The University of Wisconsin
Computer Sciences Department
1210 West Dayton Street
Madison, Wisconsin 53706

Received: August 1974

TOWARD INTEGRATED COGNITIVE SYSTEMS, WHICH MUST MAKE FUZZY DECISIONS ABOUT FUZZY PROBLEMS

by

Leonard Uhr

Computer Sciences Technical Report #222

August 1974

TOWARD INTEGRATED COGNITIVE SYSTEMS, WHICH MUST MAKE FUZZY DECISIONS ABOUT FUZZY PROBLEMS*

by

Leonard Uhr

ABSTRACT

This paper presents and gives examples of the behavior of a simple computer-programmed model (called "SEER", for semantic learner,) for an integrated, wholistic cognitive system. The system combines the cognitive functions of pattern recognition, scene description, information retrieval, deducing, and acting that are usually handled in separate programs. This forces it to make fuzzy decisions, e.g. as to what type of behavior (e.g. describe or answer) to effect, and what type of thing (e.g. word or object) is being perceived.

The paper examines a number of aspects of such a system, in order to illustrate how it begins to handle the problems of fuzziness that can no longer be avoided, and should no longer be avoided, since they appear to be at the heart of intelligence.

*Presented at Japanese-American meeting on Fuzzy Sets, Berkeley, July, 1974.

This research has been partially supported by grants from the National Institute of Mental Health (MH-12266), the National Science Foundation (GJ-36312), (NGR-50-002-160) and the University of Wisconsin Graduate School.

INTRODUCTION

When the separate cognitive processes of perception, thinking, remembering, language "understanding", acting, and learning are integrated into a single system, a variety of fuzzy problems inevitably arise. This paper examines such a computer-programmed system, one that is a first attempt to model the integrated, wholistic mind/brain.

The Search for Well-Formed Problems has Focussed "AI" Research on a Few Non-Fuzzy Problems

Artificial intelligence (AI) Research has to a great extent legislated that things be well-formed and non-fuzzy by developing separate systems to handle clear-cut problems with correct answers(e.g. the proof of a theorem, or the answer to a factual question) that can be deduced in a deterministic way. But as soon as we attempt the kinds of problems that human beings can handle (e.g. the description of a scene, or an on-going conversation in which several people exchange information) the system must constantly operate in a fuzzy domain.

AI research has attempted to reduce all cognitive processes to a search for a path from a set of Givens to a Goal, using only a set of legal Transforms. But it is rare that Givens, Goals, and Transforms are known (or, as in a game like chess or GO, easily computable). Rather, the crucial problem may be to find the set of transforms that humans, or some other intelligent entities use (as in pattern recognition), or to determine what are relevant givens (as in describing scenes or answering questions) or worthwhile goals (as in conversation, or in finding interesting new theorems).

Each area of AI uses slightly different basic transforms, overall networks, and search techniques. These are generalized in this paper, so that a single system (called a "SEER") can handle them all. This makes each separated function a good bit richer in its powers, and also a good bit fuzzier in the kinds of things that it does. It further raises new problems of fuzziness, at several ever-higher levels of integration of processes. Relevance must constantly be assessed, as a function of a wide variety of contextually interacting influences. The system must make fuzzy decisions as to the types of things perceived (e.g. objects vs. words), internal processes needed, and external acts suggested. And learning consists in a variety of types of fuzzy conjectures as to general hypotheses posited from particular pieces of experience, and how to use these to build, unbuild, and restructure the cognitive memory network.

The Ill-Formed and Fuzzy Nature of Everyday Thinking

Let's look at the kinds of simple everyday thinking that all of us spend most of our time doing. They are characterized by an intimate interaction between perceiving, deducing, searching, remembering, and acting, with constant monitoring and guidance from feedback. Rarely do we make deep or difficult deductions, or remember surprising or profound concepts. Rather, we conduct a kind of shallow and diffuse on-going "conversation" with our environment, one that finds and uses the relevant aspects of an impossibly large set of possibilities, in order to help us muddle along, often with surprising success.

Examples are the way we decide what to do on a holiday, and actually carry out all the steps needed to do it. Or choosing a dinner, or a restaurant; opening the refrigerator, getting food, baking a cake; deciding what to collect; building a bookcase.

It may feel like belaboring the obvious, but consider the subtle interactions between all the cognitive processes that go on in even the most mundane of acts:

I perceive a can which, in the context of the table it sits on and the conference I am attending, suggests it might contain a liquid to drink; so I glance about and move my head and then walk around it, to look for a spigot and an indicator - whether a sign or a telltale stain or smell - of whether it contains coffee. This arouses vague hunger needs for food, and I further look around for trays of soft objects and, remembering the time of day, for food. I sense that I like cream, deduce that it might be in a pitcher or, if powdered, a deep dish, and look about some more. I pick up a noticed up, and move it under the spigot, pushing its level with my other hand, and carefully monitor the drip. And so on.

Note that a 1-year-old infant engages in much the same interacting set of processes, albeit with fewer and simpler possibilities, e.g. when it babbles, cries, crawls, grabs, flails, pushes, bites, drools, etc., in order to get food. But I must emphasize that despite its surface simplicity, such a process is far more difficult than anything we have approached with our artificial intelligence programs.

This paper describes a model for such processes, and shows some of their many inevitably fuzzy aspects.

ARTIFICIAL INTELLIGENCE HAS SIMPLIFIED THE PSYCHOLOGICAL PROCESSES TO PIECEMEAL FUNCTIONS FOR WELL-FORMED PATH-SEARCHING

Almost all AI research has concentrated on a single, separated cognitive function, and attempted to simplify the problem being attacked to the point where it is "well-formed."

The Separate Functions of Perception, Thinking, Remembering, Acting, Language, and Learning

The separate functions being attacked are closely related to, but usually simplifications of, the traditional cognitive processes that have always interested psychologists (Figure 1):

Perception, which studies the absorbing and understanding of pertinent information from the cognitive system's external environment, has been attacked as the recognition and naming of an isolated patterned object input to the system (see e.g. Duda & Hart, 1973; Uhr, 1973a).

Thinking, which involves a variety of little-understood and only partially identified processes for deciding what are the most relevant and functional things to do in order to cope with external presses (perceived objects, imports, suggestions, commands, etc.) and internal presses (needs, desires, expectations, goals, etc.), has become deductive problem-solving in deterministic domains like games, puzzles, and logical systems (see e.g. Nilsson, 1971; Newell & Simon, 1973).

Remembering, which accesses information in the system's memory model of its world that is most relevant to its present situation and its attendant problems, becomes the search for "correct" answers to clear-cut questions that indeed have answers that can clearly be deduced to be correct (see e.g. Minsky, 1968; Simmons, 1965, 1970).

Acting, which includes a variety of things that the organism can choose to effect upon its world (e.g., touch, grasp, mix, melt, combine, heat, ingest; move its glance, eye, head, body), as a function of its percepts, thoughts and memories, becomes a very conventionalized set of actions, e.g. "move-self" or "push-object" (a specified distance in a specified direction) quite similar to the moves of the different pieces in a game like checkers or chess (see e.g. Nilsson, 1969; Uhr & Kochen, 1969; Winograd, 1971).

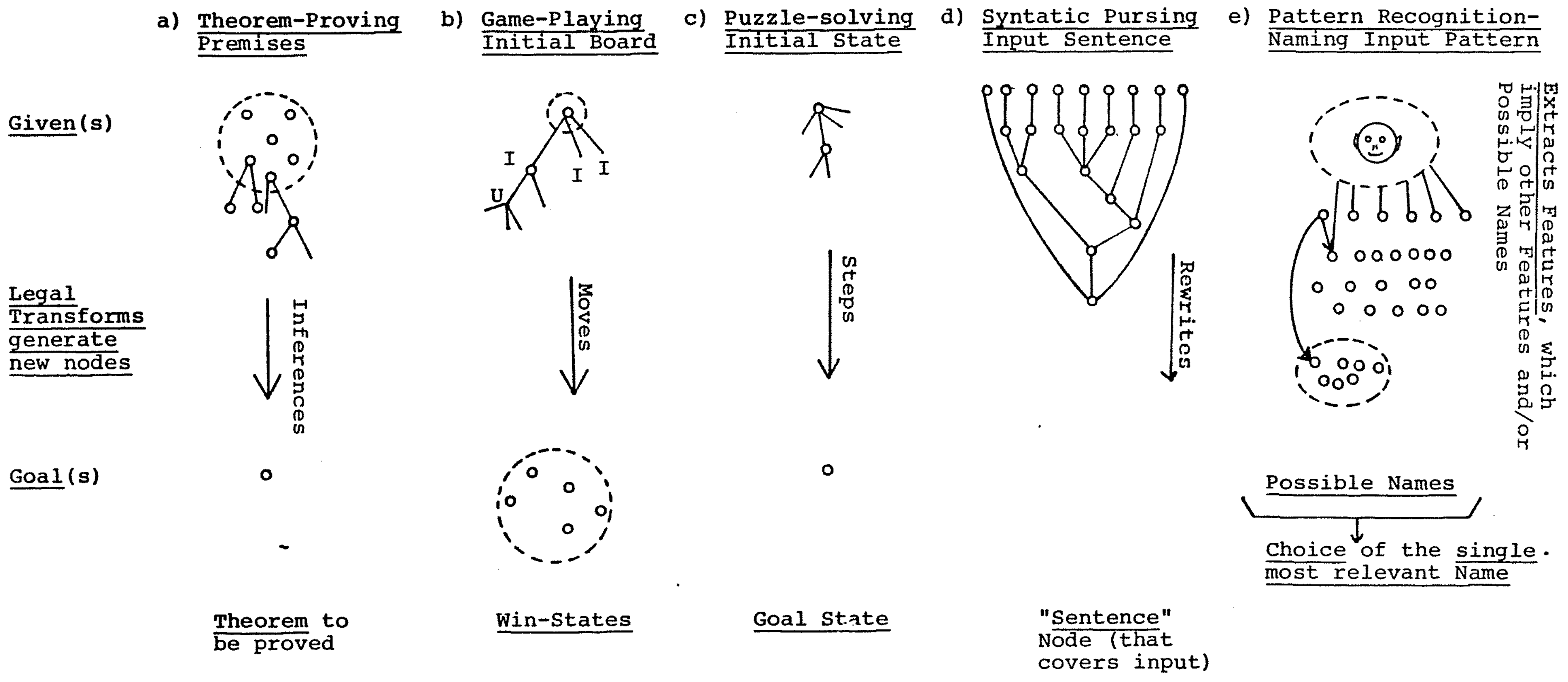
Language understanding, which is an especially mysterious process that involves symbolic reference to the import of percepts, feelings, ideas and acts, has been reduced to syntactic parsing and simple question-answering.

Learning, which must build the memory model and the system's ways of making use of that model in the first place, becomes largely inductive reweighting of the strength of already-existing connections (see e.g. Duda and Hart, 1973) and, to a slight extent, the adding of new connections and the extracting and inferring of new things to be connected (see e.g. Sauvain and Uhr, 1969; Quillian, 1969, Winston, 1970). But for work on learning that moves into ill-formed problems see Kochen, 1961; Uhr and Vossler, 1961; Hunt, 1962; Jordan, 1971; Uhr, 1964, 1973a.)

