ROBOT SIMULATION STUDIES : DESCRIPTIONS AND PLANS

by

Peter F. Rowat and Richard S. Rosenberg

April 1972

Submitted to the Canadian Computer Show Competition 1972

Table of contents

ROBOT SIMULATION STUDIES : DESCRIPTIONS AND PLANS(+)

Peter F. Rowat and Richard S. Rosenberg
Department of Computer Science
University of British Columbia
Vancouver 8, Canada.

Abstract

The problem of designing a robot-controller is approached by taking a simplified, computer-simulated, model of a robot in an environment, and writing programs to enable the robot to move around its environment in a reasonably intelligent manner. At no point is mathematical logic used. The problems of concept representation and the creation and execution of plans are dealt with in this simple system, and the problem of exploration is encountered but not satisfactorily dealt with. ROSS, an interactive computer program which simulates the robot-environment model, is described. A command language allows the user to specify tasks for the robot at various conceptual levels. Several problems are listed concerning the ways in which a robot might explore, represent, and make plans about, its environment, most of which are amenable to direct attack in this simplified model. Finally, theoretical questions concerning two-dimensional rectanguloid shapes are raised.

---

1. Introduction

The paper is organised as follows. Section 1.1 gives a general review of robot research while section 1.2 describes the nature and importance of our own contributions. Section 2 describes the design of the simulated world and the robot's computational nervous system. In section 3 a computer program which implements this design is described, and in section 4 we give a summary, some problems, and an indication of future work.

1.1 Review of robot research

The concept of a robot has been a fantasy of mankind for thousands of years and to-day, for the first time in history, attempts are being made in various centres to build real physical robots.

(+). The financial support of the National Research Council of Canada, through their grants 67-5552 and 67-7642, is gratefully acknowledged.

"Shakey", a mobile robot built at the Stanford Research Institute[13] , can navigate across a floor having several plane-faced objects in the way, and can push several such objects into a group. The Stanford University hand-eye system[5] can identify and manipulate blocks well enough to solve the "Instant Insanity" puzzle[6] . The School of Artificial Intelligence at the University of Edinburgh uses a stationary robot with mobile surroundings[2,15]. Systems consisting of a mechanical hand plus visual and/or tactile receptors that can manipulate simple objects are in use in Japan[3] and at MIT, and are proposed in Italy[1]. Recent work in robot research is fully reported elsewhere[16].

Robot research is important for many reasons. The political and social implications of the successful construction of competent mechanical men are vast and immeasurable; they cannot be entered into here. Many a science fiction writer has considered them, and, in a more serious vein, Gregory has commented on the social implications of intelligent machines[8]. We merely point out that the most obvious initial uses of robots are for jobs that man finds boring or dangerous, or for jobs in situations where man could not survive, such as planetary and deep ocean exploration. In addition, it will certainly be the responsibility of robot researchers to prevent the horrific prospect of robots being used in warfare from becoming reality.

Suppose that, some time in the future, a reasonably competent robot has been constructed. As a concrete example, consider a robot which is used at the docks. It is fully autonomous while at the docks, can load and unload packing crates of various sizes and weights from the holds of ships, and always stacks the crates in a neat and efficient manner. Necessarily, answers to the following questions will have been implemented in the design of this robot. How does the robot conceive of and reason about it's environment? How does it perceive, amongst other things, packing crates? How does it plan and carry out it's actions? In the course of providing such answers robot researchers will not be able to avoid casting considerable light on, if not solving, many fundamental problems of knowledge, thought, reason and perception that have baffled philosophers since at least the time of Plato. Thus robot research is of considerable philosophical importance.

Finally, the efforts to build real robots are important to computer science because they impose a new viewpoint or paradigm on the subject of artificial intelligence. Many problems previously tackled in isolation from one another must now be approached in a reasonably uniform manner, so that, for example, programs for problem-solving, information storage and retrieval, pattern-recognition, and language understanding, can all communicate with one another. Other problems, hard to deal with in isolation, must now be faced. These include: providing the machine with an adequate world-model; devising a good and universal method for the representation of knowledge; creating and executing plans of action; and handling the uncertainties and ill-defined, "fuzzy", problems associated with real inputs

from the external world. There is also, of course, the sheer problem of organising a complex system: "The main principle ... is the dependence of everything on everything."[5]

