32P

INFERRING CONCEPTUAL STRUCTURES FROM PICTORIAL INPUT DATA

BY

SHARON CAROLINE SALVETER

A thesis submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy
(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN--MADISON

1978

INFERRING CONCEPTUAL STRUCTURES FROM PICTORIAL INPUT DATA

by

Sharon Caroline Salveter

Under supervision of Assistant Professor Raphael A. Finkel

ABSTRACT

This thesis investigates the mechanisms a program may use to learn conceptual structures that represent natural language meaning. A computer program named Moran is described that infers conceptual structures from simulated pictorial input data. Moran is presented "snapshots" of an environment and an English sentence describing the action that takes place between the snapshots. The learning task is to associate each root verb with a conceptual structure that represents the types of objects that participate in the action and the changes the objects undergo during the action. Four learning mechanisms are shown to be adequate to accomplish this learning task. The learning mechanisms are described along with the conditions under which each is invoked and the effect each has on existing memory structures. The conceptual structures that Moran infers for seventeen senses of four root verbs are shown.

ACKNOWLEDGEMENTS

I gratefully thank the many people who have helped make Madison my home, extended their friendship and supported me during my years at Wisconsin. Gregg Oden and Greg Scragg spent many hours helping me turn vague ideas about "something to do with natural language learning" into a specific research project. Leonard Uhr has supported me as his research assistant, engaged in many fruitful conversations and helped me maintain a global view. As a friend and colleague Raphael Finkel gave me immense amounts of time, forced me to sharpen my ideas and implement everything, talked about wildflowers, watered my plants when I was gone and gave me emotional support. Steve Skedzeleski put up with living in the same office with me for two years. David DeWitt and Marvin Solomon provided humor, encouragement, lunch partners and only minimally obnoxious comments about AI. Fred and Bev Jacobson became my family, cared about me and fed me during many difficult times. Kathy Schallock and Sue Mineka cared about me and encouraged me and always were there.

I dedicate this thesis to my mother Caroline. Her continual support and enthusiasm for my endeavors has made it all possible.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

This thesis addresses the question of how complex memory organizations that can be used to represent natural language meaning may be inferred from environmental data.

A. History

1. Representation of Knowledge in Memory

Much research has been devoted to the design of meaning structures that represent knowledge in memory. [Bartlett 32] laid the groundwork some years ago for many of the representations currently being used when he proposed a representation called schemata, which are developing patterns that guide the interpretation of a person's experience. Schemata are changing organizations of past experiences that serve as a frame of reference for processing incoming information. There is a constant interaction between the schemata and incoming information. The schemata are simultaneously related and adapted to the new information.

[Minsky 75] has proposed frames as a unifying general-

ization of several representations. Frames represent stereotyped situations and contain several types of information including how to use the frame and expectations about the situation. Minsky gives an example of a frame for a room. A room frame contains information about what we expect to see such as walls, a floor and a door plus information about what different types of rooms might contain. In addition to information about objects in the room, the frame indicates relationships between the objects. Frames may also be used to represent stereotyped events such as a birthday party or shopping in a supermarket.

[Norman and Rumelhart 75] have defined Active Structural Networks that contain both data and procedures. Nodes in Active Structural Networks can represent facts or procedures that guide the interpretation of and search for new information.

The predicate calculus [McCarthy and Hayes 69], production systems [Newell and Simon 72, Davis and King 75] and procedural representation [Winograd 72] have also been used to represent knowledge for theorem proving, question answering and language understanding. In general, however, the frame-like knowledge representations have been more

successful in representing general knowledge that can be expressed in natural language.

Other theories of representation deal specifically with how to represent natural language meaning. [Quillian 68 and Quillian 69] proposed semantic networks as a way to represent entities tied together by relationships. For example, the sentence "John hit Mary" might be represented as:

```
        hit
John ---> Mary
```

Each node in the net represents an entity. Labelled arcs connecting the nodes represent relationships between entities. Additional arcs from the "John" and "Mary" nodes may represent more information about John and Mary.

[Fillmore 68] suggested case grammar to represent the semantic relationship between verbs and noun phrases by defining constraints on how verbs and noun groups form sentences. Every action has associated with it a set of cases that define the function and type of noun groups that may be combined with the verb to form sentences. For example, the verb "hit" requires an agent, an object and an instrument. Case grammars do not indicate what happens during the action, only the objects that participate in it.

[Schank 72] defines a theory of conceptual dependency as an organization of primitive actions with rules defining the things that participate in an action as well as what happens during the action. The conceptual dependency for "John hit Mary" shows John propelling some object in Mary's direction with the result that the object and Mary become in physical contact. Like case grammar, conceptual dependencies indicate the types of entities that participate in an action, like hitters, hittees and instruments. However, they also indicate what happens to the entities during the action. [Schank 73] describes the MARGIE system that can map natural language into conceptual dependency structures, make inferences and generate natural language from the conceptual dependencies.

2. Learning

Because of the immensity of data bases needed for language comprehension in all but trivial domains, many computer scientists have begun to look to programs that learn as a way to construct these data. Although computer learning has been an active research area for many years [Samuel 59, Samuel 67, Uhr and Vossler 63, Badler 75, Winston 75, Soloway and Riseman 77, Soloway 77], most of the resulting programs are designed to acquire structures

specific to particular applications and have not addressed the problem of learning generalized knowledge structures that may also be used to represent natural language meaning.

Samuel's program learns the coefficients of an evaluation polynomial used in evaluating checkerboards. Uhr and Vossler's program generates bit patterns used to identify hand-printed characters. Winston's program learns unary and binary predicates that describe geometric constructions. Badler is primarily concerned with learning descriptions of object movement. Soloway and Riseman's program learns different levels of descriptions of action-oriented games by forming a hierarchy of generalized classes that describe the similarities of the descriptions.

None of these programs seems extendible to learning generalized knowledge structures. The structures used to represent natural language meaning must be general since language can be used to discuss anything.

[Anderson 77]'s Language Acqusition System (LAS) is a purely syntactic characterization of language learning. LAS incorporates a model of how grammar is acquired that relates strings of words to network representations of their meaning. The input to LAS is a series of pairs of

sentences and semantic networks that represent the meaning of the sentences. From this input LAS acquires an augmented transition network (ATN) that maps between the sentences and their meaning structures. LAS must be given the concepts referred to in the sentences. LAS constructs an ATN for every constituent in the input sentence. Whenever a phrase can be parsed by two ATNs, the ATNs are merged. The resulting ATN is a grammar of the sentences seen so far. Unfortunately, LAS has no concept induction capability and is given the semantic meaning structures that correspond to the sentences rather than inferring them.

[Reeker 74] has also developed a theory of syntax acquisition. His mental grammar is an attempt to connect the surface structures of sentences the program hears to sentences that can be produced by an earlier mental grammar. Reeker's program assumes an initial finite state grammar. The meaning derived from an input situation and a portion of an adult sentence is used in consultation with the grammar to produce a child version of the adult sentence. Comparison of the child sentence and adult sentence yields a difference indicated by a table of connections. The table of connections and changes is used to modify the child sentence to bring it more closely in line with the adult sen-

tence. The table provides the grammatical modification that must be made to the child grammar. Like LAS, Reeker's program relates sentences to semantic structures but does not infer the semantic structures themselves.

[Hayes-Roth 76] has investigated general learning procedures. His theory of interference matching (IM) states that the knowledge underlying given examples is identified by producing a representation or abstraction that emphasizes commonalities and attenuates differences. IM identifies the common properties of two exemplars and extracts a third representation that is a template matched by the two exemplars. IM works by selecting a relation from one exemplar and putting it into correspondence with a relation from the other exemplar. Parameters that have identical properties are identified as equivalent and the resulting relation becomes the abstraction associated with a set of parameter bindings. The process continues with pairs of relations from each exemplar. As long as parameter bindings remain consistant, the relation is added to the abstraction. A heuristic hill-climbing technique is used to select the best maximal abstraction. Although the IM procedure compares event descriptions to identify commonalities, it does not infer event descriptions. IM has been most successful with events that may be described with

bit vectors and less successful with relational descriptions. The major problems are increasing the world knowledge that can be brought to bear and finding generalized binding functions to allow many-to-one mappings.

[Harris 72] has designed a comprehensive language acquisition system that divides the linguistic environment into three parts. The program is divided into three corresponding phases. In Phase 1 the program is given a sequence of word-list and concept-list pairs and establishes correlations between the word groups and concepts. The purpose of Phase 2 is to infer a grammar. The input is again a set of ordered pairs. The first component in a pair is an English sentence, the second a list of the "parts of speech" of the corresponding words in the first component. The "parts of speech" are categories of concepts built into the program. The result of Phase 2 is a set of context-free phrase structure rules and a small number of transformations. Phase 3 uses the grammar produced in Phase 2 to generate responses to input sentences. Like LAS, Harris' program is given semantic representations of utterances along with the utterances. Additionally, the teacher must also know the structure of the language in terms of the program's "parts of speech".

[Rumelhart and Norman 76] have recently begun to address the question of how Active Structural Netwoks might be learned, but they have not yet incorporated learning into their implementation. However, the three modes of learning they postulate, Accretion, Tuning and Restructuring, are compatible with the learning processes described in this dissertation. Accretion increments the knowledge base with a new set of facts. Tuning changes the categories used for interpreting new information. Restructuring devises new structures to interpret new information and imposes a different organization on the information already stored.

B. Goals

Learning involves a modification of organizational structures in memory as well as the accumulation of facts, since knowing includes the recognition of relationships among facts as well as the facts themselves. The research described here investigates how a program might learn conceptual structures that represent verb meaning from simulated perceptual and linguistic input. A computer program named Moran has been implemented that learns the meanings of verbs by distilling the changes it observes in the environment and associating those changes with root verb

words. This program shows that a small set of fairly simple and straightforward mechanisms is adequate to infer structural descriptions from environment information and a small body of set containment information.

Secondary goals are the design of a data structure for the representation of natural language meaning, the abstraction of descriptions from perceptual data, investigation of the notion of semantic similarity of graph structures, discovery of meaning units that may be viewed as primitives and the implementation of a modular program that can be used as a vehicle for experimentation.

C. Overview of Moran

Moran infers semantic graphs from perceptual input. The input to Moran is a series of two-snapshot sequences that depict an action taking place and a natural language sentence that describes what happens in each snapshot sequence. The input is provided by a human trainer. The output of Moran is a set of semantic graphs that represent the various discovered meanings of each root verb and the relations amoung them.

For each snapshot sequence, Moran first creates a reresentation for what happened by linking states in the

first snapshot with states in the second snapshot. Second, it locates the existing representation that it considers to be semantically closest to the current input. Finally, it modifies the graph structure so that the current input is represented and similarities to and differences from previous inputs are reflected. Occasionally, Moran attempts to discover global similarities in the graph structure.

D. Organization of the Dissertation

Chapter 2 describes the Conceptual Meaning Structure that Moran uses to represent the inferred verb meanings. Chapter 3 describes the information with which Moran is initialized and and the form of its input.

Chapter 4 discusses the procedures Moran uses to infer the graphs from the input. The procedures generate a semantic description from the input data, find a semantically close existing graph and modify the graph structures. Chapter 5 documents the implementation and gives extensive examples of results. Chapter 6 discusses both implementation and theoretical limitations of Moran, analyzes its contributions and gives suggestions for further research.

CHAPTER 2

THE CONCEPTUAL MEANING STRUCTURE

A. General Structure

In order to learn any new information, it is necessary to have a means of representing what is learned. Moran learns frame structures called Conceptual Meaning Structures (CMSs) that represent verb meaning. One CMS is associated with each root verb word the program learns. This root verb word is the natural language word representing a verb concept. For example, one CMS is associated with all senses of the root verb "move". The sentences:

Ristin moved the book.
John was moved by the movie.
Figaro is moving today.
Mother moved her nose.

all map to the same root verb "move".

The meaning of one sense of a root verb is composed of arguments that describe the entities that participate in the action and effects that describe the changes the arguments undergo during the action. One CMS is associated with all the senses of a root verb and consists of two

parts, arguments and effects, each of which is represented by a labeled directed acyclic graph. A complete path through a graph defines a set of nodes. One sense of a root verb is the set of arguments defined by a complete labeled path through the arguments graph plus the set of effects defined by the corresponding complete labeled path through the effects graph. For example, sense number four of a root verb is the set of arguments defined by the complete path of arcs labeled four in the arguments graph plus the set of effects defined by the complete path of arcs labeled four in the effects graph. The entire collection of CMSs is called the verb world.

B. Arguments

Each argument specifies the type of entity that may participate in an action. Each argument consists of:

1) an argument name

2) a set of restrictions that limit the type of object that may instantiate the argument. Each restriction is augmented by a frequency count that indicates how often the restriction has been met in the program's experience.

A simple example of a set of arguments associated with one sense of a root verb is:

AGENT	Human(8): Figaro(3), Ristin(5)
OBJECT	Furniture(8): Table(4), Chair(2), Lamp(2)
C1	Location(8): Loca(4), Locb(4)
C2	Location(8): Loca(2), Locb(4), Locc(2)

This sense of the verb requires four arguments: an Agent, an Object and two other arguments, C1 and C2. According to this CMS, the agent must be human. A human Agent has been seen eight times: Figaro three times and Ristin five times. The Object must be furniture. A table has been seen four times, the chair and lamp each two times. The third argument, C1, must be a location. Loca and Locb have each been seen four times. Argument C2 must also be a location: Loca has been seen two times, Locb four times and Locc two times.

Meaningful argument names like AGENT and OBJECT are used to increase output readibility. Although Moran treats input argument names somewhat differently from internally generated argument names, unlike most case systems [Fillmore 68, Bruce 75], no a priori semantic information is embodied in individual argument names.

C. Effects

A set of effects is a description of the visual effects of a sense of a verb. The effects indicate what happens to the entities described in the argument part during the action. An effect is an ordered pair of triples. The first triple is a state derived from the first snapshot of the environment, the second a state derived from the second snapshot of the environment. The pair indicates that during the action the state in the first snapshot changed into the state in the second snapshot.

