T2 = <F2, S'> from D1 to D2 such that T2 is a one element extension of T1. In other words, F2 is the same as F1 except that it maps one more token from D1 to D2 and S is a subset of S'.

Algorithm:

1. Set Df1 = {x | there is some y such that (x → y) ∈ F1} and Rf1 = {x | there is some y such that (y → x) ∈ F1}. Df1 will be a subset of I1 and Rf1 will be a subset of I2.

2. Set Sr1 to be the set of tokens of D1 that are structurally connected with Df1 and are not included in Df1. Set Sr2 to be the set of tokens of D2 that are structurally connected with Rf1 and are not included in Rf1.

3. Set St1 = {(e1 → e2) | e1 ∈ Sr1; e2 ∈ Sr2; and f1(e1) = f2(e2)}. St1 is a set of singleton admissible mapping from D1 to D2.

4. To every singleton in St1 assign a measure of a-salience as follows. If (e1 → e2) is in St1, then its a-salience is the sum of the salience of e1, the salience of e2, the structural connectivity of e1 with Df1, and the structural connectivity of e2 with Rf1.

5. Let m1 = (e1 → e2) be the singleton which has the highest a-salience in St1. Set St1 = St1 - {m1}.

6. Push St1 on the SL-STACK and push m1 on the S-STACK.

7. Set F2 = F1 ∪ {m1} and S' = S.

8. For every structural constraint X in S1, if X is transformable by F2 and X is not included in S then use Algorithm 5.3.1 or 5.3.2 to test if F2 is coherent with respect to {X}. If it is then add it to S'. Salience is used to determine the order in which structural constraints of S1 are considered for coherency with respect to F2.

The last step in this algorithm uses the pruning principle in a slightly different form to reduce unnecessary calls to the coherency checker. In this case, since we know that F1 is coherent with respect to S, then it follows that any extension of F1, in particular F2, will also be coherent with respect to S and therefore if some structural constraint is included in S then it need not be checked for coherency.

It is easy to see that the output AT-MAP of this algorithm will always have the property that it cannot be extended by adding structural constraints of the source domain alone. In other words, if the output AT-MAP is <F, S> then S will include all those structural constraints of the source domain that F is coherent with. If we assume that the input AT-MAP also has this property, which is a reasonable assumption if it was generated by the same algorithm in some previous iteration, then the Step 8 in the above algorithm can be simplified as follows. We recall that m1 = (e1 → e2) is the most recently added element to the AT-MAP.

Step 8a (Algorithm 5.4.4): For every structural constraint X in S1, if e1 occurs in X and X is transformable by F2 then use Algorithm 5.3.1 or 5.3.2 to check if F2 is coherent with respect to {X}. If it is then add it to S'. Use salience of structural constraints to determine the order in which eligibility for coherency is considered.

The reason we only consider those transformable structural constraints in which e1 appears can be explained in the following fashion. Any structural constraint that is transformable by F2 and in which e1 does not appear will also be transformable by

F1. In this case there are two possibilities, either the structural constraint is coherent with F2 or not. If it is coherent with F2 then it will also be coherent with F1 and therefore will be included in S by assumption. Thus, in this case we need not check it for coherency since it will already be present in S'. If it is not coherent with F2 then we need not consider it since it is already left out.

We now present a related algorithm that performs the backtracking function.

Algorithm 5.4.5: Backtracking in the Search for AT-MAPs

Input: An AT-MAP T1 = <F1, S> from the source domain D1 = <I1, E1, S1, d1, f1> to the target domain D2 = <I2, E2, S2, d2, f2>.

Output: An AT-MAP T2 = <F2, S'> from D1 to D2.

Algorithm:

1. Set F2 = F1.
2. If S-STACK is empty then signal error. Otherwise set m1 to be the top of S-STACK and pop S-STACK.
3. Set F2 = F2 - {m1}.
4. If SL-STACK is empty then signal error. Otherwise set Ml1 to be the top of SL-STACK and pop SL-STACK.
5. If Ml1 is empty then go to step 2. Otherwise go to the next step.
6. Set m1 to be the singleton in Ml1 that has the highest a-salience. Set Ml1 = Ml1 - {m1}.

7. Push Ml1 onto SL-STACK. Push m1 onto S-STACK.
8. Set F2 = F2 ∪ {m1}.
9. Set S′ = S.
10. Consider every structural constraint in S′ in the order of its salience. If it is not transformable by F2 then remove it from S′.
11. Consider every structural constraint in S1 in the order of its salience. If it is transformable by F2 and is not included in S′ then use Algorithm 5.3.1 or 5.3.2 to determine if F2 is coherent with respect to it. If the coherency condition is satisfied then add it in S′.

Thus any higher-level system can extend a basis AT-MAP by using a combination of Algorithms 5.4.4 and 5.4.5. Every time any of these algorithms is invoked, a new AT-MAP is passed on to the higher-level system, which can then determine whether to search further, backtrack or stop.

At this point we would like to point out that though Algorithm 5.4.3 is suggested to produce basis AT-MAPs and Algorithms 5.4.4 and 5.4.5 are suggested to produce extensions of a basis AT-MAP, there are several other possibilities. For instance, Algorithms 5.4.4 and 5.4.5 can be used directly on singleton mappings to search for AT-MAPs. Similarly, an algorithm similar to 5.4.3, that is based on structural template and matching, can be used to produce extensions of some AT-MAP.

Next we present an algorithm that performs breadth-first search for AT-MAPs and generates a set of *all* k-element AT-MAPs from the source domain to the target domain. It should only be used for small sized domains since with domains that are moderately large there will be a combinatorial explosion in the complexity of this algorithm. Also, due to its higher complexity, this algorithm is more suitable

for off-line computing of AT-MAPs. In other words, AT-MAPs from the source domain to the target domain can be computed in advance to be used later by a system such as Carbonell's [1982], or some Natural Language Generation System. It can also be used to find all possible analogies between two domains and then choose the one which is most appropriate based on some criteria.

We now present this algorithm which will be followed by explanations and comments.

Algorithm 5.4.6: Breadth-First Search for AT-MAPs

Input: Source domain D1 = <I1,E1,S1,d1,f1>; target domain D2 = <I2,E2,S2,d2,f2>; k.

Output: A set Tk of all k-element AT-MAPs from D1 to D2. Every AT-MAP in Tk is maximally coherent.

Algorithm:

1. Set n = 0 and T-new = $\emptyset$.
2. If n = k then set Tk = T-new and terminate. Otherwise set n = n + 1 and go to the next step.
3. If n = 1 then go to the next step. Otherwise go to step 5.
4. Set T-new = {{(e1 → e2)} | e1 ∈ I1; e2 ∈ I2; and f1(e1) = f2(e2)}. In other words, T-new is the set of all 1-element admissible mappings from D1 to D2. Go to step 7.

5. Set T-new = $\{M_{n-1} \cup M_1 \mid M_{n-1} \in$ T-old; $M_1 \in$ T1; and the intersection of $dom(M_{n-1})$ and $dom(M_1)$ is empty$\}$. Here by dom(M), where M is a mapping from D1 to D2, we denote the set of tokens of D1 that are mapped by M, or formally dom(M) = {e1 | there is some e2 such that (e1 → e2) ∈ M}.

6. For every mapping M_n of T-new, generate all (n-1) element submappings of M_n. There will be n such submappings. Check to see if all these submappings are in T-old. If any of the submappings do not occur in T-old then remove the corresponding M_n from T-new.

