AD-A045 790 MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTE--ETC F/G 6/4
REPRESENTING KNOWLEDGE OF LARGE-SCALE SPACE.(U)
JUL 77 B J KUIPERS N00014-75-C-0643
UNCLASSIFIED AI-TR-418 NL

1 OF 2
AD
A045790

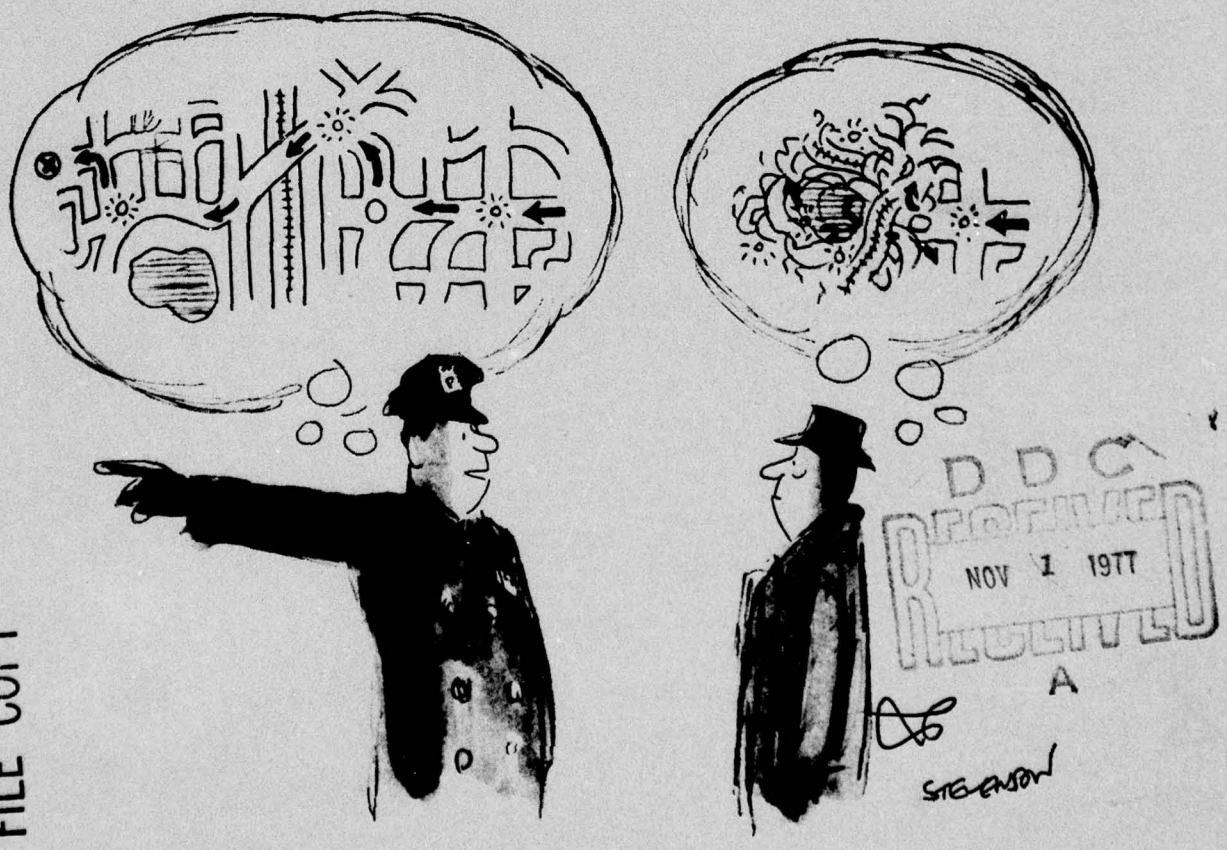
AD A045790

AI-TR-418

REPRESENTING KNOWLEDGE OF LARGE-SCALE SPACE

Benjamin Jack Kuipers

July 1977



AD No.
DDC FILE COPY

DDC
RECEIVED
NOV 1 1977
A

ARTIFICIAL INTELLIGENCE LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AI-TR-418
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:
4. TITLE (and Subtitle): Representing Knowledge of Large-Scale Space.
5. TYPE OF REPORT & PERIOD COVERED: Technical Report
6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Benjamin Jack Kuipers
8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0643
9. PERFORMING ORGANIZATION NAME AND ADDRESS:
   Artificial Intelligence Laboratory
   545 Technology Square
   Cambridge, Massachusetts 02139
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 14 p.
11. CONTROLLING OFFICE NAME AND ADDRESS:
    Advanced Research Projects Agency
    1400 Wilson Blvd
    Arlington, Virginia 22209
12. REPORT DATE: July 1977
13. NUMBER OF PAGES: 147
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):
    Office of Naval Research
    Information Systems
    Arlington, Virginia 22217
15. SECURITY CLASS. (of this report): UNCLASSIFIED
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:
16. DISTRIBUTION STATEMENT (of this Report):
    Distribution of this document is unlimited.
    Doctoral thesis
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report):
18. SUPPLEMENTARY NOTES:
    None
19. KEY WORDS (Continue on reverse side if necessary and identify by block number):
    Spatial knowledge
    Cognitive maps
    Common-sense knowledge
    Partial knowledge
    Representations
20. ABSTRACT (Continue on reverse side if necessary and identify by block number):

This dissertation presents a model of the knowledge a person has about the spatial structure of a large-scale environment: the "cognitive map." The functions of the cognitive map are to assimilate new information about the environment, to represent the current position, and to answer route-finding and relative-position problems. This model (called the TOUR model) analyzes the cognitive map in terms of symbolic descriptions of the environment and operations on those descriptions.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

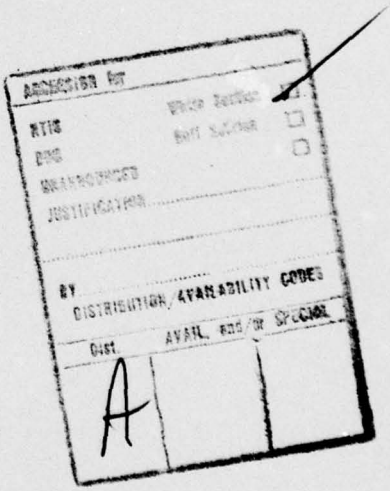
UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD A073483

AB

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Project Agency of the Department of Defense under Office of Naval Research contract N00014-75-C-0643.



REPRESENTING KNOWLEDGE OF LARGE-SCALE SPACE

by

BENJAMIN JACK KUIPERS

Massachusetts Institute of Technology

July 1977

Revised version of a dissertation submitted to the Department of Mathematics on May 5, 1977 in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Cover drawing by Stevenson; © 1976
The New Yorker Magazine, Inc.

ABSTRACT

This dissertation presents a model of the knowledge a person has about the spatial structure of a large-scale environment: the "cognitive map." The functions of the cognitive map are to assimilate new information about the environment, to represent the current position, and to answer route-finding and relative-position problems. This model (called the TOUR model) analyzes the cognitive map in terms of symbolic descriptions of the environment and operations on those descriptions.

Knowledge about a particular environment is represented in terms of route descriptions, a topological network of paths and places, multiple frames of reference for relative positions, dividing boundaries, and a structure of containing regions. The current position is described by the "You Are Here" pointer, which acts as a working memory and a focus of attention. Operations on the cognitive map are performed by inference rules which act to transfer information among different descriptions and the "You Are Here" pointer. The TOUR model shows how the particular descriptions chosen to represent spatial knowledge support assimilation of new information from local observations into the cognitive map, and how the cognitive map solves route-finding and relative-position problems.

A central theme of this research is that the states of partial knowledge supported by a representation are responsible for its ability to function with limited information or computational resources. The representations in the TOUR model provide a rich collection of states of partial knowledge, and therefore exhibit flexible, "common-sense" behavior.

Name and Title of Thesis Supervisor: Marvin Minsky, Donner Professor of Science in the Electrical Engineering and Computer Science Department.

ACKNOWLEDGEMENTS

I would like to thank Marvin Minsky, my thesis advisor, for the inspiration, ideas, and advice he provided during this project. He, Seymour Papert, and Carl Hewitt acted as the perfect thesis committee for me. I am also very grateful to Dan Bobrow for suggesting this problem. A number of people read early reports of this research and provided helpful comments. They include David Marr, Kevin Lynch, Shep White, and Mark Jeffery. Continuing discussions with Al Stevens have been pleasurable and helpful. My wife Laura and the rest of my household prevented when possible, and tolerated when not, the single-mindedness that goes with writing a thesis. I am grateful to them all.

The MIT Artificial Intelligence Laboratory provides a unique atmosphere of intellectual excitement and enjoyment which has been very important to me. Many of the ideas in this research would not have happened without the AI Playroom. The metropolitan Boston area deserves my thanks for its geography, without which this research would have been impossible. Boston's confusing street pattern highlights the partial knowledge of which spatial representations are capable. I also appreciate the patience of the people I interviewed, whose confusions and impressions I attempt to describe.

I am also grateful for financial support provided by graduate fellowships from the National Science Foundation and the Danforth Foundation, and for a research assistantship at the Artificial Intelligence Laboratory.

TABLE OF CONTENTS

I.	DESCRIBING SPATIAL KNOWLEDGE	7
II.	THE TOUR MACHINE	21
III.	ORIENTATION	49
IV.	DIVIDING BOUNDARIES	67
V.	REGIONS	84
VI.	RELATIONS TO OTHER WORK	102
VII.	CONCLUSIONS	114
	NOTES	121
	REFERENCES	125
A.	DESCRIPTIONS OF ENVIRONMENT, CURRENT POSITION, AND ROUTE	128
B.	INFERENCE RULES	133
C.	IMPLEMENTATION ISSUES	143

I. DESCRIBING SPATIAL KNOWLEDGE

"Puluwat is one of a long chain of islands covering over a thousand miles of the Pacific Ocean north of New Guinea. The sailing canoe is at the very heart of the Puluwat way of life, and skilled navigators occupy the positions of highest status. There is quite a bit of drinking among men of Puluwat. A trip to Pikelot island, one mile wide and over 100 miles distant, is often launched at a drinking party. Someone jumps up and says, 'I'm going to Pikelot. Who's coming with me?' The navigator determines his sailing plan only after he is at sea; at no time is an overall plan developed for the voyage. Yet, without fail, at an average speed of 4.5 knots, the 26-foot sailing canoe with a now-sober crew arrives two days later at a one-mile wide island in the open ocean (Gladwin, 1970). What is the nature of their knowledge about the sea and the sky and the islands and the canoe that makes this possible?

Most mornings, every reader of this paper will find his way to the office, less exuberantly than the voyagers of Puluwat but no less decisively. He or she will walk, ride a bicycle, take the train, or drive. The route may be long or short, simple or complex; it may be interrupted or accomplished around diversions. What system of habits, schemes, imageries, long- or short-term memories allows him to accomplish this?

Most mornings, thousands of 6-year-old children will find their way to school. There will be protections along the way---street-crossing policemen---but there will be little fear on anyone's part that the children will get lost. The children will find their way to the school building, within the building to their classroom, and after school, perhaps, to a friend's house. However, most 3-year-olds do not typically engage in the same process. Is it that they are incapable of following routes and will thus get lost? It is quite possible that these very young children can recognize landmarks in a large-scale terrain, but have limited 'route-knowledge,' and that adults are aware of this."

[Siegel and White, 1975, p. 10.]

What the Puluwat navigator, the morning commuter, and the kindergarten child have in common is that they find their way without aid of maps or instruments in a space that is too large to be perceived at once. People navigate in these circumstances by using their common-sense knowledge of space: they link many separate observations into a "cognitive map" and solve route-finding problems by consulting the knowledge in the cognitive map.

The term "cognitive map" as used here refers to a body of knowledge that a person has in his head about the spatial structure of an environment. This is to distinguish it from other related concepts, such as the expression of that knowledge as maps, models, or verbal directions. People also associate visual impressions or social facts with different parts of the environment, but those are not the structural aspects of the cognitive map that will concern us here.

The cognitive map is used to solve route-finding and relative location problems in a large-scale space. A large-scale space is defined by the way it is perceived, not by its physical size. Its structure is deduced from a number of observations over time, rather than being perceived from one location. The places, paths and routes of a city are the most common large-scale space in most people's experience, although of course there are others: ocean, jungle, and open field.

A basic assumption of this research is that knowledge, and the cognitive map in particular, can be viewed as a complex symbolic description which can perform a certain range of computations. At the lowest, functional level, a person's cognitive map can be considered as a black box that receives inputs describing actions and observations, and can solve route-finding and relative location problems. At the next level, the cognitive map is a learning machine: the contents of the black box, and its ability to solve problems, change as a result of the inputs it receives. At an even higher level, the ability of the cognitive map to assimilate new information changes slowly over time. These three levels of activity correspond to problem-solving, learning, and development in humans.

The research reported here deals with the cognitive map seen as a functional problem-solver

and as a learning machine. In order to describe its properties as a learning machine, we must describe the contents of the black box and how they change in reaction to the inputs. The result of this research is the TOUR model: a computational model of the cognitive map, showing how it assimilates new information and solves problems.

Mini-theories

The TOUR model is derived from a number of simple "mini-theories," each of which captures some of the properties of the cognitive map, but none of which is adequate to explain the whole. Each one can act as a valuable metaphor for some aspects of spatial knowledge. In order to construct a comprehensive model, it is useful to examine the strengths and weaknesses of each one.

The "Map in the Head" theory says that the cognitive map is isomorphic to a printed map which can be examined with the "mind's eye." A very similar theory says that the position of a place amounts to its Cartesian coordinates in a global grid. Learning in such a theory presumably occurs by "drawing" new places and paths onto the map. These theories are easily refuted by anecdotal evidence about the states of partial knowledge of which people are capable. A two-dimensional map (on paper or in the head) provides a single global frame of reference for relative positions: two vectors, however remote, can be compared. This is clearly not the case for people, who often know two sets of relative positions that they are unable to compare.

The "Patchwork Map" theory says that the cognitive map consists of a number of different

maps, of different regions and at different levels of detail, each of which is isomorphic to a printed map. This theory avoids the problem of global consistency of relative positions, but begs the question of how an appropriate map is selected when needed. It also fails to capture a very important kind of partial knowledge of relative position: the relation of a place to a dividing boundary. (See Chapter 4) These difficulties aside, the "Patchwork Map" theory is actually a very good mini-theory of cognitive maps. Chapter 3 discusses the way relative positions are defined with respect to distinct frames of reference, and Chapter 5 discusses the way appropriate regional views are selected when needed. Of course, each of these aspects of the TOUR model needs additional computational mechanisms which could not be provided by a printed map.

The "Street Network" theory says that a geographical area can be represented topologically as a network of nodes and arcs. A route-finding problem can then be stated as a "Travelling Salesman" problem, for which uniform algorithms exist. However, the distinction between easy and hard problems is quite different for such uniform algorithms and for people. Furthermore, many kinds of important common-sense spatial knowledge (e.g. a sense of direction) cannot be supported by a topological network. These are the kinds of spatial abilities addressed by the previous theories. Like the "Patchwork Map" theory, the "Street Network" theory is actually quite a good mini-theory of cognitive maps: much common-sense knowledge of space is a topological description of paths and places. Chapter 2 discusses the topological aspects of the TOUR model, and shows how the place and path descriptions differ from the graph-theoretical concepts of node and arc.

The "Hybrid Model" theory points out that if each is a good mini-theory, then perhaps some sort of hybrid of the "Street Network" theory with the "Patchwork Map" theory would be

correct. That is the assumption of this research, and the TOUR model is exactly such a hybrid. The process of expanding such a suggestion into a complete theory, however, requires extensive elaboration, as demonstrated in the rest of this thesis.

The "Assertional Database" theory says that geographical knowledge is represented as a database of atomic assertions of geographical facts. The "Semantic Network" theory says that it is represented as clusters of labelled pointers to other clusters. These are not actually theories of knowledge at all, but only suggested implementations for databases. They omit the actual descriptions that are in either kind of database, and the actual inferences which operate on them. Either suggestion may be presented as a useful method to artificial intelligence, for representing certain kinds of knowledge, but neither is a theory of spatial knowledge.

The "Urban Environment" objection is that the TOUR model is expressed in terms that are appropriate to city streets and intersections, but it would not apply to significantly different large-scale environments such as forests or open ocean. This objection can only be settled conclusively by further research. However, a review of what literature does exist on non-urban large-scale spaces suggests that the same descriptions apply. For example, see Gladwin (1970) for an analysis of the knowledge held by navigators in the South Pacific. Surprisingly, the apparently trackless ocean is seen by the experienced navigator as covered with paths and landmarks, real and imaginary, which guide him to his destination. Similarly, Lynch (1960) reports that experts familiar with trackless deserts or impenetrable jungles see them as covered with intersecting paths.

Large-scale space

Large-scale space is defined by the perceptual mechanisms for exploring the space, rather than by its physical size. A large-scale space is defined as a space which cannot be perceived at once: its global structure must be derived from local observations gathered over time. For example, a drawing is a large-scale space when viewed through a small movable hole, while a city can be small-scale when viewed from an airplane. The typical operation when learning about a large-scale space is to compare the current observations with the corresponding descriptions in the cognitive map. This requires establishing and maintaining a correspondence between the two sources of information, so that information from each source can be added to the other. Much of the meaning of an observation depends on the context provided by the cognitive map; conversely, the cognitive map is created by assimilating many observations.

Large-scale space is characterized by a very clear distinction between local observations and global structure. An important consequence of this is that local descriptions are likely to be much more accurate than global ones, a phenomena which is easy to observe in people's cognitive maps. Local inconsistencies are easy to detect and correct, and can usually be attributed to particular observations. Global inconsistencies, however, are much more difficult to pin down, and are usually due to the unwarrented extension of local observations. The TOUR model shows how local observations are extended to describe global structure, but it does not address the problem of debugging the cognitive map when errors are found.

Spatial knowledge is a good example of the more general category of common-sense knowledge, which is characterized by easy and flexible application to situations with

low-quality data and limited computational resources. Common-sense knowledge makes it possible to take a rough description of a problem and produce a correspondingly rough, but potentially refinable, solution. In order to have the flexibility of common-sense knowledge, a representation must support a rich collection of states of partial knowledge that can be meaningfully operated on. The goal of this research is to show that a certain collection of representations, because of their abilities to represent partial knowledge, can describe common-sense knowledge of space.

Spatial knowledge is a particularly fundamental kind of common-sense knowledge, for spatial metaphors are very common in representing other kinds of knowledge. For example, time is often seen as a linear spatial dimension, similarity is often judged as "near" and "far", and social relations can be described in terms of "close," "distant," "above," and "below." Spatial knowledge is also universal in that it is acquired by people in all cultures and environments, and even by most animals.

The kind of spatial knowledge which is the domain of this research is the overall structural description of large-scale space which results from assimilating local observations. This must be distinguished from the related problems of communicating such knowledge among people, either through printed maps or verbal description. Any detailed theory of map reading or map drawing, or the ability to give or understand verbal route instructions, must depend on a theory of how the knowledge is represented internally. Similarly, any detailed theory of the development of spatial abilities in children requires a way to describe what those abilities are at any given time.

Visual input is very important to the cognitive map, of course. It typically provides the local

observations which are merged into the structural descriptions. Visual memory permits the recognition of landmarks as being the same on different encounters. Without recognizing the identity of places, there can be no cognitive map, because no experience relates to any other. On the other hand, the structural descriptions of large-scale space are quite distinct from the visual descriptions of a scene. The TOUR model factors out the issues of visual input and recognition, and concentrates on how the local structural information thus provided is assimilated into a useful cognitive map. The simplified inputs it receives consist of topological connections, estimated magnitudes of turns and travels, and the egocentric heading and distance of observed remote landmarks. These are presented in more detail in chapters 2 and 3.

The TOUR model

The TOUR model includes a number of representations for different kinds of spatial knowledge. They fall into three initial categories: 1) representations for knowledge about a particular environment, 2) the description of the current position, and 3) the inference rules that manipulate knowledge of the other two kinds. The TOUR model shows how different pieces of knowledge are combined and how inference processes translate knowledge from one representation to others. Many of these inferences are organized around the central process of following a route description through the cognitive map. Since this process resembles a computer following a computer program, the collection of inference rules is known as the "TOUR machine."

The representations in the TOUR model which encode information about particular

environments are:

1. An imperative route representation that directs a traveller along a route through the cognitive map.

2. A representation for local topological properties of street networks, including the ordering of places on a street and the local geometry of streets at an intersection.

3. Frames of reference which define the relative positions of objects. Positions defined with respect to different frames of reference (or "orientation frames") may not be comparable.

4. Dividing boundaries which provide a second, qualitative representation for relative position.

5. A structure of containing regions at different scales which provides levels of abstraction for stating properties of their elements.

Throughout the TOUR model, these descriptions can be partially specified. The action of the TOUR machine, as it drives the "You Are Here" pointer through the map, is to transfer fragments of information from one description to another. The assimilation process takes place as unspecified elements of one description are filled with information from another. This theme of partial knowledge will recur throughout the discussion of representations and descriptions. Because of their states of partial knowledge, the representations in the TOUR model satisfy such principles as the Principle of Least Commitment and the Principle of Graceful Degradation. {Note Principles}

The current position of the traveller is represented by a small working memory called the "You Are Here" pointer. The "You Are Here" pointer has a number of different elements, and can describe the current position in terms of place, path, one-dimensional orientation on that path, current frame of reference, and two-dimensional orientation with respect to that frame of reference. This description may be more or less complete depending on the completeness of the descriptions of the environment and of the route that reached that point. Most

manipulations of knowledge in the TOUR model take place through an interaction between the environmental descriptions and the "You Are Here" pointer.

The inference rules that manipulate knowledge embedded in these various representations are represented as productions: simple modules that wait for a certain set of conditions to be true and then perform some simple action. The inference rules fall into categories that correspond roughly to the kinds of representations.

1. Rules which compare the current route instruction, the "You Are Here" pointer, and the topological descriptions of the environment. They can act to fill gaps in each representation with information from the others. In particular, this is how the topological description is originally created from information in the route description. (Chapter 2)

2. Rules for maintaining the current heading, or two-dimensional orientation, with respect to the current orientation frame. They compare information in the "You Are Here" pointer with the descriptions of the current place and path, again updating the knowledge in each by combinations of knowledge from the others. (Chapter 3)

3. Rules which detect special structural features of a part of the environment, such as paths which act as dividing boundaries separating places. These rules act within the "focus of attention" provided by the current route instruction and the "You Are Here" pointer. (Chapters 3 and 4)

4. Rules which solve route-finding and position-finding problems using knowledge in the hierarchy of regions, and in the descriptions of orientation frames and boundaries. (Chapters 3, 4, and 5)

The TOUR model, as described in this thesis, has been completely implemented. All the detailed examples presented below have been implemented in a program which is written in MACLISP [Moon 1974], and runs in about 100K words on the PDP-10 at the MIT Artificial Intelligence Laboratory. Appendix C discusses the main implementation technique behind the knowledge representations.

An important computational characteristic of the TOUR machine is that it never performs an

unbounded search of the cognitive map. At any point in its operation, it can access at most two route instructions, the "You Are Here" pointer, and the environmental descriptions which are directly accessible from them. This means that (with the exception of certain bounded searches) the time for performing an operation does not increase with the size of the database containing the cognitive map. On the contrary as more knowledge is represented, more powerful inference rules can be applied, and the time required to perform most operations decreases.

The TOUR model as a psychological theory

The research reported here constitutes the design of a preliminary, but comprehensive, theory of the common-sense knowledge of large-scale space to be found in the cognitive map. It is based on careful attention to the existing literature on spatial cognition, the computational requirements of the mini-theories discussed above, and interviews with selected subjects. The TOUR model is comprehensive in that it attempts to incorporate all of the different phenomena covered by the different mini-theories, and provide for their fruitful interaction. It is preliminary in that it has not yet been subjected to detailed controlled experiments and revised in the light of their results.

It is useful to consider what the effect could be of such experimental results. It is virtually certain that the detailed formats of the environmental descriptions and the productions that operate on them will need to be changed somewhat to accomodate empirical findings. In most cases, information from the literature and informal interviews do not address the detailed design issues involved, so they must be made on purely computational grounds. Experimental

results that contradict design decisions on this level, and which can be incorporated into the model by changes at this level, do not refute the underlying structure of the TOUR model.

It is possible that experimental results will contradict the structure of the environmental descriptions and the nature of their interactions: e.g. showing that the environmental descriptions are not place, path, region and orientation-frame but are some other collection, or that the topological description is not fundamental to the others. This result seems much less likely in light of existing observations, and addresses a much more fundamental aspect of the TOUR model. If refuted at this level, the TOUR model would need to be replaced by a substantially different computational model.

Most fundamentally, it is conceivable that a weight of evidence could accumulate against the position that the cognitive map can be fruitfully considered as a symbolic description and examined in terms of its computational properties. This would argue against the division of the cognitive map into environmental descriptions, manipulation procedures, and the description of the current position. This outcome would not be strictly a scientific result, in the sense of a particular empirical finding refuting a particular theory, but would be a more general conviction that this way of stating theories is not a fruitful one. It is therefore much the least likely.