Robot research, currently, remains a purely experimental science despite the efforts of Hewitt[10], Hayes[9], and others to provide some theoretical background. An experiment consists of running a program (or robot) which embodies one's idea, and observing the resultant behaviour. If the behaviour exhibits "appropriate novelty" (Gregory's dictum[8]), or compares favourably with human behaviour or with the behaviour of other similar programs (if any), then the idea is judged as useful. There are three common approaches to the setting up of experiments. Very briefly, one is to try to simulate the physiology of the human brain and body, another is to try to simulate human psychology as developed in various theories, and the third is to forget about human physiology and psychology and make a direct attack on the problems. All the hardware robot projects, and our own simple robot simulation system, take the direct approach.

Physiologists analyze at the lowest level the workings of the central nervous system; psychologists analyze human behaviour and try to synthesize the conceptual nervous system; while workers in artificial intelligence, if we may be allowed to coin a badly needed phrase, try to synthesize the computational nervous system.

1.2 Our contributions to robot research

1.2.1 Nature of the original contributions made . We have made a direct attack on the problem of designing the computational nervous system, or brain, of a robot, in what is believed to be an original manner. The approach is very simple, but indicates how one might proceed without resorting to logical, linguistic, or other Fregean modes of representing and reasoning about the world in a robot. The use of a simulated, two-dimensional, robot/environment system rather than a real system detracts little from the value of this work; in some ways it is a positive advantage.

Two new algorithms have been devised. One, the algorithm DECOMP described in section 2.3.2, takes a rectanguloid shape in two dimensions and decomposes or "parses" it into its maximal subrectangles. The other, the algorithm CONTAIN described in section 2.3.3, compares a pair of two dimensional rectanguloid shapes and decides whether one of the shapes could be moved to fit inside the other.

Perhaps the most original contribution is showing how to represent a robot's model of his world as a graph, and how the robot should use this graph to create plans of action.

1.2.2 Importance of the original contributions made . All other approaches to the problem of representing and reasoning about a robot's world are, in essence, based on John McCarthy's

"Advice Taker" program[12]. He proposed "a system that reasons verbally", where by "verbally" he really means "by utilizing first order logic". Carl Hewitt's PLANNER language[10] is based on first order logic and provides a simple and explicit way of setting up and dismissing goals and subgoals in the style of Newell,Shaw, and Simon's General Problem Solver[4](GPS). The Stanford Research Institute's STRIPS program[7] is comparable to PLANNER to the extent that it, too, is based on first order logic and incorporates in its control structure a means-end analysis in the style of GPS. Hayes proposes a "Logic of Actions"[9] based entirely on first order logic which is intended "to provide a more flexible interface between the physics of the world-model and the formal behaviour of the logic."

There are many reasons to be suspicious of any approach based on logic; rather, what is needed is an approach which captures some of our intuitive modes of thought. The importance of our contribution, then, is that it indicates an alternative approach which aims to do just this; however, only further work will show whether this alternative, intuitive, approach is really viable.

2. Design of the simulated robot/environment system

2.1 Basic approach

Suppose one is introduced to a new environment such as a large one-floor house, or a university campus. Now consider the following tasks.

Task 1 : explore and form an internal model of the environment, or in other words, learn your way about.
Task 2 : find your way from one point to another, in a reasonably efficient manner.
Task 3 : move a large object, say a table, from one point to another.

These tasks are very simple for humans, in fact so simple that we can carry them out almost unconsciously. But if asked "How do you carry out these tasks?" ,in terms of the data processing required, one is hard pressed to give an answer. Before a robot can be built that is capable of carrying out the above tasks the question of "How?" must first be answered for each of them.

The basic idea is this: take a simple, idealized, model of a robot in an environment, and see what the robot requires to enable it to carry out the above tasks. The model world should be kept as simple as possible, but not so simple that the above tasks don't make sense. Start with data structures and procedures as simple as possible for the robot's computational nervous system (CNS), and add more complex structures and procedures as required. When the model robot is able to carry

out the above three tasks then it will be possible to answer the question of "How ?" for each of them. Also, hopefully, we might have gleaned some insight into what form the computational nervous system of a real robot might take. As extra motivation for setting up procedures and data for the robot, keep in mind the following simple game which is obviously related to the games played by young children: consider an environment which contains, apart from its walls and other fixed objects, a number of movable objects of various shapes, and an equal number of fixed holes of various shapes. Let the robot wander around and discover and describe the movable objects and holes. Then it must decide which objects, if any, fit into which holes, and then, for each object which it knows fits into a certain hole, move that object through the environment and into the hole, if that is possible.