7. For every mapping M_n in T-new, set S = {X | X ∈ S1; X is transformable by M_n; and every token in $dom(M_n)$ occurs at least once in X}. Use Algorithm 5.3.1 or 5.3.2 to determine coherency of M_n with respect to S. If it is found coherent then remove it from T-new.

8. If T-new = ∅ then set Tk = ∅ and terminate. Otherwise go to the next step.

9. If n = 1 then set T1 = T-new.

10. Set T-old = T-new. Go to step 2.

In this algorithm we use the pruning principle, and the fact that the AT-MAPs are being generated in a breadth-first manner, to reduce the calls to the inconsistency-checker. From the pruning principle, described earlier in this section, we know that if a k-element mapping is a maximal AT-MAP then all its submappings will also be maximal AT-MAPs. In particular, all its (k-1)-element mappings will be maximal AT-MAPs. Since before generating all k-element AT-MAPs we would have generated all (k-1)-element AT-MAPs in the previous iteration, we could check if all (k-1)-element submappings of a k-element mapping are (k-1)-element AT-MAPs by merely testing to see if they all occur in the set of all (k-1)-element AT-MAPs. If this condition is not satisfied then we can throw that mapping away, without using a more expensive coherency check, since it is bound to be incoherent. This is

precisely what we do in step 6 in the above algorithm.

The pruning principle is exploited in another way in step 4. In forming plausible k-element AT-MAPs, we start with the set of all (k-1)-element AT-MAPs and combine them with the 1-element AT-MAPs. It is easy to verify that this process will include all k-element AT-MAPs, and possibly some other mappings that are not AT-MAPs.

Finally, step 7 in the above algorithm needs some words of explanation. We only test coherency with respect to those structural constraints that include all the tokens mapped at least once. The reason for this is as follows. Suppose M is some n-element mapping from D1 to D2 such that all its (n-1)-element submappings are (n-1)-element AT-MAPs and X is some structural constraint of D1 such that some token in dom(M), say Y, does not occur in X and X is transformable by M. Further, suppose that M′ is the (n-1)-element mapping obtained from M by removing Y from it. Then since Y does not occur in X, X will also be transformable by M′. Thus if M is incoherent with respect to X then so will be M′. Since we already know that all submappings of M are maximal AT-MAPs no such X can exist. This principle is very useful in reducing calls to the coherency checker as the mappings get bigger.

The algorithms for carrying out Augmentation and Positing Structure can be derived directly from the definitions given in Section 3.4. We will only present the algorithm for Augmentation here.

Algorithm 5.4.7: Augmenting an AT-MAP and Target Domain

Input: An AT-MAP $T = \langle F, S \rangle$ from the source domain $D1 = \langle I1, E1, S1, d1, f1 \rangle$ to the target domain $D2 = \langle I2, E2, S2, d2, f2 \rangle$; a derived token $i_d \in I1$ such that it is not mapped by F; and a derivation $D \in d1$ of i_d such that the derivation part of D is transformable by F.

Output: A new target domain $D2' = \langle I2', E2, S2', d2', f2' \rangle$ such that $I2' \supseteq I2$, $S2' \supseteq S2$, $d2' \supseteq d2$, and $f2' \supseteq f2$; and an AT-MAP $T' = \langle F', S' \rangle$ that is a one-element extension of T.

Algorithm:

1. Generate a new token X that does not occur in I2 or E2.
2. Set $I2' = I2 \cup \{X\}$.
3. Set $F' = F \cup \{(i_d \rightarrow X)\}$.
4. Set f2' to be exactly like f2 except that $f2(X) = f1(i_d)$.
5. Set $S2' = S2 \cup \{F(D)\}$ and $d2' = d2 \cup \{F(D)\}$.
6. Set $S' = S \cup \{D\}$.

The algorithm for Positing Structure can also be derived in a similar fashion from the definition given in Section 3.4. The important thing with both Augmentation and Positing Structure is not how to perform the operation which, as can be seen from the above algorithm, is rather trivial, but how to determine what structure needs to be posited or which derivation to use in augmenting the target domain. All these decisions will have to be made by the higher-level system since they are based on

the context and the purpose for which the metaphor is being employed.

Finally, at the end of this section we would like to emphasize again that the algorithms presented in this section should be taken only as suggestive. Comprehending and generating metaphors are very creative processes so that to come up with a general algorithm to generate AT-MAPs efficiently will require solutions of many problems in cognition of which we do not yet have a clear understanding. Still, as we mentioned earlier, in specific situations, when there is enough information to design some heuristics that can guide the search for AT-MAPs, it is possible to use algorithms, such as the ones presented in this section, that can generate AT-MAPs efficiently. Design of any such module that produces AT-MAPs will have to take place hand in hand with the design of the system in which it is to be embedded since they will be interacting very heavily with each other.

5.5 SOME EXAMPLES

In this section we will present some simple examples to show how the algorithms described in the previous section can be configured to model the process of interaction involved in comprehending a metaphorical utterance. We will be making several simplifying assumptions—as will become apparent later in this section. The purpose of presenting these example is not to show the way to model the process of metaphor comprehension—we have already remarked several times on the enormousness of this task—but merely to illustrate some of the mechanisms that we believe play a crucial role in understanding metaphors and that were used in the algorithms presented earlier in this chapter.

The first example we will consider is "the sky is crying". In analyzing "the sky is crying" this sentence is first analyzed by some syntactic analyzer to produce an SL sentence.[1] We assume that this analysis results in the SL sentence *cry(sky)*. Further we assume that the system can identify the respective domains of the tokens *sky* and *cry* and recognize the SL sentence *sky(cry)* as an instance of metaphor since the token *cry*, which belongs to the domain EMOTIONAL-STATE, is attributed to *sky* which belongs to the domain WEATHER. We also assume that it is decided, based upon the context, that the source domain is EMOTIONAL-STATE and the target domain is WEATHER.

In order to create an AT-MAP that can provide a possible interpretation for *cry(sky)* the first step is to access all the tokens and the structural constraints of both the source and the target domains. We present below a list of the structural constraints of EMOTIONAL-STATE and WEATHER. In presenting these domains we are not attempting a complete description and therefore the lists of constraints is only partial. The tokens that are external to the respective domains are in italics and all the remaining tokens are internal. The types of the tokens are self-explanatory from the structural constraints. In writing the constraints we introduce the following abbreviation for brevity. If X and Y are two formulas then by X Θ Y we mean the expression (X v Y) & ~(X & Y). Thus, X Θ Y is true iff either X or Y, but not both, are true. The reader familiar with logic will recognize this as the exclusive-or operation.

---

1. Cf. Bronnenberg *et al.* [1980]; Friedman & Warren [1978]; and Indurkhya [1981] for a description of programs that translate English sentences into a formal language.

DOMAIN 5: EMOTIONAL-STATE

1. cry(eyes) → *fall-down*(tear-drops, eyes)
2. cry(eyes) ↔ cry(person) & *part-of*(person, eyes)
3. cry(person) → very-happy(person) ⊖ very-sad(person)
4. laugh(person) → very-happy(person)
5. very-happy(person) → ~very-sad(person)
6. [*blue*(eyes) ⊖ *dark*(eyes) ⊖ *brown*(eyes)] & ~[*blue*(eyes) & *dark*(eyes) & *brown*(eyes)]
7. *liquid*(tear-drops)
8. *heavy*(tear-drops)

Comments: We notice here that in this representation the predicate cry can be attributed to the person as well as her eyes. This redundancy is introduced because one can say "Mary is crying" as well as "her eyes are crying". We already discussed in Chapter 3 that such redundancies of representation may be essential in comprehending metaphors.