Psychology	Artificial Intelligence		
	General Focus of Interest	Well-Formed	Ill-Formed and Fuzzy
Perception	Pattern Recognition	Name	Describe and Extract the relevant
Thinking	Deductive Problem-Solving	Path-find	Heuristics; interesting goals, everyday thinking
Remembering	Question-Answering	Parse, Path-find	Converse
Language-Understanding	Syntactic Analysis	Parse	Understand semantically
Motor Behavior	Robots	Touch, Move	"Conversational" interaction with environment
Learning	Learning	Add assertions, reweight	Induction, discovery, generalization

TRADITIONAL PROBLEM-AREAS OF PSYCHOLOGY AND ARTIFICIAL INTELLIGENCE: WELL-FORMED AND FUZZY

Figure 1



WELL-FORMED PROBLEMS OF DETERMINISTIC SEARCH FOR A SOLUTION-PATH; AND ONE ILL-FORMED AND FUZZY PROBLEM

Figure 2

The Deterministic Search for a Solution-Path Between Givens and Goal, Using only Legal Transforms

Science must simplify, and this is especially true when it studies so complex a phenomenon as the intelligent mind. And the simplified problems that Artificial Intelligence examines are important, reasonably representative, and still extremely difficult. But there seems to be an increasingly strong tendency to simplify by choosing problems that are "well-formed" in the following sense: Three sets of things are specified when a problem is posed - "Givens" (e.g. the axioms of a system of logic, the starting board in a game), "Goals" (e.g. the theorem to be proved, the winning boards) and "Legal Transforms" (e.g. the rules of inference, the moves of the game). The problem then becomes one of finding a sequence of legal transforms that forms a "Solution Path" from givens to goals. Thus we have a deductive deterministic search for a legal path between two clearly specified sets of nodes in a graph.

This may be an adequate conceptualization (see Figure 2) for deductive problem-solving, including theorem-proving, game-playing, and puzzle-solving (see Nilsson, 1971) - and these have been the major areas of interest for researchers in "artificial intelligence." It is also adequate for syntactic parsing that insists upon a tree of paths connecting the given sentence with the goal "sentence" node that roots it in a parse (Chomsky, 1957, 1965; Feldman and Gries, 1968.)

There is a greal deal of effort today to absorb other cognitive processes into the same framework (or strait-jacket). "Syntactic pattern recognition" (e.g. Narashimyan, 1964; Shaw, 1969; see Swain & Fu, 1970; and Uhr, 1971) attempts to use parsing techniques to transform an input pattern into a well-formed tree, and "robot vision" (e.g. Guzman, 1968; Waltz, 1972; see Duda and Hart, 1973) attempts something quite similar, since it uses algorithms to match stored network models of objects with the input object. Question-answerers (e.g. Thompson, 1966; Shapiro, 1971; Quillian, 1969; see Simmons, 1965, 1970) typically put two such well-formed systems side by side: First a parser extracts information from the input query that is used to access the nodes in a memory network that the query is about. Then a deductive problem-solver searches for a path to answer nodes. Robot systems (e.g. Feldman et al., 1971; Nilsson, 1969; Winston, 1972; Winograd, 1971) interface four such systems, for vision, command-parsing, deductive problem-solving, and generating actions-sequences that will effect the deduced solution.

The Fuzzy Search for Relevance

Network models seem natural and attractive. The brain is a network of neurons connected at synapses that appear to compute complex threshold functions. The mind/brain as a cognitive network that models its world is an appealing conception that has been elaborated by Peirce (1931), Craik (1952) and many others. And almost all models of intelligence seem to be network models. (This may well be a trivial consequence of the fact that any complex function - and the functions that the intelligent mind-brain must compute are nothing if not complex - is best broken down into simpler steps, and any organization of these simple steps into the complex function forms a network.)

But there is far more to using and building networks than the finding of legal solution paths between well-specified givens and goals. Rather than trying to simplify all our problems until they reduce to such a well-formed process, we should be examining the crucial - and much fuzzier - problems that arise. For example, in pattern recognition there are really no "legal transforms." Instead, any structuring of any set of feature-detectors or characterizers can be used to map the input pattern into its name (Figure 2e). The crucial problem for pattern recognition, and for perception in general, is to find from among the infinitely large set of possible transforms not the "correct" set but an adequate set, one that assigns names more or less the way we humans assign them, using our still-unknown set of transforms. In language processing the input is usually not a complete, perfectly grammatical sentence, and the sentence is always embedded in some larger perceived scene; yet we are able to extract its meaning and import, even though we cannot "parse" it. We typically must describe a scene, and converse about remembered concepts, rather than name objects and give correct answers (Figure 1).

In all these cases, givens, goals and transforms can only be inferred. Worse, there is no clear-cut search for a solution-path; rather, there is an interacting set of searches for relevant nodes, and nodes relevant to these, etc.

Even deductive problem-solving, which gives us our only well-formed paradigms, is basically fuzzy. For it is only when we are laboring in an idealized world that a theorem-goal on a game-win-state is posed. And even there we must use "heuristics" to try to direct what now becomes an essentially fuzzy search. In everyday thinking, and in the logician's and mathematician's creative work, the goal to be reached or the theorem to be proved must be judged "valuable" or "interesting" - as important and relevant enough to be worth achieving. And in general the goals of everyday thinking must be achieved by subtle and complex assessments of "relevance."

A BRIEF DESCRIPTION OF AN INTEGRATED WHOLISTIC COGNITIVE SYSTEM (SEER)

Today's robots lash together several separated systems for relatively well-formed processes. In sharp contrast, I have been trying to develop cognitive "SEER" (Semantic Learner*) systems that do a variety of cognitive processes in as integrated and wholistic a manner as possible, using as simple and general a structure as possible (Uhr, 1973b,d, 1974). This seems to me mandatory from the points of view of scientific model-building, whose canons include simplicity and elegance as well as power and fruitfulness, and of evolution-learning, where each new (and small) change must be functional and serve some purpose.

General Transforms, to Give a Unified Memory Structure

Such an integrated system needs a unified memory structure that is built up from a single general kind of transform, one that can perform the entire gamut of cognitive functions. Figure 3a surveys the kinds of transforms that have been traditionally used in separate AI systems. A parser makes the built-in assumption that nodes are concatenated (i.e., touching, in order). A deductive system has built into it the specific relation among nodes -

*or Sensed Environment Encoder, Recognizer and Responder usually co-occurrence, or ordered. Transforms for associative memory searches and feature extraction imply a whole set of unordered things (the nearby nodes in memory, the possible names to be assigned). Often they have weights or other kinds of fuzzy values associated with them. The configurational characterizer used by SEERs (Figure 3b) is general enough, since relations, weights, and other attributes can be expressed explicitly, to represent all of these. Even more important, the assignment of fuzzy values to things, features, relations and implications which is essential for fuzzy problems also allows for a natural deepening of the methods for handling problems that have been treated as though well-formed. For examples, the heuristic search for a solution path can be directed by fuzzy inferences and fuzzy contexts; parsing need not insist upon a perfectly grammatical and noise-free sentence.

Interactive Merging of Implications Across Processes

Intimate interaction among the various processes is achieved by using these general transforms and by merging the implications of transforms into a very small number of common lists. Any transform can imply implications into any of these lists and any transform can require any number of conditions. This means that any kind of contextual interaction can occur. For example, a simple feature like a curve might imply that additional curve-features be looked for, to try to build a closed loop, and also that a face be looked for, which in turn implies looking for its other features (e.g. nose, hair, ears). Similarly, an internal need, e.g. for food, might imply particular food objects which in turn imply specific features that would characterize them.

a) Specific Types of Transforms Used in Artificial Intelligence Systems

i) Parsing a sentence string

$\text{Node}_1\ \text{Node}_2 \ldots \text{Node}_N \Rightarrow \text{Replacement}$

e.g.: Article Adjective Noun ⇒ Nounphrase

ii) Deducing a move, inference or other transform

$\text{Subexpression}_1\ \text{Subexpression}_2 \ldots \text{Subexpression}_n$

$\Rightarrow \text{Transform}$

e.g.: ... P + Q ... ⇒ ... Q + P ...

iii) Searching through an associative memory net

$\text{Node}_1 \Rightarrow \text{Node}_{11}, \text{Node}_{12} \ldots \text{Node}_{IN}$

e.g.: Robin ⇒ Bird, Fly, Red-Breast, Animal, Harbinger-of-spring

iv) Extraction of Simple Features for Pattern Recognition

$\text{Feature} \Rightarrow \text{Name}_1, \text{Name}_2 \ldots \text{Name}_N$

e.g.: Small-closed-loop ⇒ P, B, R, Eye, Dumbbell, Scissors, Eyeglasses

b) A General Characterizing and Compounding Transform

$\text{Relation}, (\text{Feature}_{11}, \text{Feature}_{12}, \ldots \text{Feature}_{IN}),$
$\text{Relation}_2 (\ldots \Rightarrow (\text{Names}), (\text{Compounds}),$
(Characterizers)

e.g.: Top-right (vertical, closed-loop), Bottom-right (Vertical, closed-loop) ⇒ B, Dumbbell, Eyeglasses, Characterizers (Face)

(NOTE that weights, fuzzy values, relative locations and other attributes will also be expressed in the actual characterizing transform)

A GENERAL TRANSFORM, AND THE WAYS IT HANDLES PARSING, DEDUCING, SEARCHING, FEATURE-EXTRACTION, CHARACTERIZING AND COMPOUNDING

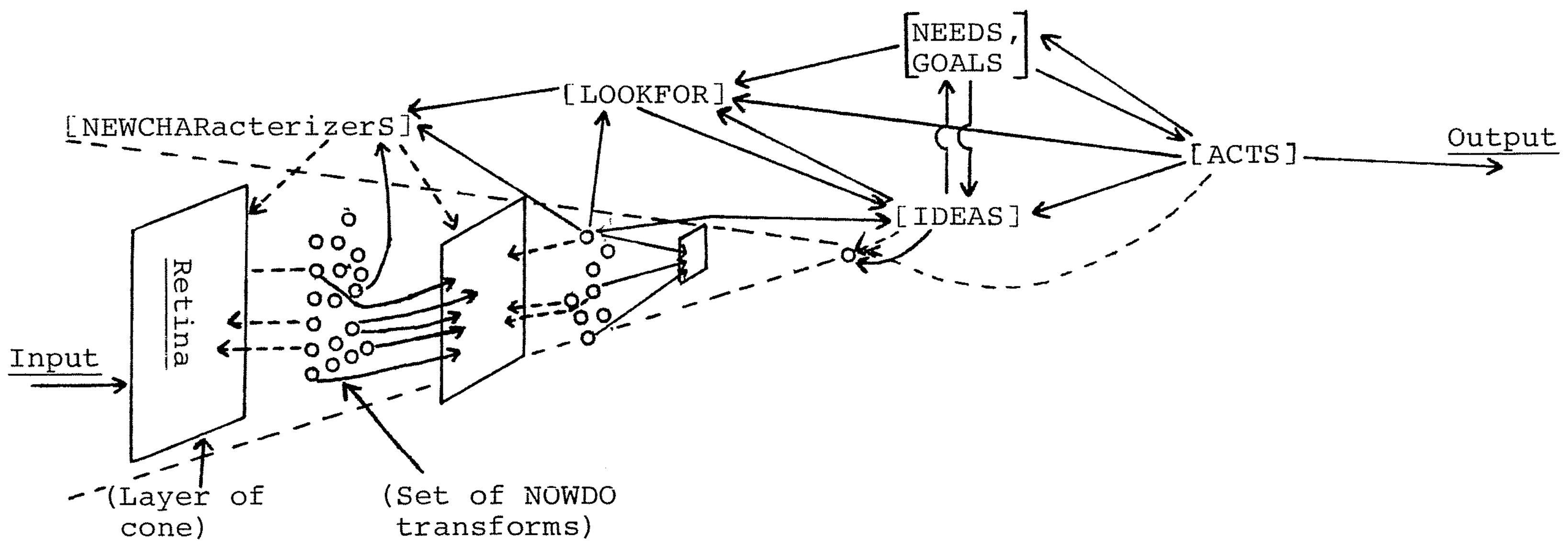
Figure 3

Overall Architecture, Processes and Behavior

SEERs are memory networks that model their "world" (including themselves) in a usable fashion. They structure transforms into several major sub-systems, for a) perceptual recognition, b) thinking, including both deduction and association, and c) generation of actions-sequences. Several successively more powerful SEERs have been coded (see Uhr, 1974.) Figure 4 sketches the structure of SEER-2, for 2-dimensional environments.