This distinction between levels of commitment in the TOUR model corresponds somewhat with Marr's (1976) levels of computational theory. The first level describes the computation that is taking place. The second level specifies the algorithm to perform that computation. There can also be a third level dealing with the mechanisms that are available for implementing the algorithm, and a fourth for the hardware (neuronal or electrical) that supports the

mechanisms.

Reading this paper

The following chapters present different aspects of the TOUR model. Each chapter focusses on one aspect of its representations or operation, pushing the others into the background. This separates operations which take place together when the model operates on a large, pre-existing body of knowledge. In the next chapter, I will describe in some detail the TOUR machine, the "You Are Here" pointer, and the route and topological representations. A detailed example of the assimilation of information from a route description into the cognitive map is included. The third chapter discusses two-dimensional orientation, or heading, and the orientation-frames with respect to which headings are defined. Chapter four explains the use of dividing boundaries in the structure of the cognitive map. It also shows how they can grow into bundles of parallel streets and grid structures. Chapter five presents the use of containing regions to summarize useful information and provide abstract descriptions of the environment. Chapter six reviews the relevant research in artificial intelligence, psychology, and urban planning. Chapter seven steps back to discuss the most important principles of the TOUR model, and what remains to be done.

Throughout this paper, I give anecdotal descriptions of the knowledge I am trying to represent. This informal description is meant to communicate a viewpoint on spatial knowledge to you, the reader, and to acquaint you with phenomena with which you might not be familiar. The psychological literature and my own interviews explore these phenomena is somewhat more detail than I can present here.

Some parts of the following chapters will be quite difficult to read. The central point of this paper is that spatial knowledge is represented in several different ways, and that important things happen when pieces of knowledge are transferred from one representation to another. After presenting anecdotes to define the knowledge I am discussing, I present the detailed forms of the descriptions that represent the knowledge and the inference rules that make the transfers. The inference rules, and the examples which illustrate their use, amount to computer code stated in (I hope) somewhat comprehensible English. They must be read carefully, as mathematics, to convince you that they actually do what I say they do. The points to remember from each chapter are the representation for that particular kind of knowledge and the general effect of the inference rules on the relationship that knowledge has with the rest of the TOUR model.

II. THE TOUR MACHINE

> *"Perhaps the most important direction for future study has been noted several times above: the lack of understanding of the city image as a total field, of the interrelations of elements, patterns, and sequences. City perception is in essence a time phenomenon, and it is directed toward an object of very large scale. If the environment is to be perceived as an organic whole, then the clarification of parts in their immediate context is only an elementary step. It will be extremely important to find ways of understanding and manipulating wholes, or at least of handling the problems of sequence and unfolding pattern."*
>
> *"But it seems likely that the core of the work will escape quantities, at least for some time, and that pattern and sequence considerations will be a primary direction. Involved in the latter is the technique of representation of complex, temporally extended patterns. Although this is a technical consideration, yet it is basic and difficult. Before such patterns can be understood or manipulated, ways must be found of representing their essence in such a way that they can be communicated without a repetition of the original experience. This is a rather baffling problem."*
>
> *[Lynch, 1960, pp. 158-159.]*

This chapter presents the TOUR machine, the route programs that drive it, and the representation of the map it moves on. The aspects of the cognitive map covered in this chapter are the route instructions and the topological description of the environment. Later chapters will extend these representations to accomodate more complex phenomena. This topological representation is the result when the "Street Network" mini-theory is exposed to the requirements of some simple, anecdotal observations about how people acquire spatial knowledge.

Some people's cognitive maps contain only descriptions of familiar routes. Such people do no spatial problem-solving using their cognitive maps. When they need a new route, they ask for explicit directions or use maps. This kind of cognitive map is often characteristic of young

children or people learning a new environment. Piaget and Inhelder (1948) observe that a young child can follow a learned route long before he learns to reverse it or restart from an intermediate point.

Other people have cognitive maps that represent only topological information about the relations between paths and places. They cannot answer questions about relative directions of invisible places. Lynch (1960) observed that while the maps drawn by his informants were typically quite distorted, they were almost always topologically correct. Errors in ordering places on a street when drawing a map or navigating, for example, are very rare and tend to be memorable when they are detected. On the other hand, when a subject is asked questions about the relative order of previously chosen places on a street, some order relations are often not known. Two minor landmarks may be recalled as between two major ones on a street, but their relative order is unknown. Thus, the order of places on a street can be a partial order.

The order of places on a path can be observed by travelling along the path. Examination of an intersection provides information about the local relation between the intersecting streets. The local geometry of an intersection tends to be accurately recalled, with some tendency to distort towards right angles. Serious distortions in the cognitive map can occur when local relations between paths at an intersection are assumed to hold globally. In the TOUR model, the basic observations are those gathered by local examination of the environment while following a route: topological connections between place and path, the ordering of places on a path, and the local geometry of intersecting paths.

Creating the cognitive map takes time. Often it takes a number of encounters with a

particular place, or a particular route, to assimilate it into the cognitive map. Rather than being used to strengthen the image of the place or route in memory, the TOUR model claims that repeated exposures are necessary to transfer different kinds of information from observation to cognitive map in a fixed sequence. For example, it is necessary to have a well developed topological description of the environment before it is possible to determine the relation of places on a route to various dividing boundaries. (See this chapter and chapter 4.)

The environmental descriptions: PLACE and PATH

Let us introduce the topological description with a contrasting analogy. Consider a robot vehicle which can travel on a featureless plane in response to two commands:

```
FORWARD <number of units>
RIGHT <number of degrees>
```

The vehicle is completely described by a "state" that consists of its position and heading on the plane. LOGO [Papert 1972] is a programming system that consists of a plane, a vehicle, and an interpreter for this simple language. The key difference between LOGO and TOUR is that the featureless plane is replaced by a highly structured, network-like cognitive map. The state of the traveller must then become correspondingly more complex. The TOUR machine executes a simple LOGO-like programming language that drives the "You Are Here" pointer through the cognitive map. A route is described by a sequence of instructions in this programming language.

The most fundamental features of an environment are the places it contains: locations to be at, which can be recognized (usually) as being the same on return. A place is primarily characterized, and recognized, by its visual characteristics. The description of a place in the

TOUR model is called a PLACE. The other fundamental features of the environment are the paths or streets, which are described by PATH descriptions.

A given description contains only part of the information which is associated with that geographical feature in the cognitive map. A geographical place may be described with a PLACE description to capture its properties as an indivisible location, and with a REGION description for a greater level of detail. Beyond the structural properties of a large scale space, a person's cognitive map includes a rich sensory image, visual and otherwise, which is associated with a place and helps him recognize it. The PLACE, PATH, and REGION descriptions in the TOUR model contain only those properties that deal with the structure of large-scale space.

A PLACE is a description of a zero-dimensional geographical object: a location or landmark. A PATH is a description of a one-dimensional geographical object: a street or a path. The PLACE description knows the paths that the place is on, and it has a local geometry to describe the relation among those paths. A PATH description includes an order relating the places on that path. This order lets us define a DIRECTION on a path: +1 or -1 meaning "with" or "against" the order on the PATH. The local geometry of a PLACE description is an arbitrary set of radial coordinates, with respect to which the headings of (PATH DIRECTION) pairs can be defined as they leave the place. These descriptions are intended to represent partial knowledge of the geography. Therefore, the order of places on a path may be a partial order, and the description a PLACE has of its local geometry may not be complete. This allows the map to represent certain quite partial states of knowledge, and to assimilate new fragments of information into existing descriptions. {Appendix A summarizes the environmental descriptions presented in this and the following chapters.}

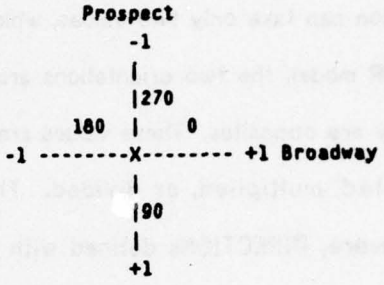
DIRECTION is an orientation defined with respect to a one-dimensional space: a PATH. This orientation can take only two values, which are arbitrarily distinguished from each other. In the TOUR model, the two orientations are represented as +1 and -1, to emphasize the fact that they are opposites. These values are not fully numerical, however, and cannot be added, subtracted, multiplied, or divided. They can only be compared and complemented. Furthermore, DIRECTIONs defined with respect to different PATHs are not comparable. Chapter 3 discusses the analogous concept of HEADING, which is a two-dimensional orientation defined with respect to a regional frame of reference.

Alternatively, one-dimensional orientation on a PATH could be represented by the partial order of places that could be seen in that direction. This representation permits the possibility that the order information associated with a PATH is not a single partial order, but can be different from different viewpoints, or facing in different directions. If there is a single partial order associated with the path, seen from different positions, then the two representations are functionally equivalent.

A description is implemented in LISP as a generated atom, with a certain set of properties whose values are the elements of the description. The properties and values are not accessed directly, but through access functions that can maintain a data structure and make simple antecedent or consequent deductions. An important role for the access function is to screen out duplicate copies from information put into a property. Throughout the TOUR model, inferences are made without checking to see if the result is already known. If so, the new copy is discarded. {Appendix C contains a more detailed discussion of this implementation.}

For example, consider the following description of a perpendicular intersection.

```
PLACE4:
        NAME:  nil
        ON:  [PATH2:  Broadway]
             [PATH3:  Prospect Street]
        STAR:  (0. PATH2 +1.)
               (90. PATH3 +1.)
               (180. PATH2 -1.)
               (270. PATH3 -1.)
```



The generated atom which holds this description is PLACE4. The location being described does not have a name of its own, like "Harvard Square", so its NAME property is empty. This location is the intersection of Broadway and Prospect Street, so the ON property of PLACE4 holds (PATH2 PATH3). The square-bracketed expression [PATH2: Broadway] is simply a helpful way of printing out a reference to the description PATH2. Parenthesized expressions such as (0. PATH2 1.) are LISP list-structures. The STAR property of PLACE4 contains the description of the local geometry of that PLACE. The radial directions are labelled clockwise, but are otherwise arbitrary and local to this PLACE. Their absolute values are irrelevant, since they are used only to compute angular differences (modulo 360) to deduce the results of turns. Just as the absolute value of the radial direction is arbitrary and irrelevant, so is the absolute choice of +1 or -1 for DIRECTION on a street.

In this case, if I am at the above intersection, on Broadway, facing in the -1 direction, the TOUR machine represents my position in the "You Are Here" pointer:

```
YOU ARE HERE:
        PLACE:  [PLACE4:  "Broadway & Prospect Street"]
        PATH:  [PATH2:  Broadway]
        DIRECTION:  -1.
```

If the description of Broadway is:

```
PATH2:
        NAME:  Broadway
        ROW:  ([PLACE1:  Harvard Square]
               [PLACE4:  "Broadway & Prospect Street"]
               [PLACE6:  Kendall Square])
              ([PLACE1:  Harvard Square]
               [PLACE7:  "Broadway & Inman Street"]
               [PLACE6:  Kendall Square])
```

then a DIRECTION of -1 might be described as "facing Harvard Square." The local geometry of PLACE4, described in its STAR property, makes it possible to predict the effect of a particular turn on the state of the TOUR machine. For example, the result of a right turn on the above state of the "You Are Here" pointer is to leave it at:

```
YOU ARE HERE:
        PLACE:  [PLACE4:  "Broadway & Prospect Street"]
        PATH:  [PATH3:  Prospect Street]
        DIRECTION:  -1.
```

The PATH description of a street contains a partial order representing partial knowledge of the total order which actually relates PLACEs on that PATH. For example, the above description of Broadway knows about four PLACEs, with partial knowledge of their relative order. The intersections of Broadway with Inman Street and Prospect Street are known to be between Harvard Square and Kendall Square, but their relation to each other is unknown, as shown in the partial order diagram below.

The access functions for the ROW property of a PATH description will add lists of PLACEs in the +1 direction to the partial order. If asked about two PLACE descriptions, they will tell whether their relative order is +1, -1, or unknown. In this case, the DIRECTION on PATH2 from PLACE7 to PLACE6 is +1, but the relation of PLACE4 to PLACE7 is unknown. The

particular data structure used to implement a partial order is not important, of course. The same external behavior of the access functions could be supported by quite different implementations.

The following schematic definitions for descriptions show what properties they can have, and what kinds of values each property can take. As more properties are introduced, these definitions will be expanded. The current PLACE and PATH descriptions are:

```
PLACE:
        NAME:  <name>
        ON:  <list of PATHs>
        STAR:  <local geometry data-structure>

PATH:
        NAME:  <name>
        ROW:  <partial order data-structure>
```

A NAME description represents the correspondence between its printed version and the environmental description(s) it refers to.

```
NAME:
        PNAME:  <list of atoms>
        NAME-OF:  <list of descriptions>
```

If the NAME-OF property holds more than one description, the natural language interface looks for contextual clues to select the appropriate referent.

The TOUR instructions: GO-TO and TURN

Knowledge of routes through an environment form an important part of a person's cognitive map. In the TOUR model, a route is a sequence of instructions to the TOUR machine, each of which instructs it to move the "You Are Here" pointer in a certain way. The two basic TOUR instructions are GO-TO and TURN, which bear a strong resemblance to their LOGO

counterparts, but with some important differences. First, a LOGO instruction specifies a change to be made to the assumed current state of the robot. A fully specified TOUR instruction specifies the initial state, the change to be made, and the resulting state of the "You Are Here" pointer. Second, a TOUR instruction need not be fully specified for TOUR to be able to execute it, while an underspecified LOGO instruction has no meaning. Third, a LOGO instruction specifies an action to be executed, while the TOUR instruction specifies a state to be achieved. The full forms of the two basic TOUR instructions are:

```
GO-TO:                    TURN:
        FROM: <place>             AT: <place>
        TO: <place>               ST1: <path>
        ON: <path>                DIR1: <direction>
        DIR: <direction>          AMT: <number of degrees>
                                  ST2: <path>
                                  DIR2: <direction>
```

GO-TO instructs TOUR to move the "You Are Here" pointer from one PLACE to another on the same PATH, and specifies the direction of travel with respect to the PATH order. TURN instructs TOUR to move the "You Are Here" pointer from one (PATH DIRECTION) pair to another at the same PLACE, and specifies the amount of the turn. {Appendix A summarizes the TOUR instructions.} A route is simply a sequence of instructions that TOUR can treat as a program, and is stored in the map so that it can be found when needed later: in an association list in the CONNECT property of its source, associated with its destination. The formats of the PLACE and ROUTE descriptions are as follows (new parts underlined).

```
PLACE:
        NAME: <name>
        ON: <list of PATHs>
        STAR: <local geometry data-structure>
        CONNECT: <list of pairs: (PLACE route) >

ROUTE:
        FROM: <PLACE>
        TO: <PLACE>
        SEQUENCE: <list of TOUR instructions>
```

Adding to the cognitive map

The amount of knowledge contained in the map most frequently increases by the addition of a new PLACE or PATH. Typically, this is done by the natural language interface as it translates the noun-phrases in a sentence into references into the cognitive map, creating new descriptions when no existing referent can be found. A newly created description has no spatial properties: perhaps a name. It acquires them from context in the route description, and from subsequent references. Thus, the cognitive map is effectively unbounded by being implicitly self-extending upon demand.

An individual TOUR instruction can be substantially underspecified. For example, the English command "Turn right" translates into a TOUR instruction containing only the amount of the turn. The current state can be obtained from the "You Are Here" pointer, and the destination PATH and DIRECTION can be obtained by examining the local geometry of the current PLACE. Naturally, it is not always possible to fill in all the gaps in the current instruction, the "You Are Here" pointer, or the current parts of the map. TOUR can move the "You Are Here" pointer along a route which is quite underspecified even when it is unable to fill in all the gaps, but there will be information about the geography implicit in the route description, which cannot then be transferred to the map.

Since the program executed by the TOUR machine can become more fully specified than the one it was given, there can be side-effects on the input route description as well as on the "You Are Here" pointer and the map. This is important because often not all of the information in the route description can be assimilated into the map on a single pass. The

knowledge about the environment which is only in the route description is not lost, but remains useful for its particular route, and can be found under the CONNECT property of its source, indexed under its destination. When the map has reached the proper state, the assimilation process can extract the remaining knowledge for the map, or fill in missing parts of the route description from the map. Thus, the role of repetition in the TOUR model is not simply to strengthen associations, but to reexamine observed information in a changed map context.

The TOUR instructions: TAKE and GET-TO

A route program is underspecified when the current position presupposed by an instruction is different from the "You Are Here" pointer. For example, if I am in Kendall Square, desiring a route to Harvard Square, and the proposed route is "Take Mass Ave from Central Square," the assumption is that I can find the route to Central Square. In this case, TOUR formulates the difference description as a GET-TO instruction that calls the problem-solving component to propose a solution. The proposed solution to such a problem may itself be an underspecified route, with smaller gaps to be filled in their turn.

We can now introduce two new TOUR instructions: TAKE and GET-TO. The TAKE instruction refers to a route, and lets it be used as a sub-program for a longer route. This can give a route program a hierarchical structure, and it allows frequently-used routes to be shared. The GET-TO instruction formulates a problem to be solved by the problem-solving component by specifying the state of the "You Are Here" pointer at source and destination. In case no solution can be found, the GET-TO instruction is left in the route description, and TOUR

continues from its destination. The forms of these instructions are:

```
TAKE:                          GET-TO:
        ROUTE:  <route>                FROM:  <place>
        FROM:   <place>                ST1:   <path>
        TO:     <place>                DIR1:  <direction>
                                       TO:    <place>
                                       ST2:   <path>
                                       DIR2:  <direction>
```

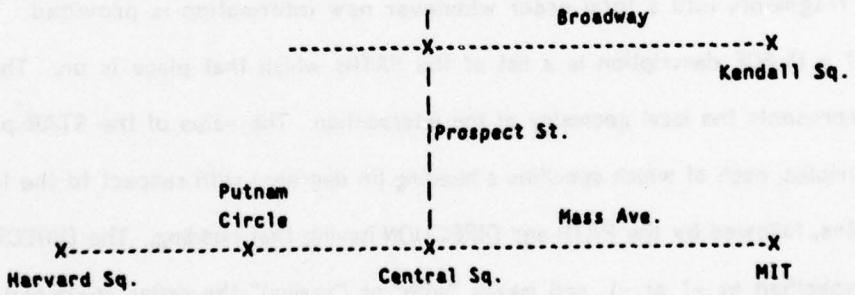
Before forging onward and adding new properties to the map and to TOUR, let us take stock of where we are. The representations for PLACEs and PATHs, which make up the map, have been described. The instructions to TOUR which make up route programs have been presented. TOUR and the "You Are Here" pointer have been described and their properties discussed. In addition, the interface to a problem-solver has been discussed. What has been presented thus far is sufficient to support an example of assimilation of knowledge into the map from a route description.

A cognitive map made from the representation as presented so far would consist primarily of stored routes, each quite well specified, with some topological knowledge about the PATHs and PLACEs involved, but few if any powerful problem-solving methods. A known route can be found under the CONNECT property of its source, indexed under its destination. Long routes can be built from shorter ones linking an intermediate goal to the desired source and destination, but the search for an appropriate intermediate goal can potentially be very long, so this is often not a practical problem-solving technique. The knowledge described in subsequent chapters will restrict this search substantially. Such a mental map would serve a person well in a situation that did not require knowledge of a huge geography, or rapid mastery or discovery of new routes.

An example of TOUR in operation

Here I present an example of the operation of the TOUR machine on a simple route, to show how information is extracted from the route program and assimilated into the PLACEs and PATHs that make up the map. The example is very simple indeed, so it is important to notice that the assimilation methods used are independent of the size of the data base. They use only information directly accessible from the current instruction and the "You Are Here" pointer, and thus would work in exactly the same way within a much larger map.

For this example of assimilation, the TOUR model will begin with a very limited knowledge of the part of Cambridge, Massachusetts shown below. It begins the example located at the intersection of Broadway and Prospect Street.



The initial state of the cognitive map for this example is listed below. Notice that the relation between Mass Ave and the intersection of Broadway and Prospect Street is not known. Neither does the cognitive map contain any description of Putnam Circle or Kendall Square.

```
PATH1:
        NAME:  Mass Ave
        ROW:  (PLACE1 PLACE2 PLACE3)

PATH2:
        NAME:  Broadway
        ROW: (PLACE4)

PATH3:
        NAME:  Prospect Street
        ROW:  (PLACE4)

PLACE1:
        NAME:  Harvard Square
        ON:  [PATH1:  Mass Ave]
        STAR:  nil

PLACE2:
        NAME:  Central Square
        ON:  [PATH1:  Mass Ave]
        STAR:  (0. PATH1 -1)
               (180. PATH1 +1)

PLACE3:
        NAME:  MIT
        ON:  [PATH1:  Mass Ave]
        STAR:  nil

PLACE4:
        NAME:  nil
        ON:  [PATH2:  Broadway]
             [PATH3:  Prospect Street]
        STAR:  nil
```

Recall that a PATH description contains a partially ordered set of PLACEs under its ROW property, as a list of totally ordered fragments. The access functions in this case attempt to merge the fragments into a total order whenever new information is provided. The ON property of a PLACE description is a list of the PATHs which that place is on. The STAR property represents the local geometry of the intersection. The value of the STAR property is a list of triples, each of which specifies a heading (in degrees) with respect to the local set of coordinates, followed by the PATH and DIRECTION having that heading. The DIRECTION on a PATH is specified as +1 or -1, and means "with" or "against" the order contained in that PATH's ROW property. Thus it is a conventional attribute of a PATH, and need not correspond to any global property.

The street directions

This example will follow the knowledge states involved in the assimilation of the route description given below. A new piece of information in a description will be underlined. A recapitulation with minor variations will illustrate further properties.

The example will begin with the "You Are Here" pointer at the intersection of Broadway and Prospect Street, on Prospect Street, with an unknown DIRECTION.

```
YOU ARE HERE:
        PLACE:  [PLACE4:  "Broadway & Prospect Street"]
        PATH:  [PATH3:  Prospect Street]
        DIRECTION:  nil
```

"Take Prospect Street to Central Square." (1)
"Turn right." (2)
"Take Mass Ave to Putnam Circle." (3)

The street directions are given in a subset of English. They are translated in a straight-forward way to instructions for the TOUR machine. {Note LINGOL} The two TOUR instructions that we shall be concerned with here are GO-TO and TURN. Thus, sentences (1) - (3) are translated into the internal instructions below. The amount of a turn is given as the clockwise change in heading (modulo 360). {Note Quantities}

```
GO-TO:                                                    (4)
        FROM:  nil
        TO:  [PLACE2:  Central Square]
        ON:  [PATH3:  Prospect Street]
        DIR:  nil

TURN:                                                     (5)
        AT:  nil
        ST1:  nil
        DIR1:  nil
        AMT:  90.
        ST2:  nil
        DIR2:  nil
```

```
GO-TO:                                                    (6)
        FROM:  nil
        TO:  [PLACE5:  Putnam Circle]
        ON:  [PATH1:  Mass Ave]
        DIR:  nil
```

while the input interface automatically creates a new place-description for instruction (6):

```
PLACE5:
        NAME:  Putnam Circle
        ON:  nil
        STAR:  nil
```

Inside the TOUR machine

All three kinds of description that we have seen can be partially specified: the map descriptions of PATH and PLACE, the "You Are Here" pointer, and the instructions to the TOUR machine. The TOUR machine communicates information between several different representations of spatial knowledge by filling in underspecified instructions and descriptions when possible. When executing an instruction, TOUR matches the current position in the "You Are Here" pointer against the current state presupposed by the instruction. Then it checks the instruction for consistency with the map, and changes the "You Are Here" pointer to represent the new position. Information in the map can fill unspecified parts of the instruction. A fully specified instruction can contribute new information to the "You Are Here" pointer or to the map, adding to the order information in a PATH or the local geometry of a PLACE.

Internally, the TOUR machine is organized as a production system [Newell & Simon 1972], consisting of modular inference rules that are activated on combinations of features of the current environment. This computational environment consists of the current instruction, the

"You Are Here" pointer, and the parts of the map they point to. The productions transfer pieces of information from one description to another, resulting in a three-way interaction among the parts of the environment.

To describe these productions, and the patterns they look for, two kinds of informal syntax are called for. First, to refer to a property of a description, I will write "<description>/<property>", referring to the value if I am reading it, or the place the value is stored, if I am writing it. In either case, access is actually mediated by access functions. Thus, in the preceding database, I may say either, "Put 'Putnam Circle' into PLACE5/NAME," or "The value (or contents) of PLACE5/NAME is 'Putnam Circle'." Furthermore, "/" is left-associative, so that "TURN/AT/ON" refers to the value of the ON property of the PLACE which is the value of the AT property of the current TURN instruction. Thus if the current instruction is a TURN taking place at the intersection of Broadway and Trowbridge Street, TURN/AT is PLACE4 and PLACE4/ON holds PATH2 and PATH3. (Rest assured that the syntax is harder to explain than to read: it means pretty much what you think it does.) The second kind of informal syntax allows me to abbreviate references to parts of the "You Are Here" pointer by referring to the values of C-PLACE, C-PATH, and C-DIRECTION (for "current place", etc.).