DOMAIN 6: WEATHER

1. *blue*(sky) ↔ sunny(sky)
2. *blue*(sky) ↔ day & *clear*(sky)
3. sunny(sky) ↔ day & *clear*(sky)
4. *gray*(sky) ↔ day & *cloudy*(sky)

5. starry(sky) → night & *clear*(sky)
6. *contains*(sky, full-moon) → night & *clear*(sky)
7. *contains*(sky, full-moon) → ~starry(sky)
8. raining(sky) → cloudy(sky)
9. raining(sky) → [*gray*(sky) & day] v [*dark*(sky) & night]
10. raining(sky) ↔ *fall-down*(rain-drops, sky)
11. snowing(sky) → cloudy(sky)
12. snowing(sky) → [*gray*(sky) & day] v [*dark*(sky) & night]
13. snowing(sky) ↔ *fall-down*(snow-flakes, sky)
14. *heavy*(rain-drops)
15. *liquid*(rain-drops)
16. *light*(snow-flakes)
17. *solid*(snow-flakes)

Comments: Here again we emphasize the redundancies in our representation. Further, since these domains are not required to be closed we are representing the significant logical consequences explicitly. Thus, the constraint 3 above is explicitly stated even though it can be directly inferred from constraints 1 and 2. This leads to the consequence that many simple inferences that can be derived from the structural constraints of a domain do not play any role in determining approximate coherency if we use the definitions presented earlier in Section 5.2. This problem does not arise if the alternate definition of approximate coherency—to be discussed in the next section—is used.

In trying to analyze "the sky is crying" from the domain representations given above, we first look at those constraints of EMOTIONAL-STATE in which *cry* appears. In the list of constraints presented earlier there are only three such constraints. Suppose that the system decides that the constraint 2 is more salient. The corresponding structural template is given below:

- $P(x) \dashrightarrow P(y)$ & *part-of*(y, x) (4)

where P, x and y are variables of the appropriate type. Now an attempt is made to match this template against the structural constraints of WEATHER. Since none of the constraints match against this template it is discarded. Suppose now the system picks up constraint 1 of EMOTIONAL-STATE. The corresponding structural template is:

- $P(x) \rightarrow$ *fall-down*(y, x) (5)

Again an attempt is made to match this template against the constraints of WEATHER. This time there are two matches since both the constraints 10 and 13 of WEATHER match against this template.

Suppose that it is decided to try constraint 13 first. This suggests the following mapping:

- EW1: {(cry → snowing); (eyes → sky); (tear-drops → snow-flakes)} (6)

An attempt is now made to verify the coherency of EW1 with respect to those constraints of EMOTIONAL-STATE that are transformable with respect to it. There

are three such constraints 1, 7 and 8. The constraint 1 is found to be transferable since it is implied trivially by constraint 13 of WEATHER. Constraints 7 and 8 are not transferable since they are contradicted by constraints 17 and 16 of WEATHER respectively.

Thus an AT-MAP is created by pairing EW1 with constraint 1 of EMOTIONAL-STATE. Let us suppose that for contextual reasons this is found to be an unacceptable interpretation by the system and therefore it backtracks.

Now another mapping is created by matching the structural template (5) above against the constraint 10 of WEATHER, which is:

- EW2: {(cry → raining); (eyes → sky); (tear-drops → rain-drops)} (7)

Again an attempt is made to verify its coherency by collecting all those constraints of EMOTIONAL-STATE that are transformable by EW2. The same three constraints 1, 7, and 8 are found to be transformable and this time they all turn out to transferable since they are trivially implied by constraints 10, 15, and 14 of WEATHER respectively. Another AT-MAP is formed at this point by coupling EW2 with these three structural constraints.

This provides the basis AT-MAP under which the SL sentence *cry(sky)* is interpreted to be *raining(sky)*. The metaphorical inferences that can be derived under this interpretation will include sentences to the effect that "the sky is cloudy" and "it is either day and the sky is gray or it is night and the sky is dark".

One may stop at this point or an attempt can be made to extend this AT-MAP by applying Positing Structure to it. Suppose that, based on constraint 2 of EMOTIONAL-STATE, it is decided to posit the existence of a token in WEATHER that corresponds to *person* of EMOTIONAL-STATE. Let us call this new token *sky-person*. Now EW2 can be extended to EW3 as follows:

- EW3: {(cry → raining); (eyes → sky);
  (tear-drops → rain-drops); (person → sky-person)} (8)

This will allow constraint 2 of EMOTIONAL-STATE to be transferred to the target domain WEATHER also. At this point one may posit new structural constraints in WEATHER, for instance the one suggested by constraint 2 of EMOTIONAL-STATE to the effect that "the sky is part of the sky-person". Also, other human attributes may be transferred to WEATHER like sadness or happiness etc.

Let us now consider "the ship plowed through the sea". The SL sentence corresponding to this utterance is *plow-a(sea, ship)*. We assume here that the higher level system can recognize that the tokens *sea* and *ship* are from the domain SAILING and that the token *plow-a* is from the domain PLOWING [Cf. Chapter 4]. Further, we assume that the context determines that PLOWING is the source domain and SAILING is the target domain.

In order to generate possible interpretations for *plow-a(sea, ship)* we first search through the structural constraints of PLOWING to find those constraints in which *plow-a* appears. All such constraints are given below:

1. *agent-of*(plow-a) = farmer
2. *instrument-of*(plow-a) = plow-o
3. plow-a(field, plow-o)
4. *moves-through*(field, plow-o)
5. *result-of*(plow-a, loose-and-broken(earth))

In forming structural templates, these constraints will be considered in the order of their salience. Suppose the system first considers the first of the above constraints. Its corresponding structural template is:

- agent-of(Q) = x (9)

where Q is a variable of the type <e,<e,t>> and x is a variable of type e.

In matching (9) against the structural constraints of SAILING we find that it matches with the following constraint:

- agent-of(sail) = crew (10)

This matching suggests the following mapping:

- PS3: {(plow-a → sail); (farmer → crew)} (11)

An attempt is now made to collect all those constraints of PLOWING that are transformable with respect to PS3 and then verify their coherency. Only the following structural constraint can be transferred:

- *agent-of*(plow-a) = farmer

The corresponding T-MAP can be passed to the higher level system or an attempt can be made to extend it. In extending PS3 there are two possible strategies that can be adapted. The first possibility is to use the structural template based on another constraint of PLOWING and then try to merge the resulting mapping with PS3 given above. For instance, suppose the second of the above structural constraints, namely "instrument-of(plow-a) = plow-o", is chosen next. Its structural template is:

- instrument-of(R) = x (12)

This template matches against the following structural constraint of SAILING:

- instrument-of(sail) = ship (13)

This mapping suggests the following mapping:

- PS4: {(plow-a → sail); (plow-o → ship)} (14)

Now PS4 can be merged with PS3 to produce PS5:

- PS5: {(plow-a → sail); (plow-o → ship); (farmer → crew)} (15)

Now we can collect those constraints of PLOWING that are transformable with respect to PS5 and verify their coherency. The important thing to note here is that we need to check only those constraints in which the newly added token *plow-o* appears. This is due to the fact that PS5 is an extension of PS3, and since we have already checked the coherency of all the constraints that are transformable with

respect to PS3 there is no point in checking them again. The following constraints of PLOWING are tested for coherency with respect to PS5:

1. controls(farmer, plow-o)
2. instrument-of(plow-a) = plow-o

Similarly by using the remaining structural constraints, and repeating this procedure, we will eventually arrive at the mapping PS2 of Chapter 4, repeated below for convenience.