Information flows into the system from the external environment, which impinges on the retina, and starts an outer-inner flow of processes. Successive sets of characterizers (called NOWDO in the SEER program) transform and coalesce information back into the next layer of the cone. But internal NEEDs and GOALs are simultaneously impelling inner-directed flows of processes. Similarly, partially recognized things can imply additional things to LOOKFOR and NEW CHARacterizerS to apply, in glancing about to gather information. Perception is thus a complex back-and-forth many-layered parallel-serial process. "Thinking" consists in the serial application of transforms from the IDEAS list, to make associative memory searches and simple deduction. A certain amount of direction is got by having the system choose the single most highly implied transform on the IDEAS list to apply next. Thus the strengths of implications, that is, of connections between one node and another (where a node can be a transforming procedure, as well as a representation of an object, class, attribute or compound) serve to heuristically direct the processes of thinking. When an act is chosen (because it is the most highly implied

(External partially recognized things, and internal NEEDS, IDEAS and chosen ACTS all imply new things to LOOKFOR, which imply new IDEAS and NEWCHARacterizerS to apply)



OVERALL ARCHITECTURE OF SEER-2, A FIRST ATTEMPT AT AN INTEGRATED, WHOLISTIC COGNITIVE SYSTEM

Figure 4

thing on IDEAS) the system will start generating the specific actions-sequence needed to carry out that act. This will usually entail further calls for needed objects to LOOKFOR and transforming IDEAS to remember and deduce what might be usable objects, which are then looked for.

SEER thus carries out a rather complex set of parallel and serial operations. We might make the loose metaphorical comment that it widens and narrows its "conscious" "attention" as a function of its problems, tending to be more parallel in its perceptual processes, more serial in its central cognitive processes. This means that (serial) time must pass, and a new program (SEER-T) handles situations in which it interacts over time with a changing environment (see Uhr, 1973c).

For fuller descriptions, see Uhr, 1974, and in preparation.

Overview of the SEER-2 Program, and its Flow of Processes

The following is a succinct description of SEER-2. The actual program is given in the Appendix, along with a brief description of the programming language, EASEy (Uhr, 1973f), an English-like variant of SNOBOL (Griswold et al., 1968) in which it is coded. Note that caps, underlines and numbers refer to the program.

(Overview of the SEER-2 program.

		Program Statement No.
START	Initialize Memory (Includes type-name to refer to lists and the Transforms repeatedly used - ERASER, NORMALIZE, PASSON.	
IN	input Memory, including NEW lists and ADDitions (The MEMory conversion program, which inputs the transforms needed (Appendix B) goes here.) (The environments - scenes and problems - to be sensed are also input here.)	1
INITialize	for the next input problem, by erasing all temporary lists.	
SENSE	Store the input as an array, in the first Layer of the recognition cone.	2-7
TRANSform	MERGE the things that POINTAT NEEDSGOALS and LOOKFOR into NEWCHARActerizers	8,9,12
TREPEAT	Initialize the LAYERS and CHARacterizerS TODO	10
T6	Get the next layer to NOWDO and its STEP-size	11
	Put the ERASER, NORMALIZER and PASSON transforms, and NEWCHARS, onto NOWDO	14
T5	Get the next TRANSform and attendant information from NOWDO. Get the bounds for applying the TRANSform, and its TYPE	15 16-18
T3	Get the pieces of its DESCRiption, and handle them as indicated by TYPE	20-24
A1	MERGE all members of the specified CLASS into the next Layer, to average or difference.	21
T1	Accumulate the TOTAL weights of the CLASS members that are GOT	22-24
T2	If TOTAL exceeds THRESHold, MERGE IMPLIEDS into the next Layer	25-26
A2	Iterate through the array until the bounds (CMAX, RMAX) have been reached.	27-28
E1	erase each cell in the next Layer, to initialize it.	29
N1	NORMALIZE each cell, to keep weights roughly constant despite convergence.	30
ITER	Go to process the next Layer, shrinking Row and Column by STEP-size, until the apex is reached	31-33 34
THINK	Initialize and go through up to 7 CYCLES	35-37
	Get NEWCHARS that POINTAT LOOKFOR and set them on IDEAS	38-40
	CHOOSE the most highly weighted THING on IDEAS, up to 100	41-45
(Cycles through IDEAS until an ACT is the most highly weighted and therefore chosen)		
I	If TYPE is I, this is an Internal transform to apply to FOUND (the apex). If the TOTAL weight of CLASS members GOT exceeds THRESHold, MERGE IMPLIEDS.	46-54
ACT	If TYPE is ACT, get the next ACTION and its ARGumentS and other ARGuments from HISTory	55-58
OUT	output the result of the act (after having completed it with routines below)	59-60

SEARCH	If the act couldn't be completed, set up a SEARCH for things that POINTAT the NEEDED PARTS, and return to apply more IDEAS.	61-66
FAIL	output that have "FAILED" to execute the chosen act, and go to the next input.	67

(The routines for the different types of acts follow.)
(Note how Describe uses name, and Move uses Find.)

D	Describe the scene, giving the objects and their parts	68-76
T	Name the single most highly weighted thing of the class specified in ARGument.	69-71
F	Find the first THING that is a member of the specified ARGument and bracket it.	77-79
M	Move all Found THINGs FROM or TO (as indicated) the TARGET thing	77-85
R	Reply to a query by CHOOSEing all THINGs belonging to the specified ARGument whose weight exceeds half the MAXWeighTed THING (i.e. the first) chosen	86-90
SEARCHR	If no Reply was chosen, MERGE associations from what already found into IDEAS and return to think some more, by applying more IDEAS.	91
C	Compute, using the specified operators (ADD, subtract and divide are shown)	92-99
SEARCHC	If the needed numbers were not recognized, return to search for INTEGERS	96
G	In a Game, make a move by replacing the old board configuration with the move	100-103

(The following are the functions used by the main program.)

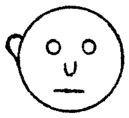
MERGE	Get the THINGs on LISTA that belong to the specified CLASS and MERGE them, combining TOTAL WeighTs and HISTories, into LISTB.	ME1-
POINTAT	Get the TOTHINGS that POINTAT each THING specified, and MERGE them into the lists specified.	PO1-
NORMALIZE	Divide the WeighT of each thing TONORMalize by NORM, erasing it if WT is below 1.	NO1-
ABS	Get the ABSsolute value of the argument (fails if the argument is not an integer).	ABS1
CHOOSE	CHOOSEs the MAXimum (or if TYPE specifies MIN the MINimum weighted thing in LISTA of the specified CLASS, getting the THING, TOTAL, LOC and HIST.	CH1-

a) To be named:

NAME

(1)

WHAT IS THIS

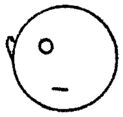


(2)

NAME
THIS

(3)

GIVE THIS
A NAME

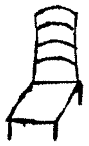


(4)

b) to be described:

DESCRIBE
THIS

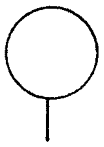
(1)

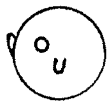


DESCRIBE

(2)

DESCRIBE





(3)

SIMPLE SCENES OF MIXED OBJECTS AND WORDS

Figure 5

Some Simple Examples of the Variety of Behavior of SEER-2

SEER must be given a set of transforms, which form the system's "cognitive memory network model" of its world. Future systems will learn these transforms from experience. But for now we can start the system with a set of transforms already in its memory, and immediately begin to examine its behavior. The following examples show the kinds of problems that SEER can handle:

Naming Objects and Describing Scenes. When input an array of information that contains one or more objects, and also verbal statements, like "NAME THIS" or "WHAT IS THIS" or "DESCRIBE", SEER will successively apply feature-extracting transforms, and then configurational characterizers that were implied by these prior transforms, and also by tentatively implied things that would be further confirmed or denied by these transforms. This process continues until high-level compounds, like words, phrases, objects, and collections of objects are got. A recognized command or suggestion will imply whether the system should name or describe (or do something else), and the system will choose a particular type of act as a function of such perceived utterances, and also any internal needs and presses.

Thus when given the inputs shown in Figure 5a, SEER-2 will output:

(1)	"CHAIR"	(3)	"TABLE"
(2)	"FACE"	(4)	"FACE"

When given the inputs shown in Figure 5b, SEER-2 will output:

(1) "CHAIR(WITH BACK;SEAT;LEGS;)

(2) "FACE(WITH VISAGE;EYE;EYE;NOSE;MOUTH;LEFTEAR;)"

(3) "BALLOON(WITH CIRCLE;STICK;) FACE(WITH EYE;NOSE; RIGHTEAR;)"

See Uhr, 1973e for a similar system that gives a wider variety of descriptive information, where a stylized interaction allows the human recipient to direct the description.

For simplicity the rest of the examples of SEER's behavior will be shown for 1-dimensional inputs, which can be handled by either SEER-1 or SEER-2 (see Uhr, 1974, for details). The first stages of the perceptual process handled by SEER-2's recognition cone become trivial, and a far simpler memory network is needed. For these examples we will use the convention @(thing name) to indicate objects, object parts and qualities (e.g. @PEAR is a symbol for an object whose name is "pear"), as a shorthand for the actual picture.

When input:
"NAME THIS @PEAR" or
"WHAT IS @PEAR THIS"
SEER will respond:
"PEAR"

When input a set of lower-level qualities, e.g.:
"SAY WHAT @STEM @YELLOW @TEARDROP YOU SEE" and
"@OVAL @STEM @RED WHAT DO YOU SEE"
SEER will respond:
"PEAR" and
"APPLE in turn.