An example of a set of effects associated with the arguments described above is:

AGENT AT C1 ---> AGENT AT C2

OBJECT AT C1 ---> OBJECT AT C2

AGENT PHYSCONT OBJECT ---> AGENT PHYSCONT OBJECT

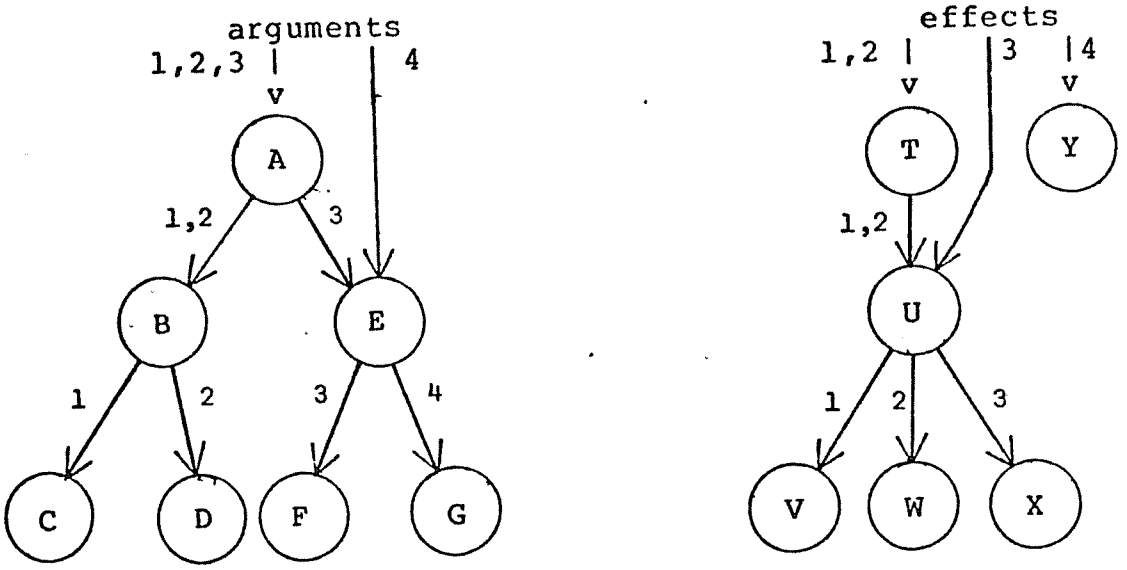
The AGENT and OBJECT moved from C1 to C2 and the AGENT and OBJECT remained in physical contact (PHYSCONT). This last entry shows that an effect may describe a constant state as well as a change in the environment.

D. Graph Structure

The CMS structure reflects similarities among dif-

ferent senses of a root verb by allowing different senses to share nodes. Both the arguments and the effects comprising all the senses of a root verb are organized into labeled directed acyclic graphs. Each node in an argument or effect graph is a set of one or more arguments or effects. This organization allows groups of arguments or groups of effects to participate in several senses of a root verb as well as in different root verbs. These similarities are explicitly represented in the CMS structure.

A complete CMS for a root verb with four senses might look like:



Nodes A through G are sets of one or more arguments and nodes T through Y are sets of one or more effects. One complete sense of this root verb is a set of arguments described by one labeled path through the arguments graph plus the set of effects described by the correspondingly labeled path through the effects graph. For example, sense three is defined as the combination of arguments found in nodes A, E and F plus the combination of effects found in nodes U and X. How the program discovers arguments and effects from the input and how the graph structure is built are described in Chapter 4.

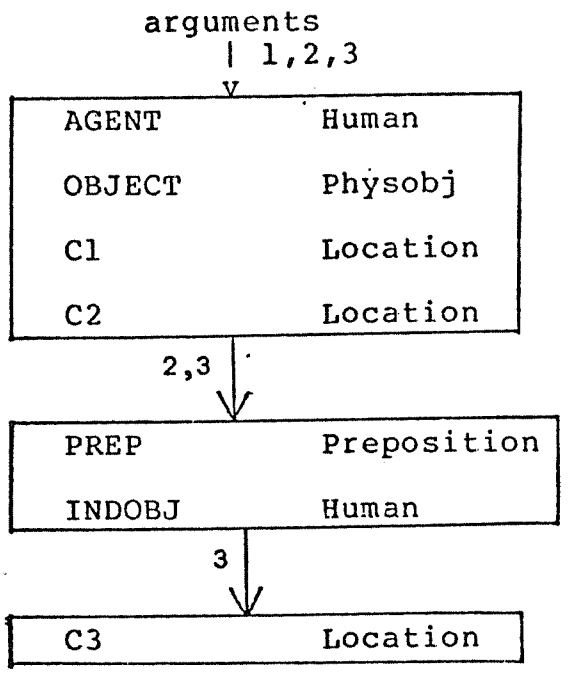
Some kinds of similarity relationships between different senses of the same root verb are explicitly represented in the graph. The structure of the graph makes

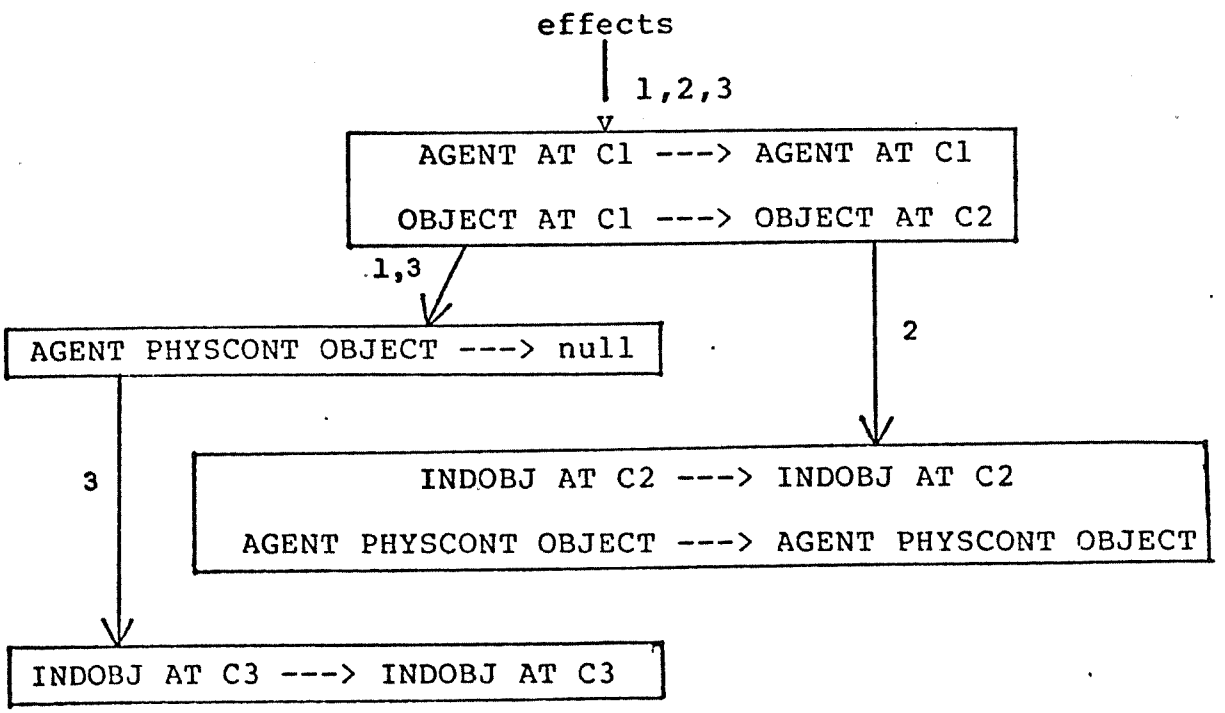
it clear that senses one and two are semantically closer to each other than senses one and four because they share more common parts. Similarly, senses one, two and three are semantically closer to each other than they are to sense four.

Nodes may also be shared across CMS boundaries. Such nodes are called global building blocks and are discussed in Section 4.D.

E. An Example

A simplified example of a possible CMS for three senses of the verb "throw" is shown below.


arguments
1,2,3
AGENT Human
OBJECT Physobj
C1 Location
C2 Location
2,3
PREP Preposition
INDOBJ Human
3
C3 Location


effects
1,2,3
AGENT AT C1 ---> AGENT AT C1
OBJECT AT C1 ---> OBJECT AT C2
1,3
2
AGENT PHYSCONT OBJECT ---> null
3
INDOBJ AT C2 ---> INDOBJ AT C2
AGENT PHYSCONT OBJECT ---> AGENT PHYSCONT OBJECT
INDOBJ AT C3 ---> INDOBJ AT C3

Sense 1 can represent "Mary throws the ball" where Mary stays at the same location, C1, the ball changes from location C1 to location C2 and Mary changes from being in physical contact (PHYS) with the ball to not being in physical contact with the ball. Sense 2 might represent "Figaro throws the ball to Ristin". As in sense 1, Figaro stays in the same location, C1, and the ball changes location from C1 to C2. In addition, the indirect object (INDOBJ), Ristin, stays at the same location C2 and the ball changes from being in physical contact with Figaro to being in physical contact with Ristin. Sense 3 could represent "Sharon threw the terminal at Raphael". It is identical to sense 1 except that the INDOBJ, Raphael, stays at the same location. Unlike sense 2, the location of the INDOBJ is not the same as the resulting location of the OBJECT.

Like Schank's Conceptual Dependencies, CMSs indicate what types of objects participate in a given action and and what happens to those entities during the action. However, unlike Conceptual Dependencies, CMSs are not composed of pre-defined primitive meaning units that specify the arguments, argument restrictions and argument behavior. Moran infers what arguments are required of an action, the restrictions on the arguments and the behavior of the arguments during the action. Since nodes may be shared across

CMS boundaries, the commonly used nodes that Moran infers may be compared with Schank's primitives to see how they correlate.

Further comparison of the expressive power of CMSs and other representations can be found in Chapter 6.

CHAPTER 3

PROGRAM INPUT AND INITIAL STATE

Moran learns the meaning of a verb (the CMS to associate with a root verb) by observing an environment, hearing a parsed surface sentence, distilling the changes it observes in the environment and then incorporating them into the CMS structure.

A. The Environment

The environment is a simulated room containing objects, people and reference locations. It may be made as detailed in information as desired. The environment is presented to the program as a list structure. This input is feasible output of a good scene recognition program [Yakimovsky 73].

An environment description list is an unordered collection of ordered triples of the form (object relation value). Each environment description list, called a <u>snapshot</u>, describes the state of the environment at one instant in time. A fairly sparse environment description list might look like:

(FIGARO AT LOCA)

(RISTIN AT LOCB)

(TABLE AT LOCD)

(BALL ON TABLE)

(PICTURE ON WALL)

(CHAIR AT LOCC)

(FIGARO PHYSCONT NEWSPAPER)

(CLOCK ON WALL)

(RISTIN PHYSCONT BOOK)

Although Moran only processes unnested triples, an inadequate general representation, it could be extended to handle triples nested to an arbitrary depth since an inference procedure operating on unnested triples can simulate nested triple representation.

The program is always presented with two successive environment description lists. The first represents a snapshot of the environment before an action begins and the second a snapshot of the environment after the action has taken place. The program assumes that the first snapshot immediately precedes the start of the action and the second immediately follows the conclusion of the action.

B. The Natural Language Sentence

In addition to the two environment snapshots, the program is given a parsed natural language sentence that describes the action that took place in the snapshot sequence. The sentence is preparsed to indicate argument names and values. The sentence

"Ristin moved the book to the table"

is given to the program as:

AGENT	Ristin
ACTION	Move
OBJECT	Book
PREP	To
INDOBJ	Table

The program can process more complicated sentences as long as they do not contain adverbs, conjunctions, relative clauses and anaphoric references. The preparsed input could be the result of almost solely syntactic parsing of the sentence [Woods 70].

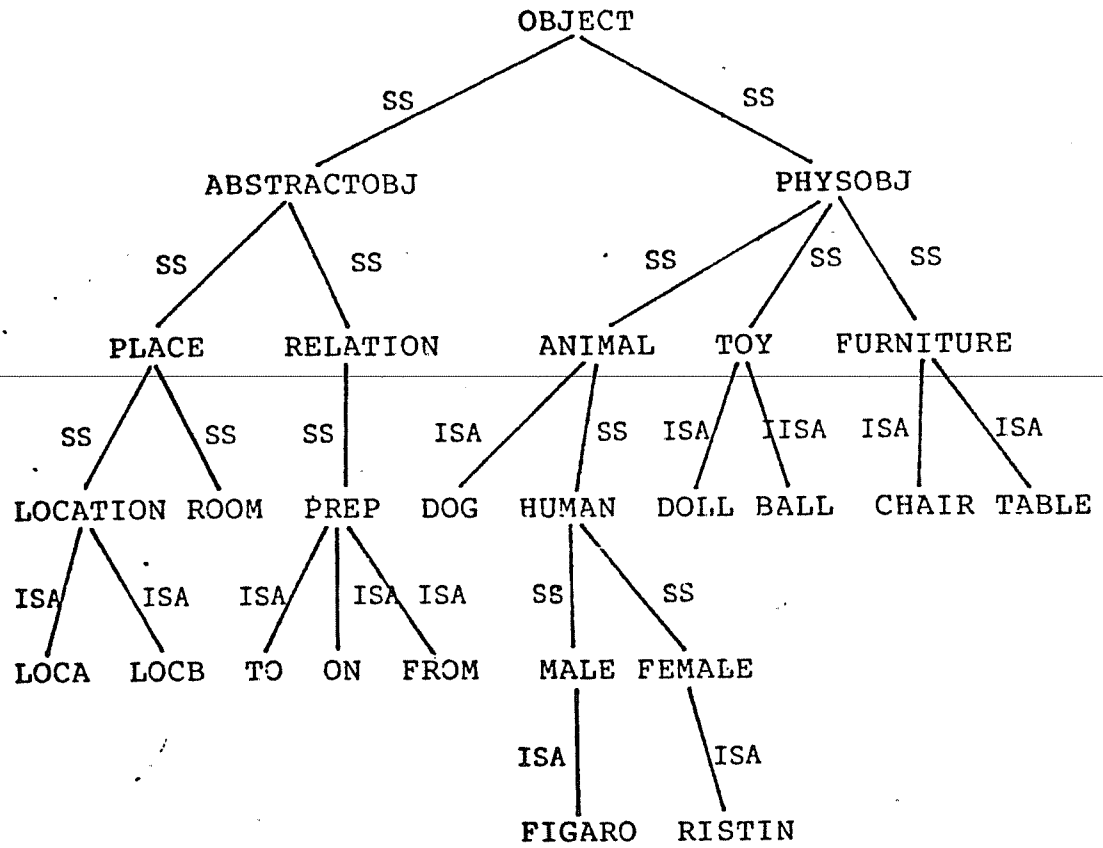
Argument names of the form AGENT and OBJECT that are given to the program as input are generally treated as more important than any internally generated name such as C1 or

C2. The words associated with input argument names in the sentence are given particular attention when looking for changes in the environment. Input arguments are also given priority over non-input arguments in the matching procedure. The input argument names could be labeled X1, X2, ... without significantly altering the performance of the program since what is important is that the word was mentioned in the input sentence, not the argument name assigned to it.