Each time an instruction is to be executed, the productions associated with the current type of instruction are activated in sequence. Unless one of the productions explicitly restarts the execution cycle each production is given a chance to act. A production may detect a conflict between the "You Are Here" pointer and the initial state assumed by the current instruction, insert a missing instruction into the route description, and restart the execution cycle. The ordering of the productions is significant. The TOUR machine will function under a variety of

different orderings of its productions, but with quite different behavior. This phenomenon may contribute to an explanation of variation in spatial cognitive style in humans. {Note Variation}

The productions that implement this behavior are given below, in a "formal" version of English. GO-1, GO-2, and GO-3 compare the description of the current state in the "You Are Here" pointer with the current state assumed by the GO-TO instruction, and fill in missing elements. GO-4 and GO-5 make the topological connection between PATHs and PLACEs implied by a GO-TO instruction. GO-5 also adds to the order knowledge represented in the PATH description. GO-6 transfers order information the other way: from the PATH description to the GO-TO instruction. GO-7 performs a very limited "look-behind" to update the previous instruction with DIRECTION information deduced by this instruction. And finally, GO-8 actually accomplishes the effect of the instruction on the "You Are Here" pointer.

GO-1: Compare GO-TO/FROM and C-PLACE. If both are empty, there is an error. If they agree, do nothing. If only one is supplied, set the other from its value. If they disagree, insert a GET-TO instruction as the new current instruction preceding this GO-TO, and restart.

GO-2: Compare GO-TO/PATH and C-PATH. If both are empty, there is an error. If they agree, do nothing. If only one is supplied, set the other from its value. If they disagree, insert a TURN instruction, and restart.

GO-3: Compare GO-TO/DIR and C-DIRECTION. If they agree, do nothing. If only one is supplied, use its value to set the other. If they disagree, insert a "Turn around" instruction.

GO-4: Send GO-TO/PATH to GO-TO/FROM/ON and GO-TO/TO/ON. (A PLACE description checks internally for redundant information.)

GO-5: If GO-TO/DIR is supplied, send GO-TO/FROM and GO-TO/TO as an ordered pair to GO-TO/PATH/ROW. If not, send them as two singletons. (The PATH description will incorporate this information into its partial order.)

GO-6: Ask GO-TO/PATH if it knows an order on GO-TO/FROM and GO-TO/TO. If not, do nothing. If so, and GO-TO/DIR is empty, set it. If so, and it disagrees with GO-TO/DIR, complain of an error. Otherwise, do nothing.

GO-7: If GO-TO/DIR is supplied, and the preceding instruction was a TURN with empty TURN/DIR2, then set that TURN/DIR2 to the value of GO-TO/DIR. (Note Look-behind)

GO-8: Set C-PLACE to GO-TO/TO, and C-DIRECTION to GO-TO/DIR.

A similar set of productions perform the action of the TOUR machine on a TURN instruction. TURN-1, TURN-2, and TURN-3 make the comparison between the "You Are Here" pointer and the current TURN instruction. TURN-4 makes the topological connection between the PATHs and PLACEs involved. TURN-5 transfers knowledge about the local geometry of the intersection from a fully specified TURN instruction to TURN/PLACE/STAR. TURN-6 and TURN-7 use the local geometry of the intersection to provide missing information in the TURN instruction. TURN-8 represents the default heuristic that a turn at a two-street intersection usually results in your being on the other street. TURN-9 provides a blank description in case all else fails to identify the street the turn puts you on. TURN-10 finally accomplishes the effect of the TURN instruction on the "You Are Here" pointer.

TURN-1: Compare TURN/PLACE with C-PLACE. If both are empty, there is an error. If they agree, do nothing. If only one is supplied, set the other from its value. If they disagree, insert a GET-TO instruction as the new current instruction preceding this TURN, and restart.

TURN-2: Compare TURN/ST1 with C-PATH. If both are empty, there is an error. If they agree, do nothing. If only one is supplied, set the other from its value. If they disagree, insert a second TURN instruction before this, comment on the oddness, and restart.

TURN-3: Compare TURN/DIR1 with C-DIRECTION. If they agree, do nothing. If only one is supplied, set the other from its value. If they disagree, there is an error.

TURN-4: Send TURN/ST1 and TURN/ST2 to TURN/PLACE/ON. Also, send a singleton order containing TURN/PLACE to TURN/ST1/ROW and TURN/ST2/ROW.

TURN-5: If TURN/ST1, TURN/DIR1, TURN/ST2, TURN/DIR2, and TURN/AMT are all supplied, then send them to TURN/PLACE/STAR, to add to the description of the local geometry of the PLACE. (The PLACE description incorporates this into the appropriate data structure.)

TURN-6: If TURN/ST1, TURN/DIR1, and TURN/AMT are all supplied, but TURN/ST2 and TURN/DIR2 are not both supplied, then ask TURN/PLACE/STAR to compute the description of the outgoing PATH from the incoming PATH and the amount of the TURN. If that is successful, use the result to set TURN/ST2 and TURN/DIR2.

TURN-7: If TURN/ST1, TURN/DIR1, TURN/ST2, and TURN/DIR2 are all supplied, but TURN/AMT is not, then ask TURN/PLACE/STAR to compute it. If that is successful, set TURN/AMT to the result.

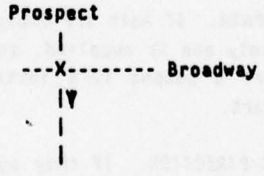
TURN-8: If TURN/ST2 remains unfilled, and TURN/PLACE/ON contains two PATH descriptions, one of which is TURN/ST1, then set TURN/ST2 to the other.

TURN-9: If TURN/ST2 still remains unfilled, then create a new PATH description to fill TURN/ST2. Send it to TURN/PLACE/ON, and send a singleton containing TURN/PLACE to the ROW property of the new PATH. (This PATH description may describe the same real street as a previously created PATH description, requiring some debugging later.)

TURN-10: Set C-PATH to TURN/ST2, and set C-DIRECTION to TURN/DIR2.

The productions which implement TAKE and GET-TO for the TOUR machine are not included here because they are not needed for the example. They can be found in Appendix B.

The first sentence



Sentence (1) is translated into instruction (4):

"Take Prospect Street to Central Square." (1)
GO-TO: (4)
FROM: nil
TO: [PLACE2: Central Square]
ON: [PATH3: Prospect Street]
DIR: nil

GO-1 provides information about the current position, producing:

```
GO-TO:                                                    (4.1)
        FROM:  [PLACE4:  "Broadway & Prospect Street"]
        TO:  [PLACE2:  Central Square]
        ON:  [PATH3:  Prospect Street]
        DIR:  nil
```

When executing a GO-TO instruction, GO-5 would like to send order information about PLACE2 and PLACE4 to PATH3. The instruction, however, does not specify their relative order, so the two fragments (PLACE2) and (PLACE4) are sent to PATH3. PATH3, via its access functions, can now define an order on the places it knows about. PLACE2 and PLACE4 are also told that PATH3 passes through them. The changed descriptions in the map are:

```
PATH3:                              PLACE2:
        NAME:  Prospect Street              NAME:  Central Square
        ROW:  (PLACE4 PLACE2)               ON:  [PATH1:  Mass Ave]
                                                 [PATH3:  Prospect Street]
                                            STAR:  (0. PATH1 -1)
                                                   (180. PATH1 +1)
```

PATH3 now has a defined direction, so GO-6 can deduce the direction of travel missing from instruction (4.1), to produce:

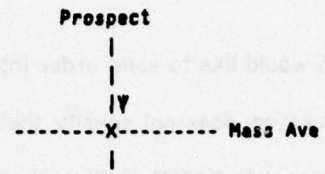
```
GO-TO:                                                    (4.2)
        FROM:  [PLACE4:  "Broadway & Prospect Street"]
        TO:  [PLACE2:  Central Square]
        ON:  [PATH3:  Prospect Street]
        DIR:  +1
```

and GO-8 changes the "You Are Here" pointer to:

```
YOU ARE HERE:
        PLACE:  [PLACE2:  Central Square]
        PATH:  [PATH3:  Prospect Street]
        DIRECTION:  +1
```

{Note Diagrams}

The second sentence



The next sentence is:

```
"Turn right."                                                    (2)
TURN:                                                            (5)
        AT:  nil
        ST1:  nil
        DIR1:  nil
        AMT:  90.
        ST2:  nil
        DIR2:  nil
```

TURN-1, TURN-2, and TURN-3 supply the initial state from the "You Are Here" pointer to produce:

```
TURN:                                                            (5.1)
        AT:  [PLACE2:  Central Square]
        ST1:  [PATH3:  Prospect Street]
        DIR1:  +1
        AMT:  90.
        ST2:  nil
        DIR2:  nil
```

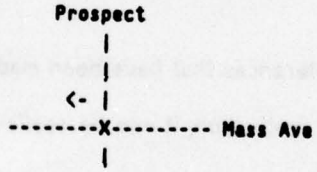
TURN-4 adds no new information to the map because the connection between PLACE2 and PATH3 is already known, and there is insufficient information in (5.1) for TURN-5 to add anything to the STAR property on PLACE2. TURN-8 then jumps to a conclusion about the street turned onto. The ON property of PLACE2 contains PATH1 and PATH3, and since the turn is made from PATH3, the destination of the turn is assumed to be PATH1. Not enough information is available to deduce the direction on PATH1. The final result is:

```
TURN:                                                    (5.2)
        AT:  [PLACE2:  Central Square]
        ST1:  [PATH3:  Prospect Street]
        DIR1:  +1
        AMT:  90.
        ST2:  [PATH1:  Mass Ave]
        DIR2:  nil
```

and

```
YOU ARE HERE:
        PLACE:  [PLACE2:  Central Square]
        PATH:  [PATH1:  Mass Ave]
        DIRECTION:  nil
```

The third sentence




```
"Take Mass Ave to Putnam Circle."                        (3)
GO-TO:                                                   (6)
        FROM:  nil
        TO:  [PLACE5:  Putnam Circle]
        ON:  [PATH1:  Mass Ave]
        DIR:  nil
```

Again, GO-1 provides the initial position, and GO-2 confirms that we are on the right street.

This produces:

```
GO-TO:                                                   (6.1)
        FROM:  [PLACE2:  Central Square]
        TO:  [PLACE5:  Putnam Circle]
        ON:  [PATH1:  Mass Ave]
        DIR:  nil
```

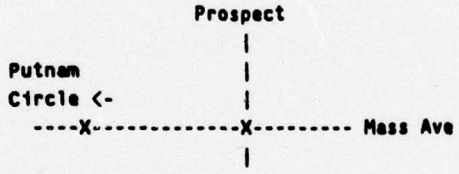
Exactly as in the first sentence, GO-5 can send only minimal order information to PATH1. This

corresponds to knowing that Putnam Circle is on Mass Ave, but not knowing where. The result on the map is:

```
PATH1:                                  PLACE5:
        NAME:  Mass Ave                         NAME:  Putnam Circle
        ROW:  (PLACE1 PLACE2 PLACE3)            ON:  [PATH1:  Mass Ave]
              (PLACE5)                          STAR:  nil
```

GO-6 cannot deduce the direction of travel for this instruction, and GO-8 finally moves the *"You Are Here" pointer.*

```
YOU ARE HERE:
        PLACE:  [PLACE5:  Putnam Circle]
        PATH:  [PATH1:  Mass Ave]
        DIRECTION:  nil
```



The route-description, extended by the inferences that have been made by TOUR, is saved for later use. Indexed under its source and destination, it can be retrieved when needed again. The final version of the route-description is:

```
GO-TO:                                                          (4.2)
        FROM:  [PLACE4:  "Broadway & Prospect Street"]
        TO:  [PLACE2:  Central Square]
        ON:  [PATH3:  Prospect Street]
        DIR:  +1

TURN:                                                           (5.2)
        AT:  [PLACE2:  Central Square]
        ST1:  [PATH3:  Prospect Street]
        DIR1:  +1
        AMT:  90.
        ST2:  [PATH1:  Mass Ave]
        DIR2:  nil

GO-TO:                                                          (6.1)
        FROM:  [PLACE2:  Central Square]
        TO:  [PLACE5:  Putnam Circle]
        ON:  [PATH1:  Mass Ave]
        DIR:  nil
```

Because each production makes only a limited, local deduction, one pass over the route description may not extract all of the available information. For example, this route-description includes the fact that, wherever Putnam Circle is located on Mass Ave, a right turn at Central Square from Prospect Street points you in that direction. This fact is not captured by the partial order in PATH1/ROW. A later assimilation of this route may be able to extract that information for the map, once the local geometry of PLACE2 is more fully specified. This possibility is examined in the second alternative below.

Further examples

It will illuminate the execution of a route by TOUR to discuss briefly a couple of alternate versions of the above example, and show how the result is different. First, consider the same route-description, but with sentence (3) replaced by:

```
"Take Mass Ave to Harvard Square."                    (3.1)
```

Since the destination was previously known, GO-6 can deduce the direction of travel for (6.2), below, from PATH1/ROW. Then GO-7 can reach back to set the preceding TURN/DIR2, to produce (5.3). This results in the following route description.

```
GO-TO:                                                (4.2)
        FROM:  [PLACE4:  "Broadway & Prospect Street"]
        TO:  [PLACE2:  Central Square]
        ON:  [PATH3:  Prospect Street]
        DIR:  +1

TURN:                                                 (5.3)
        AT:  [PLACE2:  Central Square]
        ST1:  [PATH3:  Prospect Street]
        DIR1:  +1
        AMT:  90.
        ST2:  [PATH1:  Mass Ave]
        DIR2:  -1
```

```
GO-TO:                                                    (6.2)
        FROM:  [PLACE2:  Central Square]
        TO:  [PLACE5:  Putnam Circle]
        ON:  [PATH1:  Mass Ave]
        DIR:  -1
```

The TURN instruction is now fully specified, so the next time TURN-5 is applied, on a second pass over the route, the local geometry of PLACE2 can be augmented:

```
PLACE2:
        NAME:  Central Square
        ON:  [PATH1:  Mass Ave]
             [PATH3:  Prospect Street]
        STAR:  (0. PATH1 -1)
               (90. PATH3 -1)
               (180. PATH1 +1)
```

This additional knowledge about PLACE2 will increase the power of TURN-6 and TURN-7 to deduce the result of incompletely specified TURN instructions at PLACE2.

Consider now a second alternative to the original example. Assume that sentence (2) were:

```
"Turn right onto Mass Ave toward Harvard Square."         (2.1)
```

Thus, after sentence (2.1),

```
YOU ARE HERE:
        PLACE:  [PLACE2:  Central Square]
        PATH:  [PATH1:  Mass Ave]
        DIRECTION:  -1
```

and

```
TURN:                                                     (5.4)
        AT:  [PLACE2:  Central Square]
        ST1:  [PATH3:  Prospect Street]
        DIR1:  +1
        AMT:  90.
        ST2:  [PATH1:  Mass Ave]
        DIR2:  -1
```

and TURN-5 adds an item to the local geometry of PLACE2 as described immediately above. Now, when instruction (6) is executed, GO-1 and GO-3 can fill GO-TO/FROM and GO-TO/DIR,

leaving (6.3) fully specified.

```
GO-TO:                                                    (6.3)
        FROM: [PLACE2:  Central Square]
        TO:  [PLACE5:  Putnam Circle]
        ON:  [PATH1:  Mass Ave]
        DIR:  -1
```

This, in turn, will allow GO-5 to send the ordered pair (PLACE5 PLACE2) to PATH1.

```
PATH1:
        NAME:  Mass Ave
        ROW:  (PLACE1 PLACE2 PLACE3)
              (PLACE5 PLACE2)
```

This is a much more elaborately specified partial order, ripe for the addition of the item (PLACE1 PLACE5), which will linearize the order once more.

Summary

The TOUR model creates a topological description of the environment in terms of places, paths, and routes. The most important knowledge associated with a PLACE description is the topological connection among intersecting streets, and their local geometry. The PATH description contains order information relating the places on it. A route description consists of a sequence of GO-TO, TURN, TAKE, and GET-TO instructions. All of these descriptions can be underspecified in important ways. A route description is stored in the cognitive map, associated with its source and destination.

The example shows how the TOUR machine takes an underspecified route-description, and executes it to drive the "You Are Here" pointer through the map. The productions which constitute this part of the TOUR machine, and the way they act to assimilate information, have been shown. In addition, the example shows how small variations in the correspondence

between the route-description and the cognitive map make a large difference in the amount of knowledge that can be assimilated at that time. However, nothing from the route-description is discarded: it simply waits in the relatively inaccessible route representation until the map satisfies the conditions of the inference rules that will assimilate it.

Two additional important facts are worth emphasizing. The first is that the assimilation process uses only locally available information, with no search of the map. Thus, assimilation proceeds independently of the size of the database containing the map. Second, assuming that the route description is correct, the description in the cognitive map may be partially specified, but it will not be wrong. The well-known inaccuracies in people's mental maps lie in the more global kinds of description which are presented in the following chapters. In those cases, global conclusions are drawn from local observations, so they are quite vulnerable to error.

III. ORIENTATION

> *Orient. v. tr. 1. To locate or place in a particular relation to the points of the compass:* Orient the swimming pool north and south. *2. To cause to face the east; locate or place so as to face the east. 3. To align or position with respect to a reference system. 4. To discover the bearings of. Often used reflexively:* He oriented himself by finding a familiar landmark. *5. To cause to become familiar with or adjusted to facts, principles, or a situation.*
>
> *[American Heritage Dictionary, 1973]*

Each of the next three chapters will show how a different kind of spatial knowledge can be described in terms of the TOUR model, with emphasis on the states of partial knowledge each representation can maintain. Each of these kinds of knowledge is largely based on the topological representation of Chapter 2, and they are largely independent of each other. This chapter discusses orientation and the concept of a "sense of direction." Chapter 4 deals with partial knowledge of position. Chapter 5 shows how structures of containing regions can be used to simplify route-finding and the representation of position.

Examples of orientation

A number of different kinds of knowledge fall under the term "orientation." As the definition above shows, orientation means knowing one's position with respect to some reference system. The reference system which is most often referred to explicitly is the set of cardinal directions: East, North, West, and South. These cardinal directions, however, are but a special case among more general reference systems with respect to which a person can orient himself. I hope to convince the reader by simple anecdotal examples that a person's mental map typically contains a number of more-or-less related reference systems. Some of these

anecdotes are confirmed by the literature in psychology and urban planning. The anecdotes provide a subjective definition of the phenomena which the TOUR model will describe. The bulk of this chapter is devoted to explaining the aspects of the TOUR model which describe the following characteristics of orientation and reference systems.

1. Position is often given as a relation between two places and a reference system, not an absolute property of a single place. For example, as I sit in my office at MIT, I can say, "Harvard Square is two miles west of here," or "As I face Central Square, Harvard Square is two miles ahead and to my right." Either statement defines the position of Harvard Square with respect to my current position and some reference system. However, saying only "Harvard Square is two miles away," leaves the position underspecified.

2. There are many reference systems with respect to which positions may be defined. The above example uses the cardinal directions and an egocentric reference system tied to my current position and heading. To use the latter for communication, my audience must know where I am. Often I may define my position with respect to a reference system which is tied to the environment, but which is different from the system of cardinal directions. For example, when a set of streets is laid out as a grid, as in Manhattan, I can define positions with respect to the reference system of the grid, even when I am ignorant of the cardinal directions.

3. The relation between different reference systems may be unknown. It is a common phenomenon to be quite well oriented within each of two districts, but not know the relation between them. For example, Back Bay in Boston is a rectangular grid, as is most of Palo Alto, California. Neither, however, is aligned with the cardinal directions, and even a person familiar

with both must pause to compute the relation between them. In this particular case, most people compare them by relating each to the cardinal directions and comparing the resulting headings.

4. Many people orient themselves with respect to conspicuous landmarks. In Boston, the Prudential Building and the John Hancock Building provide convenient reference points, visible from most of the city. This gives the observer knowledge of his own heading, a rough notion of the radial direction from the landmarks to his current position, and a still rougher notion of his distance. In different environments, other geographical features can play some or all of these roles. A mountain range or the seashore can often serve as such a landmark, even leading the unwary observer astray when it changes its meaning unexpectedly. For example, the Pacific Ocean is a landmark defining "west" throughout most of California, causing confusion in Santa Barbara where its true direction is south. The position of the sun acts as a time-varying but usually accessible landmark to which the cardinal directions are tied.

5. A reference system need not be tied to a large geographical structure like a mountain or a set of grid-structured streets. It may be created to represent the locations of a small set of nearby but mutually invisible places, whose positions are computed by "dead reckoning" along short routes. For example, if I reach the opposite corner of a rectangular block by following one side, making a right turn, then following a second side, I can locate my starting point as behind me and to my right, with a reasonably accurate notion of the direction and distance. Over longer or more complex routes, the accuracy of this "common-sense trigonometry" degrades considerably.

6. A "sense of direction" seems to be primarily the ability to describe where you are with

respect to other places. Thus, the "You Are Here" pointer must contain information about the current heading. It also means that the heading and the associated reference system provide access to information about the locations of a significant set of places. An additional conversational meaning of the phrase "sense of direction" is that a person has a good ability to acquire and maintain this kind of information in unfamiliar territory.

Describing orientation in the TOUR model

The TOUR model must be expanded somewhat to accomodate these phenomena. The most important addition to the TOUR model is that the "You Are Here" pointer is augmented to include the current HEADING of the traveller with respect to some reference system. A reference system is represented in the cognitive map by a description called an ORIENTATION-FRAME. Since the current HEADING is only meaningful with respect to the current ORIENTATION-FRAME, it too is included in the expanded "You Are Here" pointer. An example of the expanded "You Are Here" pointer might be:

```
YOU ARE HERE:
        PLACE:  [PLACE4:  "Broadway & Prospect Street"]
        PATH:  [PATH2:  Broadway]
        DIRECTION:  +1
        ORIENTATION-FRAME:  ORIENT2
        HEADING:  180.
```

The HEADING can take on integer values from 0 to 359 degrees. {Note Quantities} HEADING is an angular, two-dimensional orientation with respect to the current ORIENTATION-FRAME, and contrasts with DIRECTION, which is a binary specification of orientation with respect to the current PATH (a one-dimensional space). For brevity, the current HEADING and ORIENTATION-FRAME are referred to as C-HEADING and C-ORIENT, respectively.

In Chapter 2, the headings in PLACE/STAR were implicitly defined with respect to a reference

system local to that PLACE. Here that reference system is made explicit and related to more global kinds of orientation.

An ORIENTATION-FRAME is a description of a reference system with respect to which HEADINGs can be defined. It has the following properties:

```
ORIENTATION-FRAME:
        TYPE:  <one of (LOCAL, REGIONAL, CARDINAL) >
        DOMAIN:  <PLACE or REGION>
        OTHERS:  <list of pairs:  (ORIENTATION-FRAME ANGLE) >
```

If ORIENT/TYPE is LOCAL, then the reference system being described is limited to a single PLACE, held in ORIENT/DOMAIN, where it provides the basis for the local geometry. If ORIENT/TYPE is REGIONAL, then the reference system has a more extended domain, which is described by the REGION held in ORIENT/DOMAIN. (REGIONs are described below.) ORIENT/OTHERS contains pairs describing the relationship between this and other reference systems. ANGLE is the number to be added (modulo 360) to a HEADING with respect to this ORIENTATION-FRAME in order to describe the same real-world heading, but with respect to the associated ORIENTATION-FRAME.

A REGION is a description of a geographical area: a collection of PLACEs and PATHs. In this context, it describes the domain of application of an ORIENTATION-FRAME. We will encounter REGIONs again in other contexts, where we will expand the following description:

```
REGION:
        TYPE:  <one of (ORIENT) >                ; to be expanded!
        ORIENT:  <ORIENTATION-FRAME>
        PLACES:  <list of PLACEs>
        PATHS:  <list of PATHs>
```

If REGION/TYPE is ORIENT, then this REGION describes the domain of an ORIENTATION-FRAME, which is held in REGION/ORIENT. PLACEs and PATHs which are in the REGION are listed in REGION/PLACES and REGION/PATHS, respectively. When a place or path is in a region, then

the REGION description is also put into PLACE/IN or PATH/IN. We will encounter other values of REGION/TYPE in later chapters.

Once the current HEADING is known by the "You Are Here" pointer, it is quite straight-forward to maintain it while following a route. If the amount of a TURN is known, the HEADING is updated; a GO-TO instruction leaves the HEADING constant if the street is straight. If these conditions are not met, then C-HEADING is cleared, and the "You Are Here" pointer is less well oriented. We must therefore add a new element to a PATH description to describe its shape. We will define PATH/SHAPE as having a default value of STRAIGHT. Any other value will prevent the HEADING from being maintained along that street. {Note Shape Theory}

> TURN-11: If C-HEADING and TURN/AMT are set, then update C-HEADING by TURN/AMT; otherwise clear C-HEADING and C-ORIENT.
>
> GO-9: If C-HEADING is set, then check C-PATH/SHAPE. If its value is STRAIGHT, then do nothing; else clear C-HEADING and C-ORIENT.