2.2 The simulated world

Robbie lives in a chess-board type of world : an n-by-n grid of squares where each square is marked with a letter. Currently the value of n is set to 28, but the performance of Robbie in no way depends on this number. The larger the grid, the more interesting the environments we can give Robbie to work in. The letters have the following meanings :

' ' (blank)	the square is vacant.
'B'	the square is a barrier: forms part of the boundary or part of a fixed object.
'M'	the square forms part of a movable object.
'H'	the square forms part of a hole.

At any instant Robbie occupies one square, specified by coordinates (x,y), and is in one of four orientations, north, south, east, or west. An environment plus Robbie in a specific position and orientation is called a configuration (see Figure 1).

He has the following actions. He can move one square at a time in the direction he is facing, and can turn left or right or through 180 degrees while remaining on the same square. His sensory capabilities are limited : he can only sense the contents of the eight squares surrounding him. He can only occupy blank squares. A square marked 'B' or 'H' blocks his way. In general a square marked 'M' also blocks his way. If, however, he is facing an 'M' square he can pick up the whole movable object, OBJA say, of which that square is a part. He and OBJA then become a rigid body : if he takes a step or turns, OBJA goes with him, provided no collision occurs between the proposed final position of OBJA and some wall or other object. If such a collision occurs then the configuration remains as it was before the attempted step or turn. As a result of Robbie's manoeuvres, OBJA may overlap the squares of a hole. After OBJA has been picked and moved, Robbie may drop OBJA. Thereupon Robbie and OBJA cease to be a rigid body and he may walk away.

2.3 Robbie's computational nervous system

2.3.1 The ring representation of objects. We chose, for reasons of simplicity, to represent objects, movable objects, holes in the environment, and the environment itself, as a cyclic list of the edges and corners that occur when one goes round the object. The natural way to represent this computationally is as a "ring" of linked nodes where each node gives the length of an edge and the type of turn (left or right) at its end. An example is given in figure 2.

> Definition: this ring of nodes is the ring representation of an object.

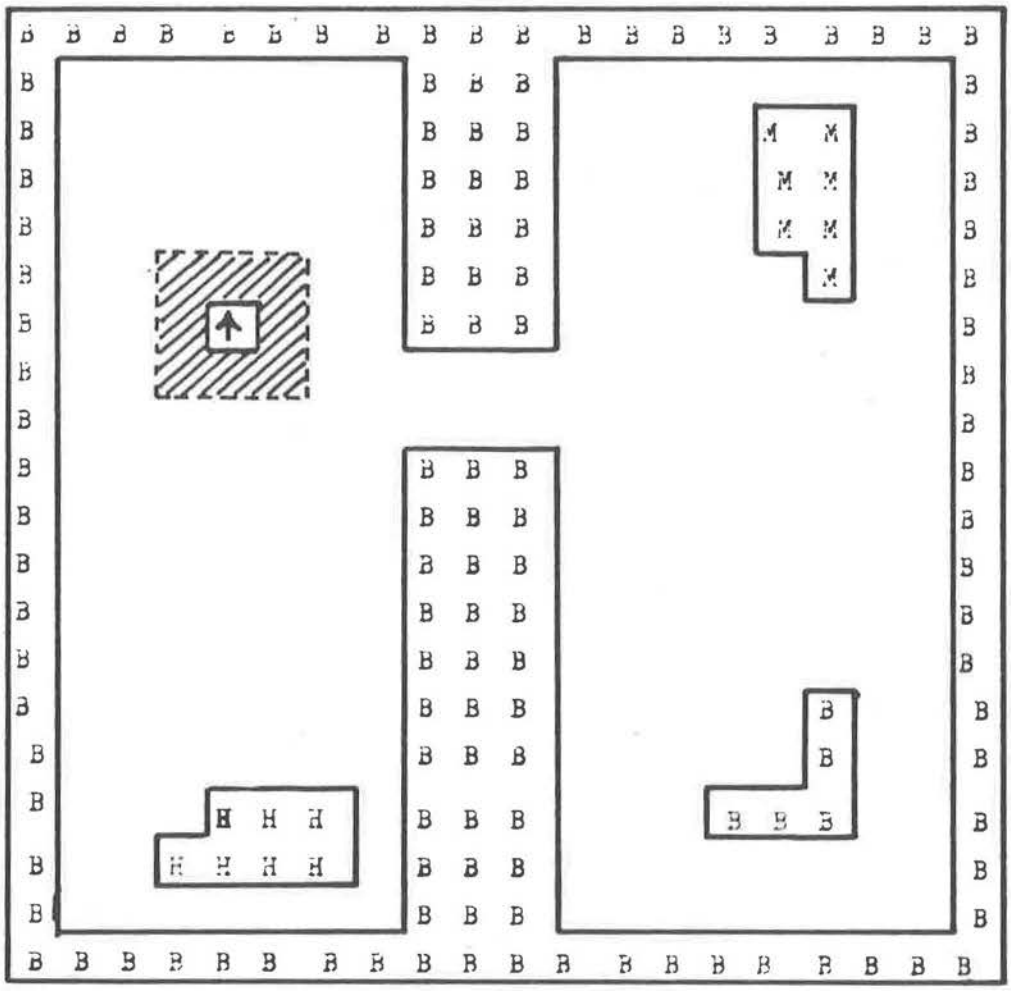
On encountering such an object in his environment, Robbie walks round the edge of the object, keeping to the left, and generates the above description. This seems to be about as simple a description as one could devise. With this description it is straightforward to compare two objects for congruence (can one be translated and rotated to lie exactly on top of the other?), for similarity (is one an expansion or contraction of the other? - see figure 3), and for corner-congruence (both the same "up to corners" - i.e. do they both have the same number of edges and corners, where the corner types must agree but the edge lengths may not? - see figure 4).