PS2: {(farmer → crew); (plow-o → ship);
(plow-a → sail); (field → sea);
(earth → water); (plow-edge → ship-bow)} (16)

The other alternative is to update the list of structural constraints every time a new T-MAP is formed. For instance, in the example above, after PS3 is formed we now collect all those structural constraints of PLOWING in which *plow-a* or *farmer* appears. This will include the five constraints presented earlier and the following ones:

1. controls(plow-o, farmer)
2. human(farmer)

The next structural template is derived from the most salient structural constraint in this larger set.

Yet another alternative is to use the principle of structural connectivity to form plausible extensions of PS3. Thus we collect all those tokens from PLOWING that are structurally connected to *plow-a* or *farmer*. In the representation of Chapter 4 there are only two such tokens, namely *plow-o* and *field*. Similarly from SAILING we collect all those tokens that are structurally connected to *sail* and *crew*. They are *captain*, *ship*, and *sea*. We now form admissible singleton mappings from these tokens. Since all of them have the type e, there will be six such mappings. These mappings are now considered in the order of their salience in forming extensions of PS3. This approach will also eventually lead to the mapping PS2 of Chapter 4 though many more mappings will be examined in the process than the first alternative which uses structural templates all along. In general, we conjecture that for computing AT-MAPs among domains that consist of clusters—where a cluster has several richly interconnected tokens—with few structural constraints relating inter-cluster tokens, this approach will focus the search within the same cluster. On the other hand, with domains that are uniformly interconnected and do not exhibit clustering this method will result in an exhaustive search.

The last example we would like to consider here illustrates the role of derivations in affecting the search for an appropriate AT-MAP. Consider the utterance "node A is sister of node B" which corresponds to the SL sentence *sister(A, B) & node(A) & node(B)*. In order to arrive at an appropriate interpretation of this metaphor one has to consider the following derivation of *sister*:

- sister(x, y) $\leftrightarrow$ female(x) & (Ez)[mother(z, x) & mother(z, y)] (17)

From this derivation, and knowing the principle of Augmentation, we can deduce that any AT-MAP from FAMILY to DAG that includes tokens *female* and *mother* can be augmented to include the token *sister*. Thus, instead of using constraints in which *sister* appears to form a structural template we can also start by using tokens *mother* and *female*. As a matter of fact a correct interpretation of this metaphor cannot be arrived at without using this derivation.

The examples we have presented here illustrate some of the mechanisms that we believe are useful in the process of comprehending metaphors. The harder question that we have not addressed is how to determine when to use which mechanism. Thus we did not provide answers to questions like "how to decide when to use a derivation and when not" or "how to determine whether to use structural template or structural connectivity in extending an AT-MAP". We hope that the future research will be able to show some light upon these matters.

5.6 AN ALTERNATIVE FORMULATION OF APPROXIMATE COHERENCY

Earlier we mentioned two major problems associated with the notion of τ-coherency as defined in Section 5.2. In this section we will suggest an alternative formulation, that is referred to as ν-coherency, that overcomes these deficiencies.

The first problem that we mentioned concerns the notion of transformability in the light of non-closure. In the example presented in the previous section, if we conjoin the constraints 1 and 3 of EMOTIONAL-STATE into one single constraint then the resulting constraint is not transformable with respect to the mapping EW2. This leads to the consequence that the first constraint of this domain ceases to

affect the coherency of EW2 even though it is still part of EMOTIONAL-STATE.

One way to overcome this problem is to unfold all conjunctions into separate constraints. In other words all the structural constraints are kept as *simple* as possible. This still suffers from the deficiency that even trivial consequences of the structural constraints are ignored. Thus, if the structural constraints of a domain included two sentences of the form X v Y and ~X and if F is any mapping such that Y is transformable with respect to it, then Y does not affect coherency of F even though it trivially follows from X v Y and ~X, unless it is also explicitly represented. Another shortcoming of this solution is that it increases the second problem that we are going to discuss now.

This second problem with the notion of τ-coherency, as it was defined in Section 5.2, is that since the structural constraints of the target domain are tested in a one-by-one fashion in determining τ-coherency, even simple consequences of its structural constraints are totally ignored. Thus, in the example we just presented, if the domain happens to be the target domain and Y trivially contradicts some transformed constraint of the source domain then it will not be detected unless Y is explicitly stated.

This problem suggests that it is best to conjoin all the structural constraints together. This immediately increases the first problem, as we just pointed out, and vice versa. A compromise can be made if we break the structural constraints of a domain as much as possible to alleviate the first problem and define τ-coherency as follows. A mapping F is said to be τ-coherent with respect to a set S of transformable constraints of the source domain if for every constraint X in S, F(X)

is τ-coherent with the conjunction of all the structural constraints of the target domain. Although this solves both the problems technically, let us look deeper and try to find the cause of these problems.

Though the previous discussion suggests that these problems reflect competing principle in the sense that they drive the representation of a domain toward opposite extremes, at a deeper level we find that they both stem from the same cause which is the non-closure of the structural constraints. We required non-closure since the knowledge that a person may have of an object is never closed. On the other hand people also do not carry all their knowledge of an object explicitly in their head. Therefore, instead of requiring that all the constraints of a domain be explicitly stated we can define a complexity bound closure in the following fashion. All the following definitions are based on the resolution-refutation [Robinson 1965] principle of theorem proving but they can easily be adapted for the theorem-provers based on the natural-deduction principle [Smullyan 1965; 1968]. In the following discussion we will be assuming some familiarity with the principle of resolution-refutation.

Given two clauses X and Y we denote the set of resolvents of X and Y by Res (X,Y). Now we recursively define a generator function G, that takes as input a set of clauses S and a natural number N and results in a set of clauses, as follows:

1. $G(S, 0) = S$
2. $G(S, N) = G(S, N\text{-}1) \cup \{X \mid X \in Res(Y,Z) \text{ where } Y, Z \in G(S, N\text{-}1)\}$

We say that a clause X occupies a level i with respect to a set of clauses S iff i is the smallest integer such that X is in G(S, i).

Next we define the notion of ν-inconsistency. Given ν to be a natural number, a set of clauses S is said to be *ν-inconsistent* iff the empty clause occupies a level less than or equal to ν with respect to S. S is said to be *ν-consistent* iff it is not ν-inconsistent.

If X is any clause then by Neg(X) we denote the set of clauses obtained by negating X and reconverting the resulting formula into its clausal form. Now we are ready to define ν-provability. A clause X is said to be *ν-provable* from a set of clauses S iff S $\cup$ Neg(X) is ν-inconsistent.

We can now present the definitions related to ν-coherency. We require that all the domains satisfy the axiom of finiteness mentioned in Section 5.2 and that the structural constraints of any domain are represented as a set of clauses. Now given an admissible mapping F from the source domain D1 = <I1,E1,S1,d1,f1> to the target domain D2 = <I2,E2,S2,d2,f2>, a set S (a subset of S1) that is transformable by F, we will say that F is *weakly ν-coherent* with respect to S iff F(S) $\cup$ S2 is ν-consistent. Further, we will say that F is *strongly ν-coherent* with respect to S iff for every clause X in G(S, ν), F(X) is ν-provable from S2. If F is weakly ν-coherent or strongly ν-coherent with respect to S then we say that F is ν-coherent with respect to S. Further, if S includes all those clauses from G(S1, ν) that are transformable with respect to F then we will say that F is maximally ν-coherent.