Here we approach an area of great individual variation in spatial cognitive styles. Some people can maintain their orientation along a curving road, while others are disoriented by any turn. It appears likely that many variants can be represented within the TOUR model by replacing GO-9 and TURN-11 with the appropriate substitutes, or by leaving them out altogether.

Interactions between HEADING and DIRECTION

When both HEADING and DIRECTION are supplied, the "You Are Here" pointer is redundant. This means that the relation between them can be stored as part of the description of the associated PATHs and PLACEs. Conversely, information from PATHs and PLACEs can be used

to infer the current HEADING or DIRECTION.

This relationship is represented by adding a new element to the PATH description to represent its HEADING, facing its +1 direction. Because a PATH may have several different HEADINGs, PATH/HEADING may contain several (ORIENTATION-FRAME HEADING) pairs. The current full form of a PATH description is as follows, with newly added parts underlined.

PATH:
NAME: <name>
ROW: <partial order data-structure>
<u>HEADING: <list of pairs: (ORIENTATION-FRAME HEADING) ></u>
<u>SHAPE: <descriptive tag, e.g. STRAIGHT></u>
<u>IN: <list of REGIONs></u>

The TOUR model is also augmented by inference rules to make deductions about these relationships. One set of inferences uses information from the current PLACE or PATH to fill in a missing ORIENTATION-FRAME, HEADING, or DIRECTION element of the "You Are Here" pointer.

TURN-12: If C-ORIENT is empty, set it to C-PLACE/ORIENT.

TURN-13: If C-ORIENT and C-DIR are both supplied, but C-HEADING is empty, and if C-ORIENT matches C-PLACE/ORIENT, then try to set C-HEADING to the heading associated with (C-PLACE C-DIR) in C-PLACE/STAR.

GO-10: If C-ORIENT and C-DIR are both supplied, but C-HEADING is not, then see if C-ORIENT is associated with a heading in C-PATH/HEADING. If so, take C-DIR into account, and set C-HEADING.

GO-11: If C-ORIENT and C-HEADING are both supplied, but C-DIR is not, then check C-PATH/HEADING to see if C-ORIENT is associated with a heading. If so, and it is close to C-HEADING or its opposite, set C-DIR accordingly. If the headings match very poorly, flag a potential error, and go on.

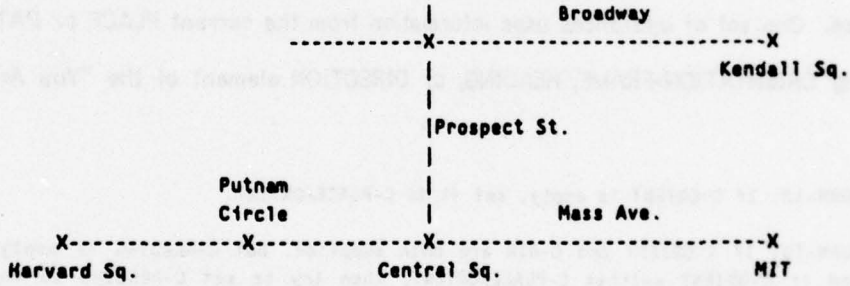
The other set of inferences add information from the "You Are Here" pointer to the current PLACE or PATH.

GO-12: If C-HEADING and C-DIR are both supplied, then let THETA be the heading of C-PATH in the +1 direction. Put (C-ORIENT THETA) into C-PATH/HEADING.

```
TURN-14: If C-HEADING and C-DIR are both supplied, and if C-ORIENT is the
same as C-PLACE/ORIENT, then put (C-HEADING C-PATH C-DIR) into C-PLACE/STAR.
```

An example with HEADING

Let us consider an example in which the HEADING is maintained, and information about it is transferred between map descriptions and the "You Are Here" pointer. This example will follow a route over the territory covered in the previous chapter. In this case, the route-description will be complete at the beginning, so all of the action will take place in the "You Are Here" pointer and in the PLACE and PATH descriptions. The route being followed is shown in the map below:




```
Start on Broadway, at the intersection of Broadway and Prospect Street,
       facing Kendall Square.
Turn right onto Prospect Street.
Take Prospect Street to Kendall Square.
Turn right onto Mass Ave.
Take Mass Ave to Putnam Circle.
```

We begin at the intersection of Broadway and Prospect Street, facing Kendall Square (PLACE6).

```
YOU ARE HERE:
       PLACE:  [PLACE4:  "Broadway & Prospect Street"]
       PATH:  [PATH2:  Broadway]
       DIRECTION:  +1
```

The relevant parts of the map description of this geography are given below.

```
PATH1:
        NAME:  Mass Ave
        ROW:  (PLACE1 PLACE2 PLACE3)
              (PLACE5 PLACE2)
        HEADING:  nil
        SHAPE:  STRAIGHT

PATH2:
        NAME:  Broadway
        ROW:  (PLACE4 PLACE6)            ; Kendall Square
        HEADING:  nil
        SHAPE:  STRAIGHT

PATH3:
        NAME:  Prospect Street
        ROW:  (PLACE4 PLACE2)
        HEADING:  nil
        SHAPE:  STRAIGHT

PLACE2:
        NAME:  Central Square
        ON:  [PATH1:  Mass Ave]
             [PATH3:  Prospect Street]
        STAR:  (0. PATH1 -1)
               (90. PATH3 -1)
               (180. PATH1 +1)
        ORIENT:  ORIENT3

PLACE4:
        NAME:
        ON:  [PATH2:  Broadway]
             [PATH3:  Prospect Street]
        STAR:  (0. PATH2 -1)
               (180. PATH2 +1)
               (270. PATH3 +1)
        ORIENT:  ORIENT2
```

The route we will follow takes two right turns from the starting point, and ends up at Putnam Circle on Mass Ave. Its formal description is:

```
TURN:
        AT:  [PLACE4:  "Broadway & Prospect Street"]
        ST1:  [PATH2:  Broadway]
        DIR1:  +1
        AMT:  90.
        ST2:  [PATH3:  Prospect Street]
        DIR2:  +1

GO-TO:
        FROM:  [PLACE4:  "Broadway & Prospect Street"]
        TO:  [PLACE2:  Central Square]
        ON:  [PATH3:  Prospect Street]
        DIR:  +1

TURN:
        AT:  [PLACE2:  Central Square]
        ST1:  [PATH3:  Prospect Street]
        DIR1:  +1
        AMT:  90.
        ST2:  [PATH1:  Mass Ave]
        DIR2:  -1

GO-TO:
        FROM:  [PLACE2:  Central Square]
        TO:  [PLACE5:  Putnam Circle]
        ON:  [PATH1:  Mass Ave]
        DIR:  -1
```

The result of the first turn is to put the "You Are Here" pointer on Prospect Street facing toward Central Square. At the same time, however, TURN-12 and TURN-13 set the orientation-frame and heading from the local geometry of PLACE4:

```
YOU ARE HERE:
        PLACE:  [PLACE4:  "Broadway & Prospect Street"]
        PATH:  [PATH3:  Prospect Street]
        DIRECTION:  +1
        ORIENTATION-FRAME:  ORIENT2
        HEADING:  270.
```

The following GO-TO takes us to Central Square, and GO-12 puts the current heading into PATH3/HEADING. This information will allow the "You Are Here" pointer to get a heading from PATH3 (via GO-10) on future routes.

```
PATH3:
        NAME:  Prospect Street
        ROW:  (PLACE4 PLACE2)
        HEADING:  (ORIENT2 270.)
        SHAPE:  STRAIGHT
```

The TURN at Central Square puts us on Mass Ave facing Putnam Circle and Harvard Square. TURN-11 updates C-HEADING with respect to the same orientation-frame, even though PLACE2/ORIENT is different.

```
YOU ARE HERE:
        PLACE:  [PLACE2:  Central Square]
        PATH:  [PATH1:  Mass Ave]
        DIRECTION:  -1
        ORIENTATION-FRAME:  ORIENT2
        HEADING:  0.
```

Finally, the GO-TO to Putnam Circle allows GO-12 to give a heading to PATH1.

```
PATH1:
        NAME:  Mass Ave
        ROW:  (PLACE1 PLACE2 PLACE3)
              (PLACE5 PLACE2)
        HEADING:  (ORIENT2 180.)
        SHAPE:  STRAIGHT
```

Simulating observations

In the previous chapter, all contact between the TOUR model and the world it tries to describe takes place through the GO-TO and TURN instructions, which provide information about the topology and the local geometry of intersections. In order to have position and orientation knowledge to represent, however, we need to simulate additional processes whereby people extract information from the environment. People find it quite easy to estimate the distance travelled (perhaps inaccurately), according to the time or effort required. People can also see landmarks in the distance, and incorporate information about distance and direction into their

mental maps.

Estimating distance while travelling is simulated in the TOUR model by adding a distance descriptor to the GO-TO instruction. GO-TO/DIST is filled automatically by a process conceptually separate from the TOUR machine. The value of GO-TO/DIST is an integer number of arbitrary distance-units, corresponding roughly to distance in meters. {Note Quantities}

```
GO-TO:
        FROM:  <place>
        TO:  <place>
        PATH:  <path>
        DIR:  <direction>
        DIST:  <integer>
```

Observing the positions of distant landmarks is simulated in the TOUR model by the NOTICE instruction.

```
NOTICE:
        FROM:  <PLACE>
        PATH:  <PATH>
        DIR:  <direction>
        REMOTE:  <PLACE>
        DISTANCE:  <distance in units>
        HEADING:  <egocentric heading in degrees>
```

The NOTICE instruction says that, if the traveller is at the given place, path, and direction, then the remote place can be observed at the given distance and heading. The heading is defined with respect to egocentric coordinates within which the heading of the observer is 0.

Each of these simulated observations provides information about the environment to the TOUR machine. The information is provided within a single TOUR instruction. Any meaning it acquires beyond that must come from the way the TOUR machine fits it into other structures. The psychologically unrealistic representation of distance as an integer is a defect in this simulation. However, the structure of the TOUR model is not dependent on the representation

of distance, so replacing the integer representation with one more psychologically valid would affect its behavior but not its underlying structure.

Representing position

The position of one place from another consists of two parts: its distance and its heading. The heading must be specified with respect to some reference system. We can use the ORIENTATION-FRAME which defines the local geometry of a PLACE for the positions of remote PLACEs as viewed from this one. The position of a remote PLACE, perhaps obtained from a NOTICE instruction, is stored in the VIEW property of a PLACE description, with other positions, as a triple:

(<remote-place> <heading> <distance>)

The heading is with respect to the ORIENTATION-FRAME kept in PLACE/ORIENT, and so is compatible with the headings of streets radiating from the intersection. The following productions execute the NOTICE instruction to compare its observational information with PLACE/VIEW.

NOTICE-1: Check that NOTICE/PLACE, NOTICE/PATH, and NOTICE/DIR match C-PLACE, C-PATH, and C-DIR. If not, complain and dismiss.

NOTICE-2: If C-ORIENT and C-HEADING are supplied, and if C-ORIENT matches C-PLACE/ORIENT, then look up the position of NOTICE/REMOTE in C-PLACE/VIEW. If none is there, add it from the NOTICE instruction, compensating for egocentric coordinates by using C-HEADING. If one is there, but fails to match the NOTICE instruction, then complain. Otherwise, do nothing.

NOTICE-3: If C-ORIENT is supplied and matches C-PLACE/ORIENT, but C-HEADING is not supplied, and if C-PLACE/VIEW has a position for NOTICE/REMOTE, then use this information to set C-HEADING.

NOTICE-4: If C-ORIENT and C-HEADING are both supplied, and there is a position for NOTICE/REMOTE in C-PLACE/VIEW, but C-ORIENT does not match C-PLACE/ORIENT, then the relation between the two orientation-frames can be

computed and put into C-ORIENT/OTHERS and C-PLACE/ORIENT/OTHERS.

Positions defined with respect to local reference systems are not very useful, however, so the TOUR model allows an ORIENTATION-FRAME to be shared among several PLACEs. The local functions of PLACE/STAR and PLACE/VIEW are not affected, and a shared ORIENTATION-FRAME allows positions from different origins to be comparable. An ORIENTATION-FRAME can be propagated from one PLACE to another along a GO-TO instruction when the appropriate features are present. This region-growing process is therefore controlled both by the geographical features of the environment, and by which paths are frequently travelled. The following productions in the TOUR machine control this propagation.

GO-13: If C-ORIENT, C-HEADING, and C-DIR are all set, and C-ORIENT and C-PLACE/ORIENT are different, then a collision of ORIENTATION-FRAMEs has taken place, and GO-14 must run to decide what to do.

GO-14: Obtain the current heading with respect to C-PLACE/ORIENT. If it cannot be found, do nothing. If it differs from C-HEADING by a multiple of 90 degrees, and if C-PLACE/ORIENT is local to C-PLACE, then run GO-15 and GO-16 to merge it with C-ORIENT. Otherwise, record the relation between the reference systems in C-ORIENT/OTHERS and C-PLACE/ORIENT/OTHERS.

GO-15: If C-ORIENT/TYPE is LOCAL, then promote it to REGIONAL and adjust the appropriate pointers.

GO-16: Put C-ORIENT into C-PLACE/ORIENT. Add C-PLACE to C-ORIENT/DOMAIN/PLACES. Change the headings in C-PLACE/STAR and C-PLACE/VIEW to the appropriate value for C-ORIENT.

Information in ORIENT/OTHERS about the relation between two orientation-frames can be used when a position is found with respect to one orientation-frame and desired with respect to another.

The TOUR model also permits a single reference system which is not associated with a local geographical feature. This allows us to represent positions defined with respect to the

cardinal directions. Since many headings are known with respect to both the cardinal directions and a local orientation frame, it is often easy to compute the relationship of different orientation frames through the cardinal directions. This is represented just as PLACE/VIEW is, but the position triple is kept in PLACE/CARDINAL.

At this point, the full form of the PLACE description is:

```
PLACE:
        NAME: <name>
        ON: <list of PATHs>
        STAR: <local geometry datastructure>
        CONNECT: <list of pairs: (PLACE route) >
        ORIENT: <ORIENTATION-FRAME>
        VIEW: <list of triples: (PLACE heading distance) >
        CARDINAL: <list of triples: (PLACE heading distance) >
        IN: <list of REGIONs>
```

Computing position from a route

One common-sense method for discovering the position of one place from another is by a process of "dead reckoning" while travelling from one to the other. Dead reckoning is a process of integrating along a route by taking into account the distances and turns along the route to maintain the current relative position from the starting place. Thus, at any point along the route, the current relative position from the starting point is known.

This is accomplished in the TOUR model by translating the distance and direction covered by each GO-TO instruction from its natural polar coordinates into rectangular coordinates with respect to the reference system of the starting place. The rectangular coordinates of the current position are kept in two temporary elements of the "You Are Here" pointer, called C-X

and C-Y.

GO-17: If C-X, C-Y, C-HEADING, and GO-TO/DIST are all supplied, then translate C-HEADING and GO-TO/DIST into rectangular coordinates and update C-X and C-Y.

GO-18: If C-X and C-Y are set, but either C-HEADING or GO-TO/DIST is empty, then clear C-X and C-Y.

Although this is an antecedent computation, it is performed to solve problems posed by questions like "Where is Harvard Square from here?" When such a problem is posed, C-X and C-Y are set up in the "You Are Here" pointer and a route is followed for the side-effect of setting C-X and C-Y to the relative position of the destination from the source of the route. At the end of the route, C-X and C-Y are converted back into polar coordinates and the position is stored in *PLACE/VIEW* or *PLACE/CARDINAL*.

Most modern navigation techniques depend on this method, but with instruments designed to supply accurate headings and distance readings, and with precise trigonometric calculations. Many of the inaccuracies in human dead reckoning are due to inaccuracies in angle and distance estimates, and to the failure to compensate correctly for curved paths. Furthermore, the human common-sense method for integrating the segments of the route surely uses something other than precise trigonometric tables. The TOUR model uses exact trigonometry for the lack of a better theory of "common-sense trigonometry." It seems plausible, however, that reasonable estimates could be made by matching any such problem to the closest of a small set of familiar triangles.

On iconic mental images

There has been much controversy about the possibility of storage of information as an iconic

mental image capable of operations isomorphic to those which are performed on physical images. (See Shepard and Metzler (1971) and Pylyshyn (1973).) An important argument against the long-term storage of iconic mental images is the vast amount of storage that would be required, perhaps even overwhelming the enormous capacity of human long-term memory. However, these arguments do not apply to the possibility of a relatively small grid (perhaps 50 by 50 points) that is used, not for long-term storage, but for intermediate computations. This is similar to what Minsky (1975) called the "global space frame."

Relative position computations or array rotations could potentially be performed by translating information from a property-value description to the grid representation, using the grid to measure relative position or perform the rotation, and translating the result back to the property-value form for permanent storage. This method would be computationally much more economical than performing the same operation directly on the original description.

The fundamental operations required of such a grid are: 1) placing two points with a given relative position in rectangular or polar coordinates, 2) reading the relative position of two given points in either rectangular or polar coordinates, and 3) mapping each grid-point to its successor under a fixed elementary rotation. The first two properties permit relative position computations and rectangular-polar conversions, and the third permits array rotation in fixed steps.

Further psychological experimentation is required to establish the actual parameters of such a device. Perhaps it can be shown that one of the above operations is not used, or that an additional one is present. The resolution of the grid could also be somewhat different. The results of Just and Carpenter (1976) suggest that the unit of elementary rotation is about 50

degrees.

Summary

The orientation frame provides a frame of reference for the definition of relative positions. These frames of reference are initially confined to representing the positions of places as viewed from a single vantage point. They gain their power, however, from being shared among a number of different places, so relative positions from different origins can be compared. The example above shows how the local inferences that maintain the heading in the "You Are Here" pointer can result in the propagation of an orientation frame to additional places along a route. Beyond this local propagation of orientation frames, dead reckoning provides a method for discovering the relative positions of the source and destination of a route. Further computations with relative positions might be facilitated by an expert array processor which acts something like an iconic mental image. Other than the iconic array, all of these aspects of the TOUR model have been implemented.

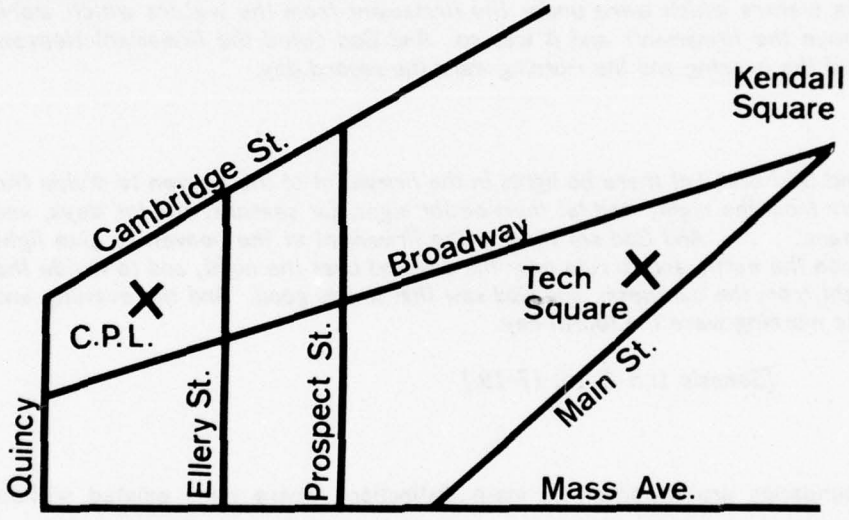
IV. DIVIDING BOUNDARIES

> *And God said, Let there be light: and there was light. And God saw the light, that it was good: and God divided the light from the darkness. And God called the light Day, and the darkness he called Night. And the evening and the morning were the first day.*
>
> *And God said, Let there be a firmament in the midst of the waters, and let it divide the waters from the waters. And God made the firmament, and divided the waters which were under the firmament from the waters which were above the firmament: and it was so. And God called the firmament Heaven. And the evening and the morning were the second day.*
>
> . . .
>
> *And God said, Let there be lights in the firmament of the heaven to divide the day from the night; and let them be for signs, for seasons, and for days, and years. . . . And God set them in the firmament of the heaven to give light upon the earth, and to rule over the day and over the night, and to divide the light from the darkness: and God saw that it was good. And the evening and the morning were the fourth day.*
>
> *[Genesis 1: 3-8, 14, 17-19.]*

Dividing boundaries are important to make distinctions where none existed without them. Thus they are important not just to mark the outlines of something, but to provide it with internal structure. Streets that provide a skeletal structure to a city are acting as dividing boundaries just as much as those which outline it.

Dividing boundaries also provide useful partial descriptions of position. For example, it is quite natural for me, sitting in my office at MIT, to describe the location of the Cambridge Public Library as "beyond Prospect Street, on this side of Mass Ave, between Broadway and Cambridge Street, closer than Quincy Street, past Ellery Street," and so on. With each phrase I add more information to my partial specification of the position.

Furthermore, such a description can be very helpful in finding a route should I want to go to the Cambridge Public Library. Getting to Prospect Street is progress in the right direction, and I should stay between Broadway and Cambridge Street. If I cross Mass Ave or Quincy Street I have gone too far and should turn back. These clues do not specify the paths that should make up my route, but they give criteria for progress and therefore could be used to guide my search.



In each clause of this description, a street acts as a boundary between two regions. Considering a street as a directed line, one can specify whether a place is to the right or the left of the street. This is a small piece of information, and easily acquired. If a route that includes the street goes to the place in question, and the turn made from the street is known, then the place can be located with respect to the street, no matter how tortuous the intervening route. (Provided, of course, that the street does not cross back over itself.)

The position of a given place is very well specified indeed when its location is known with respect to many different boundary streets. They provide a very rich collection of states of

partial knowledge about the position of a place. Furthermore, this qualitative knowledge is very resistant to error, unlike the more quantitative position descriptions of the last chapter.

The area over which a given boundary is meaningful varies considerably. Almost any place in Cambridge can be related to Mass Ave in most people's cognitive maps. Portland Street, however, has meaning only in a small region near Tech Square. The extent of such application is not defined by some second, outer boundary, but only by the travel patterns which are likely to cause boundary relations to be inferred and represented. Thus, position with respect to Mass Ave is widely known because Mass Ave is important to routes involving many different parts of Cambridge. Portland Street is only important to travel within the neighborhood of Tech Square, and so its knowledge spreads no farther.

Conversely, boundary relations are used in the selection of routes. Therefore, since Mass Ave is involved in describing the positions of many places, it will be encountered very frequently in the route-finding process. Portland Street will not. It is the number of relationships with other places that determines what is a main street, not physical size. It is a characteristic of poor or non-existent urban planning when these two concepts diverge, resulting in tiny, congested main streets and magnificent unused boulevards.

Boundaries are also very useful for the exploration of unknown territory. If a traveller knows his own heading and the heading of a familiar boundary (with respect to the same orientation frame), and knows which side of the boundary he is on, he can always steer in a direction which will bring him, sooner or later, back to familiar territory. This technique is available even in the complete absence of a cognitive map of the actual territory being explored.

Naturally, one-dimensional geographical features other than streets can function as boundaries. Rivers, railroad tracks, limited-access highways and other barriers often have this role in a city. The principles that are outlined below for streets apply equally well to other boundaries, except that their properties are not discovered by physical travel along them. The implemented version of the TOUR model considers streets as the only boundaries, but there is nothing in the representation which prohibits the inclusion of barriers as well.

Representing position with dividing boundaries

Since a PATH has a one-dimensional orientation, we can face its +1 direction and distinguish between the "right" region and the "left" region, kept in PATH/RIGHT and PATH/LEFT, respectively. The position of a PLACE is partially specified by putting it into the appropriate REGION. A REGION in one of these properties has a TYPE of BOUNDED, and a BOUNDARY property which describes its boundary. The BOUNDARY property of a REGION description contains a set of pairs (PATH DIRECTION), and means that the region is on the right of that PATH when travelled in that DIRECTION. (The current implementation does not explore regions bounded by more than one path, though the representation can describe them.) The full REGION and PATH descriptions are now:

```
REGION:
        TYPE:  <one of (ORIENT, BOUNDED) >
        BOUNDARY:  <list of pairs:  (path +1/-1) >
        ORIENT:  <ORIENTATION-FRAME>
        PLACES:  <list of PLACEs>
        PATHS:  <list of PATHs>
```

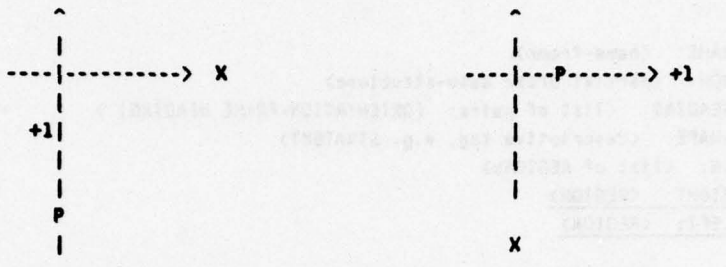
```
PATH:
        NAME:  <name-frame>
        ROW:  <partial order data-structure>
        HEADING:  <list of pairs:  (ORIENTATION-FRAME HEADING) >
        SHAPE:  <descriptive tag, e.g. STRAIGHT>
        IN:  <list of REGIONs>
        RIGHT:  <REGION>
        LEFT:  <REGION>
```

A REGION which is in PATH/RIGHT or PATH/LEFT may occasionally be referred to as a "side-region" of that PATH.