In order to decide whether one object could be moved to fit inside the other, we devised a decomposition of the ring representation which involves a deeper analysis of the shape of the object (or of the boundary of the environment itself). In fact this decomposition is basic to most of the procedures in Robbie's computational nervous system, so is of central importance.

2.3.2 Maximal subrectangles: the algorithm DECOMP . In the environment in which Robbie lives, a rectangle is the simplest kind of object. Given two rectangles, it is trivial to decide if one can fit inside the other ; also, supposing Robbie is inside a rectangular environment, it is trivial to move from place to place. The natural suggestion, then, is that a more complicated object or environment should be decomposed into a conglomeration of overlapping rectangles. For example, an "L" and a "U" are decomposed into overlapping rectangles as in figure 5. However it is not quite so obvious how an object such as in figure 6 should be decomposed into rectangles. What is needed are all the "biggest" rectangles contained in an object.

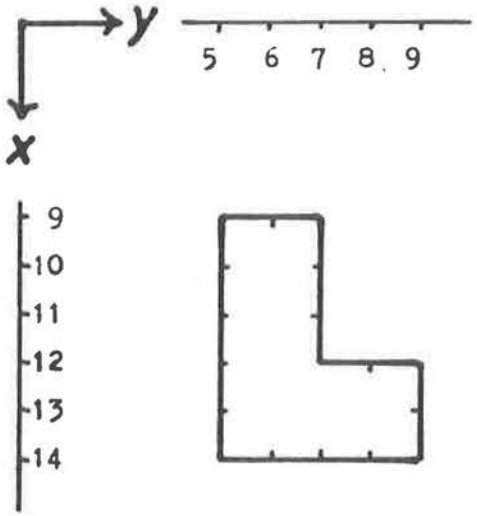
> Definition : a maximal subrectangle of an object O is a rectangle R contained in O such that each side of R has a subinterval in common with an edge of O.

The representation of an object contains, besides the representation ring, the list of all its maximal subrectangles (abbreviated "MRT"s). For instance the object in figure 6 is decomposed into the collection of MRTs in figure 7.



An example of a configuration : Robbie in an environment. The arrow designates his position and orientation. The shaded squares are all that he can "see". There is a fixed object marked with B's, a hole marked with H's, and the M's mark a movable object that could be moved to fit into the hole.

Figure 1.



This "L" - shaped object has the
the ring - representation diagrammed below.