<u>Retrieving Information, Answering Questions, and Conversing</u>.
If the input scene contains a question like:
"NAME THE FIRST PRESIDENT" or
"WHAT DOES WISCONSIN PRODUCE"
SEER will output:
"WASHINGTON" and
"BEER;CHEESE;GARBAGE;"

Verbal inputs that are not simple questions, but rather act in a more conversational way, will lead to more variable, but more or less relevant, responses.

Deducing Responses. Transforms can similarly lead to simple deductions, e.g. as to arithmetic, logic, or the moves (not necessarily brilliant, or even good) in a game. For example,

"ADD 3 + 2" will give:

"5"

This result is achieved by first recognizing the parts (letters, words, phrases), which triggers the following of the command "ADD," which points to a transform that combines the numbers to be added according to the rules of addition.

Finding and Manipulating Objects, Driven by External and/or Internal Presses. When it recognizes some part of the external scene as a command or suggestion (e.g. "TOUCH" or "FIND" or "WHY DON'T YOU TOUCH"), or some internal need as a press (e.g. a high level of HUNGER will imply EAT any perceived food object), it may choose to act (e.g. touch the indicated object). Note that "touching" is simulated inside the computer, by the placing of agreed-upon symbols around the touched objects. The scene's mixture of words and objects remains, simulating a static visual scene with written words.

Thus a command like:

"TOUCH THE @BOX PAIL @PEAR @PAIL"

(where @object (e.g. @BOX) indicates the object, not the word) will lead to the result:

"A PAIL IS FOUND-" "TOUCH THE @BOX PAIL @PEAR: @PAIL:"

(colons indicate the act that found and touched).

A command like:
"MOVE THE @PEAR @BOX FRUIT @APPLE TO THE @PAIL PAIL"
will lead to the result:
"A PEAR IS FOUND-"
"MOVE THE @BOX FRUIT @APPLE TO THE (@PAIL @PEAR) PAIL"
(where parentheses indicate that the objects within them have been moved together).

An act can also be implied from internal needs as well as external objects. E.g.
@BOX @PAIL @APPLE @SEER-MOUTH" (along with an internal need-state of hunger)
will lead to the response:
"A APPLE IS FOUND-"
"@BOX @PAIL (@SEER-MOUTH @APPLE)"
because the act of putting the @APPLE to the MOUTH is now highly implied, by both the hunger need and the apple itself.

When conflicting commands and presses are perceived, SEER will make a fuzzy choice, not necessarily a "correct" choice, among them.

Choosing Among Different Types of Behavior

These examples illustrate the range of problems that SEER can handle. By they make little use of fuzzy values and give very little feeling for the complex set of fuzzy choices it must make when the possible set of alternatives grows larger and more contextual determinants become relevant. With simple problems and appropriate transforms there is little chance for ambiguity. But when there are many distortions of possible objects in a scene and much information in memory decisions become multi-determined. This is handled by the use of fuzzy implications which are merged together into common lists where choices as to what to do next are constantly being made. Thus a variety of partial implications, as from different characteristics of a scene, and also from different sources of information such as externally sensed scenes and internally felt needs all merged together, are chosen among and determine when to choose.

It is not clear how well such a system will work as the size and complexity of its problems increase. The only way to find out is to test it on much larger sets of more difficult problems. But we need not expect it to be perfect, or even to be exceptionally good - just as people do, it can exhibit its share of mistaken, stupid and rigid behavior.

HIGHER-LEVEL FUZZY PROBLEMS

The combining of functions into a single cognitive system forces us to eliminate any rigid flow of processes. The system must now choose among alternate possibilities, and these choices confront it with inevitably fuzzy situations, including the following (see Uhr, in preparation).

Relevance Must Constantly be Assessed, in an On-Going Conversational Description

Nothing is built into the system that compels it to do something like "assign a single name"; rather, the system must develop a relevant description of its environment, in the sense that it must notice those things that will help it do what it decides to do. But "description" is an extremely fuzzy concept. We can think of a "complete description," which is far too long to be of use, or a conventionalized description, which would make such a system impossibly rigid (see Uhr, 1973c,e for examinations of "description"). Instead, the system must develop a pertinent description, as a function of external and internal presses, and of acts it has decided to try to effect, and of deductions and memories that suggest what would be helpful and relevant parts of descriptions, toward effecting those acts.

Nothing can be built-in that says "find a path from this to that," or "choose the single most highly implied name." Rather, the system must constantly assess the relevance of nodes, transforms, and flows of procedures. Nor will it be clear how to assess relevance, which will be a fuzzy function of a variety of things. It seems best to think of the system as engaging in an on-going "conversation with its external world" (within which may be objects and other systems that converse verbally). There will rarely be a "correct response," and the system will often behave in a mediocre way. But it will always be trying to act relevantly and, especially with the aid of friendly elements in its environment (e.g. parents, teachers, ripe berries and other manna from heaven), will often muddle through, and even act reasonably well.

Relevant Imports Must be Got from Mixed Words and Things

Words, phrases, statements, suggestions, questions, commands, and any kind of verbal utterance must be input through the same perceptual channels that sense, recognize and understand objects and their relations and qualities. There may indeed be several input channels, such as the two eyes, and also sensors for sound, touch, smell, or other type of impinging energy. But none can be exclusively reserved for verbal inputs, in the way that the robots' teletype inputs are known by the robot to be verbal commands, as opposed to its television inputs, which are sensed scenes of objects. Rather, the system must recognize the relevant things in its input scenes, and further recognize which combine into words and symbolic referential utterances, and what are the things to which they refer (Uhr, 1973b,d).

The System Must Choose the Type of Act to Effect

At the lowest level, any AI system must make fuzzy choices, using "heuristics" or "characterizers," in order to search through an overly-large network of possibilities. Unless the problem has been cut down to uninteresting toy size it cannot try everything.

But once we begin to ask a single system to handle a variety of different types of thing we force it to make higher-level fuzzy decisions. It must decide what is the appropriate <u>type</u> of act. E.G. it must decide whether to name, describe, draw, or touch an object; whether to treat an input recognized as a verbal utterance as a command to be followed or a statement to be responded to. This may entail several levels of fuzzy decisions. E.g. after it decides to name it must further decide whether to speak, write or print the name, and which language, e.g. French, English or American and, sometimes, which of several possible synonymous names, to use.

Certain Inputs Must be Recognized as Feedback

Just as there can be no special built-in channel for verbal utterances, there can be no special channel or signal for feedback. Rather, the system must recognize that something perceived (e.g. candy, a smile) or felt internally (e.g. pain) is feedback that refers to some previously perceived stimuli and consequent actions by the system. This entails a complex combination of hypotheses the system makes as to expected feedback consequences, which focus its attention for confirming or denying evidence, and also an ability to relate any identified feedback, whether anticipated or unanticipated, to percepts, thoughts and acts that the system must once again fuzzily infer are relevant.

The Learning of Things, Transforms and Hypotheses is Essentially Fuzzy

Learning is fuzzy in several ways.

First, the system must generalize from one, or at most a relatively small number of experiences. Such generalizations are guaranteed to be wrong a good deal if not most of the time, since the world is wondrously complex, diverse, and accidental. So the system must make an on-going experimental assessment of each tentatively-learned thing.

Second, it must generate new hypotheses in the first place, whether from experience, or by combining, refining, or in some other ways restructuring previously entertained hypotheses. Once again, there is no assurance that any such hypothesis will prove to be correct. Rather, the system must accrue evidence, through future experience, for each one of them, sifting and choosing among them as this evidence confirms and denies.

Third, a complex structure of hypotheses must be built. Once again, we have essentially fuzzy decisions as to which nodes to connect to which - e.g. what things to put into what classes, or compound into higher-level structures.

THE STEP-BY-STEP DEVELOPMENT OF MODELS OF INTELLIGENT MIND/BRAINS

A number of very difficult steps must be taken before we can hope to achieve intelligent systems. First, we must model the separate intellectual functions - perception, thinking, remembering, acting, language, and learning. But because these are so interdependent, we cannot expect to model them separately; rather, we must combine them into whole integrated systems. This paper describes a first attempt at such a system, one that begins to do a variety of cognitive tasks, where the different subsystems must interact smoothly and coherently. Such a system is just a beginning, and must be extended in a variety of ways.

It must be made to handle more things. To some extent this can be done by giving it more transforms, that is by giving it a bigger memory network. At that point extensive tests must be made, since its behavior will be far too complex to predict. This further means that we must develop some conception about the range of behavior that such a system must model, so that we can say something meaningful as to how well we are sampling, and examining. We still need to develop the basic canons for experimental test and evaluation in this new science of complex entities.

Even more important, the system must be made more powerful, with more powerful transforms and overall structure.

At the same time, it must be simplified. We must strive for the simplest possible system, at the same time that we strive for the most powerful possible system. Simplicity is not only desirable from the point of view of efficiency and of the canons for building good models, but also because the simpler the system the more likely it may resemble living systems that have evolved under nature's canons of simplicity.

In addition to generality, integration, power and simplicity, we must also worry about the fit of the model's behavior. At first we can be quite satisfied with rather general fits, for we are modelling such a wide domain of behavior - describing the whole elephant, rather than a few hairs in the left fore-legpit. But at some point - I think only after we have developed models that exhibit a good deal of generality and power - we must begin to worry about fitting details. This completes the hypothetico-deductive enterprise of predicting human behavior, comparing with experimental results and thus testing our model, and, hopefully, finding new disconfirming evidence that leads to changes that improve the model.

Finally, learning must be added. We can never hope to pre-program into such a system all the knowledge that it might need about the external world with which it must interact, that it must know about. In fact this is impossible in principle, because that external world is itself open-ended and constantly changing: New things, organisms and mutants are born, new words, concepts and other man-made things (e.g. bicycles, transistors, computers, poems) are created.

SUMMARY AND CONCLUSIONS

It is attractive to model the mind/brain as a network of neuron-like threshold elements that itself serves as a "cognitive model" of its world, including itself. Such a model must have a general kind of transform capability, one that can handle the specific types of transforms used in the typically separated Artificial Intelligence systems for perception, deductive problem-solving, language handling, remembering, acting and learning. And the different cognitive functions must be able to interact intimately, forming a well-integrated wholistic system with rich contextual influences on a constant stream of interacting decisions.

This paper examines a first step toward such a system, a programmed model called SEER that attempts to handle the various cognitive functions in as simple and integrated a way as possible.

A network model is general in that it is simply a framework for computing structures of functions of any sort. But today's AI research has over-narrowed the use of networks to the point where they make a simple well-formed search for a path, using legal transforms (e.g. rules of inference, moves in a game) from a set of givens (e.g. premises, the initial board of the game) to a goal (e.g. a theorem, a win). It may be possible to handle a few cognitive tasks in such a well-formed way. But even that seems unlikely, for when the problem of path-searching becomes difficult (as it does the minute the problem becomes difficult enough to be interesting) fuzzy and conjectural "heuristics" must be used as hunches to guide the search.