Acquiring boundary relations

The TURN instruction is crucial to the way the TOUR machine discovers a boundary relation between a place and a path. If a TURN is either preceded or followed by a GO-TO instruction involving a place not on the path, then the direction of travel on the street and the amount of the turn determine which side region that place belongs in. These inference rules, then, must examine two TOUR instructions: the current one and the previous one. In effect, these two instructions focus attention on a small part of the cognitive map which is examined for the boundary relation.

The two diagrams below illustrate this situation with a path P, travelled in the +1 direction, and a place X, not on P. In the first case, a right turn is followed by a GO-TO to X. In the second case, a GO-TO from X is followed by a right turn. In both cases, X belongs in P/RIGHT.



If the current instruction is a GO-TO and the previous one is a TURN, then we may try to determine the relation between TURN/ST1 and GO-TO/TO.

> GO-19: If the preceding instruction was a TURN onto GO-TO/PATH, with TURN/AMT and TURN/DIR1 specified, then decide which side-region of TURN/ST1 should contain GO-TO/TO, and put it there.

Similarly, if the current instruction is a TURN, preceded by a GO-TO, the relation is to be found between GO-TO/FROM and TURN/ST2.

> TURN-15: If the previous instruction was a GO-TO to TURN/AT, and TURN/AMT and TURN/DIR2 are both specified, then decide which side-region of TURN/ST2 should contain GO-TO/FROM, and put it there.

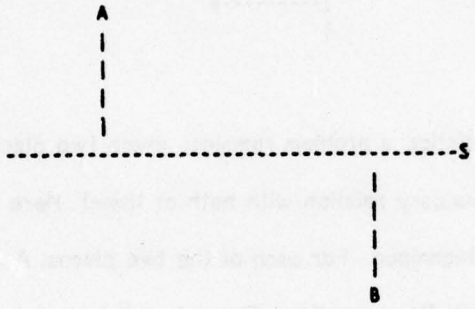
In both of these cases, the travelling instruction could potentially be a TAKE rather than a GO-TO. The relation remains true for arbitrarily long routes, provided that they do not cross back over the dividing street. The implemented version, however, uses only the productions listed.

Route-finding with dividing boundaries

There is a simple route-finding heuristic which is based on the partial position knowledge provided by dividing boundaries.

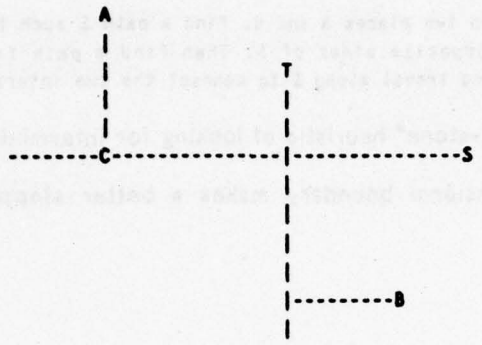
> To find a route between two places A and B, find a path S such that A and B are known to lie on *opposite sides of* S. Then find a path from A to S, another from S to B, and travel along S to connect the two intersections.

This is basically the "stepping-stone" heuristic of looking for intermediate subgoals, but recognizing that a one-dimensional boundary makes a better stepping-stone than a zero-dimensional point.



This heuristic can be extended by removing the restriction that S be selected so that A and B lie on opposite sides. The consequence of this is that routes will be found that are somewhat roundabout but which use major connecting streets, since major streets are most frequently known as boundaries.

The method can be further modified to be recursive. Suppose only one of the subgoals is achieved, and we find a path from A to S, intersecting S at C. (See figure below.) Then we can pose the new problem of finding a route from C to B, which may be solved by recourse to yet another dividing street (in this case, T).

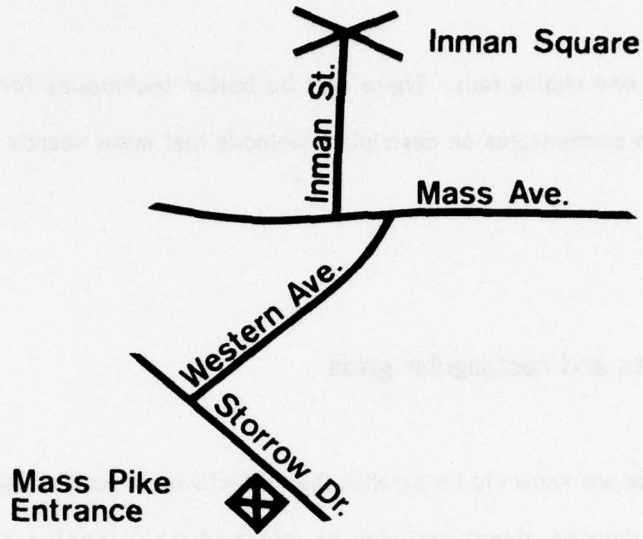


In order to implement these heuristics, a problem remains: given two places, how do we find a dividing street which has a boundary relation with both of them? Here is where the TOUR model first indulges in a search technique. For each of the two places, A and B, consider the regions of type BOUNDED in their IN properties. These two lists can be compared to find pairs of regions, one from each list, which share a boundary street. The straightforward approach is to examine all possible pairs to see if they share a boundary street. This approach examines as many pairs as the product of the lengths of the two lists. A slightly cleverer approach is:

> Put a temporary mark on the street bounding each BOUNDED region from A/IN. Then examine each BOUNDED region in B/IN, to see if its boundary is already marked. If so, this is a shared dividing boundary. When the search is concluded, erase the marks.

This technique examines the sum of the two lists, rather than the product, and is therefore not an expensive operation.

Consider using the dividing boundary strategy to find a route from Inman Square in Cambridge to the entrance to the Mass Pike, on Storrow Drive in Allston. (See the map below.) I present the example below in English because it should be clear how it translates into formal PLACE, PATH, and REGION descriptions.



Applying the above technique for finding dividing streets, we discover that Inman Square is north of Mass Ave while the Mass Pike entrance is south, so Mass Ave is our first dividing boundary. Inman Street connects directly with Inman Square as well as intersecting with Mass Ave (near the Cambridge City Hall, as it happens). However, there is no single path that connects that intersection with the Mass Pike entrance.

The smaller problem now is to find a way from Cambridge City Hall (or rather the nearby intersection) to the Mass Pike entrance. A second search for dividing boundaries reveals that Western Avenue lies between the two places, with the Mass Pike entrance to the southeast and Cambridge City Hall to the northwest. In this case, both of the desired connections can be made easily, since Mass Ave intersects Western Avenue at Central Square, and Storrow Drive leads directly from Western Avenue (at the Western Avenue Bridge) to the Mass Pike entrance.

The selection of the proper dividing boundary from among several candidates is done by

simple back-tracking if one choice fails. There may be better techniques for controlling search, but this research concentrates on descriptive methods that make search unnecessary in many cases.

Parallel streets and rectangular grids

When two or more streets are known to be parallel, their effectiveness as dividing boundaries is enhanced. A single dividing street provides an intermediate "stepping-stone" for a route-finding problem. When a bundle of parallel dividing streets is available, it provides a set of stepping stones, to be used individually or in a sequence on the way to the goal. When two orthogonal bundles of parallel streets are available, the positions of source and goal are very precisely specified in both dimensions. Orthogonal bundles also ensure that getting from one parallel street to another in the same bundle is simple, so that a route can be found very easily between any two places within the grid.

A bundle of parallel streets provides a global view of their relations which captures their local properties as boundaries, and also permits further inferences from the relations of their side-regions. In this sense a parallel bundle is similar to the order description a PATH has of a set of places: they both collect local relations into a globally useful structure. When a place is located with respect to one path, the parallel bundle extends this relation to other paths as well.

Acquiring parallel relations

A parallel relation is defined between pairs of PATHs. It is initially observed and represented when a GO-TO directs attention to the pair of cross-streets at its endpoints. If they have the same heading with respect to the same orientation-frame, they are described as parallel. Of course, this is a local description, and the same streets can intersect or diverge at some other point.

If two streets are parallel, the fact is represented in the PARALLEL property of their PATH descriptions. If P1 is parallel to P2, and their +1 directions have the same heading, then (P2 +1) is put into P1/PARALLEL, and vice versa. If their +1 directions have opposite headings, then (P2 -1) and (P1 -1) are stored. Which sides the two paths are of each other is represented by putting them in each others' side-regions.

> GO-20: Let C1 and C2 be the only cross streets at GO-TO/FROM and GO-TO/TO, respectively. If the headings of both C1 and C2 can be found with respect to the same ORIENTATION-FRAME, and if they are equal, put (C1 +1) into C2/PARALLEL, and (C2 +1) into C1/PARALLEL. If they are opposite, put (C1 -1) into C2/PARALLEL, and (C2 -1) into C1/PARALLEL.

At this point, the full form of a PATH description is:

> PATH:
> NAME: <name-frame>
> ROW: <partial order data-structure>
> HEADING: <list of pairs: (ORIENTATION-FRAME HEADING) >
> SHAPE: <descriptive tag, e.g. STRAIGHT>
> RIGHT: <REGION>
> LEFT: <REGION>
> IN: <list of REGIONs>
> PARALLEL: <list of pairs: (PATH +1/-1) >

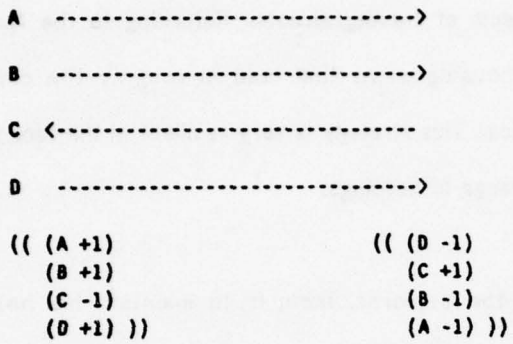
The parallel bundle

As represented in PATH/PARALLEL and the side-regions of the PATH, the the relations among parallel streets are distributed and relatively inaccessible. In order to achieve the global and more useful parallel bundle representation, we need a process which explores sets of parallel streets, following the relations between pairs of streets, and gathers them into a parallel bundle.

The parallel bundle represents two relations among parallel streets, in addition to the fact that they are parallel. First, it describes each path as a pair (PATH DIRECTION), where DIRECTION is +1 or -1 according to whether the PATH runs the same way or the opposite way as an arbitrary standard in the bundle. Second, it orders the paths in the bundle, with the order running from left to right when facing in the bundle's +1 direction. These two kinds of information are necessary in order to know the relations among the side-regions of the streets in the bundle. The bundle is held in the BUNDLE property of a REGION description whose TYPE is STRUCTURED. A grid is described by two STRUCTURED REGIONs.

```
REGION:
        TYPE:  <one of (ORIENT, BOUNDED, STRUCTURED) >
        BUNDLE:  <partial ordered set of pairs:  (PATH +1/-1) >
        BOUNDARY:  <list of pairs:  (PATH +1/-1) >
        ORIENT:  <ORIENTATION-FRAME>
        PLACES:  <list of PLACEs>
        PATHS:  <list of PATHs>
```

The following bundle of parallel streets has two possible representations, depending on which street was chosen to establish the overall DIRECTION of the bundle. The +1 direction on each street is in the direction of the arrow. The two representations are functionally identical for inferences and problem-solving.



The process of gathering isolated parallel relations between PATHs into a parallel bundle is initiated when the PARALLEL property of some PATH description contains more than one street. The gathering operation establishes the direction of the bundle as the +1 direction of the original street. Then it follows the PARALLEL links among PATHs, creating (PATH DIRECTION) pairs so that each pair faces the direction of the bundle. The side-regions of each PATH are examined to provide the ordering on the parallel streets. Finally, the fragments of order are merged by the partial order mechanism we have seen ordering PLACEs on a PATH. A REGION description is created to hold the partial order datastructure which defines the parallel bundle.

Exploring unknown territory

Qualitative knowledge of dividing boundaries can be of great value to the traveller in unfamiliar territory, where he has no cognitive map of the street network. He can always return to familiar territory if he knows what side of a familar path he is on, what the heading of that path is, and if he can maintain knowledge of his own heading with respect to that of the path. He can maintain knowledge of his own heading by careful attention to the amounts

of turns and to the gradual effects of curving streets. Returning to the familiar boundary street is simply a matter of choosing paths that lead in roughly the correct direction whenever he is faced with a choice. This strategy is very resilient in the face of errors, since his goal is accessible by a wide range of headings.

The primary task required of the explorer, then, is to maintain his heading. This is substantially easier if the new area is organized as a rectangular grid, since his heading then has only four possible values and there are no curving streets. He can also estimate his distance from his goal in terms of the number of intervening parallel streets, in case he has retained that much information about the street network.

A second way of maintaining his heading is by reference to a distant and conspicuous landmark. If he has lost track of the cumulative change to his heading from the route he is following, an observation of the landmark serves to reorient him: i.e. allow him to establish his heading with respect to his orientation-frame. In Boston, the Prudential Building and the John Hancock Tower often play this important role. This use of the orientation-frame, centered about a visible landmark, is useful for exploration but, without knowledge of the street network, cannot be used to represent the positions of currently invisible places.

Grid-based distortions

Since the description of an area in terms of parallel streets and grid structures is so effective, it is hardly surprising that many people introduce enormous distortions into the geography in order to make it conform to such a pattern. Because these representations are so resilient in

the face of errors, however, the distortions do not greatly reduce their effectiveness for route-finding. It is only in extremely anomalous cases that distortions in a person's cognitive map result in geographical paradoxes that disturb him and lead him along incorrect routes.

The most common distortions are 1) describing an intersection as a right angle when it is not, 2) describing a street as straight when it curves, and 3) describing two streets as parallel when they converge or diverge. All three of these are topological distortions of the street network which have no effect at all on the topological properties discussed in Chapter 2. They have very little effect on the roles the streets play as boundaries between regions, and even as constituents of parallel bundles and grids, because their topological properties are still the most important ones. Noticeable inaccuracies result from these distortions when routes are followed to estimate relative positions using the methods discussed in Chapter 3. A careful examination of the inference rules which make up the TOUR machine shows how little they are affected by metrical errors, as long as topological correctness is preserved.

Thus we get the curious fact that an incorrect description of geographical reality is more effective than a correct one, because it abstracts out the metrical features and leaves the topological features which permit powerful descriptions and problem-solving techniques. Of course, extreme anomalies will cause geographical paradoxes to arise. Fortunately (for this research), Boston contains many such extreme anomalies, and the cognitive maps of Boston residents are correspondingly full of geographical paradoxes.

For example, Beacon Street and Commonwealth Avenue are essentially parallel over most of their length, but they switch sides by crossing at Kenmore Square. Images of Cambridge differ as to whether Mass Ave and Memorial Drive are parallel or perpendicular, according to

the end of Cambridge at which they were first encountered. Perhaps the most famous geographical paradox in Boston arises from the fact that Boston Common has five sides while each corner appears square. This leads to incorrect estimates of the headings of the bounding streets, which in turn eliminates the space occupied by whole sectors of the city.

Such paradoxes aside, the use of default values like straight streets and right-angle turns is justified on practical grounds, if not those of strict accuracy. A description which adheres to the defaults is more economical to create and to store, and most inferences that operate on it are not adversely affected by the inaccuracies. Furthermore, a description of an area in terms of parallel bundles and grid structures is more likely if the places and paths are described as regular. That structure confers so much problem-solving power that minor carelessness with the facts appears justified.

Summary

The role of a one-dimensional geographical feature as a dividing boundary underlies a number of important parts of the cognitive map. Dividing boundaries are the basis of an important kind of partial knowledge of position. They can be recognized easily from travel along well-specified routes, and they can be easily applied to route-finding problems. When a number of such boundaries are gathered into a parallel bundle, their power for describing positions and specifying routes is greatly increased. Two orthogonal parallel bundles constitute the grid representation which is so useful in organizing urban geography. This derivation is important because it shows that the grid representation is not a primitive element of the cognitive map, but is built from simpler elements with richer states of partial

knowledge. The dividing boundary and the orientation frame can also be used to define a strategy for exploring areas where the topology of the street network is still unknown. Except for the exploration strategy, the use of dividing boundaries is completely implemented in the TOUR model.

V. REGIONS

> *"Rather than a single comprehensive image for the entire environment, there seemed to be sets of images, which more or less overlapped and interrelated. They were typically arranged in a series of levels, roughly by the scale of area involved, so that the observer moved as necessary from an image at street level to levels of a neighborhood, a city, or a metropolitan region." (pp. 85-86)*
>
> *"The physical characteristics that determine districts are thematic continuities which may consist of an endless variety of components: texture, space, form, detail, symbol, building type, use, activity, inhabitants, degree of maintenance, topography. In a closely built city such as Boston, homogeneities of facade---material, modeling, ornament, color, skyline, especially fenestration---were all basic clues in identifying major districts. Beacon Hill and Commonwealth Avenue are both examples. The clues were not only visual ones: noise was important as well. At times, indeed, confusion itself might be a clue, as it was for the woman who remarked that she knows she is in the North End as soon as she feels she is getting lost." (pp. 67-68)*
>
> *[Lynch, 1960.]*

Regions play an important role in the cognitive map by allowing groups of geographical features to be referred to collectively. First, they allow sets of geographical features to be referred to as units, for effective indexing and economical information storage. Second, they provide levels of abstraction, so that references to the geography can take place with different amounts of detail. They also provide a linguistic context for such context-dependent terms as "near" and "far".

Storage economy is achieved by allowing certain pieces of information to be reconstructed when required, rather than being represented explicitly. When similar properties are shared by a number of geographical places, it can be convenient to collect the places into regions and store the properties with the regions. The properties of the places can then be reconstructed when needed. For example, the fact that the West Coast of the United States is

3000 miles west of the East Coast may be stored explicitly. When I ask the distance from MIT to Stanford University, the fact that they are (roughly) 3000 miles apart can be reconstructed from the more general assertion and their containment relations with the two coasts. The same general assertion replaces countless similar assertions about the distance between particular places on the two coasts. An even more common example is the use of trunk routes to connect large regions. Routes connecting places in two regions can be constructed quite easily by first selecting the appropriate trunk route, and then connecting its endpoints with the desired source and goal.

Regions also function as levels of abstraction in describing the geography. I may describe my current location in a number of different ways: I am at MIT, in Cambridge, on the East Coast, and so on, depending on what I want to associate with that description. A generally useful piece of information is often best represented as relating two places which are in fact abstractions of large regions. Thus, when I say, "The West Coast is 3000 miles west of the East Coast," I am treating the two large regions as abstract places, capable of being related by a well-defined distance. Similarly, a trunk route from New England to Northern California might be stated as, "Take Interstate 90 from Boston to Cleveland. Then turn onto Interstate 80 and follow it to San Francisco." The route is stated at one level of abstraction, treating cities as simple locations, and it is indexed at another level: the regions New England and Northern California.

These levels of abstraction reflect the representation of information in the cognitive map, not simply the attempt to express the relations in English. However, there are linguistic effects for which the use of regions is important. For example, whether two places are described as "near" or "far" is dependent on the context provided by some region. Also, the way a place

will be described in response to a question is dependent on the circumstances of the question.

In this chapter, regions are treated primarily as levels of abstraction for indexing and summarizing frequently used information. However, they can also be used for their spatial extent and internal structure. This facet of the region description is linked with representations of visual space that are used in map reading, and with the use of the iconic mental image for inference (Chapter 3). When a region is treated as an extended shape, several new operations become meaningful. It becomes possible to subdivide a region into subregions: e.g. Western Massachusetts, or the "thumb" of Michigan. It also becomes possible to describe the position of a place as "central" or "in the northwest corner" with respect to the region.

The use of regions as levels of abstraction has been completely implemented in the TOUR model. The implementation supports the retrieval of general trunk routes for specific route-finding problems and the shift in levels of detail that is required to fill in the gaps at the endpoints and make a complete route. The use of a region as an extended shape has not yet been implemented.

A hierarchy of regions

Rumelhart (1974) explored the use of a collection of containing regions for summarizing information and providing useful levels of abstraction for explanation. His "ROOM Theory" was proposed to explain the answers to "where" questions and the use of relative distance terms like "near" and "far." The ROOM is the smallest region that contains the reference position of

the conversation (usually its location) and the places of interest. "Near" and "far" are comparisons of the requested distance with the size of the ROOM. This theory explains why, from Rumelhart's position in La Jolla, the San Diego airport is far away, while New York and Boston are near each other. The context-dependent answer to a "where" question is the least precise region that is still informative: the largest region which is smaller than the ROOM, and which contains the place of interest but excludes the reference position.

Stevens (1976) has tested the ability of a model of this sort to account for certain psychological phenomena. He found that it explained a class of common position distortions, the relative difficulty in answering pairs of position-estimation questions, and the facilitation of one such question by a previous one about related parts of the region structure. This is the only research I know of that is sufficiently detailed to distinguish among alternate accessing strategies for a hierarchy of regions. The design of the operators presented below for searching the region structure is consistent with Stevens' results.

As Lynch (1960) notes at the beginning of this chapter, the characteristics which define a region are often not spatial in the sense of large-scale space used in this paper. Furthermore, most named regions are learned by explicit telling, either in school or in response to questions like "Where is the South End, anyway?" Thus, in this chapter we will not be concerned with the creation of regions, but only with the way that they relate to the rest of the cognitive map.

Most of the regions that we use in this way have names, and can be referred to explicitly. However, occasionally a region which is clearly defined by some perceived property may not have a name, but will have the same role as a named regions in the cognitive map. Important

unnamed regions are often regions in the sense of Chapter 4, where they are defined by their boundaries, like "the other side of the tracks," or "the area between Mass Ave and the river." An important area for further research is regions which combine the properties discussed in chapters three, four, and five.

Representing the regions

Regions are related by containment. In most of the familiar cases, such as cities, states, and sections of the country, regions have well-known names and politically defined boundaries. These regions fall into a clear hierarchical structure by containment. For example, Boston and Springfield are both cities in Massachusetts, while Hartford and New Haven are in Connecticut. Massachusetts and Connecticut are both in New England, which is in the East Coast, which is in the United States, which is in North America, and so on. In this case, the set of containing regions, or "superiors", of a given place can be represented as a sequence from the smallest to the largest.

Regions can overlap without containment, however. A region like the Berkshire area in Western New England overlaps Massachusetts, Connecticut, and Vermont without being contained in any of them. Thus a place like the Tanglewood Music Center is both in Massachusetts and in the Berkshires, without a meaningful ordering on the two. So in general, rather than being a sequence, the set of superiors of a given place may be a partial order.

The fundamental representation for these relations among regions is that a given place knows about some regions which contain it. Thus, the description of the Tanglewood Music Center

can mention explicitly that it is in the Berkshires, in Massachusetts, and in New England. This description does not include information on the relations among those three containing regions. The mention of New England is redundant since containment is transitive, and both the Berkshires and Massachusetts are described as being in New England. Furthermore, if the full structure of the containing regions is compiled, Tanglewood can inherit containment in the East Coast and the United States from the description of New England.

The structure of the containing regions is distributed among many local containment relations, pointing from each region to its superior. In order to compile this structure into a single datastructure, useful for problem-solving, the elementary upward pointers must be followed and merged into a partial order. This merge of local information into a global datastructure is done by the same mechanism that orders places on paths. The containment relations are local in order to keep the cost of adding new regions as low as possible, and allow them to participate in the problem-solving process as soon as they are added.

A region and a place are different ways of describing the same piece of geography. Therefore, a PLACE description is often coupled with a REGION description for the same area at a more detailed level. To represent the correspondence between these two descriptions, we add a REGION property to the PLACE description, which points to its corresponding region, and a CORRESPONDS property to the REGION description, which points to its corresponding place.

It is convenient to represent the containment relations among regions as a property of the corresponding PLACE descriptions. The CONTAINED property of a PLACE description holds a set of PLACEs, which correspond to the superior regions of the given place. The rest of this

chapter describes the techniques for representing and using the structure of the set of superiors of a given place. PLACE/CONTAINED has only upward pointers from a place to its superiors. We will discuss downward pointers in a later section.