C. World Knowledge

Moran is initialized with information about objects in its world. It needs this information to learn complicated interrelationships among these objects. This information is in a separate, self-contained body of knowledge called world knowledge. World knowledge is limited to a tree of nodes expressing set membership with ISA and SUPERSET/SUBSET links. The generalization process to be described later is based on the set information provided in world knowledge. Although a much richer body of knowledge may theoretically be stored in world knowledge, the program only uses set containment information in the generalization process.

A small world knowledge base is shown in below.



PREP - preposition

ABSOBJ - abstract object

PHYSOBJ - physical object

SS - superset/subset

D. An Example

An example of an input sequence given to Moran is

shown below. The human trainer designs the snapshot triples by looking at pictures in [Richards 61] and encoding the visual information as triples. The following input corresponds to "Figaro threw the doll to Ristin".

Sentence:

AGENT	Figaro
ACTION	Throw
OBJECT	Doll
PREP	To
INDOBJ	Ristin

Snapshot1	Snapshot2
(FIGARO AT LOCB)	(FIGARO AT LOCB)
(RISTIN AT LOCA)	(RISTIN AT LOCA)
(FIGARO PHYSCONT DOLL)	(RISTIN PHYSCONT BALL)
(DOLL AT LOCB)	(DOLL AT LOCA)
(CLOCK ON WALL)	(CLOCK ON WALL)
(BOOK ON TABLE)	(BOOK ON TABLE)
(TABLE AT LOCC)	(TABLE AT LOCC)

Extensive input examples are given in Chapter 5.

CHAPTER 4

INFERRING CONCEPTUAL MEANING STRUCTURES

Moran infers labeled directed acyclic graphs that represent different senses of root verbs from input composed of a snapshot of the environment before the action, a snapshot of the environment after the action and a parsed natural language sentence describing the action that took place. This chapter describes the processes Moran uses to learn graphs from snapshots.

In response to each input, the program first associates states described in the first snapshot with states described in the second snapshot and creates a one-sense dummy CMS that represents what happened in the snapshot sequence. Second, it locates the CMS in the verb world that is semantically closest to the input situation. Finally, it modifies the verb world to incorporate the new knowledge into the existing knowledge. Occasionally, Moran inspects the entire verb world to look for similarities across CMS boundaries.

A. Comparing Snapshots

Moran first compares the two input snapshots and

creates a one-sense dummy CMS that represents the action that took place in the input sequence. Several mechanisms are required:

1) Filtering information not relevant to the action from the snapshots.

2) Generating effects from the triples in each snapshot.

3) Creating a dummy CMS that describes the input.

1. Filtering Data

The snapshot data are complete descriptions of the room and its contents. The procedure DIRECTATTENTION reduces the amount of information to a more manageable size by filtering from each snapshot those triples not considered to be relevant to a description of the action.

There are many possible ways to filter the data. If Moran had an "eye", only the information within its field of view might be considered [Zeigler 78]. If its world knowledge base were richer in general semantic information, it might use this information to try to make a decision about the significance or importance of each data item. The program currently filters the snapshot information by

directing its attention to those entities mentioned in the input sentence. Thus, the trainer prescribes what entities will be noticed by mentioning those entities in the sentence.

Each snapshot is filtered by removing any triple that does not contain at least one word mentioned in the input sentence. The resulting modified snapshots, consisting only of triples that contain input words, are compared by component COMPARISON.

This procedure is not adequate in all cases since objects are sometimes not mentioned in the input sentence even though they are critical to the meaning of the verb. For example, Moran was given the sentence "Figaro threw a party" with two snapshots showing many people in the room and food and drink on the table. Moran removed all but one triple from each snapshot: the triple showing Figaro at a location in the room. It is not realistic to expect Moran to require a surface sentence like "Figaro threw a party with vegetables, dip, potato chips, beer and wine for Joe, Marv, Sue, Kathy, Hanna, Eric, Dave, ...". However, the current filtering process works fairly well for sentences that mention concrete objects (like Raphael).

2. Discovering Effects

Moran determines what has taken place during the snapshot sequence by comparing snapshot1 with snapshot2. Component COMPARISON discovers the effects that describe the event by linking a triple from snapshot1 (possibly a null triple) with a triple from snapshot2 (possibly null). Each triple in snapshot1 is compared with each triple in snapshot2 and the linkup is assigned a value that indicates the goodness of the association.

Associations of triples can be ambiguous. Change can be seen as occuring in the first position, the "object" position, the third position, the "value" position or the second position, the relation position, of a pair of triples. The program considers differences in the relation or value positions of a pair of triples to be more valuable in explaining the event than differences in the object position. If the program has as environment information:

snapshot1	snapshot2
(BOOK AT LOCA)	(BOOK AT LOCB)
(CUP AT LOCB)	(CUP AT LOCA)

then the associations chosen as effects will be:

(BOOK AT LOCA) ---> (BOOK AT LOCB)

(CUP AT LOCB) ---> (CUP AT LOCA)

rather than:

(BOOK AT LOCA) ---> (CUP AT LOCA)

(CUP AT LOCB) ---> (BOOK AT LOCB)

In other words, it is considered to be more important that objects have changed values than that values have changed objects.

The result of COMPARISON is a list of associations that link each triple in snapshot1 with at most one triple in snapshot2, and vice versa. The mapping is one-to-one; a triple in one snapshot cannot be linked with more than one triple in the other snapshot. The mapping is optimal in the sense that any other mapping would have a lower sum of individual match worths.

3. Creating a CMS that Describes the Input Situation

In the final stage of the snapshot comparison process, component NEWCMS creates a one-sense dummy CMS that describes the input situation. The arguments graph consists of all argument names and values given in the input sentence and internally generated argument names that are assigned to words in the triples but not in the input sentence. The effects graph is composed of the associations of triples discovered in COMPARISON.

4. An example

An example will illustrate the procedures DIRECTATTENTION, COMPARISON and NEWCMS. Suppose Moran is initially given the following environment description lists and input sentence:

snapshot1	snapshot2
(RISTIN AT LOCA)	(RISTIN AT LOCB)
(TABLE AT LOCA)	(TABLE AT LOCA)
(BOOK ON TABLE)	(BOOK ON CHAIR)
(PICTURE ON WALL)	(PICTURE ON WALL)
(CLOCK ON MANTLE)	(CLOCK ON MANTLE)
(CHAIR AT LOCB)	(CHAIR AT LOCB)
(LAMP AT LOCC)	(LAMP AT LOCC)

Sentence:

AGENT	Ristin
ACTION	Move
OBJECT	Book
PREP1	From
INDOBJ1	Table
PREP2	To
INDOBJ2	Chair

Process DIRECTATTENTION removes from each snapshot those triples that do not contain at least one word from

the input sentence. The filtered snapshots look like:

snapshot1	snapshot2
(RISTIN AT LOCA)	(RISTIN AT LOCB)
(TABLE AT LOCA)	(TABLE AT LOCB)
(BOOK ON TABLE)	(BOOK ON CHAIR)
(CHAIR AT LOCB)	(CHAIR AT LOCB)

Process COMPARISON compares the two filtered snapshots and makes the following triple associations:

(RISTIN AT LOCA)	--->	(RISTIN AT LOCB)
(TABLE AT LOCA)	--->	(TABLE AT LOCA)
(BOOK ON TABLE)	--->	(BOOK ON CHAIR)
(CHAIR AT LOCB)	--->	(CHAIR AT LOCB)

A one-sense dummy CMS is then created that describes the event:

arguments
|
v

AGENT	Ristin
OBJECT	Book
PREP1	From
INDOBJ1	Table
PREP2	To
INDOBJ2	Chair
C1	Loca
C2	Locb

effects
|
v

(AGENT AT C1)	--->	(AGENT AT C2)
(INDOBJ1 AT C1)	--->	(INDOBJ1 AT C1)
(OBJECT ON INDOBJ1)	--->	(OBJECT ON INDOBJ2)
(INDOBJ2 AT C2)	--->	(INDOBJ2 AT C2)

B. Finding the Semantically Closest Sense in the Verb World

The process COMPARE2CMS finds the semantically closest existing sense of the input root verb by comparing the dummy CMS created by NEWCMS with each sense currently

represented in the CMS for that root verb word. If there is no CMS associated with that root verb, no sense is closest. Otherwise, the semantically closest existing sense is that sense of the root verb whose arguments and effects are most nearly satisfied by the input situation. The program attempts to instantiate the arguments of each candidate sense with values from compatible arguments in the dummy CMS and checks if the effects in the candidate sense are satisfied by the effects in the dummy CMS. A value is assigned that reflects the overall goodness of the match. The highest valued match is chosen as the sense closest to the input situation.

1. Instantiating Arguments

The arguments of a candidate sense are instantiated with values of compatible arguments from the dummy CMS. Identical argument names given as input are compatible, as are the internally generated argument names that are of the form Cn, n=1,2,... For example, the AGENT and OBJECT arguments of a candidate sense may only be instantiated by, respectively, the AGENT and OBJECT arguments of the dummy CMS. Argument names of the form Cn in the candidate CMS may be instantiated by any argument with the name Cn in the dummy CMS. Therefore, the program must iterate through all

such combinations to find the best one. The best set of instantiations cannot be determined until the instantiated candidate sense effects are compared against the dummy effects that describe the input situation.

2. Comparing Effects

A recursive co-routine, MATCH, compares effects in the candidate sense with effects in the dummy CMS and returns a value that indicates how well the instantiated candidate effects match the dummy CMS effects. This process is repeated for each possible instantiation of candidate arguments with dummy CMS argument values. The instantiation with the highest value is returned as the best one.

Each candidate effect is compared with each dummy CMS effect. The goodness value of an instantiation ranges between 0 and 1 and is the product of the goodness values of each linkup between one candidate effect and one dummy CMS effect. Each effect is an ordered pair of ordered triples. The goodness of the linkup is computed by positionally comparing the six values in the two six-tuples. Input argument names, those not of the form Cn, and the relation names in positions two and five must match exactly, otherwise an amount is subtracted from the goodness value of the

linkup. For a match of Cn argument names to occur, the value of the argument in the dummy CMS must be identical or a subset of the argument value in the dummy CMS.

For example, suppose a candidate effect is:

(AGENT AT C1) ---> (AGENT AT C2)

where the argument restrictions are:

AGENT	Human
C1	Location
C2	Location.

Then the dummy CMS effect:

(AGENT AT C42) ---> (OBJECT AT C24)

where

AGENT	Human
OBJECT	Physobj
C42	Loca
C24	Locb

is a better match to the candidate effect than the dummy CMS effect:

(AGENT AT C16) ---> (AGENT PHYSCONT OBJECT)

where

AGENT	Human
C16	Locc
OBJECT	Physobj

since the first effect matches in all but one position, the first position of the second triple, and the second effect matches in only four positions. In the case of ties, the first matched effect is arbitrarily chosen.

Since no semantic information is used to give extra weight to a match of input argument names or relation names over internally generated names, MATCH may not always find the best semantic match. For example, for the same candidate effect shown above, the dummy CMS effect:

(AGENT AT C18) ---> (AGENT PHYSCONT OBJECT)

where

AGENT	Human
C18	Locc
OBJECT	Physobj

is not considered a better linkup than:

(OBJECT AT C20) ---> (OBJECT AT C19)

where

OBJECT	Physobj
C20	Loca
C19	Locd

because they both have two non-matches, even though the first effect seems to have more related to the candidate because they both concern an AGENT, whereas the second effect is dealing with OBJECTs. In actual practice, the linkups that MATCH makes usually agree very well with intuition; when considered in combination with other effect linkups rather than individually, the best <u>overall</u> match filters out most spurious individual linkups.

The result of COMPARE2CMS is a pointer to that sense of the input root verb that the program considers to be semantically closest to the input situation, plus argument instantiation information and flags that indicate which effects in the candidate and dummy CMS were matched and which were not matched.

Moran does not solve the general problem of finding the semantically closest sense. However, the current program employs modules that implement one way of determining environmental differences and one way to select the struc-

ture that is the semantically closest to the input. Although these are very important problems, they are not the crux of this research. Future research will address the problem of making these processes more sophisticated.

C. Modifying the Verb World in Response to Input

Now that Moran has a sense it considers to be semantically close to the input situation, procedure GENERALIZECMS decides how to alter the verb world to incorporate the new knowledge into the existing knowledge of verb senses. The primary concern of this research is the investigation of the set of mechanisms the program needs in order to modify a given graph structure to represent the similarities and differences of meaning under a larger set of input situations.

In response to individual input situations, GENERALIZECMS operates on a single sense of a root verb, the one chosen as closest to the input situation. The learning mechanisms embodied in GENERALIZECMS are called Accretion, Minor Adjustment and Splitting. They respectively build initial graphs for root verbs, generalize argument values and build up the graph structure for many senses of a root verb. The following sections discuss when each type of

learning is invoked and the effects it has on the verb world.

1. Accretion

Accretion has the least drastic effect on existing verb world knowledge. It adds knowledge to the verb world without refining existing information. GENERALIZECMS invokes Accretion when the root verb has not been seen before or when the semantically closest sense of the root verb is not considered sufficiently close in meaning to the input situation.