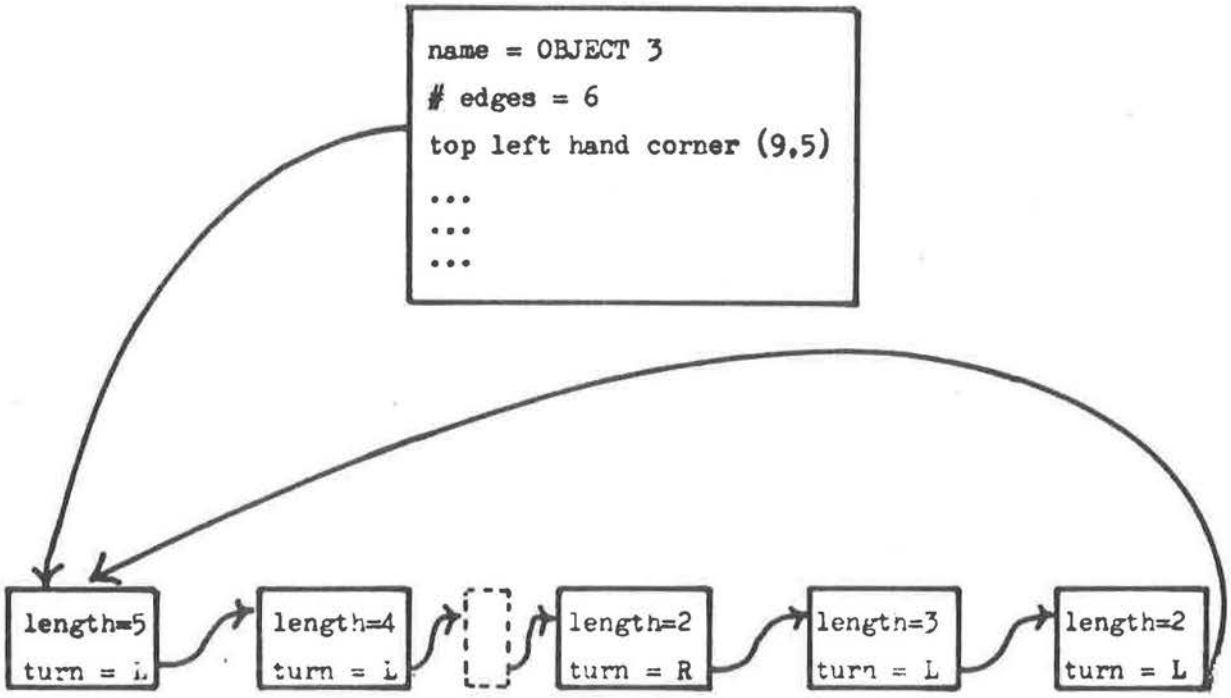
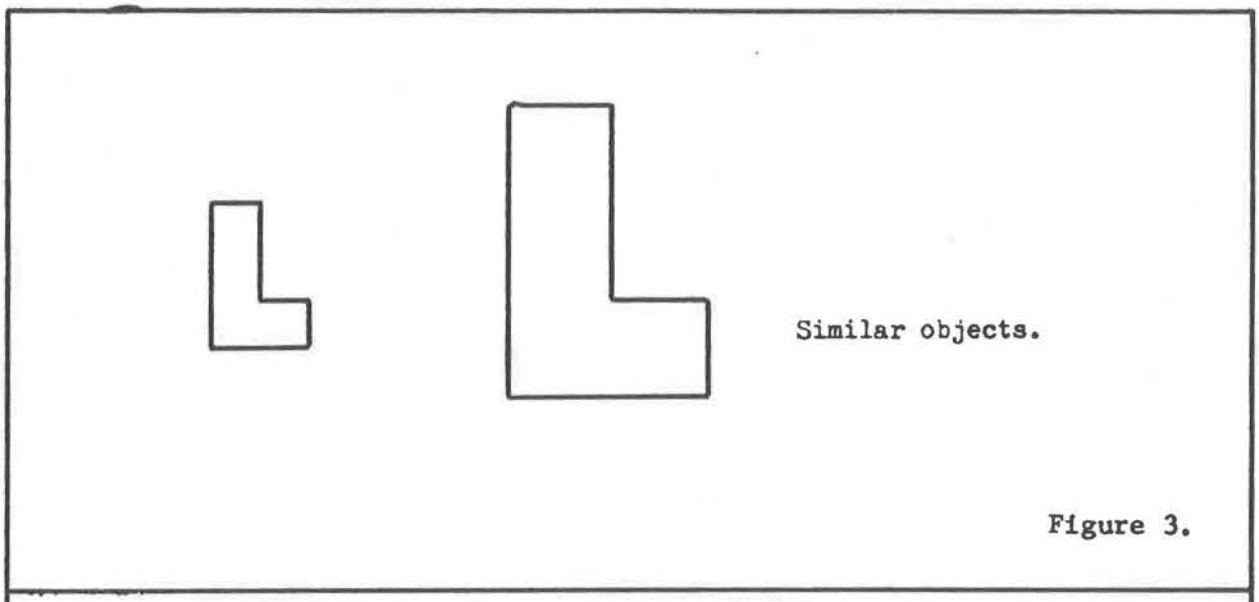