With these additions, the PLACE and REGION descriptions now have the following properties.

```
PLACE:
        NAME: <name>
        ON: <list of PATHs>
        STAR: <local geometry datastructure>
        CONNECT: <list of pairs: (PLACE route) >
        ORIENT: <ORIENTATION-FRAME>
        VIEW: <list of triples: (PLACE heading distance) >
        CARDINAL: <list of triples: (PLACE heading distance) >
        IN: <list of REGIONs>
        REGION: <region>
        CONTAINED: <list of PLACES>

REGION:
        TYPE: <one of (ORIENT, BOUNDED, STRUCTURED, NAMED) >
        NAME: <NAME>
        BUNDLE: <partial ordered set of pairs: (PATH +1/-1) >
        BOUNDARY: <list of pairs: (PATH +1/-1) >
        ORIENT: <ORIENTATION-FRAME>
        PLACES: <list of PLACEs>
        PATHS: <list of PATHs>
```

Notice that PLACE/IN and PLACE/CONTAINED both refer to containing regions, but ones that play different roles with respect to the PLACE.

Using regions in problem-solving

The kinds of problems that can be solved using the region structure are essentially search problems: there is a piece of information (a route or relative position) relating two places in the cognitive map but it may be indexed under superiors of the two places, so the problem is to find it. Since the region hierarchy is not typically very deep, this search is quite well

constrained. Furthermore, the piece of information, once found, may be stated in terms of (possibly different) superiors of the given places. Relating the solution as found to the problem as stated is the topic of the next section.

For example, consider the problem of finding a route from MIT to Stanford University. The key to the solution is the trunk route we saw above, indexed under New England and Northern California. This key route is stated as going from Boston to San Francisco. In order to use a regional structure to solve such problems, it is necessary to show how each piece of the solution is found, and how they can be knit together to solve the complete problem.

There are three operations involved in searching the region structure for the solution to a problem. First, finding and representing the set of superiors of the given places involved. Second, comparing these sets to find the smallest region which includes all the places of interest: Rumelhart's ROOM. Third, looking at the places contained in the ROOM to find the desired relation: a route from source to goal, or a position relating the two places of interest. These operations are combined in the appropriate ways by the specialists in route-finding, position-estimating, or answering "where" questions.

> UPWARD-ORDER takes a PLACE description, and returns the partial order datastructure which results from merging the upward pointers in the CONTAINED property of the given PLACE and those pointed to.
>
> SHARED-REGIONS takes two sequences of nested PLACEs, and returns three values: the smallest common PLACE and the two longest disjoint sequences of nested PLACEs.
>
> RELATED-REGIONS takes two disjoint sequences of nested PLACEs and a property-name, and looks for a relation indexed under that property-name and a PLACE in each sequence. If several such relations exist, it returns the relation associated with the smallest pair of PLACEs.

Note that although UPWARD-ORDER returns a partial order of superiors of the given PLACE, the other two operations take only sequences. The sequence used by the latter two

operations is the longest totally ordered subset of the partial order produced by UPWARD-ORDER. Not examining the other regions in the partial order can result in two kinds of errors. First, SHARED-REGIONS may select as the smallest common PLACE one which is larger than necessary. Second, RELATED-REGIONS may miss a relation which is present because it was indexed under a PLACE that was discarded from its partial order.

To observe these operations in action, let us consider the following nested region structure.

The United States
East Coast
New England
Massachusetts
Cambridge
MIT
Tech Square
Boston
Logan Airport
Connecticut
Central States
New York
Pennsylvania
West Coast
California
Northern California
San Francisco
San Francisco Airport
Palo Alto
Stanford University
Southern California
Los Angeles
San Diego
Oregon
Washington

(Since the representation of this structure in terms of CONTAINED properties of PLACE descriptions is straightforward but lengthy, I am omitting it.)

We are attempting to find a trunk route which is indexed under the PLACE description of Northern California in the CONNECT property of the PLACE describing New England. The

result of the operation UPWARD-ORDER when applied to MIT and Stanford University is:

```
((The United States                ((The United States
  East Coast                         West Coast
  New England                        California
  Massachusetts                      Northern California
  Cambridge                          Palo Alto
  MIT))                              Stanford University))
```

SHARED-REGIONS, applied to these two sequences, identifies the United States as being the smallest common region, and returns the rest of the two sequences as their disjoint sets of superiors.

RELATED-REGIONS steps down the two sequences, looking for a PLACE in the first sequence with a route in its CONNECT property indexed under a PLACE in the second sequence. Since the region structure is relatively shallow, an exhaustive search of possible pairs is not unreasonably computationally expensive. However, if we assume a certain amount of regularity in the structure that puts comparable regions at roughly the same distance (within the sequence) from a common containing region, the search is restricted considerably. It seems plausible that the procedures that create REGION descriptions would attempt to maintain such regularity to aid the problem-solving process. In the implementation, RELATED-REGIONS examines only pairs of PLACEs whose distance from the common containing region is the same or differs by one. In this case, the desired route is found in the CONNECT property of New England, indexed under Northern California.

Exactly the same strategy works to find position information relating two places at the most precise level of detail. RELATED-REGIONS then examines the information stored under the property-name CARDINAL rather than CONNECT.

Relating different region descriptions

The technique of the last section starts with the problem statement and finds where an applicable route description is indexed. So far, we have not examined the route description itself, or attempted to relate it to the original problem statement. The remaining issue is to relate information stated at one level of abstraction with questions stated at another.

Assume that the route that was found in the previous section said:

```
Take Interstate 90 from Boston to Cleveland.
Turn onto Interstate 80 and take it to San Francisco.
```

(It should be clear how this corresponds to three TOUR instructions.) The difficulty is that the endpoints of this route are Boston and San Francisco, which are at a more general level of abstraction than the source and goal of the original problem: MIT and Stanford University. We need to change the focus of attention to more precise descriptions of the endpoints. This mismatch is detected by the TOUR machine when the inference rule GO-1 notices that the "You Are Here" pointer does not match the assumed initial position of the first GO-TO instruction, and inserts a GET-TO instruction.

The very nature of abstraction makes this difficult, of course: an abstract place represents many specific ones. However, we seldom need a more precise description of position unless we are planning a route, in which case the current position should be more fully specified. The cognitive map contains certain correspondences between an abstract (PLACE PATH DIRECTION) description, and a more specific one. For example, although being in Cambridge does not correspond to being at any particular intersection in Cambridge, being in Cambridge on the Mass Pike corresponds to a particular heading at the intersection of Storrow Drive and the River Street Bridge (in Allston!).

AD-A045 790 MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTE--ETC F/G 6/4
REPRESENTING KNOWLEDGE OF LARGE-SCALE SPACE.(U)
JUL 77 B J KUIPERS N00014-75-C-0643
UNCLASSIFIED AI-TR-418 NL

2 OF 2
AD A045790

END
DATE FILMED
11-77
DDC

Such a downward mapping of position descriptions may not exist for all possible abstract position descriptions. They are created by the processes which build the region descriptions. A more detailed theory of the discovery of such correspondences must wait for further research.

The correspondence is represented by the DOWNWARD property of the higher REGION description, which contains a list of 6-tuples. Each 6-tuple represents the correspondence between a higher (PLACE PATH DIRECTION) triple and a lower one.

```
REGION:
        TYPE:  <one of (ORIENT, BOUNDED, STRUCTURED, NAMED) >
        NAME:  <NAME>
        DOWNWARD:  <list of 6-tuples:  (HPLACE HPATH HDIR LPLACE LPATH LDIR) >
        BUNDLE:  <partial ordered set of pairs:  (PATH +1/-1) >
        BOUNDARY:  <list of pairs:  (PATH +1/-1) >
        ORIENT:  <ORIENTATION-FRAME>
        PLACES:  <list of PLACEs>
        PATHS:  <list of PATHs>
```

When a change has been selected in the focus of attention between different descriptions of the same place, it is communicated to the TOUR machine through the UP and DOWN instructions. These instructions change the "You Are Here" pointer without corresponding to any physical travel from one place to another. They have the following format:

```
UP:                              DOWN:
        FROM:  <place>                   FROM:  <place>
        ST1:  <path>                     ST1:  <path>
        DIR1:  <direction>               DIR1:  <direction>
        TO:  <place>                     TO:  <place>
        ST2:  <path>                     ST2:  <path>
        DIR2:  <direction>               DIR2:  <direction>
```

The following productions incorporate the UP and DOWN instructions into the normal TOUR machine problem-solving process. GET-5 and GET-6 recognize when a route-finding problem is not well-formed because one of the endpoints is a more abstract description of the other.

In that case, the more abstract one must be mapped onto a more specific place, and a new, more specific route-finding problem posed. UP-1 and DOWN-1 recognize when their preconditions are not yet satisfied, and insert a GET-TO instruction before the current instruction. UP-2 and DOWN-2 actually change the focus of attention by changing the "You Are Here" pointer to the alternate description of the same position.

> GET-5: If GET-TO/FROM is a superior of GET-TO/TO, then map downward from (GET-TO/FROM GET-TO/ST1 GET-TO/DIR1) to a lower description (LPLACE LPATH LDIR) of the same position. Replace this GET-TO with two instructions: a DOWN to the lower state, and a GET-TO from the lower state to the goal.
>
> GET-6: If GET-TO/TO is a superior of GET-TO/FROM, then map downward from (GET-TO/TO GET-TO/ST2 GET-TO/DIR2) to a lower description (LPLACE LPATH LDIR) of the goal state. Replace this GET-TO with two instructions: a GET-TO from the current state to the lower goal state, and an UP instruction to the higher goal state.
>
> UP-1: If the current position in C-PLACE, C-PATH, C-DIR doesn't match the current position in UP/FROM, UP/ST1, and UP/DIR1, insert a GET-TO instruction before the UP. Dismiss.
>
> UP-2: Reset C-PLACE, C-PATH, and C-DIR to UP/TO, UP/ST2, and UP/DIR2, respectively. Clear the rest of the "You Are Here" pointer.
>
> DOWN-1: If the current position in C-PLACE, C-PATH, C-DIR doesn't match the current position in DOWN/FROM, DOWN/ST1, and DOWN/DIR1, insert a GET-TO instruction before the DOWN. Dismiss.
>
> DOWN-2: Reset C-PLACE, C-PATH, and C-DIR to DOWN/TO, DOWN/ST2, and DOWN/DIR2, respectively. Clear the rest of the "You Are Here" pointer.

To conclude the example of getting from MIT to Stanford University, the ends of the trunk route that was found must be patched to match the more specific starting places of the problem. MIT is recognized as being a more specific description than (Boston I-90 nil), which is the initial state of the trunk route, so the Boston location is mapped to a specific intersection: the Mass Pike entrance on Storrow Drive in Allston. The connecting route can now be found easily. Note that recognizing that MIT is a more specific place description than Boston requires the additional information discussed in the following section.

Getting from (San-Francisco I-80 nil) to Stanford University is a similar problem: the endpoint of the trunk route is mapped to a more specific location which is connected by the Bayshore Expressway to Palo Alto. Palo Alto on the Bayshore Expressway can in turn be mapped to a more specific location that is connected with Stanford University, completing the route.

Notice that the "turn" from I-90 to I-80 at Cleveland has been left in the more abstract form. If it must be specified in more detail, the two abstract positions (Cleveland I-90 nil) and (Cleveland I-80 nil) can be mapped downward to more specific place descriptions, which may be a significant distance apart. Thus, what appears to be a simple turn at one level may become a substantial route upon closer examination.

Generic constraints on containment

A region's first defining characteristic is usually that it contains some particular place. The process of elaborating its description consists of adding containment relations with other regions. For example, if I am told that my current position is "in Tech Square," and this is my first acquaintance with Tech Square, then I would like to relate it to Massachusetts, Cambridge, MIT, this building, my office, and other regions that also contain my current position. If I know that Tech Square is a building complex, however, then I can conclude that it is contained in Cambridge and that my current building is contained in it. Since both MIT and Tech Square are building complexes, I can't conclude anything about their relationship. Since Boston is a city while MIT is a building-complex, it can be seen to be at a more abstract level of description in the previous example, and call for the downward mapping.

Jeffery (1977) suggests that this inference can be represented by a generic hierarchy of region-types which can add constraints to the containment structure. This generic type-hierarchy and the rules for using it are:

> continent > country > state > city > building-complex > building > room
>
> Two regions of different generic types cannot overlap without strict containment.
>
> The containment relations between regions of different generic types cannot contradict the type-hierarchy.

The type-hierarchy fills many of the functions of a "scale" property for a region, and thus allows knowledge of a region's generic type to function as partial knowledge of its size.

Answering questions about "near" and "far"

We can use the structure-searching operators defined above to implement Rumelhart's room theory question-answering with context-dependent usage of "near" and "far". The process of finding the distance between two places involves identifying the smallest common containing region: Rumelhart's ROOM. If the diameter of the region is known, then the distance can be declared to be "near" or "far" with respect to the size of the ROOM. The diameter of a region is stored in the REGION description's SCALE property. {Note Quantities} In the current implementation, this property is acquired by direct telling, rather than being inferred from internal distance information.

```
REGION:
        TYPE:  <one of (ORIENT, BOUNDED, STRUCTURED, NAMED) >
        NAME:  <NAME>
        GENERIC-TYPE:  <one of (CONTINENT, COUNTRY, STATE, ... ) >
        SCALE:  <integer>
        DOWNWARD:  <list of 6-tuples:  (HPLACE HPATH HDIR LPLACE LPATH LDIR) >
        BUNDLE:  <partial ordered set of pairs:  (PATH +1/-1) >
        BOUNDARY:  <list of pairs:  (PATH +1/-1) >
        ORIENT:  <ORIENTATION-FRAME>
        PLACES:  <list of PLACEs>
        PATHS:  <list of PATHs>
```

Denofsky (1976) has made a naturalistic study of the use of terms like "near" and "far." He concludes that, when a distance is estimated within a containing interval, "near" applies to distances up to 1/8 of the interval, "far" applies to distances over half of the length of the interval, and neither term applies to intermediate values. In answering questions about "near" and "far", the TOUR model follows Denofsky's criterion.

Spatial extent and internal structure

Although generic type and scale provide a partial description of the internal structure of a region, they omit much of importance. Two-dimensional shape requires a much more complex description than a scale factor. The internal structure of such a region can be divided into a number of sub-regions *without precisely defined boundaries*. These subregions are standard for different kinds of regions, are are referred to with modifiers like "East," "Central," "in the northeast corner," and so on. The position of a place within such a region can be partially described by assigning it to one of those subregions. It could be valuable to investigate the relation between this notion and that of iconic mental images (Chapter 3).

This kind of internal structure permits an interaction between the relative positions of pairs

of places within the region and the scale of the region as a whole. For example, if Springfield is 150 miles west of Boston, and Massachusetts has a diameter of about 200 miles, then it is safe to conclude that Springfield is in Western Massachusetts, while Boston is in Eastern Massachusetts. Conversely, membership in different subregions allows an estimate of relative position, and a range of typical relative distances within a region permits an estimate of the region's diameter.

Summary

Regions in the TOUR model exist primarily to provide levels of abstraction for indexing and summarizing spatial information that applies to large sets of places. The relations among region descriptions are initially stored as local containment relations from a region to its superiors. These are not constrained to fit into a strict hierarchy, for a place can refer to several superiors whose containment relations are not known. When a problem is to be solved, however, the local containment pointers are gathered into a partial order. The largest sequences from two such partial orders can be compared to find the largest disjoint containing regions about the places of interest. It is then straightforward to search for a relationship at a number of levels of detail. When such a relationship is found, it may be necessary to shift the level of detail at the endpoints in order to fill in the gaps between the general solution that was found and the specific question that was asked.

A generic type-hierarchy can add constraints to the structure of containing regions, and acts as a partial description of the scale of a region. The scale is used to provide the appropriate context-dependent level of detail for answers to "where" and "near/far" questions. A

substantially different view of regions considers their extent and internal structure, and makes it possible to describe sub-regions and positions within the region. The use of regions as levels of abstraction has been completely implemented. Their use for spatial extent and internal structure has not.

VI. RELATIONS TO OTHER WORK

What has been is what will be,
and what has been done is what will be done;
and there is nothing new under the sun.

Is there a thing of which it is said,
"See, this is new"?
It has been already,
in the ages before us.

There is no remembrance of former things,
nor will there be any remembrance
of later things yet to happen
among those who come after.

[Ecclesiastes 1: 9-11.]

The cognitive map, or mental image of space, has been of interest to psychologists since the work of Trowbridge (1913) on "imaginary maps." Research since then has indicated that the mental image of space does not have the characteristics of a pictorial image, but can be decomposed into a number of different symbolic elements. This research on the cognitive map has been surveyed by Lynch (1960), Downs and Stea (1973), Siegel and White (1975), and Moore and Golledge (1976). Artificial intelligence research, meanwhile, has focussed on ways of representing different kinds of knowledge as symbolic descriptions. This chapter reviews the work in artificial intelligence which has led to the representations in the TOUR model, and the work in environmental psychology which has outlined the kind of knowledge to be incorporated.

Symbolic descriptions

The field of artificial intelligence is based on the assumption that knowledge consists of symbolic descriptions and that intelligence consists of procedures for manipulating such descriptions. Over the past two decades techniques have been developed for decomposing the knowledge involved in certain activities that require intelligence, determining the functional characteristics of each kind of knowledge, and devising symbolic representations and mechanisms to perform those functions.

representation of knowledge as descriptions made up of property-value pairs was first investigated by Raphael (1968) in his program SIR ("Semantic Information Retrieval"). Raphael showed how such descriptions could represent facts about class membership, class inclusion and part-of relationships, and how inferences could be made by examining and modifying the relationships among the descriptions. Evans (1968) used a similar representation to solve geometric analogy problems, demonstrating that the property-value representation is applicable to domains that are not obviously propositional. Quillian (1968) developed the "spreading activation network search" which permitted unanticipated relationships to be found between two descriptions. Winston (1975) showed how a system based on symbolic descriptions could learn from short sequences of carefully chosen examples. The result of all this research is a body of techniques for representing knowledge as descriptions composed of property-value pairs, and for performing inferences on collections of these descriptions.

Complementing the research on representing descriptions, Hewitt (1971) investigated the procedural embedding of knowledge in inference rules. He distinguished between two kinds of inference rules: antecedent rules, which draw conclusions from a given antecedent, and

consequent rules, which propose potential subgoals for a desired consequent. Most of the inferences in the TOUR model are antecedent rules, which operate to draw conclusions from newly provided information.

Subsequent work by Hewitt and his students [Hewitt and Smith, 1974; Greif, 1975] reexamined the distinction between procedure and datastructure, and proposed to describe computations in terms of "actors": entities with prescribed behavior in response to messages. Parts of this notion have been incorporated into the TOUR model by making the descriptions into active datastructures (see Appendix C). Procedures associated with a description, and activated by its access functions, can perform inferences which depend only on information locally accessible from that description.

While the computational properties of the active descriptions are influenced by these developments, their semantic properties are inspired by Minsky's (1975) concept of "frames." A frame is a rich description of a stereotyped class of situations which can be selected and instantiated to describe a particular instance of that situation. It provides a pre-established set of terms for stating the description, along with defaults, constraints, and relationships among the different parts of the description. A frame can also have procedural aspects which specify how its instantiation should be carried out or how to handle certain anomalous observations. The idea of frames contrasts with the position that knowledge is primarily organized into assertions which are to be compared with each other individually. The descriptions in the TOUR model are "frames" in this sense. They provide a fixed set of terms in which each geographical feature is to be described. Access functions associated with each type of description can perform local inferences based on the information directly accessible from that description.

Rule-based systems

Much of the knowledge in the TOUR model is procedurally embedded in the inference rules. These rules are composed of tests and actions, where the tests are applied to information that is accessible from a small working memory. This kind of structure is modelled after the "production systems" of Newell and Simon (1972). A production system consists of a large long-term memory, a small short-term or working memory, and a collection of inference rules (productions) which can examine the contents of working memory. One major advantage of a production system is that procedural knowledge is represented in a modular way, requiring that interactions between productions must take place through the working memory, so new productions can be added without seriously disturbing the others. This means that expertise can potentially be acquired by the incremental addition of more and more expert productions, rather than requiring a monolithic program to be rewritten when new techniques are learned. Baylor (1971) and Moran (1973) examined how production systems could model human visual imagery. Howe and Young (1976) discuss how a production system can model individual variation in task performance by varying the collection of productions that make up the set of inference rules.

The working memory in the TOUR model consists of the "You Are Here" pointer, which describes the current position, and the current and immediately preceding route instructions. An inference rule in the TOUR machine can examine the properties of the descriptions that are directly accessible from the working memory. Although this restricts the extent of any search into the long-term memory, an inference rule may perform a substantial computation to

determine its applicability, rather than simply matching a pattern. Essentially the working memory acts as a "focus of attention" for the inference rules.

Partial knowledge

Little direct attention has been given in the artificial intelligence literature to the states of partial knowledge supported by a representation. The classification hierarchy, consisting of more and less general categories, can be seen as a structure for the partial states of knowledge in the recognition process. The knowledge attached to each category is the set of assertions warranted by that state of partial knowledge of identity. This position considers the structure of the hierarchy to follow from the recognition process, rather than from the inherent taxonomy of the objects categorized. This view is not widely held in the literature, with the exception of Collins, et al (1975) who use a classification hierarchy as the primary kind of incomplete knowledge in their geography tutor.

Sacerdoti (1973, 1975) has exploited different aspects of partial knowledge very fruitfully. Sacerdoti (1973) generates a hierarchy of abstraction spaces for planning by using partially specified operators in the more abstract spaces. An operator is partially specified by omitting one or more preconditions for its applicability. The key concept in his later work [Sacerdoti 1975] is that the partial state of a plan should be represented as a partial order rather than a total order. By representing the appropriate collection of states of partial knowledge of the planning process, he is able to state simple planning heuristics that are much more powerful than were previously possible. They act by adding constraints to the partial order and decomposing operators until the plan becomes a sequence of executable operations. The use

of a partial order to model states of partial knowledge of a sequence has been very important in this research.

Partial knowledge is often seen in popular descriptions of human memory as the "hologram" metaphor. The optical hologram has the feature that removing part of the hologram still allows a complete picture to be reconstructed, albeit with somewhat less resolution. This is compared to the ability of human memory to withstand the destruction of large parts of the brain without losing identifiable functions. Beyond this single characteristic, however, the metaphor must be viewed with considerable caution, since the Fourier transform representation of an optical image as a hologram is not a serious candidate for a memory representation.

The search for adequate memory representations must focus on the actual states of partial knowledge that memory exhibits, of which the classification hierarchy is a conspicuous example. The states of spatial knowledge described in this research are another. The next chapter discusses how the characteristics of common-sense knowledge depend on the states of partial knowledge supported by the representations.

Environmental psychology

Kevin Lynch's delightful book *The Image of the City* [Lynch 1960] was responsible for the recent surge of interest in mental images of the environment. One central point is that the mental image can be decomposed into symbolic elements, and is not necessarily pictorial in nature or properties. He distinguishes between the qualities of imageability and legibility of

an environment. *Imageability concerns the sensory aspects of the environment* and how its landmarks evoke strong reactions. Legibility concerns the structural aspects of the environment and how a traveller can find his way around in it. The properties that determine legibility are the ones addressed by this research.

Although controversy still rages among cognitive psychologists about the pictorial nature of mental imagery [Pylyshyn 1973, 1976; Kosslyn and Pomerantz, 1977], primarily motivated by the striking mental rotation experiments of Shepard and his students [Shepard and Metzler, 1971; Cooper and Shepard, 1973; Shepard and Judd, 1976] the consensus among geographers and environmental psychologists [Downs and Stea, 1973; Siegel and White, 1975; Moore and Golledge, 1976] is that the cognitive map of large-scale space is a symbolic description, quite unlike a graphical map, which makes it possible to answer spatial questions. Except for possible use of an iconic image as an inference device (Chapter 3), the two kinds of knowledge (of visual space and large-scale space) are quite disjoint.

Lynch decomposed the knowledge in the cognitive map into five elements: landmarks, nodes, paths, edges, and districts. Their functions and relationships correspond roughly with those of places, paths, and regions in the TOUR model. Lynch's concern with imageability led him to focus on those elements of the cognitive map that evoke a strong and distinct response. This led him to neglect the role of inconspicuous landmarks, nodes, and edges. For example, he identifies as edges only those linear objects whose boundary properties are so conspicuous that they dominate their path properties. In the TOUR model, any street can act as both path and boundary, to the extent to which it orders the places on it and divides the places to each side. Similarly, the place description in the TOUR model applies to any location a person has knowledge of, not just those that act as conspicuous landmarks.

An important methodological contribution of Lynch's research is his technique for drawing qualitative conclusions about the structure of the cognitive map from observations about individual variation. His basic assumption is that if two kinds of knowledge are present or absent independently in different cognitive maps, then they have independent representations, even if the representations cannot be specified precisely. Variation in cognitive maps can take place among different individuals, within the same individual in different environments, and in the same individual over time as he learns or forgets a particular environment. This technique was used to supply the anecdotal descriptions of spatial knowledge in the preceding chapters.