Accretion is like rote learning in that it adds the dummy CMS that represents the input situation as a completely new sense of the root verb without modifying other senses of the verb, if they exist. If Moran has never experienced this root verb before, the CMS is added to the verb world as the only sense of that root verb. The graphs of the root verb will each consist of one node that contains all the arguments or all the effects. If the root verb has been seen before but no sense is sufficiently close, then the dummy CMS is added as a separate sense to the existing CMS for that root verb. The closest exisiting sense is not considered to be sufficiently close unless it

shares at least a ten percent overlap with the input situation. The new sense has a single argument and effect node and does not share any nodes of the graphs with other senses.

The two Accretion situations are schematically represented below.

If the input situation is represented by:

```
arguments                effects
    |                        |
    v                        v

    X                        Y
```

where X is a set of arguments and Y is a set of effects, then if there is no CMS in the verb world for the input root verb, the addition to the verb world will look like:

```
arguments                effects
    | 1                      | 1
    v                        v

    X                        Y
```

If there does exist a CMS for the root verb that looks like:

```
arguments                effects
    | 1                      | 1
    v                        v

    A                        B
```

where A and B are acyclic directed graphs of arguments and effects, then the resulting state of the verb world will be:

```
arguments        effects          arguments        effects
   | 1              | 1              | 2              | 2
   v                v                v                v

   A                B                X                Y
```

Accretion learning becomes increasingly infrequent as the verb world increases in knowledge, since radically new situations become rarer and rarer. More frequently, modification of the sense closest to the input situation is the proper course of action.

2. Minor Adjustment

Minor Adjustment learning uses world knowledge to generalize the restrictions on argument slots so that a larger class of objects may fill a slot. It allows only semantic, not structural, changes to a CMS. Minor Adjustment occurs when the input situation is identical to an existing sense or when there is a partial match of the arguments in the dummy CMS and the closest sense of the root verb.

The information for broadening the argument slot res-

trictions comes from the set containment information expressed as ISA and SUPERSET/SUBSET links in world knowledge. Each matched argument restriction in the dummy CMS and the closest sense is processed to find the least common superset of the two restrictions.

In order to limit overgeneralization of all argument restrictions to OBJECT, the root node of the world knowledge, procedure GENERALIZECMS always asks procedure ALLOWGEN for permission to generalize an argument restriction. How to limit generalization is a difficult problem. Rosch [Rosch 76a, 76b] suggests levels of abstraction for categories that yield the most information for the least cognitive effort. It might be interesting to use these categories as generalization limits.

The current implementation uses a simpler method. ALLOWGEN will allow generalization of two argument restrictions to their least common superset if:

1) no more than two ISA or SUPERSET/SUBSET links had to be traversed to find the least common superset or

2) the least common superset is not OBJECT, PHYSOBJ (physical object) or ABSTRACTOBJ (abstract object).

Thus, generalization to the highest level categories in the world knowledge tree is allowed only if both restrictions are already "close" to these categories: no more than two links away. Any number of links may be traversed to get the least common superset if the least common superset is not OBJECT, PHYSOBJ or ABSOBJ.

Suppose the verb world contains the following CMS for the verb "move" after having experienced one example of "Ristin moved the chair".

```
arguments
    | 1
    v

AGENT      Ristin

OBJECT     Chair

C1         Loca

C2         Locb

effects
    | 1
    v

(AGENT AT C1) ---> (AGENT AT C2)

(OBJECT AT C1) ---> (OBJECT AT C2)

(AGENT PHYSCONT OBJECT) ---> (AGENT PHYSCONT OBJECT)
```

The next input situation is:

Snapshot1	Snapshot2
(FIGARO AT LOCA)	(FIGARO AT LOCD)
(TABLE AT LOCD)	(TABLE AT LOCD)
(FIGARO PHYSCONT TABLE)	(FIGARO PHYSCONT TABLE)

Sentence:

AGENT	Figaro
ACTION	Move
OBJECT	Table

The dummy CMS created to describe the input is:

arguments
|
v

AGENT	Figaro
OBJECT	Table
C3	Loca
C4	Locd

effects
|
v

(AGENT AT C3) ---> (AGENT AT C4)

(OBJECT AT C3) ---> (OBJECT AT C4)

(AGENT PHYSCONT OBJECT) ---> (AGENT PHYSCONT OBJECT)

The closest sense of the root verb "move" is the one sense currently associated with "move". It is an excellent

match except for the restrictions on the argument slots. Using the world knowledge base shown in Section 3.C, Minor Adjustment generalizes the closest sense in the CMS for "move" to:

arguments
|
v

AGENT	Human(2): Figaro(1), Ristin(1)
OBJECT	Furniture(2): Table(1), Chair(1)
C1	Loca(2)
C2	Location(2): Locd(1), Locb(1)

effects
|
v

(AGENT AT C1) ---> (AGENT AT C2)

(OBJECT AT C1) ---> (OBJECT AT C2)

(AGENT PHYSCONT OBJECT) ---> (AGENT PHYSCONT OBJECT)

There is still only one sense of "move" and it accounts for both inputs the program has experienced of "moving".

The closest sense and the current input most commonly match very well. In the example above, if some of the arguments and effects had not matched and/or ALLOWGEN had disallowed the generalization of at least one argument slot, Splitting as well as Minor Adjustment would need to

be invoked.

3. Splitting

Splitting makes structural changes in the CMS graph structure. It reorganizes the arguments and effects graphs so that matched arguments and effects become parent nodes to offspring nodes that are the arguments and effects not matched in the closest sense and the dummy CMS. Splitting may operate on both the arguments or effects graphs or only on the arguments graph or only on the effects graph.

GENERALIZECMS traverses the closest sense of the root verb CMS and the dummy CMS, marking each argument and each effect as to whether or not it is to be split off. Arguments are marked to be split off if either they are not matched or ALLOWGEN will not authorize the generalization. Effects are marked to be split off if they are not matched perfectly. If any argument or effect is marked to be split off, procedure SPLITCMS is invoked. SPLITCMS modifies the graphs so that existing difference and similarity relationships are maintained and the new information is incorporated to reflect similarities to and differences from the existing knowledge.

Suppose the verb world contains the generalized one-

sense CMS for "move" derived in the previous section. The program then experiences:

Snapshot1	Snapshot2
(TIM AT LOCB)	(TIM AT LOCA)
(PICTURE AT LOCB)	(PICTURE AT LOCD)
(TIM PHYSCONT PICTURE)	

Sentence:

AGENT Tim

ACTION Move

OBJECT Picture

This input is described by the dummy CMS:

arguments
|
v

AGENT	Tim
OBJECT	Picture
C5	Locb
C6	Loca
C7	Locd

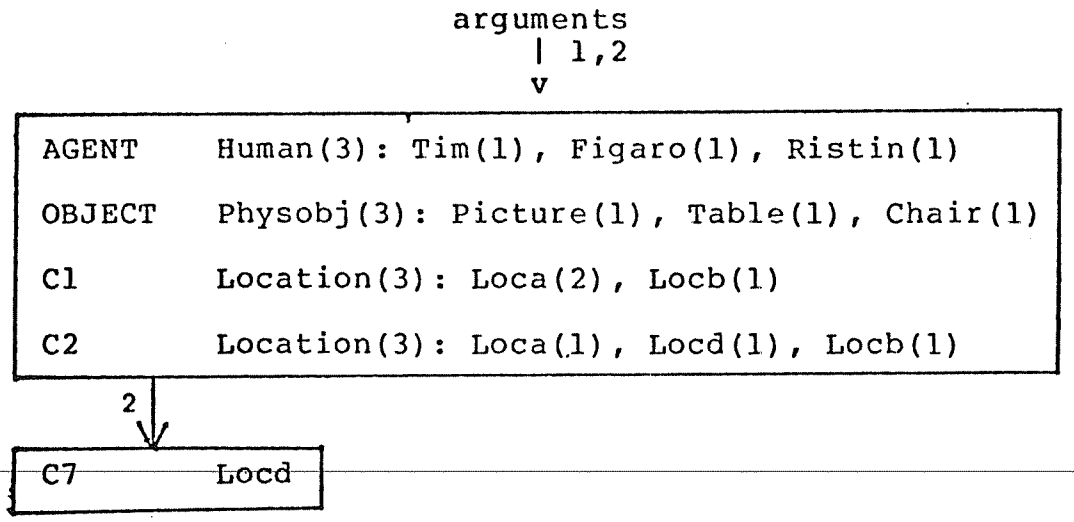
effects
|
v

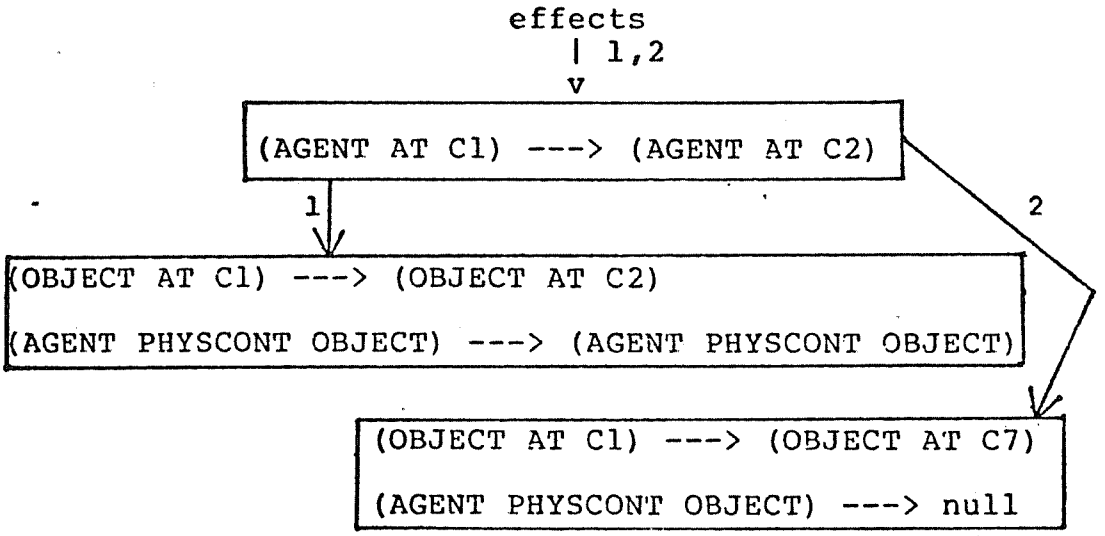
(AGENT AT C5) ---> (AGENT AT C6)

(OBJECT AT C5) ---> (OBJECT AT C7)

(AGENT PHYSCONT OBJECT) ---> null

The closest sense is the only existing sense of "move". However, there is only a partial match of both arguments and effects. A combination of Minor Adjustment and Splitting results in the following two-sense CMS for "move".

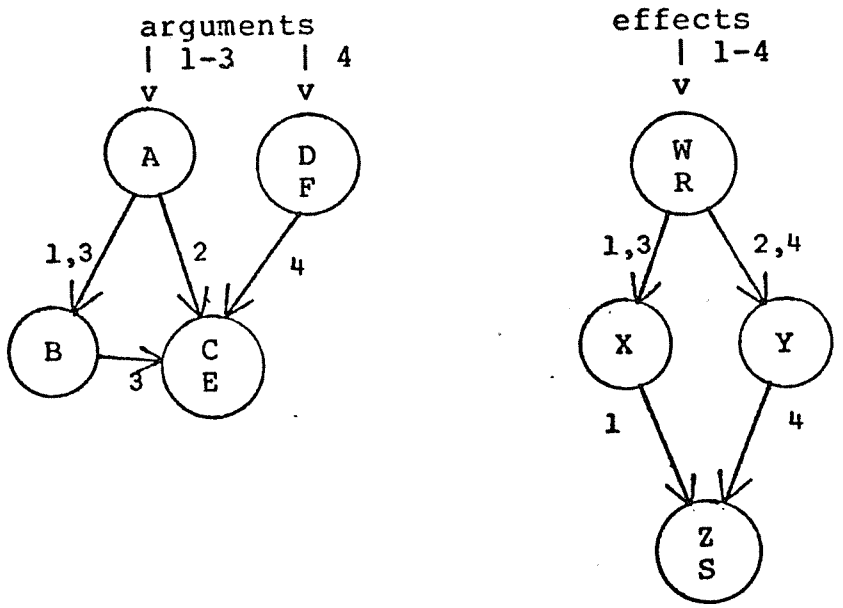




Minor Adjustment generalizes argument C1 to Location from the specific location Loca. Splitting puts C7, the argument required by sense two but not sense one, in an offspring node of the node required by both senses. The effects are also split into a parent node shared by both senses and offspring nodes for each sense.

A more complex example of splitting CMSs is schematically depicted below. Capital letters represent sets of arguments or effects.

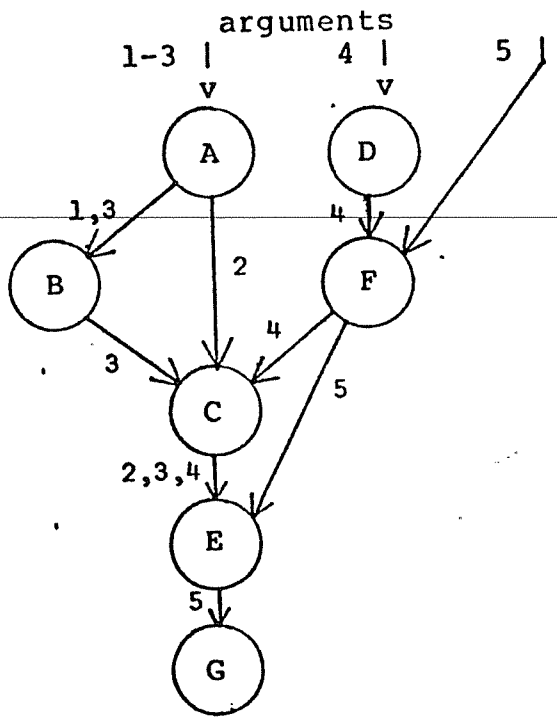
Suppose the verb world contains the following labelled directed acyclic graph for some root verb.