This alternative formulation of approximate coherency does not suffer from any of the deficiencies that we mentioned at the beginning of this section. It has another elegant feature that as ν becomes large, ν-coherency approaches the logical coherency defined in Chapter 3. In the limiting case both these notions become the

same. The price that we have to pay for this is the increased complexity. Compared to τ-coherency it has the disadvantage that the complexity of ν-coherency increases exponentially with the size of the domain, as well as with ν. The combinatorial explosion inherent in this formulation, in the absence of any heuristic, is apparent in the following non-deterministic algorithm, which is analogous to Algorithm 5.3.1, for determining weak ν-coherency of a mapping with respect to a set of structural constraints.

Algorithm 5.6.1: Checking Approximate Weak Coherency

Input: An admissible mapping F from the source domain D1 = <I1, E1, S1, d1, f1> to the target domain D2 = <I2, E2, S2, d2, f2>; and a set $S \subseteq S1$ such that every structural constraint in S is transformable by F. All the structural constraints are presented in clausal form.

Output: TRUE if <F, S> is a weakly ν-coherent AT-MAP and FALSE otherwise.

Algorithm:

1. If S or S2 is empty then return TRUE and terminate. Otherwise go to the next step.
2. Set $G_0 = S2 \cup F(S)$; and $G_i = \emptyset$ for $1 \leq i \leq \nu$.
3. If $G_i = \emptyset$ for all $0 \leq i < \nu$ then return TRUE and terminate. Otherwise go to the next step.
4. Set i to be some value between 0 and (ν-1).

5. If G_i is empty go to the previous step. Otherwise choose any two clauses X and Y, not necessarily different, from G_i.

6. Set $G_i = G_i - \{X, Y\}$; $Z = Res(X, Y)$.

7. If Z contains the empty clause then return FALSE and terminate. Otherwise set $G_{i+1} = G_{i+1} \cup Z$ and go to Step 2.

Similarly the other algorithms presented in Sections 5.3 and 5.4 will also have their non-deterministic counterparts. The exponential growth of their complexity makes it impossible for them to be implemented on any existing serial machine. Further research is needed in encoding these algorithms in some parallel language [Cf. Sridharan 1984] as well as providing principles and heuristics to constraining the search space before algorithms based on ν-coherency can be implemented.

5.7 SUMMARY

In this chapter we addressed the problem of computability and finite representability of CST, by developing a theory of Approximate Semantic Transference. In AST closure condition on the structural constraints of a domain is removed and the coherency condition is replaced with approximate coherency, which is coherency within a specified complexity bound. We also presented several algorithms, that can be useful in designing computational models of metaphor comprehension and analogical reasoning, for checking coherency and computing approximate T-MAPs, called AT-MAPs, from the domain knowledge. Some of the mechanisms that were used in these algorithms were illustrated by means of some simple examples.

We mentioned some deficiencies of this definition of approximate coherency due to non-closure. To overcome them, we formulated another version of approximate coherency that is based on a complexity bound closure. This version, referred to as ν-coherency, has a higher complexity measure and leads to non-deterministic algorithms.

"... Would you tell me, please, which way I ought to go from here?"

"That depends a good deal on where you want to go," said the Cat.

"I don't much care where—" said Alice.

"Then it doesn't matter which way you go," said the Cat.

"—so long as I get somewhere," Alice added as an explanation.

"Oh, you're sure to do that," said the Cat, "if you only walk long enough."

— Lewis Carroll, "Alice's Adventures in Wonderland".

C H A P T E R VI

CONCLUSIONS AND FURTHER RESEARCH

6.1 CONCLUSIONS

In this thesis our concern was to develop a theory of metaphors and analogies that is precise, computational, and not tied to specific characteristics of a particular domain or some artifact of representation. Further, it was desired to characterize many cognitive features of metaphors and analogies in the theory.

In developing such a theory we started with many assumptions concerning the nature of metaphors and analogies. The assumptions were based on the insights provided by many philosophers, psychologists, linguistics, and computer scientists who had studied metaphors and analogies from different perspectives. Some of the more important ones are given below:

1. A metaphor consists of an application of one or more terms of the source domain to one or more terms of the target domain.

2. All metaphors are based on some structural analogy between the source and the target domains.

3. The notion of *coherency* plays a very central role in determining the aptness of a structural analogy.

Thus, in developing a theory of metaphors we focused on the notion of structural coherency. Before formulating it precisely we needed a formalism for representing domain knowledge.

In order not to base our formalism on specific characteristics of certain domains, and keep it as general as possible, we decided to start from the language of the first order predicate calculus. In our formalism, which is called SL, the domains are characterized by a set of tokens, which form the vocabulary of the domain, and a set of structural constraints, which interrelate the meanings of the tokens from each other. Though SL is very similar to the predicate calculus there are some significant points that we emphasize. Two of the more important ones are:

1. SL allows multiple derivations—which correspond to definitions in predicate calculus—of the derived terms.

2. SL allows multiple choices of primitives in describing a domain. In other words, a domain can be described in several ways by using different primitives.

Having decided how to represent domains we turned back to formalizing coherency. We first introduced the notion of *mapping* which are partial function from the tokens of the source domain to the tokens of the target domain. A mapping allows some of the structural constraints of the source domain, namely those that interrelate the tokens included in the mapping, to be *transformed* to the target domain by replacing all the source domain tokens with their counterparts in

the target domain, as given by the mapping. We call a mapping *coherent* with respect to a set of structural constraints of the source domain if the transformed structural constraints are consistent with the structural constraints of the target domain. All these notions are put together in the definition of a T-MAP which is a pair consisting of a mapping and a set of structural constraints such that the mapping is coherent with respect to the structural constraints.

We also introduced two operators, called Augmentation and Positing Structure, that allow one to create new structure in the target domain. Whereas Augmentation uses derivations to create new tokens in the target domain, Positing Structure is more general and allows tokens and structural constraints to be created in the target domain constrained by the structure of the source domain.

The theory just outlined was called Constrained Semantic Transference [CST]. We demonstrated how CST can be a theory of metaphors by characterizing interpretations of a metaphor in terms of T-MAPs. This characterization can explain several properties of metaphors like metaphorical highlighting and downplaying. We also showed how analogies and scientific models can be construed as T-MAPs. Further, in CST we can formally define the notion of metaphorical inference and truth. Our definitions have the novelty that the criterion of deciding metaphorical truth is the same as that of literal truth.

Although many properties of metaphors and analogies can be explained in CST, it has some severe limitations. For instance, it does not address the issue of meaning change that is normally associated with metaphors. Moreover, it is not computational since coherency is not algorithmically decidable. We removed the

second of these limitations in the theory of Approximate Semantic Transference [AST] in which the notion of coherency is replaced with that of *approximate coherency*, which is coherency within a specified complexity bound. Correspondingly, the definition of AT-MAPs is derived from that of T-MAPs by replacing the coherency requirement on T-MAPs by approximate coherency. We gave two alternative formulations of approximate coherency, one of which leads to deterministic algorithms for computing AT-MAPs and the other to non-deterministic algorithms. We presented several algorithms for computing AT-MAPs in different situations. These algorithms are intended to be used in a complete system that understands or generates metaphors and analogies.

6.2 FURTHER RESEARCH

The research presented in this thesis raises many important questions. We will briefly discuss a few in this section.

One of the limitations of CST that was left unresolved in AST, is that it does not address the problem of meaning change that is associated with metaphors. Earlier in Chapter 4 we made a suggestion that a model of meaning change [Cf. Hill 1982] can be built on top of CST (or AST) by associating confidence or credibility factors with structural constraints and then using a truth maintenance system [Cf. Doyle 1979] to make changes in the source and the target domains. However, to put this in practice will require that the notion of coherency be also redefined to reflect the credibility factors of the constraints. We consider this to be one of the major lines along which CST should be expanded.