Most cognitive problems are ill-formed and fuzzy. The mathematician doesn't prove theorems posed to him in an MIT exam; rather he must find theorems worthy of proof. A perceiver does not name single isolated easily discriminable objects (and even that is a fuzzy problem); rather he makes note of the relevant aspects of a scene of interacting objects. We cannot expect to be given only simple verbal questions, for which there is a "correct" answer; rather, we must usually engage in a conversation, to which we respond with hopefully relevant comments.

In just about every separated area of cognition the real problems are ill-formed and fuzzy, and it is over-simplifying to reduce them to the point where they can be handled by a search for a solution path.

Even more important, the moment we ask a system to handle a variety of cognitive functions at the same time - which is the typical process for human adults, and even for infants and higher animals, e.g. when we find, stalk, capture and prepare food - we force it to make a constant interacting stream of fuzzy decisions. Externally perceived objects and internally perceived needs fuzzily suggest acts that might be effected and memories and deductions that might help carry out these acts. Conversely, remembered objects, qualities, and procedures suggest objects that should be looked for and actions (e.g. glancing about, crawling around) that might help in gathering more information and testing the potential value of possible acts. In general, each cognitive function calls upon and affects all the others. There is no "correct" sequence of procedures. Rather, multi-determined fuzzy decisions are constantly being made to guide a fuzzy interacting set of sets of fuzzy processes.

Finally, several higher-level fuzzy problems emerge. Relevance must constantly be assessed. All perceived things are mixed together in the sensory input channels, and must be sorted out. Thus verbal utterances must be recognized as structures over perceived things, and then as having symbolic referential import. Feedback must also be recognized as feedback, and as probably relevant to fuzzily conjectured previous stimuli, acts, and hypotheses. And learning must make fuzzy decisions as to what to learn, and how to build, unbuild, or restructure the cognitive network memory.

Appendix A: The SEER-2 Program

The following program (coded in EASEy, see Appendix C and Uhr, 1973f, and therefore able to run on any computer that has a translator for SNOBOL4, see Griswold et al.,1968) handles the examples given in the paper, among many others. It also handles far more complex problems, e.g. perception of scenes of distorted objects, since it merges together multiple fuzzy implications, and chooses among them.

AI programs are too complex to describe fully, accurately and fairly. The typical paper usually doesn't give many details, but rather extolls a program's virtures. But we cannot really begin to understand one another's programs, so that we can begin to borrow from and built upon one another, until we can observe them clearly. Think what monographs on mathematics would be without proofs, or books on architecture without pictures and diagrams.

The EASEy programming language was developed as a first attempt to bridge this communication gap. A program coded in EASEy is still cumbersome and hard to read and understand. It is like a complicated and messy proof in a peculiar and fussy notation. But it is precise and complete. For the reader who want to dig in and see exactly what is happening, it is the thing itself. This is still difficult - but because of the intrinsic complexity of the model, not the peculiarities of the programming language.

Following the program is a set of characterizing transforms sufficient to handle all the examples given in both this paper and Uhr (1974) along with many others. The overview and description given in the paper should help when digging into the program.

```
(Program SEER-2)                                                    RECODES-1       SEER-2
                                                                    (see Uhr,1973e)

(Initialize
START    set  ERASER = 'E 0 0 R C E '
              NORMALIZE = 'N 0 0 R C N '
              PASSON = 'P 0 0 R C P1 '
              P1 = 'A%D=  0 0 ]%I=%'
         set  SPOTSIZE = 1
              F = 'FOUND'
              N = 'NEWCHARS'
              L = 'LOOKFOR'
              I = 'IDEAS'
(INput and go to TYPE (initialize memory, SENSE or TRANSform)
IN       input TYPE DESCR % [+to $('M' TYPE -end) ]                  1          1
(MEMory  input and format routine goes here-only the beginning is shown.  .1    M
(input      NEW lists or ADDed information on old lists
MNEW     from DESCR get NAME =
         set  $NAME = CONTENTS [to IN]                               .2         M
MADD     from DESCR get NAME =                                       .3         M
         on $NAME set CONTENTS [IN]                                  .4         M
MINIT    erase R, L, EXTERNAL, TOOUT, NEWCHARS, LOOKFOR, IDEAS, ACTIVEWT   M(N+2)   M
(Input and SENSE a new scene into Layer 1 (the retina)
MSENSE   erase C                                                     2          2
         on EXTERNAL list DESCR                                      .1         3
S1       from DESCR get and call SPOTSIZE symbols SPOT erase [-to S2]  3.V      4
         list  $(L;R;C) = 'BRIGHT QUAL :BRIGHT ' SPOT ]              4.V        5
         C = C + 1 [to S1]                                           5          6
S2       R = R + 1 [to IN]                                           6          7
MTRANS   output DESCR ' SCENE HAS BEEN INPUT, IS BEING TRANSFORMED'  7          8
(NEEDS and GOALS imply things to LOOKFOR that POINTAT them
         MERGE(POINTAT(NEEDSGOALS),L)                                .1         9
TREPEAT  TODO = LAYERS CHARS                                         8          10
T6       from TODO get STEP NOWDO % = [-TREPEAT ]                    9          11
(LOOKFOR implies NEWCHARacterizers that point at them.
         MERGE(POINTAT(LOOKFOR),N)                                   .1         12
         erase LOOKFOR                                               .2         13
(To ERASE next LAYER,NORMALIZE, apply NEWly implied CHARacterizers, and PASSON
(found things
   at start of NOWDO set ':ERASER X ]:NORMALIZE X ]' NEWCHARS ':PASSON X ]'   10.V   14
(Get each TRANSform from NOWDO (to be applied at this layer)
T5       from NOWDO get CLASSES : TRANS WT HIST ] = [-ITER]          11.V       15
         from $TRANS get RA CAA RMAX CMAX DO                         12         16
T7       CA = CAA                                                    13         17
T4       from $DO get TYPE THRESH '%D=' DESCR '%I=' IMPLIEDS %       14.V       18
         erase GOT TOTAL                                             15         19
T3       from DESCR get : CLASS TVAL DR DC ] = [ +$(TYPE 1) - $TYPE 2) ]   16.V     20
(TYPE A transform Averages or differences.
A1       MERGE($(L;RA+DR;CA+DC),'$(L+1;RA/STEP; CA/STEP)',,TVAL,CLASS) [T3]   17.V   21

(TYPE I transform characterizes
I1       from $(L;RA+DR;CA+DC) get # that CLASS # REST : THING VAL [-T3]   18.V    22
         is TVAL lessthan VAL ? yes. TOTAL = TOTAL + 10 [-T3]        19.V       23
         on GOT list CLASS THING  [T3]                               20.V       24
(the characterizer succeeds if the TOTAL weight reaches THRESHold.
I2       is THRESH greaterthan TOTAL ?  [+ A2 ]                      21.V       25
         MERGE(IMPLIEDS, '$(L+1;RA/STEP;CA/STEP)', GOT)              22         26
```

```
(keeps applying this TRANSform till its upper bounds are reached.         RECODES-1  SEER-2
A2       is CA lessthan $CMAX? yes - CA = CA + 1  [+T4 ]                   23         27
         is RA lessthan $RMAX? yes - RA = RA + 1  [+T7-T5]                 24         28
(Erases the next layer, to re-initialize.
E1       erase $(L+1; RA/STEP ; CA/STEP) [A2]                              25         29
(Normalizes, dividing by 5 - roughly assuming a STEP-size of 2 plus a bit more.
N1       $(L;RA;CA) = NORMALIZE($(L;RA;CA),5) [ A2 ]                       .1         30
(Converge to the next layer.
ITER     L = L + 1                                                         26         31
         R = R / STEP                                                      27         32
         C = C / STEP                                                      28         33
(If the apex has been reached, start to "THINK" (apply IDEAS)
         is R lessthan 1 ? is C lessthan 1 ?[ -T6]                         29         34

(CYCLES 7 times applying 100 transform from IDEAS to FOUND (the apex)
THINK    CYCLES = 7                                                                   35
TCYCLE   CYCLES = CYCLES - 1                                                          36
         is CYCLES lessthan 1 ? [ + FAIL ]                                            37
         MERGE(POINTAT(LOOKFOR),N)                                                    38
            on IDEAS set NEWCHARS                                                     39
         erase LOOKFOR, NEWCHARS                                                      40
         set TRIES = 100                                                              41
TMORE    TRIES = TRIES - 1                                                            42
         is TRIES lessthan 1 ? [+ TCYCLE ]                                            43
(CHOOSES most highly weighted TRANSform from IDEAS
         from IDEAS get CHOOSE(IDEAS) = [ -TCYCLE]                         9.V        44
(Applies the TRANSform to the FOUND in the apex
         from $THING get TYPE THRESH '%D=' DESCR '%I=' IMPLIEDS % [$TYPE]  11.V       45
I        set FOUND = $(L;0;0)                                              10.V       46
(Loops if.All indicated (should use LOCs and get nearest.)
TH1      from DESCR get : CLASS WT ]= [-EVAL]                              12.V       47
         from CLASS get '$' ALL = [ +THA ]                                            48
           ALL = 1                                                                    49
THA      from FOUND get LEFT # that CLASS # RIGHT : THING WTF = [-TH1]     15.V       50

         on GOT list CLASS THING                                           16.V       51
         TOTAL = TOTAL + WT * WTF [ $('TH' ALL ) ]                         17.V       52
EVAL     is THRESH greaterthan TOTAL ? [-TMORE ]                           18.V       53
         MERGE(IMPLIEDS,F,GOT) [TMORE]                                                54
(ACTs, because an act was chosen from IDEAS.
ACT      from IMPLIEDS get ACTION ARGS ; = [- OUT]                                    55
ACT2     from HIST get CLASS ARG = [+ $ACTION ]                                       56
           ARG = ARGS  [$ACTION]                                                      57
(OUTputs its actions-sequence
OUT      is TOOUT sameas EMPTY ? [+ SEARCH ]                                          58
         output        TOOUT                                                          59
OUT2     output        EXTERNAL [IN]                                       44.V       60
(Initiates a SEARCH to help in completing the frustrated act.
SEARCH   from $ARG get '%I=' IMPLIEDS % [- FAIL ]                                     61
SEARCH2  from IMPLIEDS get IMPLIED WT ; = [- FAIL]                                    62
         from IMPLIED get NEEDED '$' [-SEARCH2 ]                                      63
         from $NEEDED get '%D=' PARTS %                                               64
         MERGE(POINTAT(PARTS),N)                                                      65
```

```
                                                                          RECODES-1   SEER-2
RETURNACT  on NEWCHARS set CHOOSE [TMORE]                                              66
FAIL       output 'FAILED' EXTERNAL  [IN]                                              67
D          [T]                                                                         68
(Names the most highly implied object of class specified in ARGument
T          from $(L;0;0) get CHOOSE($(L;0;0),ARG) = [-ACT ]                 36-43      69
(TOTAL weight must be above 5.
           is TOTAL greaterthan 5 ? [-ACT]                                             70
           on TOOUT list THING       [$('AC' ACTION) ]                                 71
ACD        on TOOUT set '( WITH '                                                      72
(Describes the scene
D3         from HISTC get CLASS THINGH =  [-D2 ]                                       73
           from  $(L;0;0) get : that THINGH REST = [-D3]                    46.V       74
           on TOOUT list THINGH ; [D3]                                      49.V       75
D2         on TOOUT set ' )  '  [T ]                                                   76
(Finds the first THING for each ARGument.
F          from  $(L;0;0) get # that ARG # REST : THING MORE] = [-ACT]      52.V       77
(Finds THING only if without variations in EXTERNAL input
           from EXTERNAL get that THING = : THING : [-ACT]                  53.V       78
           on TOOUT list  'A ' THING ' IS FOUND- '  [ACT]                   54.V       79
(Moves all Found things as PREPosition (TO or FROM) indicates
M          is CLASS sameas  'PREP' ? [ -F ]                                            80
           from HIST get CLASS ARGT     [-ACT ]                                        81
           from  $(L;0;0) get # that ARGT # REST : TARGET  [-SEARCH ]       56.V       82
M2         from EXTERNAL get : THINGA : = [+ $('M' ARG) ]                   57.V       83
MTO        from EXTERNAL get that TARGET = '(' TARGET THINGA ')' [M2 ]                 84
MFROM      from EXTERNAL get LEFT that TARGET RIGHT =                       58.V       85
+             : THINGA : LEFT RIGHT TARGET  [M2 ]
(Replies. If nothing about ARG, SEARCH associates out some more.
(needs more directed and conscious search
R        - from $(L;0;0) get CHOOSE($(L;0;0), ARG) = [-SEARCHR]             60.V       86
              MAXWT = TOTAL                                                            87
R2         on TOOUT list THING ;                                                       88
           from $(L;0;0) get CHOOSE($(L;0;0),ARG) = [ -ACT ]                           89
           is TOTAL lessthan MAXWT / 2 ? [+ACT - R2 ]                                  90
SEARCHR    MERGE($(L;0;0),IDEAS) [ RETURNACT ]                                         91
(Computes
C          from ARGS get OP CLASSA CLASSB                                              92
           from $(L;0;0) get # that OP # REST : OP                                     93
           from $(L;0;0) get # that CLASSA  # REST : ARGA [ - SEARCHC ]                94
           from $(L;0;0) get # that CLASSB # REST : ARGB [ + $OP ]                     95
SEARCHC    MERGE(POINTAT(INTEGERS) [RETURNACT ]                                        96
(ADD,- etc. are OPS.
ADD        on TOOUT list ARGA + ARGB  [ACT]                                            97
-          on TOOUT list ARGA - ARGB  [ACT]                                            98
/          on TOOUT list ARGA / ARGB  [ACT]                                            99

(Game move (Needs to look deeper, choose with parallel heuristics)
G          from ARGS get OLD NEW                                                      100
           from $(L;0;0) get # that OLD # REST : OLD [-G]                             101
           from $(L;0;0) get # that NEW # REST : NEW [-G]                             103
           from EXTERNAL get that OLD = NEW [OUT2]                                    104
```

```
                                                                        RECODES-1 SEER-2
(Functions used by the main program
(MERGE two lists, combining weights and HISTories
MERGE      DEFINE:  MERGE(LISTA,LISTB,HISTA,WT,CLASS)                   34.V     ME1
           is CLASS sameas EMPTY ? [-ME1]                                          2
           from LISTA get LEFT : THING WTA HIST ] = [ - return + ME3]              3
ME1        from LISTA get LEFT # that CLASS # REST : THING WTA HIST] =[ -return] 35.V   ME4
ME3        from WTA get '$' = TOTAL                                     36.V     ME5
(Can optionally specify LISTB as part of THING
           from THING get '$' INTOLIST [+ ME4]                            .1     ME6
           INTOLIST = LISTB                                             37.V     ME7
ME4        from INTOLIST get : that THING TOTALB HISTB =                38.V     ME8
+            NAME TOTALB+ (WT+1) * WTL HISTA HISTB ]   [+ME1 ]
           from $THING get '%C=' CLASSES %  [+ME2 ]                       .1     ME9
             erase CLASSES                                                       ME10
ME2        on $LISTB list THING CLASSES : THING (WT+1) * WTL LISTB HISTA ] [ME1] 39.V   ME11

(Back-links only to transforms with names
POINTAT    DEFINE: POINTAT(THINGS)                                               PA1
PA1        from THINGS get CLASSES : THING HIST]  = [- return ]                  PA2
           from $THING get '%D=' TOTHINGS % [- PA1]                              PA3
           MERGE(TOTHINGS;N)       [ PA1 ]                                       PA4
(NORMALIZE to keep weights roughly constant even though converging passed-on things

NORMALIZE  DEFINE: NORMALIZE(TONORM,NORM)                                        NO1
NORM1      from TONORM get LEFT : THING WT RIGHT ] = [- return]                  NO2
           WT = WT / NORM                                                        NO3
           is WT lessthan 1 ? [+ NORM1]                                          NO4
           on NORMALIZE list LEFT : THING WT RIGHT ] [ NORM1 ]                   NO5
(Get ABSolute value
ABS      - DEFINE: ABS(ABS)                                                      AB1
ABS1       at start of ABS get '-' = [return]                                    AB2
(CHOOSE MAX or MIN weighted (MAX if no type is specified)
CHOOSE     DEFINE: CHOOSE(LISTA,CLASS,TYPE)                             41       CH1
(CLASS can be a specific thing, or empty (in which case all things are chosen among

CH1        from LISTA get CLASSES : FIRST THWT HIHIST ] = [ - freturn ]  42.V     CH2
           is CLASS sameas EMPTY ? [+ CH2]                              43       CH3
           from CLASSES get # that CLASS # [-CH1 ]                      44.V     CH4
           list CHOOSE = CLASSES: FIRST THWT L ; RA ; CA HIHIST         50.V     CH5
CH2        from LISTA get ORCLASSES : ORTHING ORWT ORHIST ] = [+CH4]    45.V     CH6
           from CHOOSE get CLASSES : THING TOTAL LOC HIST ] [return ]   45.V     CH7
CH4        is CLASS sameas EMPTY ? [+ $('CH'TYPE) ]                     46       CH8
           from ORCLASSES get # that CLASS #  [+$('CH' TYPE)  -CH2 ]    47.V     CH9
CHMIN      is ORWT lessthan THWT ? yes - THWT = ORWT [+CH3 - CH2]       48.V     CH10
CH         [CHMAX ]                                                      .1      CH11
CHMAX      is THWT lessthan ORWT ? yes- THWT = ORWT  [+CH3 - CH2]        .2      CH12
CH2        list CHOOSE = ORCLASSES : ORTHING ORWT LISTA HIHIST ] [CH2]  49.V     CH13
end
```