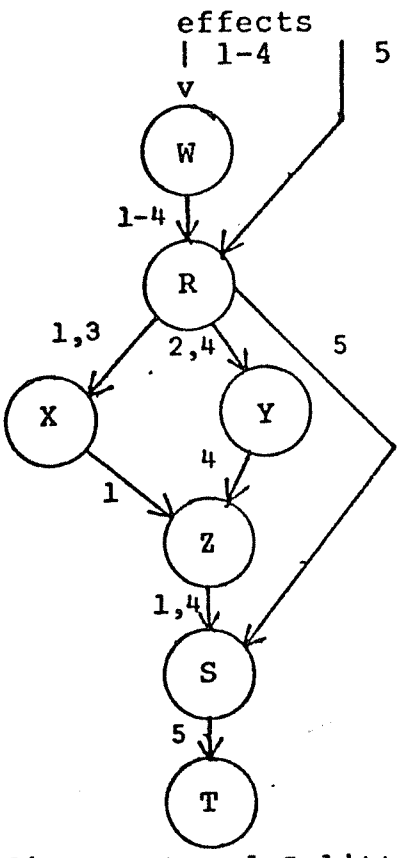


The dummy CMS:

describes the input situation. Sense four of the root verb is determined to be the semantically closest sense. Sense

four and the dummy CMS share sets of arguments E and F, and sets of effects R and S. The CMS produced by GENERALIZECMS for the root verb will have five senses:





Accretion, Minor Adjustment and Splitting are all invoked in response to individual input situations. The changes they are allowed to make to the CMS structures are limited unless there is a very close match between the current input and an existing sense of the input root verb. Similarities between different root verbs are not recognized since the three learning mechanisms may only operate on the CMS associated with the input root verb.

D. Modifying the Verb World Across CMS Boundaries

The fourth learning mechanism, <u>Global Building Block</u>

Extraction, operates across CMS boundaries and identifies those effects that are regularly used in describing the meaning of many verb senses. Each such effect and its restrictions is called a Global Building Block (GBB).

The purpose of a GBB is to:

1) Allow the verb world representation to recognize similiar components used in different verb senses within a CMS as well as across CMS boundaries.

2) Facilitate a more compact memory representation since each GBB is stored only once, rather than in each sense where it is used.

3) Be the basis for future recognition of larger chunks of effects used in representing many verb senses.

4) Aid in the identification of primitive verb meaning components. Those configurations of GBBs that are repeatedly used to describe many verb senses may help shed some light on the question of the existance of primitive meaning components.

The GBB Extraction process EXTRACTGBB identifies semantically equivalent effects used in different verb

senses and decides which, if any, are to be used as GBBs. Two effects are semantically equivalent if:

1) each input argument name is identical and the restrictions on their arguments are the same

2) relation names are identical and

3) the restrictions on internally generated argument names are identical.

For example, the two effects:

(AGENT AT C7) ---> (AGENT AT C8)

(AGENT AT C18) ---> (AGENT AT C19)

where

AGENT	Human
C7	Location
C8	Location
C18	Location
C19	Location

are semantically equivalent. But the two effects:

(AGENT AT C7) ---> (AGENT AT C8)

(AGENT AT C18) ---> (AGENT AT C18)

violate rule three and are not equivalent since the first

effect assumes two different locations, C7 and C8, and the second effect assumes only one location, C18. The two effects:

(OBJECT AT C20) ---> (OBJECT ON INDOBJ)

(OBJECT AT C3) ---> (OBJECT ON INDOBJ)

where the first effect requires:

OBJECT	Physobj
INDOBJ	Furniture
C20	Location

and the second effect requires:

OBJECT	Physobj
INDOBJ	Physobj
C3	Location

are not equivalent because because the restrictions on the INDOBJ argument do not match and violate rule one.

It is too time-consuming to completely traverse each sense of each CMS looking for semantically equivalent effects since graphs tend to be both deep and bushy. Procedure TRAVERSECHANGES instead traverses the effects graph of eash sense only to a depth of three levels. It computes how many times each triple and its semantic equivalents are

seen. Procedure PICKCANDIDATE decides which of these triples are to be used as triples in potential GBB effects. The current criterion is that a triple must occur in at least one third of the total number of triples traversed. If TRAVERSECHANGES traverses one hundred triples in the first three levels of the verb world, then a triple must occur at least 34 times in order to be chosen. The result of PICKCANDIDATE is a list of those triples, not effects, that are frequently used in many verb senses.

Procedure CANDIDATEFREQ then completely traverses the graph of each sense of each CMS, counting the occurrence of each possible combination of candidate triples. Even though two passes must be made through the verb world, one by TRAVERSECHANGES and one by CANDIDATEFREQ, the first pass is only to a depth of three levels. The second pass completely traverses each graph, but the amount of computation is drastically reduced since most triples will not match in the first few positions and further matching within that effect may be abandoned. The result of CANDIDATEFREQ is a list of candidate effects composed of all possible pairs of triples chosen by PICKCANDIDATE, a count of how often each candidate effect was used in all senses in the verbworld and a list of pointers to each matched effect.

Procedure PICKGBB uses the information produced by CANDIDATEFREQ to select those candidate effects that are to be designated GBBs. The current criterion is that a candidate effect must participate in at least one third of the verb senses represented in the verb world.

Suppose, for example, that PICKCANDIDATE selects triples a, b and c. Each of these triples comprises at least one third of all triples in the first three levels of the verb world effects graphs. CANDIDATEFREQ traverses each complete path through each effects graph and compiles frequency and occurrence information on the following candidate effects:

a ---> a

a ---> b

a ---> c

b ---> a

b ---> b

b ---> c

c ---> a

c ---> b

c ---> c

Each candidate effect used in at least one third of the senses in the verb world is designated a GBB. Examples of

the actual GBBs selected by the program from a rich verb world are shown in Chapter 6.

The current implementation only identifies the effects to be used as GBBs and lists those verb senses that use each GBB. A future version of Moran will reorganize the verb world with respect to these GBBs and filter all further input through them: each input description will be inspected to see if any GBBs are used in the description. If so, the incorporation of the input knowledge into the verb world will be done with respect to the GBBs so its similarities to all other verb senses are reflected.

GBB Extraction is currently invoked by the trainer at any arbitrary time. A more interesting approach is to have Moran decide when GBB Extraction should be done. Possible stimuli are:

1) Memory is getting full; GBB Extraction may result in memory consolidation since each GBB is stored only once.

2) A sufficient amount of information is now available; more global generalization and organizational decisions may be made more safely.

A discussion of the implementation and extensive exam-

ples of Accretion, Minor Adjustment, Splitting and GBB Extraction are given in Chapter 5.

CHAPTER 5

IMPLEMENTATION AND RESULTS

This chapter describes the implementation of Moran and gives examples of the results to date.

A. Implementation

Moran is implemented in SIMULA 67 on a UNIVAC 1110. It is approximately 3700 lines long. Each major component is implemented as a class object that contains both data and the procedures that may operate on the data. The major classes and procedures are listed below.

1) Class SEMANTICNET creates a semantic net that describes both the world knowledge base and input descriptions of the environment snapshots.

2) Class SENTENCEPARSE inputs the parsed natural language sentence.

3) Procedures DIRECTATTENTION, COMPARISON and DIFFERENCES operate on the input to filter the environment snapshots and form effects by linking a triple in the first snapshot with a triple in the second snapshot.

4) Class NEWCMS creates a CMS structure that

represents a new input verb and contains all procedures that modify a CMS structure.

5) Procedure COMPARE2CMS compares an existing CMS associated with a root verb with a dummy CMS that describes another instance of that root verb and returns the sense that it considers to be semantically closest to the input situation.

6) Procedure ADDSENSE adds a completely new sense to an existing CMS when the closest sense is not close enough to be modified.

7) Procedure GENERALIZECMS invokes Accretion, Minor Adjustment, and/or Splitting as required to modify the closest sense of the input root verb to account for previous as well as current input.

8) Procedure EXTRACTGBB inspects the entire verb world and selects commonly used effects that are to be used as GBBs in the verb world.

Moran invokes the above classes and procedures approximately in the order they are listed. SEMANTICNET is first invoked to read in and build a semantic net that represents the world knowledge. The first seven classes and procedures are then invoked for each learning cycle. A learn-

ing cycle consists of inputting a two-snapshot sequence and a parsed natural language sentence and then incorporating the information embodied in the input into the verb world.

1. Processing the Input

During each learning cycle the input is first processed. SEMANTICNET is invoked twice to build semantic nets that represent each snapshot. SENTENCEPARSE inputs the parsed natural language sentence. DIRECTATTENTION filters each snapshot by keeping for further consideration only those triples that contain at least one word contained in SENTENCEPARSE. COMPARISON computes goodness values for potential linkups of triples in the first snapshot with triples in the second snapshot. DIFFERENCES selects the best linkup and computes additional information needed to create effects from triples. Finally, NEWCMS creates a dummy CMS that represents the action that took place in the input sequence.

2. Finding the Closest Existing Sense

Once the input is describwed as a sense of a CMS, it may be compared with other information in the verb world. Procedure COMPARE2CMS compares the input described in the

dummy CMS with each sense associated with the CMS for the input root verb word. The selection of the existing sense that is closest to the input situation is done both structurally and semantically. The structurally closest sense is the one whose graphs may be modified in the least drastic way to account for the input situation. This sense is found by choosing the existing graph that maximally covers the input graph. The semantic selection is done by using the set containment information in world knowledge to select the maximally covering graph that describes objects that are the most semantically similiar to those seen in the input. COMPARE2CMS returns the sense it considers to be the best description of the input situation along with information about how well each linkup of arguments and effects in the dummy CMS and the closest sense matched.

3. Modifying Individual CMSs

If no CMS exists for the input verb word, ADDSENSE will create an entry in the verb world for the input verb word that points to the dummy CMS. If the closest sense returned by COMPARE2CMS is a very poor match, then ADDSENSE adds a pointer to the dummy CMS at the root node level of the CMS for the input root verb.

When Moran considers the closest sense chosen by COMPARE2CMS to be close enough to warrant modification, procedure GENERALIZECMS modifies that sense in the least drastic way so that it describes the new as well as the old input. First it tries Minor Adjustment to loosen the restrictions on the arguments to the least common superset of two arguments. This generalization is allowed only if the least common superset is semantically close to the two instances, as defined by world knowledge. If Minor Adjustment is not sufficient to make the closest sense account for the input, then Splitting will alter the CMS graph strucures so that matched effects or arguments generalized by Minor Adjustment in each node are shared by both senses and the unmatched information resides in offspring nodes that are local to the individual senses.

These procedures are repeated for each learning cycle. The learning cycles terminate whenever no more input situations are provided or when the trainer invokes GBB Extraction to look for similarities across CMS boundaries.

4. Finding Commonalities in the Verb World

The purpose of EXTRACTGBB is to identify information shared by many verb senses across CMS boundaries. If Moran

is ever to be able to recognize synonyms and/or classes of semantically similar verbs, it must be able to recognize how the meanings of different verb words are similar and different as well as how different senses of the same verb word are similar and different. The problem of recognizing similarities across CMS boundaries is combinatorially large since each effect must be compared with each other effect. Moran limits this combinatorial explosion by limiting the depth of initial searches for equivalent effects.

The first pass through the verb world searches for equivalent triples rather than equivalent effects and is only allowed to look to a depth of three levels in each effects graph. The most frequently used triples are then linked in all possible combinations to form candidate GBBs. Only the final pass through the verb world that looks for matches to these candidate GBBs traverses each path through each effects graph to its maximum depth. The frequency and location of matches to the candidate GBBs are recorded. Then the most frequently used candidate GBBs are chosen as GBBs.

B. Results to Date

This section presents examples of CMS graphs that

Moran has learned. The test data Moran has been given consists of seventeen senses of four different root verbs. The environment snapshots and corresponding natural language sentences are shown below. Moran was presented with four senses of the root verb "throw", six of "carry", eight of "move" and three of "buy".

The abbreviation PHYSCONT stands for "physical contact" and HAP for "has as part".

"Figaro bought the picture from Ristin"

Snapshot1	Snapshot2
(FIGARO PHYSCONT MONEY)	(FIGARO PHYSCONT PICTURE)
(FIGARO AT LOCB)	(FIGARO AT LOCB)
(RISTIN AT LOCA)	(RISTIN AT LOCA)
(RISTIN PHYSCONT PICTURE)	(RISTIN PHYSCONT MONEY)

Sentence:

AGENT	Figaro
ACTION	Buy
OBJECT	Picture
PREP	From
INDOBJ	Ristin

"Ristin bought the table from Figaro"

Snapshot1	Snapshot2
(RISTIN PHYSCONT MONEY)	(FIGARO PHYSCONT MONEY)
(RISTIN AT LOCA)	(RISTIN AT LOCA)
(FIGARO AT LOCB)	(FIGARO AT LOCB)
(TABLE AT LOCC)	(TABLE AT LOCC)

Sentence:

AGENT	Ristin
ACTION	Buy
OBJECT	Table
PREP	From
INDOBJ	Figaro

"Ristin bought the book"

Snapshot1	Snapshot2
(RISTIN PHYSCONT MONEY)	(RISTIN AT LOCD)
(RISTIN AT LOCB)	(RISTIN AT LOCD)
(BOOK AT LOCC)	(BOOK AT LOCC)

Sentence:

AGENT Ristin

ACTION Buy

OBJECT Book

"Figaro carried the book"

Snapshot1	Snapshot2
(FIGARO AT LOCA)	(FIGARO AT LOCB)
(BOOK AT LOCA)	(BOOK AT LOCB)
(FIGARO PHYSCONT BOOK)	(FIGARO PHYSCONT BOOK)

Sentence:

AGENT Figaro

ACTION Carry

OBJECT Book

"Figaro carried the book to the table"

Snapshot1	Snapshot2
(FIGARO AT LOCC)	(FIGARO AT LOCD)
(BOOK AT LOCC)	(BOOK AT LOCD)
(FIGARO PHYSCONT BOOK)	(FIGARO PHYSCONT BOOK)
(FIGARO HAP MOUSTACHE)	(FIGARO HAP MOUSTACHE)

Sentence:

AGENT	Figaro
ACTION	Carry
OBJECT	Book
PREP	To
INDOBJ	Table

"Figaro carried the pin with the tweezers"

Snapshot1	Snapshot2
(FIGARO AT LOCA)	(FIGARO AT LOCB)
(PIN AT LOCA)	(PIN AT LOCB)
(TWEEZERS AT LOCA)	(TWEEZERS AT LOCB)
(TWEEZERS PHYSCONT PIN)	(TWEEZERS PHYSCONT PIN)
(FIGARO PHYSCONT TWEEZERS)	(FIGARO PHYSCONT TWEEZERS)