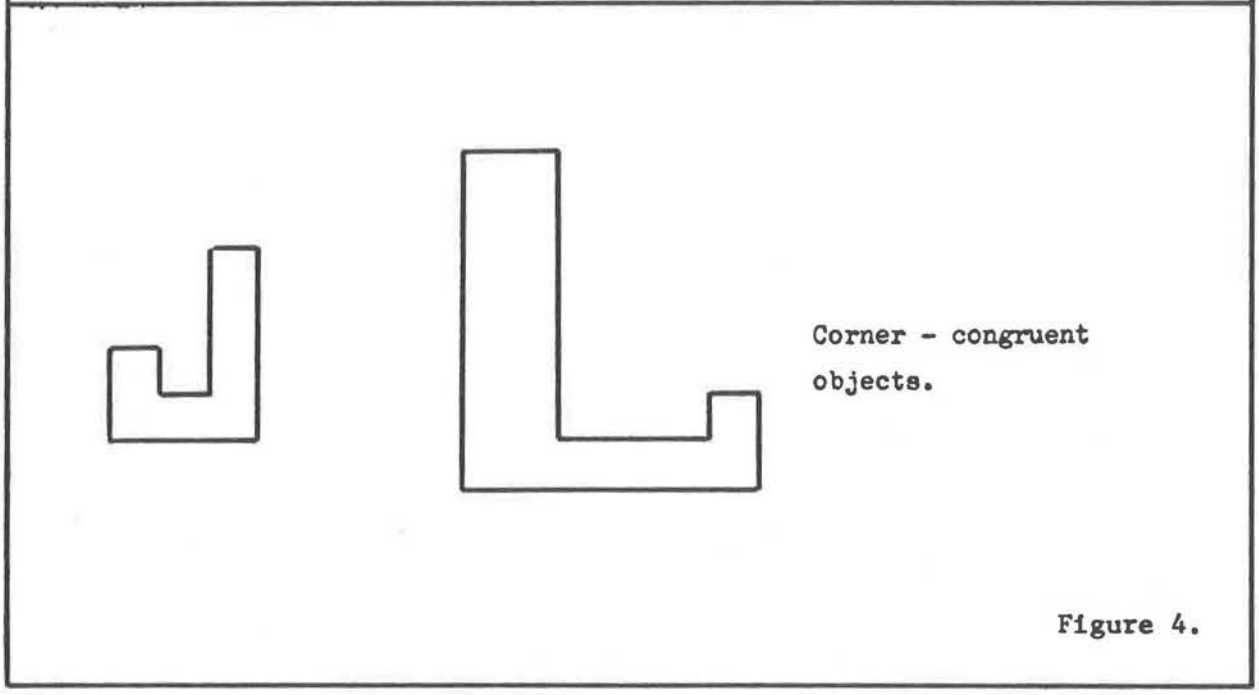