Appendix B: Characterizing Transforms that Form SEER's Memory Network

Memory networks input in the following form are automatically input and converted (see Uhr, 1974 for further description of details of transforms).

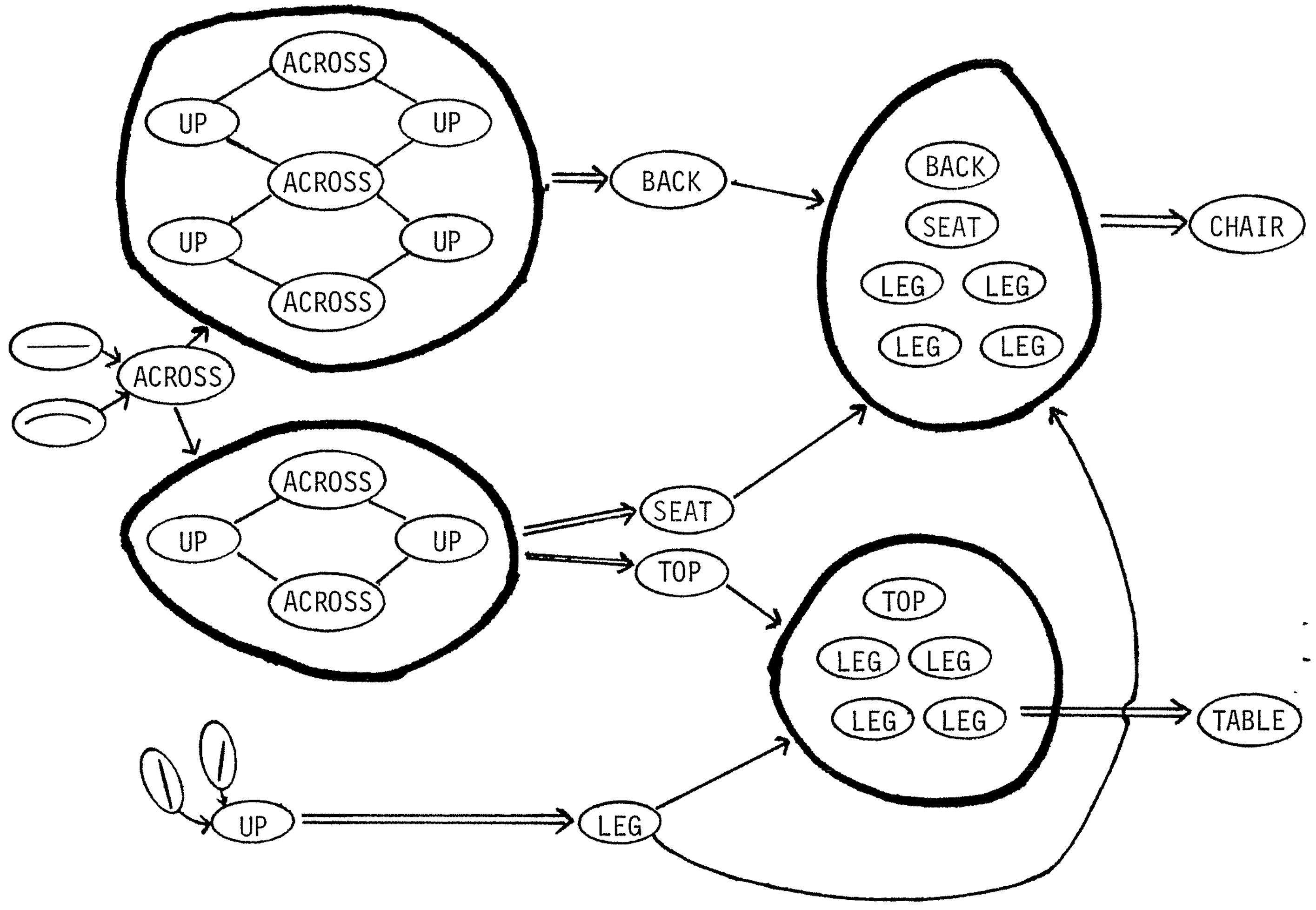
Section 1 gives the memory nodes needed to handle the 2-dimensional pattern recognition problems. Several layers of the recognition cone are set up, giving averaging, differencing and transforming characterizers rich enough to begin to handle a variety of distorted and shaded inputs. Many more such transforms would be needed to handle a wider range of objects, but these examples should indicate how they can rather routinely be described.