Sentence:

AGENT	Figaro
ACTION	Carry
OBJECT	Pin
PREP	With
INDOBJ	Tweezers

"Ristin carries a wallet"

Snapshot1	Snapshot2
(RISTIN AT LOCA)	(RISTIN AT LOCA)
(WALLET AT LOCA)	(WALLET AT LOCA)
(RISTIN PHYSCONT WALLET)	(RISTIN PHYSCONT WALLET)

Sentence:

AGENT	Ristin
ACTION	Carry
OBJECT	Wallet

"Ristin carried on a conversation"

Snapshot1	Snapshot2
(RISTIN AT LOCA)	(RISTIN AT LOCA)
(FIGARO AT LOCB)	(FIGARO AT LOCB)
(RISTIN TALKTO FIGARO)	(RISTIN TALKTO FIGARO)

Sentence:

AGENT	Ristin
ACTION	Carry
PREP	On
OBJECT	Conversation

"Figaro carries a torch"

Snapshot1	Snapshot2
(FIGARO AT LOCB)	(FIGARO AT LOCB)
(FIGARO LOVE RISTIN)	(FIGARO LOVE RISTIN)

Sentence:

AGENT Figaro

ACTION Carry

OBJECT Torch

"Figaro threw the ball"

Snapshot1	Snapshot2
(FIGARO AT LOCA)	(FIGARO AT LOCA)
(BALL AT LOCA)	(BALL AT LOCB)
(FIGARO PHYSCONT BALL)	

Sentence:

AGENT Figaro

ACTION Throw

OBJECT Ball

"Figaro threw the ball to Ristin"

Snapshot1	Snapshot2
(FIGARO AT LOCB)	(FIGARO AT LOCB)
(RISTIN AT LOCA)	(RISTIN AT LOCA)
(FIGARO PHYSCONT BALL)	(RISTIN PHYSCONT BALL)
(DOLL AT LOCB)	(DOLL AT LOCA)

Sentence:

AGENT	Figaro
ACTION	Throw
OBJECT	Doll
PREP	To
INDOBJ	Ristin

"Ristin threw the book at Figaro"

Snapshot1	Snapshot2
(RISTIN AT LOCA)	(RISTIN AT LOCA)
(FIGARO AT LOCB)	(FIGARO AT LOCB)
(RISTIN PHYSCONT BOOK)	(BOOK AT LOCC)
(BOOK AT LOCA)	

Sentence:

AGENT	Ristin
ACTION	Throw
OBJECT	Book
PREP	At
INDOBJ	Figaro

"Ristin threw a tantrum"

Snapshot1	Snapshot2
(RISTIN AT LOCA)	(RISTIN AT LOCA)
(RISTIN HAVE CALM)	(RISTIN HAVE ANGER)
	(RISTIN HAVE TEARS)

Sentence:

AGENT	Ristin
ACTION	Throw
OBJECT	Tantrum

"Ristin moved the book"

Snapshot1	Snapshot2
(RISTIN AT LOCA)	(RISTIN AT LOCB)
(BOOK AT LOCA)	(BOOK AT LOCB)
(RISTIN PHYSCONT BOOK)	(RISTIN PHYSCONT BOOK)

Sentence:

AGENT Ristin

ACTION Move

OBJECT Book

"Figaro moved the pencil"

Snapshot1	Snapshot2
(FIGARO AT LOCB)	(FIGARO AT LOCC)
(PENCIL AT LOCB)	(PENCIL AT LOCC)
(FIGARO PHYSCONT PENCIL)	(FIGARO PHYSCONT PENCIL)

Sentence:

AGENT Figaro

ACTION Move

OBJECT Pencil

"Figaro moved the table"

Snapshot1	Snapshot2
(FIGARO AT LOCA)	(FIGARO AT LOCB)
(TABLE AT LOCA)	(TABLE AT LOCC)
(FIGARO PHYSCONT TABLE)	

Sentence:

AGENT Figaro

ACTION Move

OBJECT Table

"Ristin moved the pencil"

Snapshot1	Snapshot2
(RISTIN AT LOCA)	(RISTIN AT LOCB)
(PENCIL AT LOCA)	(PENCIL AT LOCB)
(RISTIN PHYSCONT PENCIL)	(RISTIN PHYSCONT PENCIL)
(RISTIN HAP PONYTAIL)	(RISTIN HAP PONYTAIL)

Sentence:

AGENT Ristin

ACTION Move

OBJECT Pencil

"Ristin moved the pencil"

Snapshot1	Snapshot2
(RISTIN AT LOCB)	(RISTIN AT LOCC)
(RISTIN PHYSCONT PENCIL)	(RISTIN PHYSCONT PENCIL)

Sentence:

AGENT	Ristin
ACTION	Move
OBJECT	Pencil

"Ristin moved the book to the table"

Snapshot1	Snapshot2
(RISTIN AT LOCA)	(RISTIN AT LOCC)
(BOOK AT LOCA)	(BOOK AT LOCB)
(RISTIN PHYSCONT BOOK)	(BOOK PHYSCONT TABLE)
(TABLE AT LOCB)	(TABLE AT LOCB)

Sentence:

AGENT	Ristin
ACTION	Move
OBJECT	Book
PREP	To
INDOBJ	Table

"Ristin moved to adjourn"

Snapshot1	Snapshot2
(RISTIN AT LOCA)	(RISTIN AT LOCA)
(PEOPLE IN ROOM)	(RISTIN HOLD GAVEL)
(RISTIN HOLD GAVEL)	

Sentence:

AGENT Ristin

ACTION Move

OBJECT Adjourn

"Figaro moved Ristin"

Snapshot1	Snapshot2
(FIGARO AT LOCA)	(FIGARO AT LOCA)
(RISTIN AT LOCB)	(RISTIN AT LOCB)
(RISTIN MENTAL NEUTRAL)	(RISTIN MENTAL HAPPY)

Sentence:

AGENT Figaro

ACTION Move

OBJECT Ristin

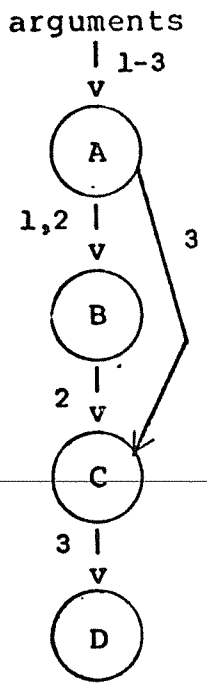
From the input above Moran inferred a verb world that consists of eight non-connected directed acyclic graphs, two for each root verb, and selected a set of GBBs. These results are shown below. The surface sentences that resulted in each sense are listed before each graph. The program required 32K and 28.5 seconds for execution. Garbage collection was performed 36 times.

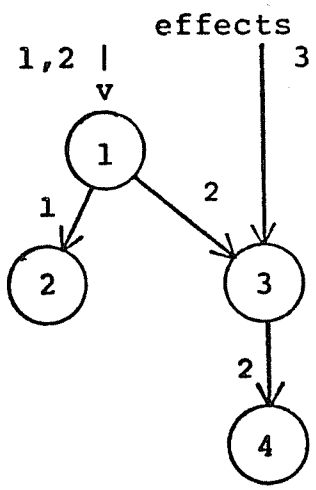
Moran inferred the following three-sense CMS from three instances of the verb "buy":

Sense1: "Figaro bought the picture from Ristin"

Sense2: "Ristin bought the table from Figaro"

Sense3: "Ristin bought the book"





where the arguments are:

A:	Agent	Human
	Object	Physobj
	C1	Money
	C2	Location

B:	Prep	From
	Indobj	Human
	C3	Location
C:	C5	Location
D:	C10	Locb

and the effects are:

1: Indobj at C3 ---> Indobj at C3

2: Agent physcont C1 ---> Agent physcont Object
Agent at C2 ---> Agent at C2
Indobj physcont Object ---> Indobj physcont C1

3: Agent physcont C1 ---> Agent at C5
Agent at C5 ---> null
Object at C2 ---> Object at C2

4: null ---> Indobj physcont C1

Sense one describes a human Indobj remaining at a location during the action. A human Agent changes from being in physical contact with money to being in physical contact with an Object that is a physical object. The Agent remains at a location that is different from the location

of the Indobj. The Indobj changes from being in physical contact with the Object to being in physical contact with money.

Moran viewed six instances of "carrying" and inferred the following six sense CMS:

Sense1: "Figaro carried the book"

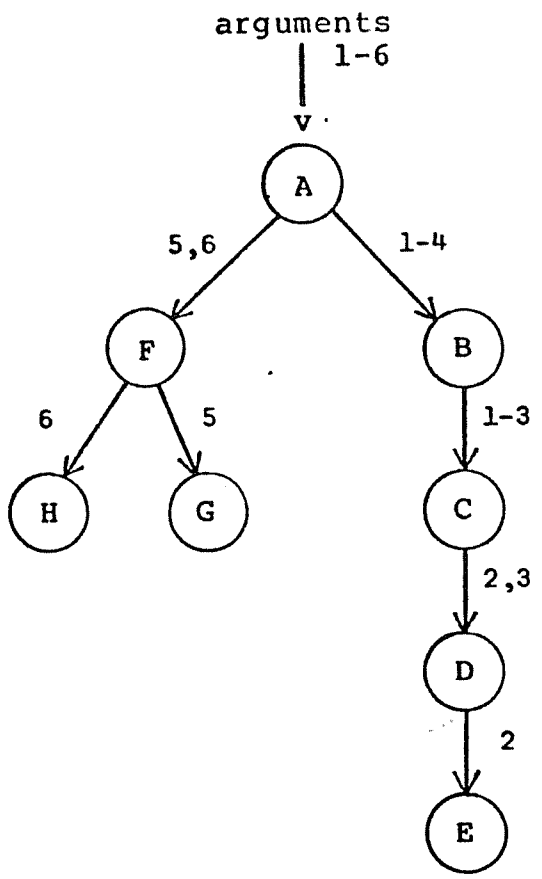
Sense2: "Figaro carried the book to the table"

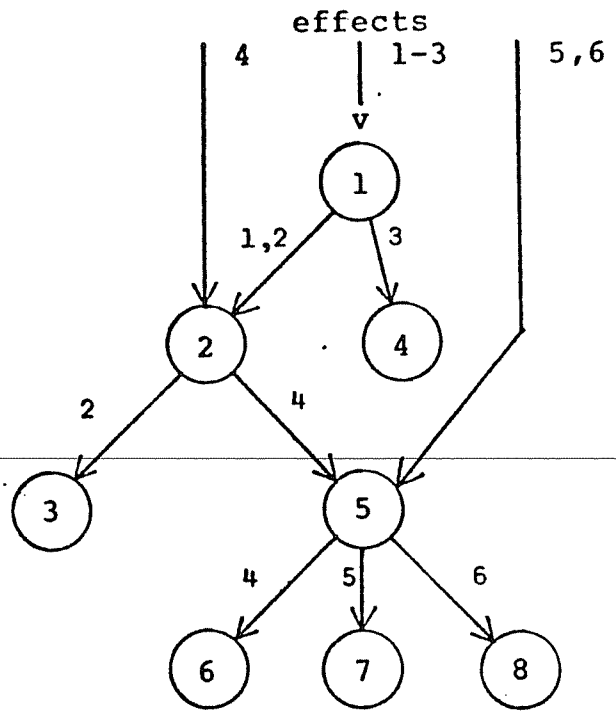
Sense3: "Figaro carried the pin with the tweezers"

Sense4: "Ristin carries a wallet"

Sense5: "Ristin carried on a conversation"

Sense6: "Figaro carries a torch"


arguments
1-6
A
5,6
1-4
F
B
6
5
1-3
H
G
C
2,3
D
2
E



where the arguments and effects are:

A:	Agent	Human
	C1	Location
B:	Object	Physobj
C:	C2	Location
D:	Prep	Preposition
	Indobj	Physobj
E:	C5	Moustache
F:	C10	Human
G:	Prep	On

Object Conversation

H: Object Torch

1: Agent at C1 ---> Agent at C2
Object at C1 ---> Object at C2

2: Agent physcont Object ---> Agent physcont Object

3: Indobj at C2 ---> Indobj at C2
Agent hap C5 ---> Agent hap C5

4: Indobj at C1 ---> Indobj at C2
Agent physcont Indobj ---> Agent physcont Indobj
Indobj physcont Object ---> Indobj physcont Object

5: Agent at C1 ---> Agent at C1

6: Object at C1 ---> Object at C1

7: Agent talkto C10 ---> Agent talkto C10

8: Agent love C10 ---> Agent love C10

The first three senses describe variations on the sense of "carrying" where the Agent moves the Object from one location to another. Sense four describes a sense of carrying where the Agent has a physical object on his or

her person. Senses five and six are idiomatic and were given to Moran to demonstrate how radically different senses of the same root verb are incorporated into the graph structure.

The eight instances of "moving" resulted in the following seven sense CMS since the first two snapshot sequences mapped to the same structure.