Earlier in Chapter 1 we criticized several AI models of analogies on the grounds that they are too domain-specific and it is very hard to separate general principles of analogies from the description of their programs. In the work presented in this thesis we have started from the other extreme and did not include *any* domain specific principles. In order to lend more strength to our theory the next step is to build computational models of analogy and metaphor in specific domains based on AST. This will also be in keeping with the paradigm we outlined in Chapter 1.

Although, our chief motivation in developing CST was to capture as many cognitive properties of metaphors and analogies as possible, we believe that the notions of coherency and T-MAPs play a central role in some other areas of cognition as well. For instance, we notice an interesting parallel between T-MAPs and the semiotic morphisms of Goguen & Linde [1984], which were introduced to formalize systems of translation from one medium of symbols to another. Exactly what role is played by T-MAPs in such translation is a subject of further research. At this point we only conjecture that phenomena like highlighting and downplaying—that can be characterized by using T-MAPs—can also be noticed in the process of translating from one system to another. For instance, in producing a verbal description of a scene the choice of words and sentences emphasizes certain aspects of the scene while downplaying certain others.[1] We also conjecture that the generalized notions of types and admissibility, which we defined at the end of

---

1. Cf. Conklin [1983] for a discussion of issues involved in producing a verbal description of a scene.

Chapter 3, will be found useful in formalizing such translation mechanisms.

Another major line of research concerns the second formulation of approximate coherency, based on ν-consistency presented at the end of Chapter 5. We find this version more attractive due to the fact that in the limiting case—as the bound ν on the approximation approaches infinity—this notion of approximate coherency becomes exact coherency. Since, as we pointed out briefly in Chapter 5, the complexity of the algorithms based on this formulation increases exponentially in the absence of any heuristics, further research need be done to develop more tractable algorithms for determining ν-coherency. There are two possible ways in which this problem can be tackled. Firstly, one can design domain-dependent principles and heuristics to constrain the search space. Secondly, one can develop parallel-algorithms for a faster execution [Cf. Sridharan 1984]. Though an actual implementation of such algorithms will have to await the coming of a general purpose large-scale parallel computer, we can benefit a great deal by simulating these algorithms on a serial machine and studying their complexity.

"... Just look along the road, and tell me if you can see either of them."

"I see nobody on the road," said Alice.

"I only wish I had such eyes," the King remarked in a fretful tone. "To be able to see Nobody! And at that distance too! Why, it's as much as I can do to see real people, by this light!"

— Lewis Carroll, "Alice Through the Looking Glass".

BIBLIOGRAPHY

1. Arbib M.A. and Hesse M.B., 1984, *The Construction of Reality*, To appear.

2. Beardsley M.C., 1962, The Metaphorical Twist, *Philosophy and Phenomenological Research* 22, No. 3, pp. 293–307; reprinted in Johnson 1981, pp. 105–122.

3. Berger P.L. and Luckmann T., 1966, *Social Construction of Reality: A Treatise in Sociology of Knowledge*, Double Day & Co. Inc.; Anchor Book Edition 1967.

4. Black M., 1962, *Models and Metaphors*, Cornell University Press, Ithaca.

5. Black M., 1962a, Metaphor, in Black 1962, pp. 25–47; reprinted in Johnson 1981, pp. 63–82.

6. Black M., 1962b, Models and Archtypes, in Black 1962, pp. 219–243.

7. Black M., 1979, More about Metaphor, in Ortony 1979, pp. 19–45.

8. Boyd R., 1979, Metaphor and Theory Change: What is "Metaphor" a Metaphor for?, in Ortony 1979, pp. 356–408.

9. Boyer R.S. and Moore J.S., 1979, *A Computational Logic*, Academic Press.

10. Brachman R.J., 1978, *A Structural Paradigm for Representing Knowledge*, BBN Report No. 3605, Bolt Beranek & Newman, Cambridge, Mass.

11. Brachman R.J., Fikes R.E., and Levesque H.J., 1983, Krypton: A Functional Approach to Knowledge Representation, *Computer 16*, No. 10, Oct. 1983, pp. 67–73.

12. Bronnenberg W.J.H.J., Bunt H.C., Landsbergen S.P.J., Scha R.J.H., Schoenmakers W.J., and Van Utteren E.P.C., 1980, The Question Answering System PHILQA1, in L. Bolc (ed.) *Natural Language Question Answering System*, MacMillan, London, pp. 217–305.

13. Carbonell J.G., 1982, Metaphor: An Inescapable Phenomenon in Natural Language Comprehension, in W.G. Lehnert & M.H. Ringle (eds.) *Strategies for Natural Language Processing*, Lawrence Erlbaum Associates, pp. 415–433.

14. Carbonell J.G. and Minton S., 1983, Metaphor and Common-Sense Reasoning, Technical Report No. CMU-CS-83-110, Dept. of Computer Science, Carnegie-Mellon Univ., Pittsburg, PA.

15. Carnap R., 1958, *Introduction to Symbolic Logic and its Applications*, Dover Publications, Inc., New York.

16. Cassirer E., 1955, *The Philosophy of Symbolic Forms: Vol. 1*, translated by R. Manheim, Yale Univ. Press, New-Haven.

17. Chang C.C and Keisler H.J., 1973, *Model Theory*, North-Holland Publishing Company, Amsterdam.

18. Conklin E.J., 1983, *Rapid and Limited Text Generation Using Salience*, Ph.D. Dissertation, Dept. of Computer and Information Science, Univ. of Massachusetts, Amherst, Mass.

19. Doyle J., 1979, A Truth Maintenance System, *Artificial Intelligence 12*, pp. 231–272.

20. Eberle R., 1970, Models, Metaphors, and Formal Interpretations, appendix to Turbayne C.M. *The Myth of Metaphor*, Univ. of South Carolina Press, Columbia.

21. Evans T.G., 1963, *A Heuristic Program to Solve Geometric-Analogy Problems*, Ph.D., Dissertation, Dept. of Mathematics, M.I.T., Cambridge, Mass.

22. Foss M., 1949, *Symbol and Metaphor in Human Experience*, Princeton Univ. Press.

23. Friedman J. & Warren D.S., 1978, A Parsing Method for Montague Grammars, *Linguistics and Philosophy 2*, pp. 347–372.

24. Frye N., 1981, *The Great Code: The Bible and Literature*, Harcourt Brace Jovanovich, New-York.

25. Gentner D., 1982, Are Scientific Analogies Metaphors?, in D.S. Miall (ed.) *Metaphor: Problems and Perspectives*, Harvester Press Ltd.

26. Gentner D., 1983, Structure-Mapping: A Theoretical Framework for Analogy, *Cognitive Science 7*, pp. 155–170.

27. Gentner D. and Gentner D.R., 1983, Flowing Waters or Teeming Clouds: Mental Models of Electricity, in D. Gentner & A.L. Stevens (eds.) *Mental Models*, Lawrence Erlbaum, Hillside N.J., pp. 99–129.

28. Green T.F., 1979, Learning without Metaphor, in Ortony 1979, pp. 462–473.

29. Goodman N., 1968, *Languages of Art*, The Bobbs-Merrill Company.

30. Goodman N., 1978, *Ways of Worldmaking*, Hackett Publishing Co.

31. Gougen J. and Linde C., 1984, Optimal Structures for Multimedia Instruction, Final Report, SRI Project 4778, SRI International, Menlo Park, Calif.