Section 2 shows the additional memory nodes needed to handle all other problems. With 1-dimensional inputs perceptual recognition becomes much easier (although difficulties can arise with misspellings), and either SEER-1 or SEER-2 can handle these problems.

Figures 6 and 7 give a sketchy idea of two portions of the memory network into which the conversion routine transforms these inputs.

1) Memory 1. For 2-Dimensional Inputs

Type	Description	Implieds	Classes	Name
L	L1:3]			
MEM	A: 121:			BRIGHT]
Y	242]			
Y	121]			
L	L2:3]			
MEM	A: AAA:			BRIGHT]
Y	A8A]			
Y	AAA]			
L	L3:3]			
MEM	00000(3;7):	HOR*2;ACROSS$N;:	STROKE;:	HOR]
Y	11111]			
Y	00000]			
MEM	01(2;6):	VERT*2;UP$N*3;LEGS$N;:	EDGE;:	VERT]
Y	01]			
Y	01]			
Y	01]			
MEM	0001(4;7):	DIAGA*2;UP$N;:	STROKE;:]	
Y	0010]			
Y	0100]			
Y	1000]			
MEM	1000(4;5):	DIAGB*2;:	STROKE;:]	
Y	1100]			
Y	0110]			
Y	0011]			
MEM	00111100(3;6):	CURVA*2;ACROSS$N;ENCLOSURE$N;:	STROKE;:]	
Y	01000010]			
Y	10000001]			
MEM	110(3;9):	RLOOP;LEFTEAR;FACE$N;:	STROKE;:]	
Y	001]			
Y	110]			
MEM	011(3;3):	LLOOP;RIGHTEAR;FACE$N;:	STROKE;:]	
Y	100]			
Y	111]			
MEM	101(1;7):	DLOOP;NOSE$2;:	STROKE;:]	
Y	101]			
Y	111]			
Y	010]			
MEM	010(3;9):	CIRCLE;EYE*3;FACE$N;:	OBJECT;:	EYE]
Y	101]			
Y	010]			
MEM	10001(2;7):	DISH;MOUTH;:	STROKE;:]	SAUCER]
Y	01110]			
MEM	10000001(3;6):	SAUCER;ENCLOSURE$N;:	STROKE;OBJECT;:	
Y	01000010]			
Y	00111100]			



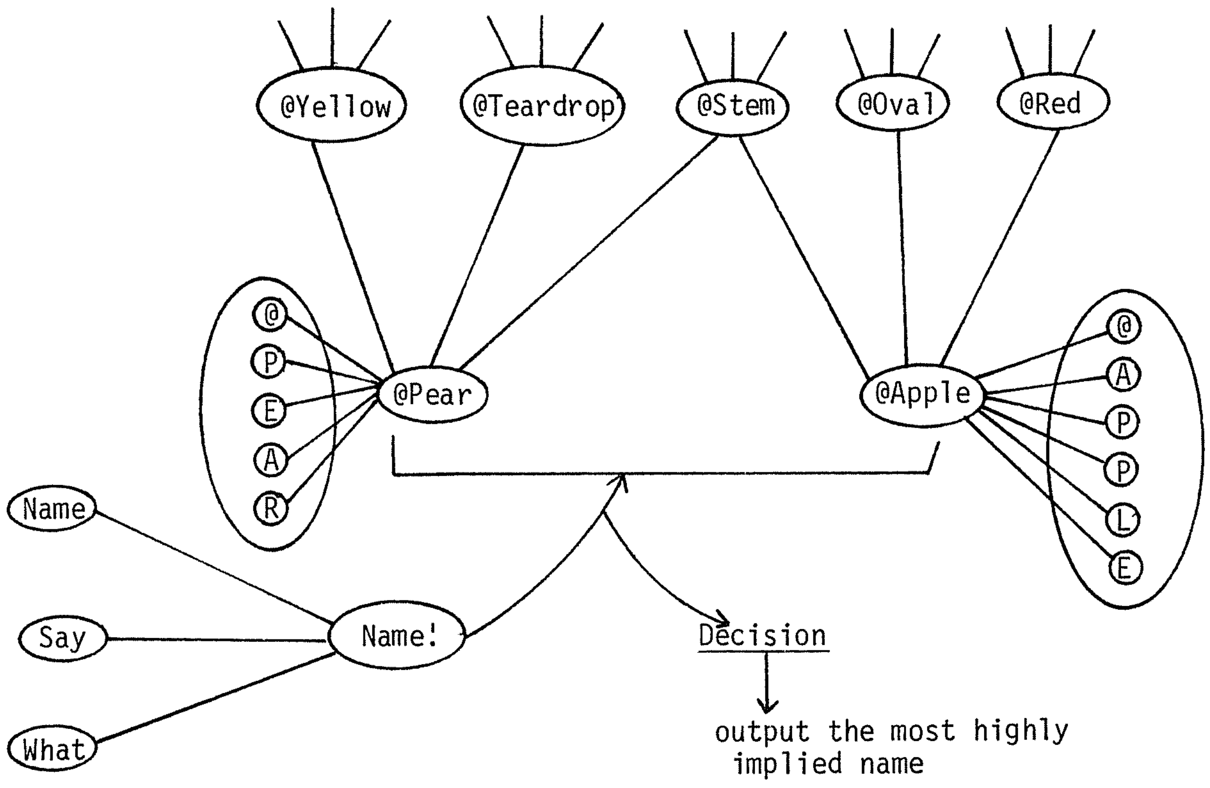
A GRAPHIC REPRESENTATION OF THE PART OF MEMORY THAT IS NEEDED TO RECOGNIZE CHAIRS AND TABLES. NOTE THAT MANY DETAILS, LIKE WEIGHTS, THRESHOLDS, AND HOW POINTERS ARE USED, ARE NOT SHOWN.

FIGURE 6

```
Type   Description           Implieds                              Classes            Name

L      CHARS:]

MEM    HOR;CURVEA;(1):       ACROSS*6;BACK$N*4;:                                      ACROSS]

MEM    VERT;DIAGA;(1):       UP*3BACK$N*4;LEVEL$N*6;STICK;:                           UP]
L      IDEAS]
MEM    !  !ACROSS(7):        BACK*9;CHAIR$N*9;:                                       BACK]
Y      !!UP!    !UP!]
Y      !!UP!    !UP!]
Y      !   !ACROSS!]
Y      !!UP!    !UP!]
Y      !!ACROSS!]

MEM    !O!ACROSS!(7):        SEAT*14;CHAIR$N*8;TOP*13;TABLE$N*8;:                     LEVEL]
Y      !O!UP!   !UP!]
Y      !O!HOR!]

MEM    VERT;:VERT;VERT;VERT;]
         (7):]               LEGS*8;CHAIR$N*6;:                                       LEGS]

MEM    !!CURVEA!:            VISAGE;FACE$N*3;BALLOON$N;
                               CIRCLE;:                            FIGURE;:           ENCLOSURE]
Y      !!SAUCER!]            ((in IDEAS))
( If FACE succeeds, BALLOON is implied with a negative weight, to negate it.]
MEM    !!VISAGE*5!(6):       FACE*35;PERSON$N*4;BALLOON*-7;: OBJECT;:                 FACE]
       !! RIGHTEAR!!LEFTEAR!]
       !    !EYE! !EYE!]
       !      !NOSE!]
       !    !MOUTH!]

MEM    !!CIRCLE!:            BALLOON*16;:                          OBJECT;:           BALLOON]
       !!STICK!]

MEM    !!BACK!(9):           CHAIR*25;COUCH$L;:                    FURNITURE;OBJECT;: CHAIR]
Y      !!SEAT!]
Y      !!LEGS!]

MEM    !!TOP!(9):            TABLE*21;CHAIR$L;:                    FURNITURE;OBJECT;: TABLE]
Y      !!LEGS!]
```

1) MEMORY 2. For 1-Dimensional Inputs. (to be inserted, as indicated, into the proper layer, ordered to fit examples in the text.)

Type	Description	Implieds	Classes	Name
(The following go in Layer 1 (right after L L1:3]) (For Pattern Recognition)				
(The word PEAR implies "look for a pear (@PEAR)" and "Internally associate about pear)				
MEM	' PEAR '(1;7):	PEAR;@PEAR$N;IPEAR$N;:	WORD;TWORD;:	PEAR]
MEM	' @PEAR '(1;7):	@PEAR;TACT$N;IPEAR$N*2;PEAR;:	OBJECT;FRUIT;:	@PEAR]
MEM	NAME(1;(7):	NAME;TONAME$N*8;TACT$N*2;:	WORD;COMMAND;:	NAME]
MEM	WHAT:	TONAME$N;TOFIND$N;:	WORD;:	THIS]
MEM	NAME;SAY;WHAT;(1):	TACT$N*35;:	COMMAND;:	TONAME]
(The following goes in IDEAS (right after L IDEAS:])				
MEM	ACT:	T OBJECT;:		TACT]
(The following go in Layer 1)				
MEM	@TEARDROP(1;7):	@PEAR2$N;@TEARDROP;:	QUAL;:	@TEARDROP]
MEM	@STEM(1;5):	@APPLE2$N;@PEAR2$N;@STEM;STEM;	QUAL;:	@STEM]
(A 2d characterizer of @PEAR. Note variations (e.g. @YELLOW didn't point to it).)				
MEM	@YELLOW;@TEARDROP*2; @STEM;(4):	@PEAR*3;NACT$N;IPEAR$N;:	OBJECT;FRUIT;:	@PEAR2]
MEM	@OVAL(1;6):	@APPLE2$N;@OVAL;:	QUAL;:	
MEM	@RED;OVAL*2;STEM;(4):	@APPLE*4;NACT$N;IAPPLE$N;:	OBJECT;FRUIT;:	@APPLE2]
(For Describing (The following go in L1)				
MEM	DESCRIBE(1;6):	DACT$N*99;RACT$N*97;:	COMMAND;:	DESCRIBE]
MEM	ALL;:	DACT$N3;:	ADJ;:	ALL]
(The following goes in IDEAS)				
MEM	ACT:	D OBJECT;:		DACT]
(For retrieving information) (The following go in Layer 1)				
MEM	PRESIDENT(1;7):	PRESIDENT;IPRESIDENT$N*9; PNOW$N*5;PFIRST$N*5;:	NOUN;VIP;:	PRESIDENT]
MEM	BEER;:	BEER;:	PROD;:	BEER]
MEM	CHEESE;:	CHEESE;:	FOOD;PROD;:	CHEESE]
MEM	GARBAGE;:	GARBAGE;:	PROD;:	GARBAGE]
MEM	PROD;:	PROD;:	CLASS;:	PROD]
MEM	APPLE;:	RED*89;FRUIT*93;OVAL*83;	FOOD;:	IAPPLE]
(The following go in IDEAS)				
MEM	PRESIDENT;FIRST;(9):	WASHINGTON*99;RACT$N*50;:	COMMAND;:	PFIRST]
MEM	ACT:	R ;:		RACT]
MEM	PRESIDENT;:	FORD*48;RACT$N*23;:	COMMAND;:	PNOW]
MEM	WISCONSIN(1;7):	WISCONSIN;IWIS$N*5;IWIS2$N*4;:	WORD;STATE;:	WISCONSIN]
MEM	WISCONSIN;CITIES;:	MILWAUKEE*23;MADISON*18; RACINE*15;RACT$N*25;:	COMMAND;:	IWIS]
MEM	PRODUCE(1;6):	PROD;IWIS2$N;GARBAGE*23;:	:	
MEM	WISCONSIN*3;PROD;:	BEER*27;CHEESE*27;RACT$N*25;:	COMMAND;:	IWIS2]
MEM	WISCONSIN;TELL;:	COLD;COWS;LIBERAL;RACT$N*19;:	COMMAND;:	IWIS2]