Sense1: "Ristin/Figaro moved the book/pencil"

Sense2: "Figaro moved the table"

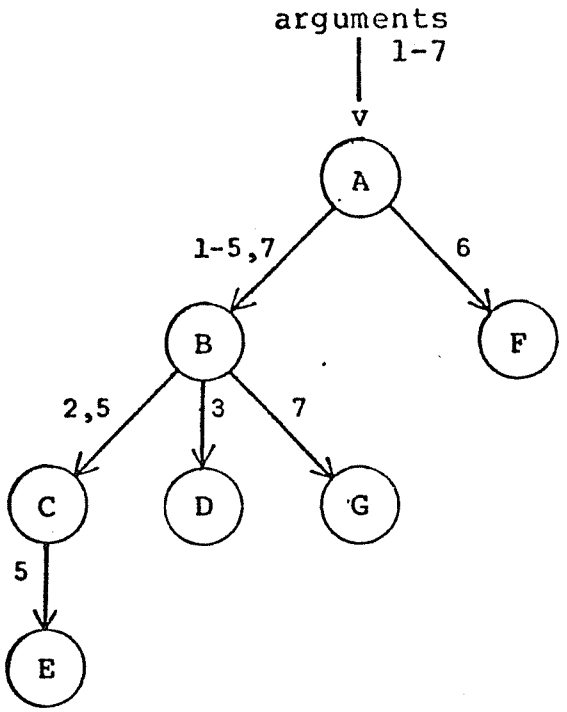
Sense3: "Ristin moved the pencil"

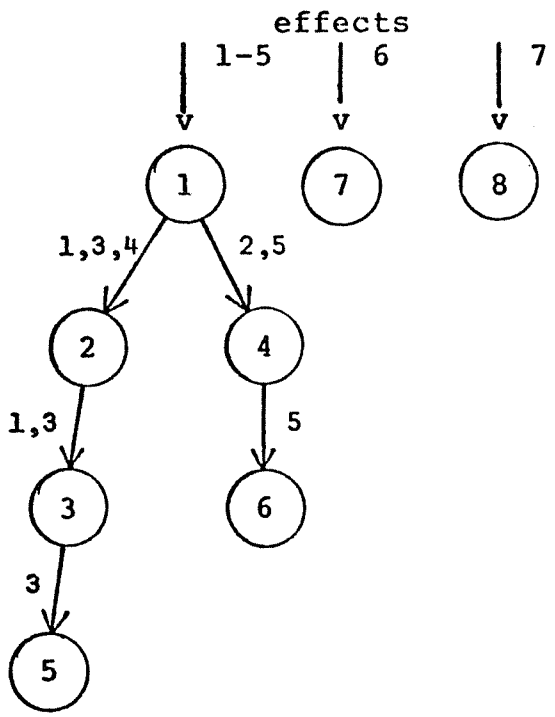
Sense4: "Ristin moved the pencil"

Sense5: "Ristin moved the book"

Sense6: "Ristin moved to adjourn"

Sense7: "Figaro moved Ristin"





The arguments and effects are:

A:	Agent	Human
	C1	Location
B:	Object	Physobj
	C2	Location
C:	C7	Location
D:	C10	Ponytail
E:	Prep	Preposition
	Indobj	Table
F:	Object	Adjourn
	C17	Gavel
G:	C20	Neutral
	C21	Happy

1: Agent at C1 ---> Agent at C2

2: Agent physcont Object ---> Agent Physcont Object

3: Object at C1 ---> Object at C2

4: Object at C1 ---> Object at C7
Agent physcont Object ---> null

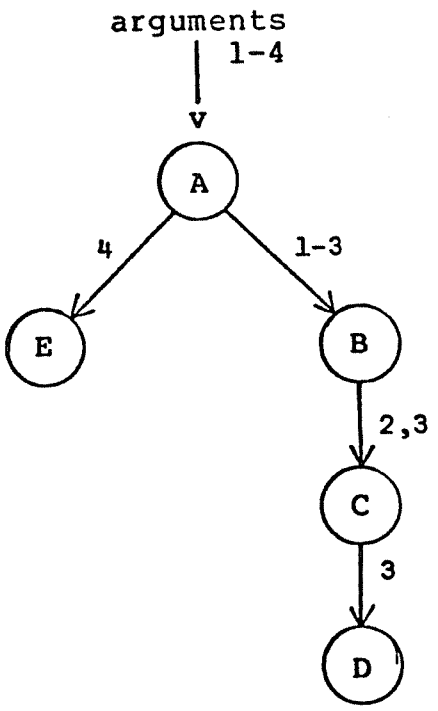
5: Agent hap C10 ---> Agent hap C10

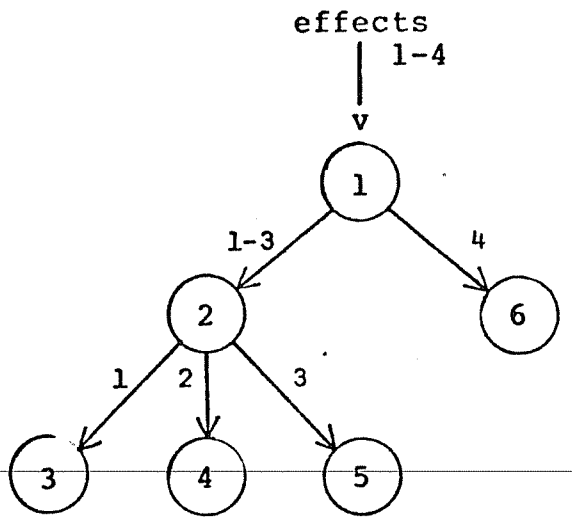
6: Indobj at C7 ---> Indobj at C7

null ---> Object physcont Indobj

Senses one through five are very similar and represent the sense of "move" where the Agent changes location with the Object. Senses six and seven are idiomatic and were given to Moran to show how radically different senses are incorporated into the graph structure.

The following four sense CMS was inferred from the four instances of "throwing".





The arguments and effects are:

A:	Agent	Human
	C1	Location
B:	Object	Physobj
	C2	Location
C:	Prep	Preposition
	Indobj	Human
D:	C6	Locb
E:	Object	Tantrum
	C9	Calm
	C10	Anger
	C11	Tears

1: Agent at C1 ---> Agent at C1

2: Object at C1 ---> Object at C2

3: Agent physcont Object ---> null

4: Indobj at C2 ---> Indobj at C2
Agent physcont Object ---> Indobj physcont Object

5: Indobj at C6 ---> Indobj at C6
Agent physcont Object ---> null

6: Agent have C9 ---> Agent have C10
null ---> Agent have C11

Senses one through three represent variations on the sense of "throw" where the Agent causes the Object to change location. Sense four represents a change in the mental state of the Agent.

The graphs that Moran inferred for the twenty one senses of four root verbs are bushy as well as deep relative to the amount of data presented. This is because snapshots of the same action may vary in some details and Moran always represents all the information in the filtered snapshots. For example, the only difference between sense two and sense three of "carry" is that in the second sense

the Agent is shown as having a moustache. This problem will be compounded as Moran is presented with more snapshots of actions since the chances of snapshots of the same action being exactly the same is very small. A future version of Moran will have to include mechanisms for deciding when groups of senses that share enough "important" information are actually representations of the same general sense of a root verb.

The following four effects were chosen as GBBs.

1. AGENT PHYSCONT OBJECT ---> AGENT PHYSCONT OBJECT

AGENT	Human
OBJECT	Physobj

2. AGENT AT CASEA ---> AGENT AT CASEA

AGENT	Human
CASEA	Location

3. AGENT AT CASEA ---> AGENT AT CASEB

AGENT	Human
CASEA	Location
CASEB	Location

4. OBJECT AT CASEA ---> OBJECT AT CASEB

OBJECT	Physobj

CASEA Location

CASEB Location

The first GBB describes a human agent remaining in contact with an object that is a physical object. It was used in seven of the twenty one senses. The second GBB describes a human agent staying in the same location, while the third GBB describes a human agent changing location. The second GBB was used ten times, the third nine times. The final GBB describes an object that is a physical object changing location. It was used eleven times. The next most frequently used effect was seen only four times.

CHAPTER 6

LIMITATIONS, EXTENSIONS AND CONCLUSIONS

This research limits its scope to investigating learning overt physical actions that occur in the environment when a verb "takes place". The current implementation and the theory are both limited in certain important ways. Most of the limitations are highly interrelated. This chapter identifies the major limitations and suggests how they might be remedied, discusses the significance of Moran's results, and suggests directions for future research.

A. Implementation Limitations

1. Two-Snapshot Sequence

Restricting program input to a two-snapshot sequence severely limits what the program can learn about action verbs, since it is not rich enough in information to allow the program an opportunity to extract subtle detail. All changes in the environment are perceived to be discrete, instantaneous and of equal magnitude. The intermediate subactions that make up an action are invisible.

Take, for example, one simple sense of the verb "move"

as in

"Figaro moved the book".

For this overt physical action the program may be presented with a first snapshot showing Figaro and the book in physical contact at some location, and a second snapshot showing Figaro and the book in physical contact at a second location. A more realistic illustration of one sense of the verb "move" might show a person gradually approaching an object, coming into contact with the object by grasping it, changing location with it, releasing it and perhaps moving away from it. The series of snapshots might look something like:

t0) Figaro and the book at different location, say Loc1 and Loc2

t1) The book at the same location, Loc2, and Figaro having moved closer to it

t2) Figaro in physical contact with the book at Loc2

t3) Figaro and the book in physical contact at some new location, Loc3.

t4) The book at Loc3 and Figaro at some other location, Loc4.

This sequence is richer in detail and begins to provide the program with sufficient information for inferring the events that occur in "moving".

While an longer snapshot sequence provides additional information, it also results in a combinatorial explosion of processing complexity. In order to segment the action into subactions, each snapshot in the sequence must be analyzed with respect to others in the sequence [Soloway 77, Badler 75]. It is also extremely difficult to compare one N-snapshot sequence with another N-snapshot sequence since the time slice viewed during the two actions may not be the same and the intervals between snapshots in the sequence may also be different. Additionally, the CMS would need to be able to represent time.

2. An Interactive Component

In order to acquire the information necessary to refine the CMSs as accurately as inherently possible for the representation, it would be useful for Moran to ask questions of a human trainer. Since the program has no access to an episodic memory, a record of actual snapshots seen, it may be unable to discover necessary distinctions between different verb senses. Environmental information that was

determined to be unimportant for learning a particular verb sense may prove vital in distinguishing that sense from another sense that occurs later in its experience.

The component DIRECTATTENTION discussed in Section 4.A.1 filters the incoming perceptional data. Once information is filtered out, it can never be recovered. It is therefore necessary for the program to ask discriminating questions. For example, if Moran has identical or very similiar representations for some senses of different root verbs, it might ask if these senses are synonyms. If so, they could be marked as such. If not, Moran could ask how they are different and appropriately modify each CMS.

An interactive component would also aid in learning non-action verbs and inferring causality. A discussion of causality appears below. Both problems require information that is not directly expressed as environmental changes. With the addition of an interactive component, the program would be able to be told information that it may be unable to extract from visual analysis of the environment. For example, it might inquire how "owning" differs from "holding". It might also ask for confirming information for its hypotheses. If, for example, it notices that Agents are always or almost always human, it could ask if indeed

Agents are always human and perhaps why that is so, possibly illiciting information about causality if the answer indicates that Agents cause actions and Causers must be human.

B. Theoretical Limitations

A CMS cannot express all types of verbs. A verb whose meaning involves repetitive changes or continuous time cannot be fully represented. Argument co-occurrence restrictions are not represented and the concept of causality is not addressed.

1. Argument Restrictions

The program currently learns the arguments to associate with a verb sense and the restriction placed on each argument. There can be only one restriction on each argument. It is the least common superset of all the instances of the argument. Although instances of the arguments are recorded, they do not function as argument restrictions; they are maintained for future use in forgetting operations where such statistical information is required.

To reduce overgeneralization, it is desireable to allow a restriction to be a logical combination of other res-

trictions. For example, perhaps the Object argument must always be a green physical object or a C82 argument must be a non-bald man with one pierced ear. Logical expressions are necessary if argument restrictions are to be other than broad general classes.

The semantic constraints on the argument slots are too local with respect to past experience. The frequency of occurrence of each argument is recorded, but not co-occurrence between arguments. Suppose, for example, the program experiences

Cat says meow

and

Dog says bark.

The arguments of the CMS for the verb SAY might be:

AGENT	Animal(2): cat(1), dog(1)
OBJECT	Sound(2): meow(1), bark(1)

The CMS correctly represents the fact that animals "say" sounds. It also implies that cats may say "bark" and dogs may say "meow". Storing an episodic memory could help solve this problem. Another approach is to store semantic constraint information that limits what may fill an argument given that other arguments are already instantiated.

It may be argued, however, that the purpose of the argument part of a CMS is not to indicate semantic constraints between argument slots, but only to describe legitimate fillers of individual argument slots. The semantic constraint information would be contained in world knowledge (and possibly learned concurrently). If the program should ever experience a cat saying "bark", it would still correctly find the appropriate sense of "say". Other semantic constraint information would be needed to register surprise at what the cat said [Oden 78].

2. Causality

Moran is unable to determine, incorporate or represent the cause of an action, as opposed to the visible content of an action. What caused a verb to happen is part of the overall meaning of the verb [Schank 72, Norman and Rumelhart 75] and should be part of the representation of the meaning of the verb. When asked what a particular verb "means", people are able to indicate what or who causes the event to happen or at least to make a good guess as to the most likely cause as well as likely effects. For example, in the sentence

"John moved the book"

it is clear that John did something that caused the book to move. This program will learn who can do the moving and what can be moved, but not that the person who can do the moving also causes the object to be moved.

Causality appears to be especially important to the meaning of non-action verbs where the "occurrence" of the verb can be viewed as a change of internal state as the result of some visible action. An essential part of the meaning of such verbs is knowledge about the action that caused the change of state to occur. For example, the sentence

"Ristin comforted John"

means that Ristin did something that caused John to feel comforted. Similarly,

"Ristin prevented the accident"

means that Ristin performed some action that caused the accident not to happen.

It is not clear exactly what type of information is needed to enable learning causality or how causality should be represented. We first need an understanding of the types of causality and classes of actions.

Two types of knowledge that would be helpful for inferring causality are:

1) Knowledge about physical forces and how they operate in the world.

2) Knowledge and understanding of internal motivations of potential actors.

Knowledge of physical forces provides information such as the fact that inanimate objects cannot move of their own volition but animate objects can. We can include in this category "animate forces" like wind, rain and waves. This type of knowledge is necessary to avoid deductions like "The book caused Figaro to move". It is true that this type of knowledge may be implicitly embedded in the program by assuming the agent of the sentence is always (or most likely to be) the cause of the action. However, such information should either be learned or explicitly represented within the world knowledge so that it can be used whenever necessary.

Knowledge of the internal motivations of potential actors is also necessary in order to determine causation. Often the only way of knowing (or inferring) what caused an action is to know that an actor desired some final state to

be reached and therefore set in motion a sequence of events to cause that state to come about. For example, in

"Figaro hurt Ristin"

some knowledge about Figaro's internal state, such as the fact that he felt angry at Ristin, can give credence to the deduction that Figaro did something that caused Ristin to become hurt as well as help predict Figaro's next action, like apologizing. Also, from knowledge about what Figaro did, it may be possible to infer that he must have been angry.