32. Guenther F., 1975, On the Semantics of Metaphor, *Poetics 4*, pp. 199–220.

33. Hays P.J., 1975, The Naive Physics Manifesto, Dept. of Computer Science, Univ. of Essex, Colchester, Essex, U.K.

34. Henle P., 1958, Metaphor, in P. Henle (ed.) *Language, Thought, and Culture*, Univ. of Michigan Press, pp. 173–195; reprinted in Johnson 1981, pp. 83–104.

35. Hesse M.B., 1959, On Defining Analogy, *Proceedings of the Aristotelian Society*, pp. 79–100.

36. Hesse M.B., 1963, *Models and Analogies in Science*, Sheed and Ward.

37. Hesse M., 1974, *The Structure of Scientific Inferences*, Univ. of California Press.

38. Hesse M., 1980, *Revolutions and Reconstructions in the Philosophy of Science*, Indiana Univ. Press.

39. Hill J.A.C., 1982, *A Computational Model of Language Acquisition in the Two-Year-Old*, Ph.D. Dissertation, Dept. of Computer and Information Science, Univ. of Massachusetts, Amherst, Mass.

40. Hobbs J.R., 1979, Metaphor, Metaphor Schemata, and Selective Inferencing, Tech. Note 204, SRI International, Menlo Park, Calif.

41. Indurkhya B., 1981, *Sentence Analysis Programs based on Montague Grammar*, Master's Thesis, Philips International Institute of Technological Studies, Eindhoven, The Netherlands.

42. Johnson M. (ed.), 1981, *Philosophical Perspectives on Metaphor*, Univ. of Minnesota Press, Minneapolis.

43. Kling R.E., 1971, *Reasoning by Analogy with Application to Heuristic Problem-Solving: A Case Study*, Ph.D. Dissertation, Dept. of Computer Science, Stanford University, Palo Alto, Calif.

44. Kuhn T.S., 1979, Metaphor in Science, in Ortony 1979, pp. 409–419.

45. Lakoff G. and Johnson M., 1980, *Metaphors We Live By*, Univ. of Chicago Press, Chicago.

46. Levesque H.J., 1982, A Formal Treatment of Incomplete Knowledge Bases, Tech. Report 3, Fairchild Laboratory for Artificial Intelligence Research, Palo Alto, Calif., Feb. 1982.

47. Levin S.R., 1979, Standard Approaches to Metaphor and a Proposal for Literary Metaphor, in Ortony 1979, pp. 124–135.

48. MacCormac E.R., 1976, *Metaphor and Myth in Science and Religion*, Duke Univ. Press, Durham, North Carolina.

49. MacCormac E.R., 1982, Metaphors and Fuzzy Sets, in *Fuzzy Sets and Systems 7*, pp. 243–256.

50. Mansour H., 1983, A Structural Approach to Analogy, AIM 747, Artificial Intelligence Laboratory, MIT, Cambridge, Mass.

51. Mendelson E., 1979, *Introduction to Mathematical Logic*, 2nd ed., D. van Nostrand Company.

52. Miller G.A., 1979, Images and Models, Similes and Metaphors, in Ortony 1979, pp. 202–250.

53. Milner R., 1978, A Theory of Polymorphism in Programming, *Journal of Computer and System Sciences 17*, pp. 348–375.

54. Minsky M., 1975, A Framework for Representing Knowledge, in P.H. Winston (ed.) *The Psychology of Computer Vision*, McGraw-Hill, New York.

55. Montague R., 1974, *Formal Philosophy: Selected Papers of Richard Montague*, Edited by R.H. Thomason, Yale University Press, New Haven.

56. Mooij J.J.A., 1976, *A Study of Metaphor*, North-Holland.

57. Nunberg G.D., 1978, *The Pragmatics of Reference*, Indiana Univ. Linguistic Club, Bloomington, Ind.

58. Ortony A. (ed.), 1979, *Metaphor and Thought*, Cambridge Univ. Press.

59. Ortony A., 1979a, The Role of Similarity in Similes and Metaphors, in Ortony 1979, pp. 186–201.

60. Partee B.H., 1979a, Montague Grammar, Mental Representations, and Reality, in P.A. French *et al* (eds.) *Contemporary Perspectives in the Philosophy of Language*, Univ. of Minnesota Press, Minneapolis, pp. 195–208.

61. Partee B.H., 1979b, Semantics–Mathematics or Psychology ?, in R. Bauerle *et al* (eds.) *Semantics from Different Points of View*, Springer-Verlag, pp. 1–14.

62. Petrie H.G., 1979, Metaphor and Learning, in Ortony 1979, pp. 438–461.

63. Piaget J., 1936, *The Origin of Intelligence in the Child*, translated by Margaret Cook (1953), Penguine (1977).

64. Piaget J., 1951, *Play, Dreams and Imitation in Childhood*, translated by Gattegno and Hodgson, Norton (1962).

65. Quine W.V., 1953, *From a Logical Point of View*, Harvard Univ. Press, Mass. Cambridge.

66. Ricoeur P., 1977, *The Rule of Metaphor*, translated by Robert Czerny *et al*, Univ. of Toronto Press.

67. Ricoeur P., 1978, The Metaphorical Process as Cognition, Imagination, and Feeling, *Critical Inquiry 5*, No. 1, pp. 143–159; reprinted in Johnson 1981, pp. 228–247.

68. Robinson J.A., 1965, A Machine-Oriented Logic Based on the Resolution Principle, *Journal of the ACM 12*, No. 1, Jan. 1965, pp. 23–41.

69. Russell S.W., 1982, Formalizing Factors in Metaphorical Extensions, Tech. Report, Computer Science Dept., Univ. of New Hampshire, Durham.

70. Scha R.J.H., 1981, Distributive, Collective and Cumulative Quantification, in J.A.G. Groenendijk *et al* (eds.) *Formal Methods in the Study of Language*, Part 2, Amsterdam Mathematisch Centrum, pp. 483–512.

71. Schon D.A., 1979, Generative Metaphor: A Perspective on Problem-Setting in Social Policy, in Ortony 1979, pp. 254–283.

72. Shibles W.A., 1971, *Metaphor: An Annotated Bibliography and History*, Wisconsin Language Press, Whitewater.

73. Smullyan R.M., 1965, Analytic Natural Deduction, *Journal of Symbolic Logic 30*, No. 2, June 1965, pp. 123–139.

74. Smullyan R.M., 1968, *First-Order Logic*, Springer-Verlag.

75. Sridharan N.S., 1984, A Semi-Applicative Language for Artificial Intelligence Programming, Privately Circulated Draft

76. Suppes P., 1957, *Introduction to Logic*, D. van Nostrand Co., Princeton.

77. Tourangeau R. and Sternberg R.J., 1982, Understanding and Appreciating Metaphors, *Cognition 11*, No. 3, May 1982, pp. 203–244.

78. Turbayne C.M., 1962, *The Myth of Metaphor*, Yale Univ. Press, New-Haven; revised edition with appendix by R. Eberle "Models, Metaphors, and Formal Interpretations", Univ. of South Carolina Press, Columbia, 1970.

79. Urban W.M., 1939, *Language and Reality: The Philosophy of Language and the Priniciples of Symbolism*, George Allen & Unwin Ltd., London, U.K.

80. Van Dijk T., 1975, Formal Semantics of Metaphorical Discourse, *Poetics 4*, pp. 173–198.

81. Wheelwright P., 1962, *Metaphor and Reality*, Indiana Univ. Press, Bloomington.

82. Whorf B.L., 1956, *Language, Thought, and Reality: Selected Papers of Benjamin Lee Whorf*, Edited by J.B. Carroll, M.I.T. Press, Cambridge, Mass.