Among other qualitative observations, this technique allowed Lynch to conclude that the cognitive map tends to be topologically correct, even when it is seriously distorted metrically. He also observed that the cognitive map of a large area would typically consist of highly structured regions, loosely connected. Furthermore, the highly structured descriptions were built up over time from loose, topological descriptions [Appleyard 1970]. Another observation about the process of learning a new environment was that novices tend to use prominant landmarks to orient themselves, whereas people familiar with an area use local features of the street network.

Long before Kevin Lynch began his research, Jean Piaget and his colleagues had been studying the development of many different cognitive abilities in children. The scope of this chapter does not permit a detailed review of their work (see Hart and Moore (1973)). However, it is possible to draw a rough parallel between parts of the TOUR model and the developmental stages proposed by Piaget and Inhelder (1948) for the concept of space. They

subdivide spatial abilities into three sequentially developed stages: topological, projective, and euclidean. In the topological stage, a child is able to follow routes and narrate sequences of observed landmarks. In the projective stage, he can reproduce the scene from familiar observation points. In the euclidean stage, he can merge information from several projective views into a consistent and reasonably accurate set of metrical relationships among the places observed.

In the TOUR model, the topological stage corresponds to the process of learning route descriptions and assimilating them into a street network of paths and places (Chapter 2). The projective stage is characterized by representing positions relative to orientation frames local to each observation point. The euclidean stage occurs when an orientation frame can be shared among a number of different observations points, so relative positions from different origins can be made comparable. (See chapter 3.) Much more work is necessary before this sketch can be turned into a detailed explanation of the extensive experiments and observations of the Piagetian school.

Considerable amounts of research has been reported since these pioneering works. Downs and Stea (1973) collect a substantial range of different approaches to environmental psychology, including a number of attempts to create a common conceptual framework. This search for a conceptual framework characterizes much of the research they survey. Siegel and White (1975) review research focussing specifically on large-scale spaces. They discuss the philosophical underpinnings of the problem, compare the acquisition of spatial knowledge in adults with the development of spatial abilities in children, and present a neurologically-based learning theory. Most recently, Moore and Golledge (1976) identify the emerging theoretical schools of thought on the appropriate terms for studying cognitive maps and

include representative papers from each.

Of these schools of thought, the information processing view presented by Kaplan (1973, 1976) is most compatible with the TOUR model. Kaplan discusses the importance of examining the information processing that is taking place inside a person's cognitive map, and argues that this computational process can be described in terms of levels of symbolic descriptions and their manipulations. However, he does not carry his theory much beyond this general level. The TOUR model provides a substantially more detailed description of the computations which make up the cognitive map.

New questions to ask

The studies reviewed above make an important contribution by identifying, describing and categorizing the knowledge that makes up a cognitive map. However, they lack an adequate technical language for stating their theories in. The TOUR model attempts to provide such a technical language which makes it possible to state (and sometimes answer) questions beyond the scope of less formal descriptions. In order to state deep questions and answers about the cognitive map, the technical language must recognize the existence of the objects, such as processes and datastructures, which are necessary to describe the computations involved.

Consider the question "How is the current position represented?" A meaningful answer requires a theory of what is in the cognitive map, and of what roles the current position plays in the operations on it. The "Map in the Head" theory says that the current position is represented as a point on the map and a heading. The "Street Network" theory says that the

current position consists of a node and an arc. The heading and the arc are required because the current position must change in response to route instructions that don't specify their destination. Each of these theories is inadequate by itself because it doesn't contain a sufficiently rich collection of datastructures for the range of knowledge in the cognitive map.

An analysis of this kind, focussing on the computational requirements the current position is called on to fulfill, suggested the structure of the "You Are Here" pointer in the TOUR model. The distinction between one-dimensional orientation on a path and two-dimensional orientation with respect to an orientation-frame arose from the computational requirements of following a sequence of route instructions over a street network. This distinction solves a dilemma encountered by Siegel and White (1975) who were unable to formulate a model of navigation in a street network without global orientation. Global two-dimensional orientation, however, is usually acquired much later than the ability to navigate in a street network.

The "You Are Here" pointer plays a number of roles in addition to describing the current position. It acts as a working memory for the inference rules, as a context for incomplete instructions, as a source and destination for knowledge in the assimilation process, and as a focus of attention for structure-building specialists. Computational constraints apply in both directions as these processes make demands on the "You Are Here" pointer, while it determines what they can do. An example of such a constraint is that the focus of attention within which a structure-building specialist works is limited to the contents of the "You Are Here" pointer and the current and previous route instruction.

With the TOUR model, it is becoming possible to state currently unanswered questions about the relation between mental descriptions of visual space and large-scale space, between the

cognitive map and the verbal and graphical descriptions people give of it, and between the cognitive map and spatial metaphors. These questions could be stated, or answered, only in the most general terms without a technically adequate description of the knowledge in the cognitive map.

VII. CONCLUSIONS

> *"There once were two watchmakers, named Hora and Tempus, who manufactured very fine watches. Both of them were highly regarded, and the phones in their workshops rang frequently---new customers were constantly calling them. However, Hora prospered, while Tempus became poorer and poorer and finally lost his shop. What was the reason?*
>
> *The watches the men made consisted of about 1,000 parts each. Tempus had so constructed his that if he had one partly assembled and had to put it down---to answer the phone, say---it immediately fell to pieces and had to be reassembled from the elements. The better his customers liked his watches, the more they phoned him and the more difficult it became for him to find enough uninterrupted time to finish a watch.*
>
> *The watches that Hora made were no less complex than those of Tempus. But he had designed them so that he could put together subassemblies of about ten elements each. Ten of these subassemblies, again, could be put together into a larger subassembly; and a system of ten of the latter subassemblies constituted the whole watch. Hence, when Hora had to put down a partly assembled watch in order to answer the phone, he lost only a small part of his work, and he assembled his watches in only a fraction of the man-hours it took Tempus."*
>
> *[Simon, 1969, pp. 90-91.]*

One pervading theme of this paper is the importance of considering the states of partial knowledge which a representation can support. These are the states that can be stored over a long period, as opposed to the knowledge represented in the transient state of a computation. The transition between two states of a knowledge representation requires two kinds of resources: the information which is prerequisite for that transition and the computation time required to perform it. A representation which permits more states of partial knowledge will make smaller demands on those two resources in order to make a permanent change in what is represented.

Consider a representation for knowledge of a sequence which can represent only sequences.

The information requirements for a transition from one state of the representation to another are quite high, because a new element must be related to every other element before it can be added at all. Furthermore, the costs are associated with merging newly acquired information into the representation, rather than with retrieval, so if computational or information resources are lacking at that time, new information is lost forever.

Compare this with a partial order representation for partial knowledge of a sequence. Any fragment of order knowledge can be incorporated into the representation, so the information demands are as small as possible. Furthermore, significant amounts of computational resources are expended only on retrieving information from the partial order and on maintaining the datastructure in an efficient format. Disaster does not result from the failure of either task: the worst consequence of interruption is that certain pairs of places might be seen as lacking an order relation when in fact the information is present. When resources are available again, no knowledge has been lost, and performance is restored to an optimal level. An extra computational cost is paid when order information is retrieved in order to give the representation vastly more expressive power, and to make it relatively insensitive to resource limitations.

This example illustrates how a rich collection of states of partial knowledge allows a representation to satisfy the "principle of graceful degradation" proposed by Norman and Bobrow (1975). This principle states that when an intelligent process is given insufficient resources, there should be a smooth degradation of task performance rather than a calamitous failure. Bobrow and Norman also discuss the different implications of limitations in information and computational resources.

A first implication of this principle is that assimilation of new information should be very easy, even at the expense of retrieval or problem-solving, because it is more calamitous to lose a piece of information entirely than to fail to solve a particular problem. The entire assimilation process may be long and expensive, but it should have many states of partial knowledge so that small amounts of new information can be incorporated with small amounts of computation. A representation which is lacking in states of partial knowledge can make learning too expensive to be possible.

A second implication is that it is better to assimilate new information (perhaps through a large number of intermediate states) into a rich and complex representation which simplifies problem-solving than to rely on an expensive search when a problem is presented. This is because the state of a search process is very transient and is therefore very vulnerable to interruptions or resource limitations. On the other hand, resources can be used to good effect in assimilation whenever they are available.

A rich and complex representation can be structured so that solutions to the problems that are normally expected to arise are very easily found with a minimum of search, once the environment has been described. This description, ideally, can be produced by a collection of simple inference rules, each requiring few resources, which elaborate the description when the information and computational resources are available.

The TOUR model has been designed in accordance with these principles and has shown, I believe, how such representations support the flexibility, power, and tolerance of inadequate information or processing resources that is characteristic of common-sense knowledge.

Further development of the TOUR model

In some aspects of the TOUR model, the knowledge representations should be improved, a new kind of knowledge should be incorporated, or new manipulation techniques should be developed. Some of these are straight-forward extensions to the model, while others are substantial research problems in their own right.

1. The TOUR model should include states of partial knowledge (or partial progress) for the problem-solving processes. These are potentially present already in the GET-TO and TAKE instructions. It should be possible to examine the result of a partially-successful problem-solving attempt, or to compare two partial solutions to a problem to see which should be refined to a complete solution.

2. It should be possible to make inferences by examining the history of internal operations associated with parts of the cognitive map. For example, repeated confusion or errors associated with a particular geographical feature may suggest that its description is incorrect and leading to paradoxes. In general, the TOUR model lacks processes for checking for inconsistencies and for debugging the cognitive map.

3. There are flaws in the PATH description and the way it interacts with the PLACE and REGION descriptions. When corrected, the abstraction process should be smoother, and the kinds of partial orders which are permitted will be more restricted.

4. Metrical information is assumed to be accurate and precise, an assumption which is known

to be false of people in interesting ways. In particular, a number has no states of partial knowledge about the magnitude of a quantity.

5. There are a number of properties of REGIONs that bear further investigation. What role is played by regions with several boundaries, especially in relating to containing single-boundary regions? How is the geometrical shape of a boundary represented? What kind of internal structure does it have? How should a region be represented when more than one type is appropriate?

Related problems

There are, of course, many aspects of the enormous and complex domain of spatial knowledge which are not directly addressed by the TOUR model. Many related activities involve translating information to or from the cognitive map. Research into these activities would seem to require a detailed understanding of the representations which make up the cognitive map.

1. There are interesting linguistic conventions for describing routes and relative positions. These apparently involve a sense of the appropriate level of detail required to relate what is said to the hearer's cognitive map. This research would treat both the processes for translating information from the cognitive map representation into English and the description a speaker has of a hearer's cognitive map.

2. Map reading and map drawing are two forms of communication with the cognitive map

which use highly constrained media. Map reading exploits a correspondence between the cognitive map and conventionally determined graphical symbols. The cognitive map permits many states of partial knowledge, while the graphical map permits powerful visual search and association methods. Map drawing encourages dramatic inferences as the under-constrained cognitive map representation is translated into the over-constrained graphical representation. These inferences depend on the symbolic feedback which occurs as a person reads the map as he draws it.

3. Visual and kinesthetic space differ from large-scale space in that information is provided at a much greater rate and that the distinction between local evidence and global structure is less clear, though just as important. Since the three spatial representations interact at many points, they must have much in common. Research into the structure of visual space has been taking place (e.g. Marr, et al.), but relatively little into common-sense kinesthetic space or into the interactions.

4. The "flash of comprehension" is a dramatic subjective event where a single new geographical fact initiates the reorganization of a substantial part of the cognitive map. Often this is triggered by the realization that two separate PLACE descriptions actually refer to the same physical place. Rather than occuring in an instant, however, the "flash" seems to take place over hours or even days.

5. Spatial metaphors play a prominent role in memory and problem-solving in non-spatial domains. One striking example is the common-sense representation of time. It seems initially plausible to describe time in terms of partial orders of events, along with a structure of intervals linked by containment relations. Some intervals would be defined by boundary

events, while others would be defined in terms of common characteristics of the contained events. Part of the partial ordering would come from the ordering on associated dates, when those are known. Other parts would come from causal links.

Summary

This research is based on the assumption that the representation of knowledge in the cognitive map, and the methods by which new information is assimilated into that representation, are fundamental to understanding spatial cognition. Other aspects of spatial cognition can only be fruitfully investigated on the basis of a prior theory of representation. More specificially, a model consisting of symbolic descriptions and computational processes is an adequate way to describe the cognitive map.

The TOUR model is presented as such a model, whose detailed structure is designed to correspond to the evidence that is available about human common-sense knowledge of large-scale space. The fundamental principle behind its design is that the states of partial knowledge supported by the representations are responsible for the desirable characteristics of common-sense knowledge. The goal that it attempts to achieve is to provide a precise descriptive language for spatial knowledge that will make it possible to state and answer new and interesting questions.

NOTES

{Note Diagrams}

It is clear that diagrams of the knowledge state would help an example like this. Unfortunately, existing graphical conventions for drawing maps make it difficult to represent partial geographical knowledge without committing outselves to something actually false.

{Note LINGOL}

The natural language input to the TOUR model implementation is provided by a small *context-free grammar parsed* by LINGOL [Pratt 1973]. Its grammar has 86 rules, and it has a vocabulary of 123 words, to which it can add *place-names learned from the input*. Semantic interpretation is accomplished by a decision-tree, branching on the verb, the surface cases which appear in the sentence, and the type of object filling that case. Referents are found by procedures called from the leaves of the semantic-interpretation tree, which attempt to match a name or noun-phrase with the appropriate environmental description. They create new descriptions when *no pre-existing referent can be found*. Responses are generated from templates, with names or descriptions of the relevant geographical features filled in. A sentence can typically be read, parsed, interpreted, assimilated, and a response printed in less than a second.

{Note Look-behind}

Throughout most of the TOUR model, only the current instruction is accessed by a production. GO-7, GO-19, and TURN-15 are the only productions which violate this rule, and they only examine the immediately preceding instruction.

{Note Principles}

David Marr (1975) has collected four design principles for the organization and representations used by complex symbolic processes. These have been expressed in various forms in the Artificial Intelligence literature for a number of years.

1. The principle of explicit naming: Whenever a collection of data is to be described, discussed or manipulated as a whole, it should first be given a *name*. This forms the data into an entity in its own right, permits properties to be assigned to it, and allows other structures and processes to refer to it.

2. The principle of modular design: Any large computation should be split up and implemented as a collection of small sub-parts that are as nearly independent of one another as the overall task allows.

3. The principle of least commitment: One should never do something that may later have to be undone.

4. The principle of graceful degradation: Whenever possible, degrading the data should not prevent one from delivering at least some of the answer.

{Note Quantities}

It is clear that the representations which people use for quantities are quite different from the integers in both accuracy and precision. The integer representation is used for lack of one with more states of partial knowledge. There are some indications, however, of the qualities which such a representation should have. We must distinguish between: 1) quantities like distance which are unbounded, 2) quantities which occur on a bounded interval, and 3) quantities like angle which are always on the same scale. A more realistic distance representation might have a fixed resolving ability applied to a variable base-line distance.

In the implementation of the TOUR machine, the only headings that can be directly read from input sentences are multiples of 45 degrees. Other headings can be computed by dead reckoning from these, of course. Furthermore, two headings will match on retrieval from the local geometry of an intersection if they are within 20 degrees. It appears that little of the structure of the TOUR model would be affected if distance were represented in a different way.

{Note Shape Theory}

The TOUR model does not include a theory to account for the shapes of streets. One version of such a theory would begin with simple descriptors for the street such as CURVING-RIGHT, CURVING-LEFT, or TWISTY. The next level of description would estimate the cumulative amount of curvature over a segment of the street. Here we encounter further problems with

representing quantity. We also encounter problems with segmenting the street description.

{Note Variation}

The kinds of variation supported by a system based on a collection of inference rules offer an initially plausible explanation for human individual variation. First, a given task may be performed by a number of different rules that vary somewhat in the conditions of their applicability. Second, variations in ordering of a given collection of rules can result in variations in behavior. Third, adding more rules to handle more cases generally improves performance, and thus may correspond to developmental learning. Fourth, different rules which accomplish a given task may vary in their ability to support additional rules to expand their applicability.

The important consequence of these facts is that substantial amounts of behavioral variation can be produced by relatively minor changes to the set of rules within a fixed overall system architecture. Substantial individual variation does not depend on fundamentally different processing strategies.

REFERENCES

Appleyard, D. 1970. Styles and methods of structuring a city. *Environment and Behavior* 2: 100-118.

Baylor, G. W. 1971. A treatise on the mind's eye: an empirical investigation of visual mental imagery. Unpublished Ph.D. thesis, Carnegie-Mellon University, Department of Psychology.

Collins, A., E. H. Warnock, N. Aiello, and M. Miller. 1975. Reasoning from incomplete knowledge. In D. G. Bobrow and A. Collins. (Eds.) 1975. *Representation and Understanding.* New York: Academic Press.

Cooper, L. A., and R. N. Shepard. 1973. Chronometric studies of the rotation of mental images. In W. G. Chase. (Ed.) 1973. *Visual Information Processing.* New York: Academic Press.

Denofsky, M. E. 1976. How near is near? MIT Artificial Intelligence Laboratory Memo 344.

Downs, R. M., and D. Stea. (Eds.) 1973. *Image and Environment.* Chicago: Aldine Publishing Company.

Evans, T. G. 1968. A program for the solution of geometric-analogy intelligence Test questions. In [Minsky, 1968].

Gladwin, T. 1970. *East is a Big Bird.* Cambridge: Harvard University Press.

Greif, I. 1975. Semantics of communicating parallel processes. MIT Project MAC Technical Report 154.

Hart, R. A., and G. T. Moore. 1973. The development of spatial cognition: a review. In [Downs and Stea 1973].

Hewitt, C. 1971. Procedural semantics. In R. Rustin. (Ed.) 1971. *Natural Language Processing.* New York: Algorithmics Press.

Hewitt, C., and B. Smith. 1975. Towards a programming apprentice. *IEEE Transactions on Software Engineering* SE-1: 26-45.

Howe, J. A. M., and R. M. Young. 1976. Progress in cognitive development. University of Edinburgh Department of Artificial Intelligence Research Report 17.

Jeffery, M. 1977. Personal communication.

Just, M. A., and P. A. Carpenter. 1976. Eye fixations and cognitive processes. *Cognitive Psychology* 8: 441-480.

Kaplan, S. 1973. Cognitive maps in perception and thought. In [Downs and Stea, 1973].

Kaplan, S. 1976. Adaptation, structure, and knowledge. In [Moore and Golledge, 1976].

Kosslyn, S. M., and J. R. Pomerantz. 1977. Imagery, propositions, and the form of internal representations. *Cognitive Psychology* 9: 52-76.

Lynch, K. 1960. *The Image of the City*. Cambridge: MIT Press.

Marr, D. 1975. Early processing of visual information. MIT Artificial Intelligence Memo 340.

Marr, D. 1976. From computational theory to psychology and neurophysiology -- a case study from vision. MIT Artificial Intelligence Working Paper 131.

Minsky, M. 1968. *Semantic Information Processing*. Cambridge: MIT Press.

Minsky, M. 1975. A framework for representing knowledge. In P. H. Winston. (Ed.) 1975. *The Psychology of Computer Vision*. New York: McGraw-Hill.

Moon, D. A. 1974. *Maclisp Reference Manual*. Cambridge: MIT Project MAC.

Moore, G. T., and R. G. Golledge. (Eds.) 1976. *Environmental Knowing: Theories, Research, and Methods*. Stroudsburg, PA: Dowden, Hutchinson and Ross.

Moran, T. P. 1973. The symbolic imagery hypothesis: a production system model. Unpublished Ph.D. thesis, Carnegie-Mellon University, Department of Computer Science.

Newell, A., and H. A. Simon. 1972. *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.

Norman, D. A., and D. G. Bobrow. 1975. On data-limited and resource-limited processes. *Cognitive Psychology* 7: 44-64.

Quillian, M. R. 1968. Semantic memory. In [Minsky, 1968].

Papert, S. 1972. Teaching children to be mathematicians versus teaching about mathematics. Int. J. Math. Educ. Sci. Technol. 3: 249-262.

Piaget, J., and B. Inhelder. 1967. *The Child's Conception of Space*. New York: Norton. (First published in French, 1948.)

Pratt, V. 1973. A linguistics oriented language. Proceedings of the Third International Joint Conference on Artificial Intelligence, 1973.

Pylyshyn, Z. W. 1973. What the mind's eye tells the mind's brain: a critique of mental imagery. *Psychological Bulletin* 80: 1-24.

Pylyshyn, Z. W. 1976. Imagery and artificial intelligence. In W. Savage. (Ed.) 1976. *Minnesota Studies in the Philosophy of Science*, Vol. IX. Minneapolis: University of Minnesota Press.

Raphael, B. 1968. SIR: a computer program for semantic information retrieval. In [Minsky

1968].

Rumelhart, D. E. 1974. The room theory. Unpublished computer listing. La Jolla, CA: University of California, San Diego.

Sacerdoti, E. D. 1974. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence* 5: 115-135.

Sacerdoti, E. D. 1975. A structure for plans and behavior. Stanford Research Institute Artificial Intelligence Center Technical Note 109.

Shepard, R. N., and S. A. Judd. 1976. Perceptual illusion of rotation of three-dimensional objects. *Science* 191: 952-954.

Shepard, R. N., and J. Metzler. 1971. Mental rotation of three-dimensional objects. *Science* 171: 701-703.

Siegel, A. W., and S. H. White. 1975. The development of spatial representations of large-scale environments. In H. W. Reese. (Ed.) 1975. *Advances in Child Development and Behavior*, Vol. 10. New York: Academic Press.

Simon, H. A. 1969. *The Sciences of the Artificial*. Cambridge: MIT Press.

Stevens, A. L. 1976. The role of inference and internal structure in the representation of spatial information. Unpublished doctoral dissertation. University of California at San Diego, Psychology Department.

Trowbridge, C. C. 1913. Fundamental methods of orientation and imaginary maps. *Science* 88: 888-897.

Winston, P. H. 1975. Learning structural descriptions from examples. In P. H. Winston. (Ed.) 1975. *The Psychology of Computer Vision*. New York: McGraw-Hill.

Appendix A: Descriptions of Environment, Current Position, and Route

Descriptions of the environment

A PLACE is a description of a zero-dimensional geographical object. It includes the PATHs which that PLACE is on, and a local geometry defining the radial headings of (PATH DIRECTION) pairs leaving the PLACE, with respect to an arbitrary coordinate system. The description need not be complete.

```
PLACE:
        NAME:  <name>
        ON:  <list of PATHs>
        STAR:  <local geometry datastructure>
        CONNECT:  <list of pairs:  (PLACE route) >
        ORIENT:  <ORIENTATION-FRAME>
        VIEW:  <list of triples:  (PLACE heading distance) >
        CARDINAL:  <list of triples:  (PLACE heading distance) >
        IN:  <list of REGIONs>
        REGION:  <region>
        CONTAINED:  <list of PLACES>
```

A PATH is a description of a one-dimensional geographical object. It includes a partial order on the set of PLACEs on that PATH. The DIRECTION on a PATH can be +1 or -1, meaning "with" or "against" the order on the PATH.

```
PATH:
        NAME:  <name-frame>
        ROW:  <partial order data-structure>
        HEADING:  <list of pairs:  (ORIENTATION-FRAME HEADING) >
        SHAPE:  <descriptive tag, e.g. STRAIGHT>
        RIGHT:  <REGION>
        LEFT:  <REGION>
        IN:  <list of REGIONs>
        PARALLEL:  <list of pairs:  (PATH +1/-1) >
```

A REGION is a set of PLACEs, to which a global description can be applied; for example, nature of grid or shape of outline.

```
REGION:
        TYPE:  <one of (ORIENT, BOUNDED, STRUCTURED, NAMED) >
        NAME:  <NAME>
        GENERIC-TYPE:  <one of (CONTINENT, COUNTRY, STATE, ... ) >
        SCALE:  <integer>
        DOWNWARD:  <list of 6-tuples:  (HPLACE HPATH HDIR LPLACE LPATH LDIR) >
        BUNDLE:  <partial ordered set of pairs:  (PATH +1/-1) >
        BOUNDARY:  <list of pairs:  (PATH +1/-1) >
        ORIENT:  <ORIENTATION-FRAME>
        PLACES:  <list of PLACEs>
        PATHS:  <list of PATHs>
```

An ORIENTATION-FRAME is a set of PLACEs, along with a system of coordinates with respect to which distances and directions can be defined between pairs of those PLACEs. A HEADING is a direction (in degrees) defined with respect to an ORIENTATION-FRAME.

```
ORIENTATION-FRAME:
        TYPE:  <one of (LOCAL, REGIONAL, CARDINAL) >
        DOMAIN:  <PLACE or REGION>
        OTHERS:  <list of pairs:  (ORIENTATION-FRAME ANGLE) >
```

A ROUTE is a description of the route, holding the source and destination of the route, and a sequence of route instructions.

```
ROUTE:
        FROM:  <place>
        TO:  <place>
        SEQUENCE:  <list of TOUR instructions>
```

A NAME is a description which associates a print-name, consisting of a list of LISP atoms, with one or more referents, which are descriptions.

```
NAME:
        PNAME:  <list of atoms>
        NAME-OF:  <list of descriptions>
```

The map is a set of PLACEs, PATHs, ORIENTATION-FRAMEs, and REGIONs.