Since internal states cannot be directly observed, it is necessary to understand that one event may place an actor in an emotional state that will later express itself by an action. The complex event of purchasing a book can be understood as a causal chain that starts with availability of book and money and desire for the book and concludes with actions of tendering by one party causing tendering by the other.

As further discussed in Section 4 of this chapter, internal states and motivations of others are hard to represent and to infer from experience with one's own internal drives. We want a way to determine that Figaro

caused the book to move and not vice versa from knowledge that Figaro desired the book to be in a different place from its present one.

Internal motivations are especially important for determining "reverse causation", where the surface language treats the cause as temporally following the result. An example of reverse causation is

"The coffee was bad, so Figaro must have made it".

Although Figaro made the coffee before it was discovered that the coffee was bad, the fact that the coffee was bad causes one to believe that Figaro must have made it. Inferring this type of causality requires understanding thought processes of the speaker. Whenever the reason for doing something is emotional, the deductive processes must deal with what is happening inside a person's head.

Moran does not deal with causality because the necessary incorporation and utilization of mental states, drives and emotions is an extremely complicated and unexplored area. Basic and powerful learning processes can still be explored aside from most aspects of causality.

3. Integration of the Verb World and World Knowledge

Moran maintains two separate bodies of knowledge. One, the verb world, is a set of directed acyclic labeled graphs that represent the set of CMSs the program infers as a result of its experience with the environment. The other, world knowledge, is a tree of concept nodes given to the program. It currently represents only set membership with ISA and SUPERSET/SUBSET links, although the implementation is capable of representing general semantic nets.

World knowledge and the verb world are separate for practical rather than theoretical reasons. Although in this program they have different formats, most theorists agree that they should be connected [Norman & Rumelhart 75]. However, since this program needs some given knowledge in order to learn other knowledge, the two knowledge bodies are maintained separately so that:

1) It will be clear what the program has itself learned as opposed to what it is given.

2) It is easy to determine and record the world knowledge used while the verb world knowledge is learned.

Integration of the two knowledge bases would require

redesign of the basic data element constituting a node in the net so that it may represent any knowledge rather than a CMS, noun concept or set concept. Additionally, the program would need procedures to access the more complex knowledge and decide what information should be stored as restrictions in the CMS. In the current implementation, if Raphael and Sharon are encountered as agents, then the restriction on the Agent argument becomes the least common superset of Raphael and Sharon. If a great deal of information is stored in world knowledge about both Raphael and Sharon, then the program must decide if the intersection of all their attributes, such as the facts that they both have curly hair, ride bicycles and are computer scientists, are relevant restrictions on the Agent argument.

4. Non-Action Verbs

Moran does not learn CMSs that represent non-action verbs, for which there may be no overt physical or directly observable manifestation and whose causality is nebulous. Examples of non-action verbs are love, hate, threaten, believe, describe, desire, expect, feel, fear, forget, imagine, know, learn, please, predict, remember, suspect, understand, want, be and stand. Many of these verbs share some factors:

1) An internal state or motivation is necessary to describe what the verb means.

2) There may be no directly observable manifestation of the verb.

3) It is not possible to determine the cause of the verb directly.

Although people obviously are able to extract such information from the environment, they undoubtedly apply a great deal of processing and inference on the raw perceptual data and internally generated data. A great deal of information also comes from hearing rather than seeing. The sentence,

John said the toast was burned so Joe must have made it.

provides a wealth of information. It is not clear exactly <u>what</u> information should or can be extracted.

Several types of information might be needed to solve this problem. First, it is necessary to determine what information is embodied in an "internal state". There may be hundreds of factors that contribute to a person's internal state: stability, security, happiness, fear, anger and desire are some.

Even if an internal state can be decomposed into factors, there remains the problem of describing the relative "quantity" of each factor. Assigning such quantities seems inadequate for several reasons.

1) The value associated with each factor may spontaneously shift very frequently.

2) The physics of the values is unknown. That is, they cannot always be increasing. When do they go up? How do they come down? Are there certain things that cause relatively permanent increases or decreases in the values?

3) It is not clear which verbs are reflections of changes in what factors.

Assigning one overall numerical value to an internal state also seems inadequate because there are probably a great many such factors. To combine all values to get a resultant "internal state value" would very likely discard any meaning embodied in the individual states.

Perhaps the only way descriptions of non-action verbs can be obtained is by induction from the observer's internal state to another's internal state. That is, a child doesn't have to move a book to learn the basic meaning of

what it means for someone else to move a book. But a child does have to experience fear to understand what it means for someone else to fear.

The source of the problem is that these verbs are not actions one performs on an object. They are the result of an action rather than an action itself. Some, possibly unknown, action causes a given internal state.

In any case, given the appropriate information either in the environment or as a result of extensive inference on the information in the environment, Moran should be able to learn the meanings of non-action verbs by the same processes it uses for learning action verbs.

Consider, for example, the verb "threaten", in the sense to threaten bodily harm with a physical object, as an example of a non-action verb. A simplified CMS we would like the program to learn might look something like:

```
        AGENT      Human
        OBJECT     Human
        INSTR      Physobj

    AGENT CAUSE *

    ( OBJECT IMAGINE *'
*   { OBJECT HAPPINESS value1 --> OBJECT HAPPINESS (value1-d1)
    ( OBJECT FEAR value2 --> OBJECT FEAR (value2+d2)

*'  INSTR MAY.HARM OBJECT  if OBJECT DO *''

*'' undefined action
```

The vast majority of this information in the CMS is not explicitly observable in the environment: CAUSE, BELIEVE, HAPPINESS and FEAR are functions of many factors. All this information must be inferred from the overt physical action taking place in a two snapshot sequence which may look something like:

Snapshot1	Snapshot2
RISTIN AT LOCA	RISTIN AT LOCA
FIGARO AT LOCB	FIGARO AT LOCB
RISTIN PHYSCONT BRICK	RISTIN YELL-AT FIGARO
	BRICK CLOSE-TO FIGARO
	RISTIN PHYSCONT BRICK

"Ristin threatened Figaro"

Nothing about causing, fearing, happiness or fear is present in these snapshots. It must all be deduced from expressions on Ristin's and Figaro's faces and a dictionary of societal formalities about "threatening behavior" [Searle 69].

Although a solution to the problem of learning non-action verbs would certainly be valuable, the insights gained are not necessary for a solution to the problem addressed here: the processes needed to infer semantic graphs. How mental states are inferred from subtle clues and how to represent these mental states are topics for further research.

C. Conclusions

The implementation of Moran has shown that a small set

of well-defined processes are adequate, to a certain point, to infer semantic graph structures from pictorial input data. The semantic graphs represent both the meaning of individual input situations and the similarities and differences between different input situations.

Two types of similarities and differences are inferred. The first type is within the categories that segment the collection of graphs, in this case root verb words. Moran infers graphs where shared nodes that contain groups of arguments and effects reflect similarities between different senses of the same root verb. The second type reflects similarities across graph boundaries by identifying Global Building Blocks that are effects used in the meaning of many senses of different root verbs. A GBB may be viewed as a type of primitive that will strongly influence subsequent incorporation of information into the graph structure. It is significant that the GBBs are discovered by the program rather than given to it.

Moran shows that learning can be viewed as a formalizable set of mechanisms and interactions between these mechanisms that can learn general structural descriptions, rather than an ad hoc method that is limited to learning structures and relations specific to one application. The

fact that Moran is implemented and has generated semantic graphs from pictorial data demonstrates that the learning mechanisms discussed work.

D. Future Directions

Further research is required to refine the learning mechanisms discussed here and to define other learning processes needed for further refinement of the CMS structures.

The fourth learning mechanism, GBB Extraction, should be part of a more general non-input directed process, Global Reorganization. The function of Global Reorganization will be to modify the verb world in order to produce a "better" organization.

Global Reorganization needs to operate within individual CMSs as well as across CMS boundaries. CMS graphs tend to become both deep and bushy because information incorporated into the graphs is never discarded. A procedure is needed that will forget information that is not required for verb sense discrimination. Another procedure that would operate within an individual CMS would reorganize the arguments and effects graphs so that the most commonly shared nodes in each graph are at the top levels of the

graphs for as many senses as possible. Such an 'organization would facilitate heuristic search pruning since two graphs that did not match at the higher levels would not need to be matched further.

Like GBB Extraction, other Global Reorganization processes need to operate across CMS boundaries. The verb world must be reorganized with respect to the GBBs Moran discovers. Additionally, higher level GBBs that consist of groups of effects and their restrictions should be found as well as the single-effect GBBs. These higher level GBBs may be viewed as primitive meaning units and compared with primitive meaning units that have previously been suggested [Schank 72].

Two further problems to be investigated are the conditions under which Global Reorganization should be invoked and how the results of the Global Reorganization process should affect the processing of input in subsequent learning cycles.

Finally, research must be pursued regarding the semantic similarity of graph structures. At least two problems are to be initially addressed: what does it mean for two graphs to be semantically similiar and what types of information are necessary to determine semantic similarity?

Although Moran is a large, complex program with interactions between many procedures, it has been modularly designed to facilitate experimentation and further research on these issues.

REFERENCES

[Anderson 77] J.R. Anderson, "Induction of Augmented Transition Networks", Cognitive Science, 2, 1977,125-157.

[Badler 75] N.I. Badler, Temporal Scene Analysis: Conceptual Descriptions of Object Movements, University of Toronto Computer Science Dept. Technical Report #80, February, 1975.

[Bartlett 32] F.C. Bartlett, Remembering: A Study in Experimental and Special Psychology, Cambridge University Press, 1932.

[Bruce 75] B. Bruce, "Case Systems for Natural Language", Artificial Intelligence, 6, 1975,327-360.

[Davis and King 75] R.Davis and J. King, An Overview of Production Systems, Stanford University Computer Science Dept. Report AIM-271, October, 1975.

[Fillmore 68] C. Fillmore, "The Case for Case", in Bach and Harms (eds), Universals in Linguistic Theory, Holt, Rinehart and Winston, 1968.

[Harris 72] L.R. Harris, A Model for Adaptive Problem Solving Applied to Natural Language Acquisition, PhD

Thesis, Computer Science Dept., Cornell University, 1972.

[Hayes-Roth 76] F. Hayes-Roth, "Patterns of Induction and Associated Knowledge Acquisition Algorithms", in C. Chen (ed), Pattern Recognition and Artificial Intelligence, New York: Academic Press, 1976.

[McCarthy and Hayes 69], "Some Philosophical Problems from the Standpoint of Artificial Intelligence", in Meltzer and Michie (eds), Machine Intelligence 4, Edinburgh University Press, 1969.

[Minsky 75] M. Minsky, "A Framework for Representing Knowledge", in P.H. Winston (ed), The Psychology of Computer Vision, McGraw-Hill, 1975.

[Newell and Simon 72] A. Newell and H. Simon, Human Problem Solving, Prentice Hall, 1972.

[Norman and Rumelhart 75] D. Norman, D. Rumelhart and the LNR Research Group, Explorations in Cognition, Freeman, 1975.

[Oden 78] G.C. Oden, On the Use of Semantic Constraints in Guiding Syntactic Analysis, University of Wisconsin Human Information Processing Program, WHIPP Report #3, January, 1978.

[Quillian 68] M.R. Quillian, "Semantic Memory", in M. Minsky (ed), Semantic Information Processing, The MIT Press, 1968.

[Quillian 69] M.R. Quillian, "The Teachable Language Comprehender", CACM, 1969, 12 ,459-475.

[Reeker 74] L. Reeker, A Problem Solving Theory of Syntactic Acquisition, PhD Thesis, Computer Science Dept., Carnegie-Mellon University, 1974.

[Richards 61] I. A. Richards et. al., English Through Pictures, Washington Square Press, 1961.

[Rosch 76a] E. Rosch, "Classifications of Real World Objects: Origins and Representations in Cognition", in S. Ehrlich and E. Tulving (eds), La Memaire Semantique, Bulletin de Psychologie, 1976.

[Rosch 76b] E. Rosch et. al., "Basic Objects in Natural Categories", Cognitive Psychology, 8, 1972,382-439.

[Rumelhart and Norman 76] D. Rumelhart and D. Norman, Accretion, Tuning and Restructuring: Three Modes of Learning, University of California at San Diego Center for Human Information Processing Report 63, August, 1976.

[Samuel 59] A.L. Samuel, "Some Studies in Machine Learning

Using the Game of Checkers", IBM Journal of Research and Development, 3, 1959.

[Samuel 67] A.L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers-Recent Progress", IBM Journal of Research and Development, 6, 1976.

[Schank 72] R. Schank, "Conceptual Dependency: A Theory of Natural Language Understanding", Cognitive Psychology, 3, 1972,552-631.

[Schank 73] R. Schank, "MARGIE": Memory, Analysis, Response, Generation and Inference on English", Advance Papers on the Third International Joint Conference in Artificial Intelligence, Stanford, California, 1973.

[Searle 69] J. Searle, Speech Acts, London: Cambridge University Press, 1969.

[Soloway 77] E. Soloway, Knowledge Directed Learning Using Multiple Levels of Description, PhD Thesis, Computer Science Dept., University of Massachusetts, 1977.

[Soloway and Riseman 77] E. Soloway and E. Riseman, Knowledge-Directed Learning, COINS Technical Report 77-6, University of Massachusetts, Amherst, 1977.

[Uhr and Vossler 63] L. Uhr and C. Vossler, "A Pattern

Recognition Program that Generates, Evaluates and Adjusts its own Operators", in E. Feigenbaum and J. Feldman (eds), Computers and Thought, McGraw-Hill, 1963.

[Winston 75] P.H. Winston, "Learning Structural Descriptions from Examples", in P.H. Winston (ed), The Psychology of Computer Vision, McGraw-Hill, 1975.

[Woods 70] W. Woods, "Transition Network Grammars for Natural Language Analysis", CACM, 13, 1970, 591-606.

[Yakimovsky 73] Y. Yakimovsky, Scence Analysis Using a Semantic Base for Region Growing, Stanford AI Lab Memo AIM-209, June, 1970.

[Zeigler 78] S. Zeigler, Learning by Pattern Induction, University of Wisconsin Computer Sciences Dept. Technical Report #319, April, 1978.