A GRAPHIC REPRESENTATION OF PART OF THE MEMORY FOR CHARACTERIZING AND NAMING A FEW OBJECTS

FIGURE 7

Type	Description	Implieds	Classes	Name
(For deducing responses) (The following go in Layer 1)				
MEM	2;:	2*5;CACT$N;:	NUMBER;:	2]
MEM	TWO;:	2*5;:	NUMBER;:	TWO]
MEM	3 ;:	3*5;:	NUMBER;:	3]
MEM	ADD ;:	CACT$N*99;ADD*8;:	COMMAND;:	ADD]
MEM	PLAY(1;7):	GACT$N*36;:	COMMAND;:	PLAY]
(The following goes in IDEAS)				
MEM	ACT:	C COMMAND NUMBER NUMBER ;:		CACT]
MEM	ACT :	G Y.Y YXY X.X XXX(IWIN);:		GACT]
(For Finding and Moving) (The following go in L1)				
MEM	@BOX(1;7):	@BOX;BOX;:	CONTAINER;:	@BOX]
MEM	BOX(1;6):	BOX;@BOX$N;:	WORD;TWORD;:	BOX]
MEM	@PAIL(1;6):	@PAIL;PAIL;:	CONTAINER;:	@PAIL]
MEM	PAIL(1;6):	PAIL;@PAIL$N;:	WORD;TWORD;:	PAIL
MEM	@APPLE(;7):	@APPLE;APPLE;EAT$N;DACT$N*5;:	OBJECT;FRUIT;:	@APr
MEM	APPLE(1;6):	IAPPLE$N*99;APPLE;@APPLE$N; RACT$N*5;:	WORD;TWORD;:	AP'
MEM	FRUIT(1;6):	FRUIT;:	WORD;TWORD;CLASS;:	F
MEM	TOUCH(1;7):	TOUCH;FACT$N*37;FIND;:	COMMAND;:	
MEM	FIND(1;7):	FIND;FACT$N*37;:	COMMAND;:	
MEM	' TO ':	TO;:	PREP:	
NEW L	NEEDSGOALS NEEDS]	:N1 10]:N2 10]%		
MEM	FOOD;:	@HUNGER*9;EAT$N;:	OBJECT;:	
MEM	FRUIT;CANDY;(1):	FOOD;@HUNGER;EAT$N;:		
MEM	MOVE(1;7):	MOVE;TOMOVE$N*30;:	COMMAND;:	Mu.
MEM	@SEER-MOUTH(1;7):	@SEER-MOUTH;EAT$N;TO;:	SELF;:	
(The following go in IDEAS)				
MEM	OBJECT;TO;OBJECT;:	MACT$N*99;		TOMOVE]
MEM	ACT:	F OBJECT;:		FACT]
MEM	ACT:	M ;:		MACT]
MEM	@HUNGER;FRUIT;CANDY; TO;@SEER-MOUTH;(6):	EACT*99;:	COMMAND;:	EAT]
MEM	ACT:	M ;		EACT]

Appendix C: A Note on EASEy Programs (See Uhr, 1973f for details)

1. Numbering at the right identifies statements, and allows for comparisons between programs. M indicates initializing Memory statements: I indicates cards that are Input by the program. .V indicates a Variant, .1 an additional statement.

2. A program consists of a sequence of statements, an end card, and any data cards for input. (Statements that start with a parenthesis are comments, and are ignored.) Statement labels start at the left; gotos are at the right, within brackets (+ means branch on success; - on failure; otherwise it is an unconditional branch). + signifies a continuation card.

3. Strings on capitals are programmer-defined. Strings in underlined lower-case are system commands that must be present (they would be keypunched in caps to run the program). These include input, output, erase, set, list, get, start, call, that, and the inequalities. Other lower-case strings merely serve to help make the program understandable; they could be eliminated.

4. EASEy automatically treats a space following a string as though it were a delimiter; it thus automatically extracts a sequence of strings and treats them as names.],:,;, and % act similarly as a delimiter, but the programmer must specify it. The symbol # is used to stand for any delimiter (a space,], : , ;, % or #)

5. The symbol $stringI is used to indicate "get the contents of string I, and treat it as a name and get its contents" (as in SNOBOL).

6. Pattern-matching statements work just as in SNOBOL statements: there are a) a name, b) a sequence of objects to be found in the named string in the order specified, c) the equal sign (meaning replace), and d) a replacement sequence of objects (b, c, and/or d can be absent). that stringI means "get that particular object" - otherwise a new string is defined as the contents of stringI, which is taken to be a variable name.

7. size(...) is a built-in function that counts the symbols in the string(s) named within parentheses (its argument). integer(...) succeeds if its argument is an integer.

8. DEFINE: defines a programmer-coded function. The function is executed whenever it is specified, FUNCTIONNAME (ARGUMENTS), in the program. It ends in success or failure when it reaches a [return] or [-return] goto.

REFERENCES

Chomsky, N. Syntactic Structures, The Hague: Mouton, 1957.

Chomsky, N. Aspects of the Theory of Syntax, Cambridge: MIT, 1965.

Craik, K. J. W. The Nature of Explanation, New York: Cambridge, 1952.

Duda, R. O. and Hart, P. E. Pattern Classification and Scene Analysis, New York: Wiley, 1973.

Feldman, J. and Gries, D. Translator writing systems, Comm. ACM, 1968, 11, 77-113.

Feldman, J. A. et al. The Stanford hand-eye project. Proc. 2nd Int. Joint Conf. on Artificial Intell., 1971, 521-526.

Griswold, R. E. et al. The SNOBOL4 programming language Englewood-Cliffs: Prentice-Hall, 1968.

Guzman, A. Decomposition of visual scenes into three-dimensional bodies, Proc. AFIPS EJCC, 1968, 33, 291-304.

Hunt, E. B. Concept Formation: an Information Processing Problem New York: Wiley, 1962.

Jordan, S. R. Learning to use contextual patterns in language processing, Unpubl. Ph.D. Thesis, University of Wisconsin, Madison, 1971.

Kochen, M. An experimental program for the selection of disjunctive hypotheses, Proc. AFIPS WJCC, 1961, 19, 571-578.

Minsky, M (Editor) Semantic Information Processing, Cambridge: MIT, 1968.

Narasimhan, R. Labelling schemata and syntactic descriptions of pictures, Information and Control, 1964, 9, 151-179.

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

Nilsson, N. Problem-Solving Methods in Artificial Intelligence, New York: McGraw-Hill, 1971.

Nilsson, N. J. A mobile automaton: an application of A.I. techniques, Proc. 1st Int. Joint Conf. on Artificial Intell., 1969, 509-520.

Peirce, C. S. Collected Papers, Cambridge: Harvard, 1931-1958.

Quillian, M. R. The teachable language comprehender: a simulation program and theory of language. Comm. ACM, 1969, 12, 459-476.

Sauvain, R. and Uhr, L. A teachable pattern describing and recognizing program, Pattern Recognition, 1969, 1, 219-232.

Shapiro, S. A net structure for semantic information, storage deduction and retrieval, Proc. 2nd Int. Joint Conf. on Artificial Intell., London, 1971.

Shaw, A. C. A formal picture description scheme as a basis for picture processing systems, Information and Control, 1969, 14, 9-52.

Simmons, R. F. Answering English question by computer: a survey, Comm. ACM, 1965, 8, 53-70.

Simmons, R. F. Natural language question-answering systems: 1969, Comm. ACM, 1970, 13, 15-30.

Swain, P. H. and Fu, K. S. Nonparametric and linguistic approaches to pattern recognition, Elect. Engin. Tech Rept. TR-EE 70-20, Purdue Univ., Lafayette, 1970.

Thompson, F. English for computers, Proc. AFIPS FJCC, 1966, 28, 349-356.

Uhr, L. and Vossler, C. A pattern recognition program that generates, evaluates and adjusts its own operators, Proc. AFIPS WJCC, 1961, 19, 555-570. (Reprinted, with additional results, in E. Feigenbaum and J. Feldman, Eds., Computers and Thought, New York: McGraw-Hill, 1963.)

Uhr, L. Pattern-string learning programs, Behavioral Science, 1964, 9, 258-270.

Uhr, L., and Kochen, M. MIKROKOSMs and robots, Proc. 1st Int. Joint Conf. on Artificial Intell., 1969, 541-556.

Uhr, L. Flexible linguistic pattern recognition, Pattern Recognition, 1971, 3, 363-384.

Uhr, L. Layered "recognition cone" networks that pre-process, classify and describe, IEEE Trans. Computers, 1972, 21, 758-768.

Uhr, L. Pattern Recognition Learning and Thought, Englewood-Cliffs: Prentice-Hall, 1973(a).

Uhr, L. Recognizing, "understanding," deciding whether to obey, and executing commands, Computer Sci. Dept. Tech. Rept. 173, Univ. of Wisconsin, 1973(b).

Uhr. L. The description of scenes over time and space, Proc. AFIPS NCC, 1973, 42, 509-517(c).

Uhr, L. DECIDER-1: a system that chooses among different types of acts, Proc. 3d Int. Joint Conf. on Artificial Intell., 1973, 396-401(d).

Uhr, L. Describing, using "recognition cones", Proc. 1st Int. Joint Conf. on Pattern Recognition, 1973(e). (Also Computer Sci. Dept. Tech. Rept. 176, Univ. of Wisconsin, Madison, 1973.

Uhr, L. EASEy-2: An English-like program language, Computer Sciences Dept. Tech. Rept. 178, Univ. of Wisconsin, 1973(f).

Uhr, L. A wholistic cognitive system (SEER-1) for integrated perception, action and thought. Computer Sci. Dept. Tech. Rept., Univ. of Wisconsin, 1974.

Uhr, L. Semantic Learning, in preparation.

Uhr, L. An integrated cognitive system (SEER-2) that interacts with scenes that change over time, in preparation.

Waltz, D. L. Gene rating semantic descriptions from drawings of scenes with shadows, Unpubl. Ph.D. Diss., MIT, Cambridge, 1972 (AI TR-271).

Winograd, T. Procedures as a representation for data in a computer program for understanding natural language, Unpubl. Ph.D. Diss., MIT, Cambridge, 1971. (Also "Understanding Natural Language, New York: Academic, 1972).

Winston, P. H. Learning structural descriptions from examples, Unpubl. Ph.D. Diss., MIT, Cambridge, 1970.

Winston, P. H. The MIT robot, In: Machine Intelligence 7 (B. Melzer and D. Michie, Eds.) Edinburgh Univ. Press, 1972.