83. Winston P.H., 1978, Learning by Creatifying Transfer Frames, *Artificial Intelligence 10*, pp. 147–172.

84. Winston P.H., 1981, Learning New Principles from Precedents and Exercises, AIM 632, Artificial Intelligence Laboratory, MIT, Cambridge, Mass.

85. Winston P.H., 1982, Learning by Augmenting Rules and Accumulating Censors, AIM 678, Artificial Intelligence Laboratory, MIT, Cambridge, Mass.

86. Zadeh L.H., 1965, Fuzzy Sets, *Information and Control 8*, pp. 338–353.

"... And I wish you wouldn't keep appearing and vanishing so suddenly: you make one quite giddy!"

"All right," said the Cat; and this time it vanished quite slowly, beginning with the end of the tail, and ending with the grin, which remained some time after the rest of it has gone.

"Well! I've often seen a cat without a grin," thought Alice; "but a grin without a cat! It's the most curious thing I ever saw in all my life!"

— Lewis Carroll, "Alice's Adventures in Wonderland".

A P P E N D I X

PROOF OF THEOREM 1

In this appendix we will formally prove Theorem 1 that was stated in Section 3.3. The theorem is as follows:

Theorem 1: If T is an invertible maximal T-MAP from the source domain D1 to the target domain D2 and both D1 and D2 satisfy the axiom of closure, then T^{-1} is a maximal T-MAP from D2 to D1.

We will prove this theorem by proving a more general result. The more general theorem is stated in a form that will make the proof easier to present. Let L1 and L2 be sets of tokens such that there is a type associated with every token. T1 is a set of sentences such that every token in every sentence in T1 is in L1, and T2 is a set of sentences such that every token in every sentence in T2 is in L2. We say that T1 and T2 are *theories* of L1 and L2 respectively. We assume that both T1 and T2 are closed and consistent. Let $L \subseteq L1$ and F be a total injective function from L into L2 such that it preserves types of the token, i.e. if $X \in L$ then both

X and F(X) have the same type. By T1 | L we mean the set of sentences of T1 such that none of the sentences contain any token that is not in L. In other words, T1 | L is the maximal subset of T1 that is transformable with respect to F. F(T1 | L) is the set of sentences obtained from T1 | L by transforming every sentence by F. Obviously, F(T1 | L) is a theory of L2. F(L), a subset of L2, is the image of L under F. F^{-1}, a function from F(L) into L1, is the inverse of F. Also, if S is a sentence then we use S1 |= S to mean that S can be deduced from S1, where S1 is a sentence or a set of sentences. Now the generalized version of Theorem 1 is as follows:

Theorem 1a: If T1 is consistent and F(T1 | L) ∪ T2 is consistent then F^{-1}(T2 | F(L)) ∪ T1 is also consistent.

We will first prove Theorem 1a and then show how Theorem 1 follows from Theorem 1a.

The proof of Theorem 1a is based on Craig's Interpolation Lemma [Chang & Keisler 1973, pp. 84–85], which is restated as follows:

Proposition 1 (Craig's Interpolation Lemma): Let S1, S2 be sentences such that S1 |= S2. Then there exists a sentence S such that,

1. S1 |= S and S |= S2, and
2. For every token X that occurs in S, X also occurs both in S1 and in S2.

Now we state another proposition and then prove a lemma that will be used in the proof of Theorem 1a.

Proposition 2: Let L1 be a set of tokens and T1 be a closed theory of L1. Further, let $L \subseteq L1$ and $T1 \mid L$ be T1 restricted to L, as defined above. For any sentence S of L the following is true:

if $T1 \models S$ then $(T1 \mid L) \models S$

Proof of Proposition 2: If $T1 \models S$ then $S \in T1$ (since T1 is closed). Therefore, $S \in (T1 \mid L)$, and $(T1 \mid L) \models S$.

[Proof Complete].

Lemma: Let L1 be a set of tokens and $L \subseteq L1$. Further let T1 be a closed and consistent theory of L1 and T be any theory of L. Then,

if $(T1 \mid L) \cup T$ is consistent then $T1 \cup T$ is consistent.

Proof of Lemma:

1. Suppose $T1 \cup T$ is inconsistent, then there exist finite subsets $\Sigma 1$ (a subset of T1) and Σ (a subset of T) such that $\Sigma 1 \cup \Sigma$ is inconsistent.
2. Let $\sigma 1$ be the conjunction of all the sentences in $\Sigma 1$ and σ be the conjunction of all the sentences in Σ. Then, from the inconsistency of $\Sigma 1 \cup \Sigma$ it follows that $\sigma 1 \models \sim\sigma$.
3. From Proposition 1 above, this implies that there is a sentence ζ such that $\sigma 1 \models \zeta$ and $\zeta \models \sim\sigma$; and every token in ζ occurs both in $\sigma 1$ and in σ. Since $\Sigma \subseteq T$ and T is a theory of L, this implies that ζ is a sentence of L.

4. From $\sigma 1 \models \zeta$ we get $\Sigma 1 \models \zeta$ and $T1 \models \zeta$. By Proposition 2 above, this entails that $(T1 \mid L) \models \zeta$ (since T1 is closed by assumption).
5. From $\zeta \models \sim\sigma$ we get $\sigma \models \sim\zeta$ and $T \models \sim\zeta$.
6. The last two steps (4 & 5) entail that $(T1 \mid L) \cup T$ is inconsistent which contradicts the assumption of the lemma.

[Proof Complete]

Now we are ready to present a proof of Theorem 1a.

Proof of Theorem 1a:

1. If $F(T1 \mid L) \cup T2$ is consistent then $F(T1 \mid L) \cup (T2 \mid F(L))$ is also consistent.
2. If $F(T1 \mid L) \cup (T2 \mid F(L))$ is consistent then $F^{-1}(F(T1 \mid L)) \cup F^{-1}(T2 \mid F(L))$ is also consistent.
3. If $F^{-1}(F(T1 \mid L)) \cup F^{-1}(T2 \mid F(L))$ is consistent then $(T1 \mid L) \cup F^{-1}(T2 \mid F(L))$ is also consistent (since F^{-1} is the inverse of F).
4. Since $F^{-1}(T2 \mid F(L))$ is a theory of L, by the lemma above this implies that $T1 \cup F^{-1}(T2 \mid F(L))$ is also consistent.

[Proof Complete]

Finally we will show how Theorem 1 is an instance of Theorem 1a and therefore follows from it. Let the source domain D1 be $\langle I1, E1, A1, d1, f1\rangle$ and the target domain D2 be $\langle I2, E2, A2, d2, f2\rangle$, and $T = \langle F, S\rangle$ be an invertible maximal T-MAP from D1 to D2. We now define L1, L2, L, T1, T2, and F′ as follows:

1. $L1 = I1 \cup E1$
2. $L2 = I2 \cup E2$
3. L is the set of tokens such that every token in S occurs in some sentence in S. Thus $L \subseteq L1$.
4. $T1 = A1$
5. $T2 = A2$
6. F' is a function from L into L2 that is defined as follows. If $X \in L$ is such that $X \in I1$ then $F'(X) = F(X)$, otherwise $F'(X) = X$.

Now from Theorem 1a it follows that $\langle F^{-1}, S' \rangle$, where S' is the maximal subset of A2 that is transformable with respect to F^{-1}, is a T-MAP from D2 to D1.