Describing the current position

The "You Are Here" pointer describes the current position of the TOUR machine in the map. It includes the current PLACE, PATH, DIRECTION, ORIENTATION-FRAME, and HEADING. Some of these may be left unspecified.

```
YOU ARE HERE:
        PLACE:  <place>
        PATH:  <path>
        DIRECTION:  <direction (+1, -1) >
        ORIENTATION-FRAME:  <orientation-frame>
        HEADING:  <integer (0, 360) >
        X:  <integer>
        Y:  <integer>
```

The X and Y elements of the "You Are Here" pointer are for temporary storage when a route is being followed to compute a position. They hold the current rectangular coordinates with respect to the current orientation-frame and the starting point.

Route instructions

GO-TO instructs TOUR to move the "You Are Here" pointer from one PLACE to another on the same PATH, and specifies the direction of travel with respect to the PATH order. GO-TO/DIST is automatically filled with the "observed" distance of travel.

```
GO-TO:
        FROM:  <place>
        TO:  <place>
        ON:  <path>
        DIR:  <direction>
        DIST:  <integer>
```

TURN instructs TOUR to move the "You Are Here" pointer from one (PATH DIRECTION) pair to another at the same PLACE, and specifies the amount of the turn.

```
TURN:
        AT:  <place>
        ST1:  <path>
        DIR1:  <direction>
        AMT:  <number of degrees>
        ST2:  <path>
        DIR2:  <direction>
```

The GET-TO instruction formulates a problem to be solved by the problem-solving component by specifying the state of the "You Are Here" pointer at source and destination. In case no solution can be found, the problem-statement is left as an instruction in the route program, and TOUR continues from its destination.

```
GET-TO:
        FROM:  <place>
        ST1:  <path>
        DIR1:  <direction>
        TO:  <place>
        ST2:  <path>
        DIR2:  <direction>
```

The TAKE instruction says that a particular route should be used as a sub-program of the primary route, going from TAKE/FROM to TAKE/TO.

```
TAKE:
        ROUTE:  <route>
        FROM:  <place>
        TO:  <place>
```

The NOTICE instruction says that, if the traveller is at the given place, path, and direction, then the remote place can be observed at the given distance and heading. The heading is defined with respect to an egocentric set of coordinates within which the heading of the observer is 0.

```
NOTICE:
        FROM:  <PLACE>
        PATH:  <PATH>
        DIR:  <direction>
        REMOTE:  <PLACE>
        DISTANCE:  <distance in units>
        HEADING:  <egocentric heading in degrees>
```

When a change has been selected in the focus of attention between different descriptions of the same place, it is communicated to the TOUR machine through the UP and DOWN instructions. These instructions change the "You Are Here" pointer without corresponding to any physical travel from one place to another. They have the following format:

```
UP:                              DOWN:
        FROM:  <place>                   FROM:  <place>
        ST1:  <path>                     ST1:  <path>
        DIR1:  <direction>               DIR1:  <direction>
        TO:  <place>                     TO:  <place>
        ST2:  <path>                     ST2:  <path>
        DIR2:  <direction>               DIR2:  <direction>
```

RESET clears the "You Are Here" pointer and sets it to a new PLACE, PATH, and DIRECTION, ready to execute a new program.

```
RESET:
        PLACE:  <place>
        PATH:  <path>
        DIR:  <direction>
```

Appendix B: Inference Rules

These inference rules are stated in English in order to eliminate inessential details from the actual computer code, and to make them understandable to those not conversant with LISP. Each rule checks a number of conditions and, if they are satisfied, performs an action. The system is modular in that the model will function, although with somewhat different properties, with virtually any selection and ordering of the given rules.

Each rule is associated with an instruction type, and is active (allowed to check its conditions and run if successful) when the current instruction is of that type.

To clarify the notation: "GO-TO/FROM" refers to the FROM property of the GO-TO description; and "C-PLACE" refers to the PLACE element of the "You Are Here" pointer. Similar terms have similar meanings.

GO-TO

GO-1, GO-2, and GO-3 check the current position as represented in the "You Are Here" pointer with the current position assumed by the GO-TO instruction. Information from one can be used to fill a gap in the other. If the two descriptions disagree, an intermediate instruction may be added to get from the current position to what the instruction assumes.

GO-1: Compare GO-TO/FROM and C-PLACE. If both are empty, there is an error. If they agree, do nothing. If only one is supplied, set the other from its value. If they disagree, insert a GET-TO instruction as the new current instruction preceding this GO-TO, and restart.

GO-2: Compare GO-TO/PATH and C-PATH. If both are empty, there is an error.

If they agree, do nothing. If only one is supplied, set the other from its value. If they disagree, insert a TURN instruction, and restart.

GO-3: Compare GO-TO/DIR and C-DIRECTION. If they agree, do nothing. If only one is supplied, use its value to set the other. If they disagree, insert a "Turn around" instruction.

GO-4, GO-5, and GO-6 transfer topological connection and order information between the GO-TO instruction and its path and endpoints.

GO-4: Send GO-TO/PATH to GO-TO/FROM/ON and GO-TO/TO/ON. (A PLACE description checks internally for redundant information.)

GO-5: If GO-TO/DIR is supplied, send GO-TO/FROM and GO-TO/TO as an ordered pair to GO-TO/PATH/ROW. If not, send them as two singletons. (The PATH description will incorporate this information into its partial order.)

GO-6: Ask GO-TO/PATH if it knows an order on GO-TO/FROM and GO-TO/TO. If not, do nothing. If so, and GO-TO/DIR is empty, set it. If so, and it disagrees with GO-TO/DIR, complain of an error. Otherwise, do nothing.

GO-7 transfers direction information, which can be obtained by a GO-TO instruction from the PATH description, to the destination part of the preceding TURN instruction. This is one of the *few rules that accesses more than one instruction.*

GO-7: If GO-TO/DIR is supplied, and the preceding instruction was a TURN with empty TURN/DIR2, then set that TURN/DIR2 to the value of GO-TO/DIR. (Note look-behind)

GO-8 and GO-9 perform the change on the "You Are Here" pointer which is the primary effect of a GO-TO instruction. One sets the current position to the destination of the instruction, and the other permits the current heading to remain constant unless the path curves.

GO-8: Set C-PLACE to GO-TO/TO, and C-DIRECTION to GO-TO/DIR.

GO-9: If C-HEADING is set, then check C-PATH/SHAPE. If its value is STRAIGHT, then do nothing; else clear C-HEADING and C-ORIENT.

The relationship between C-HEADING and C-DIR can be represented in C-PATH/HEADING, indexed under the current orientation frame. GO-10, GO-11, and GO-12 use this relationship to fill empty parts of the "You Are Here" pointer or the current PATH description.

GO-10: If C-ORIENT and C-DIR are both supplied, but C-HEADING is not, then

see if C-ORIENT is associated with a heading in C-PATH/HEADING. If so, take C-DIR into account, and set C-HEADING.

GO-11: If C-ORIENT and C-HEADING are both supplied, but C-DIR is not, then check C-PATH/HEADING to see if C-ORIENT is associated with a heading. If so, and it is close to C-HEADING or its opposite, set C-DIR accordingly. If the headings match very poorly, flag a potential error, and go on.

GO-12: If C-HEADING and C-DIR are both supplied, then let THETA be the heading of C-PATH in the +1 direction. Put (C-ORIENT THETA) into C-PATH/HEADING.

Two orientation-frames can be compared when C-ORIENT is different from the orientation-frame associated with C-PLACE. This situation is recognized by GO-13 and analyzed by GO-14. As a result, either the relation between the two is stored, or one orientation-frame propagates into the territory of the other and replaces it.

GO-13: If C-ORIENT, C-HEADING, and C-DIR are all set, and C-ORIENT and C-PLACE/ORIENT are different, then a collision of ORIENTATION-FRAMEs has taken place, and GO-14 must run to decide what to do.

GO-14: Obtain the current heading with respect to C-PLACE/ORIENT. If it cannot be found, do nothing. If it differs from C-HEADING by a multiple of 90 degrees, and if C-PLACE/ORIENT is local to C-PLACE, then run GO-15 and GO-16 to merge it with C-ORIENT. Otherwise, record the relation between the reference systems in C-ORIENT/OTHERS and C-PLACE/ORIENT/OTHERS.

GO-15: If C-ORIENT/TYPE is LOCAL, then promote it to REGIONAL and adjust the appropriate pointers.

GO-16: Put C-ORIENT into C-PLACE/ORIENT. Add C-PLACE to C-ORIENT/DOMAIN/PLACES. Change the headings in C-PLACE/STAR and C-PLACE/VIEW to the appropriate value for C-ORIENT.

The integrated change in position accomplished by following a route can be computed by updating C-X and C-Y by the incremental change resulting from the GO-TO instruction. This process occurs only when C-X and C-Y are non-NIL.

GO-17: If C-X, C-Y, C-HEADING, and GO-TO/DIST are all supplied, then translate C-HEADING and GO-TO/DIST into rectangular coordinates and update C-X and C-Y.

GO-18: If C-X and C-Y are set, but either C-HEADING or GO-TO/DIST is empty, then clear C-X and C-Y.

The relation of a place and a boundary can be computed from a TURN instruction followed by a GO-TO.

> GO-19: If the preceding instruction was a TURN onto GO-TO/PATH, with TURN/AMT and TURN/DIR1 specified, then compute which side-region of TURN/ST1 should contain GO-TO/TO, and put it there.

Two streets can be found to be locally parallel if they have equal or opposite headings with respect to the same orientation frame. Two streets are examined if they fall within the focus of attention of a single GO-TO instruction.

> GO-20: Let C1 and C2 be the only cross-streets at GO-TO/FROM and GO-TO/TO, respectively. If the headings of both C1 and C2 can be found with respect to the same ORIENTATION-FRAME, and if they are equal, put (C1 +1) into C2/PARALLEL, and (C2 +1) into C1/PARALLEL. If they are opposite, put (C1 -1) into C2/PARALLEL, and (C2 -1) into C1/PARALLEL.

TURN

TURN-1, TURN-2, and TURN-3 compare the current position in the "You Are Here" pointer with what is assumed by the current TURN instruction. Parts of each can be used to fill gaps in the other. If they disagree sufficiently, an extra instruction may be inserted to get from the current position to that assumed by the TURN instruction.

> TURN-1: Compare TURN/PLACE with C-PLACE. If both are empty, there is an error. If they agree, do nothing. If only one is supplied, set the other from its value. If they disagree, insert a GET-TO instruction as the new current instruction preceding this TURN, and restart.
>
> TURN-2: Compare TURN/ST1 with C-PATH. If both are empty, there is an error. If they agree, do nothing. If only one is supplied, set the other from its value. If they disagree, insert a second TURN instruction before this, comment on the oddness, and restart.
>
> TURN-3: Compare TURN/DIR1 with C-DIRECTION. If they agree, do nothing. If only one is supplied, set the other from its value. If they disagree, there is an error.

TURN-4 adds the topological connection between place and path assumed by the instruction. TURN-5, TURN-6, TURN-7, TURN-8, and TURN-9 implement the relationship between the TURN instruction and the local geometry of the current place, transferring information to where it is needed.

> TURN-4: Send TURN/ST1 and TURN/ST2 to TURN/PLACE/ON. Also, send a singleton order containing TURN/PLACE to TURN/ST1/ROW and TURN/ST2/ROW.
>
> TURN-5: If TURN/ST1, TURN/DIR1, TURN/ST2, TURN/DIR2, and TURN/AMT are all supplied, then send them to TURN/PLACE/STAR, to add to the description of the local geometry of the PLACE. (The PLACE description incorporates this into the appropriate data structure.)
>
> TURN-6: If TURN/ST1, TURN/DIR1, and TURN/AMT are all supplied, but TURN/ST2 and TURN/DIR2 are not both supplied, then ask TURN/PLACE/STAR to compute the description of the outgoing PATH from the incoming PATH and the amount of the TURN. If that is successful, use the result to set TURN/ST2 and TURN/DIR2.
>
> TURN-7: If TURN/ST1, TURN/DIR1, TURN/ST2, and TURN/DIR2 are all supplied, but TURN/AMT is not, then ask TURN/PLACE/STAR to compute it. If that is

successful, set TURN/AMT to the result.

TURN-8: If TURN/ST2 remains unfilled, and TURN/PLACE/ON contains two PATH descriptions, one of which is TURN/ST1, then set TURN/ST2 to the other.

TURN-9: If TURN/ST2 still remains unfilled, then create a new PATH description to fill TURN/ST2. Send it to TURN/PLACE/ON, and send a singleton containing TURN/PLACE to the ROW property of the new PATH. (This PATH description may describe the same real street as a previously created PATH description, requiring some debugging later.)

TURN-10 and TURN-11 accomplish the primary effect of the TURN instruction on the "You Are Here" pointer: updating the current path, direction, and heading.

TURN-10: Set C-PATH to TURN/ST2, and set C-DIRECTION to TURN/DIR2.

TURN-11: If C-HEADING and TURN/AMT are set, then update C-HEADING by TURN/AMT; otherwise clear C-HEADING and C-ORIENT.

TURN-12, TURN-13, and TURN-14 implement the relationship between the current heading and orientation frame and the local geometry of the current place.

TURN-12: If C-ORIENT is empty, set it to C-PLACE/ORIENT.

TURN-13: If C-ORIENT and C-DIR are both supplied, but C-HEADING is empty, and if C-ORIENT matches C-PLACE/ORIENT, then try to set C-HEADING to the heading associated with (C-PLACE C-DIR) in C-PLACE/STAR.

TURN-14: If C-HEADING and C-DIR are both supplied, and if C-ORIENT is the same as C-PLACE/ORIENT, then put (C-HEADING C-PATH C-DIR) into C-PLACE/STAR.

The relationship between a place and a boundary can be computed from a GO-TO instruction followed by a TURN.

TURN-15: If the previous instruction was a GO-TO to TURN/AT, and TURN/AMT and TURN/DIR2 are both specified, then compute which side-region of TURN/ST2 should contain GO-TO/FROM, and put it there.

NOTICE

The NOTICE instruction simulates an observation of a remote landmark. NOTICE-1 checks that the preconditions of the instruction are satisfied. NOTICE-2 adds a description of the observation to the current PLACE description. NOTICE-3 can use the observation to determine the current heading. NOTICE-4 can use the observation to compute the relationship between two orientation frames.

NOTICE-1: Check that NOTICE/PLACE, NOTICE/PATH, and NOTICE/DIR match C-PLACE, C-PATH, and C-DIR. If not, complain and dismiss.

NOTICE-2: If C-ORIENT and C-HEADING are supplied, and if C-ORIENT matches C-PLACE/ORIENT, then look up the position of NOTICE/REMOTE in C-PLACE/VIEW. If none is there, add it from the NOTICE instruction, compensating for egocentric coordinates by using C-HEADING. If one is there, but fails to match the NOTICE instruction, then complain. Otherwise, do nothing.

NOTICE-3: If C-ORIENT is supplied and matches C-PLACE/ORIENT, but C-HEADING is not supplied, and if C-PLACE/VIEW has a position for NOTICE/REMOTE, then use this information to set C-HEADING.

NOTICE-4: If C-ORIENT and C-HEADING are both supplied, and there is a position for NOTICE/REMOTE in C-PLACE/VIEW, but C-ORIENT does not match C-PLACE/ORIENT, then the relation between the two orientation-frames can be computed and put into C-ORIENT/OTHERS and C-PLACE/ORIENT/OTHERS.

TAKE

The TAKE instruction refers to an embedded route, which is executed by the TOUR machine in a recursive call in TAKE-5. The other rules check for strange conditions: TAKE-1 inserts a GET-TO if a gap exists; TAKE-3 checks for correct formats; and TAKE-2 and TAKE-4 replace the take instruction with its referent if the embedded route is only one instruction long.

TAKE-1: If TAKE/from is not equal to C-PLACE, then insert a GET-TO. Restart.

TAKE-2: If TAKE/route is a single instruction (a GO-TO), then replace the TAKE with the GO-TO. Restart.

TAKE-3: If TAKE/from and TAKE/to do not agree with TAKE/route/from and TAKE/route/to, complain of a contradiction.

TAKE-4: If TAKE/route/sequence is only one instruction long, replace the TAKE with that instruction. Restart.

TAKE-5: Else make a recursive call to TOUR to execute TAKE/route. Use the embedded value of C-TRIP to update TAKE/route/sequence, and put the updated TAKE instruction on the outer C-TRIP.

GET-TO

GET-TO allows problems to be posed as part of a route-description. It also allows route-descriptions with missing instructions to be completed. GET-1 and GET-2 detect the cases where yet another instructions is required. GET-3 calls the problem-solving component of the TOUR model to propose a solution to the problem. GET-4 handles the case where no solution is found.

> GET-1: If GET-TO/from is not equal to C-PLACE, then insert yet another GET-TO, and restart.
>
> GET-2: If GET-TO/from equals GET-TO/to, then this GET-TO can be replaced with a *TURN*. *Restart*.
>
> GET-3: If none of the above apply, call PROPOSE-ROUTE to get a route from GET-TO/from to GET-TO/to. If this succeeds, make a recursive call to TOUR to execute the resulting route, which will go on C-TRIP as a TAKE instruction.
>
> GET-4: If GET-3 fails, simply jump to GET-TO/to, explaining the problem to the user, and put the GET-TO instruction on C-TRIP, hoping that the problem will be easier some other time.

GET-5 and GET-6 implement the cases where the source and goal of the instruction are simply descriptions of the same place at different levels of detail. The more abstract description must be mapped down to a more particular one, and a different GET-TO instruction provided to link the more particular places (Chapter 5). These rules must actually appear between GET-2 and GET-3 when the TOUR machine runs.

> GET-5: If GET-TO/FROM is a superior of GET-TO/TO, then map downward from (GET-TO/FROM GET-TO/ST1 GET-TO/DIR1) to a lower description (LPLACE LPATH LDIR) of the same position. Replace this GET-TO with two instructions: a DOWN to the lower state, and a GET-TO from the lower state to the goal.
>
> GET-6: If GET-TO/TO is a superior of GET-TO/FROM, then map downward from (GET-TO/TO GET-TO/ST2 GET-TO/DIR2) to a lower description (LPLACE LPATH LDIR) of the goal state. Replace this GET-TO with two instructions: a *GET-TO from the current state to the lower goal state*, and an *UP instruction* to the higher goal state.

UP and DOWN

The UP and DOWN instructions accomplish changes in the focus of attention between abstract and particular descriptions of the same place. UP-1 and DOWN-1 check to ensure that the preconditions are met. UP-2 and DOWN-2 change the "You Are Here" pointer to the appropriate position.

> UP-1: If the current position in C-PLACE, C-PATH, C-DIR doesn't match the current position in UP/FROM, UP/ST1, and UP/DIR1, insert a GET-TO instruction before the UP. Dismiss.
>
> UP-2: Reset C-PLACE, C-PATH, and C-DIR to UP/TO, UP/ST2, and UP/DIR2, respectively. Clear the rest of the "You Are Here" pointer.
>
> DOWN-1: If the current position in C-PLACE, C-PATH, C-DIR doesn't match the current position in DOWN/FROM, DOWN/ST1, and DOWN/DIR1, insert a GET-TO instruction before the DOWN. Dismiss.
>
> DOWN-2: Reset C-PLACE, C-PATH, and C-DIR to DOWN/TO, DOWN/ST2, and DOWN/DIR2, respectively. Clear the rest of the "You Are Here" pointer.

Appendix C: Implementing descriptions

The descriptions that the TOUR model uses to represent knowledge about the environment are implemented in LISP as property-lists of generated atoms. Access to the property-value pairs, however, is not through the LISP functions GET and PUTPROP, but through special access functions GETSLOT and PUTSLOT.

GETSLOT and PUTSLOT act as an interface with access functions associated with the property-name of that type of description. Thus, the ROW property of a PATH description has a PUT-function which adds a new piece of order information to the partial order datastructure held in the ROW property. In cases where the appropriate access to a property is not analogous to a GET or PUTPROP, the access function is called explicitly, rather than through GETSLOT or PUTSLOT. An example of this is WH-ORDER, the function which asks a partial-order datastructure for the order relation between two given objects.

Descriptions belong to classes. All of the descriptions in a given class share the same properties and access functions (though of course the values of those properties are different for different descriptions).

Consider the particular example of PATH descriptions. A new PATH description, with its properties set to default values, is created by a call to the function CONS-PATH.

```
PATH0:
        NAME:  nil
        ROW:  nil
        HEADING:  nil
        SHAPE:  STRAIGHT
        RIGHT:  nil
        LEFT:  nil
        IN:  nil
        PARALLEL:  nil
        CLASS:  CONS-PATH
```

Subsequent calls to CONS-PATH will produce PATH descriptions on the property-lists of the generated atoms PATH1, PATH2, and so on. Notice that PATH/SHAPE has the initial default value of STRAIGHT until further information is provided. PATH/CLASS refers to the description of the class, where the access functions for PATH properties are to be found.

The class description has the following properties:

```
CONS-PATH:
        EXPR:  <function for creating PATH descriptions>
        GETFNS:  (DEFAULT GET
                  RIGHT GET-SIDE-REGION
                  LEFT GET-SIDE-REGION)
        PUTFNS:  (DEFAULT PUTPROP
                  NAME PUTNAME
                  ROW ADD-TO-ORDER
                  HEADING PUTLIST
                  IN PUTLIST
                  PARALLEL PUTLIST)
```

Notice that default access functions, in this case GET and PUTPROP, are provided in case nothing is specified for a particular property. GET-SIDE-REGION returns the REGION description if one is in the appropriate property, and creates one with the proper linkages if not. PUTLIST is a useful general-purpose PUT-function which adds a new element to a list, checking first for duplications. We have already seen ADD-TO-ORDER as the access function for the partial-order datastructure in the ROW property. PUTNAME simply sets up the proper back-pointers from the NAME description to the thing named.

The class of PATH descriptions is created by a call to DEFRAME which supplies the structure of the description and the access functions.

```
(DEFRAME CONS-PATH                          ; class description
         PATH                               ; prefix for generated atoms
         (NAME NIL                          ; properties and initial values
          ROW NIL
          HEADING NIL
          SHAPE STRAIGHT
          RIGHT NIL
          LEFT NIL
          IN NIL
          PARALLEL NIL)
         (GETFNS (DEFAULT GET               ; access functions
                  RIGHT GET-SIDE-REGION
                  LEFT GET-SIDE-REGION)
          PUTFNS (DEFAULT PUTPROP
                  NAME PUTNAME
                  ROW ADD-TO-ORDER
                  HEADING PUTLIST
                  IN PUTLIST
                  PARALLEL PUTLIST)))
```

GETSLOT and PUTSLOT simply look up their property-name under GETFNS or PUTFNS on the class description to find the access function for that property. The definition of GETSLOT is representative. It is written in MACLISP [Moon 1974].

```
(DEFUN GETSLOT (DESCRIPTION PROPERTY)
       ((LAMBDA (ACCESS)
                (OR ACCESS (FAULT))
                (FUNCALL (OR (GET ACCESS PROPERTY)
                             (CAR ACCESS))
                         DESCRIPTION
                         PROPERTY))
        (CDR (MEMQ 'DEFAULT
                   (GET (GET DESCRIPTION 'CLASS) 'GETFNS)))))
```

DEFRAME takes the information provided by its arguments and sets up the properties of the class description. It also substitutes particular values into a template to create the function which generates new descriptions of that class.

```
(DEFUN DEFRAME FEXPR (ARGS)
       (PUTPROP (CAR ARGS)
                (SUBLIS (LIST (CONS '^IMEM (CADDR ARGS))
                              (CONS '^TYPE (CADR ARGS))
                              (CONS '^ILIST (CAR ARGS))
                              (CONS '^CMEM (CAR ARGS)))
                        '(LAMBDA ()
                                 (PROG (SELF)
                                       (SETQ SELF (GEN-FRAME-NAME '^TYPE))
                                       (SETQ ^ILIST (CONS SELF ^ILIST))
                                       (SET SELF SELF)
                                       (SETPLIST SELF
                                                 (APPEND '^IMEM
                                                         '(CLASS ^CMEM)
                                                         (PLIST SELF)))
                                       (RETURN SELF))))
                'EXPR)
       (PUTPROP (CAR ARGS)
                (CADR (MEMQ 'GETFNS (CADDDR ARGS)))
                'GETFNS)
       (PUTPROP (CAR ARGS)
                (CADR (MEMQ 'PUTFNS (CADDDR ARGS)))
                'PUTFNS)
       (SETQ FRAME-LISTS (CONS (CAR ARGS) FRAME-LISTS))
       (SET (CAR ARGS) NIL)
       (CAR ARGS))
```

GEN-FRAME-NAME generates the atoms to which new property-lists are attached. FRAME-LISTS is a global variable which lists the current class descriptions. The value of the class-description atom is a list of its instances.