Figure 2.



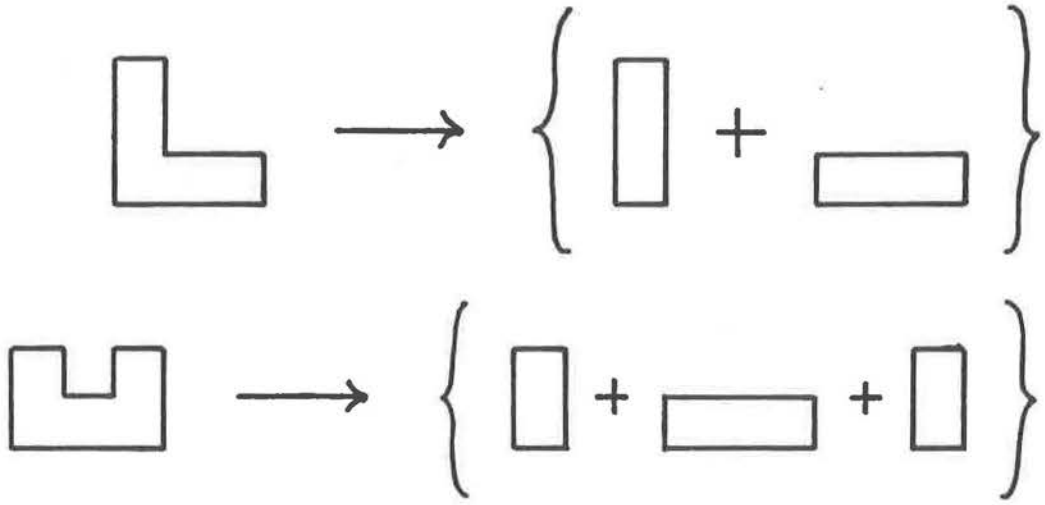
Similar objects.

Figure 3.



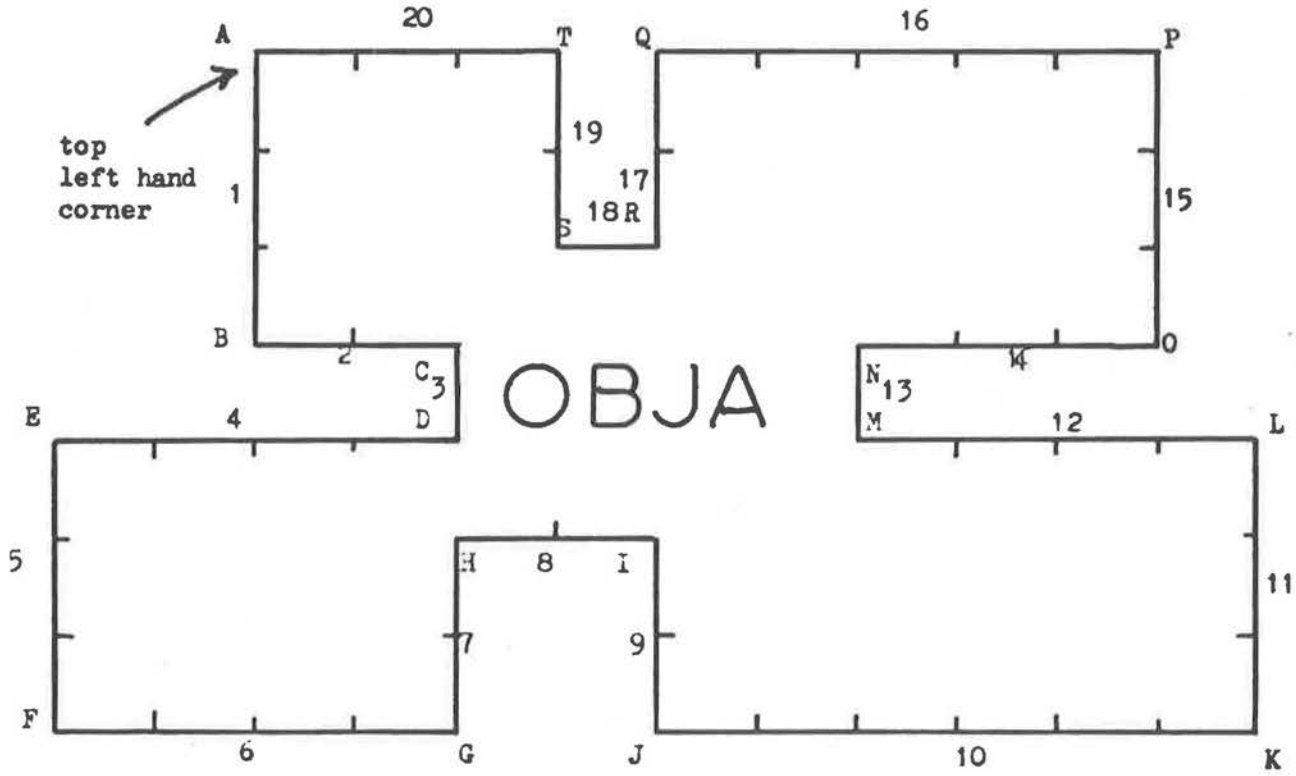
Corner - congruent objects.

Figure 4.



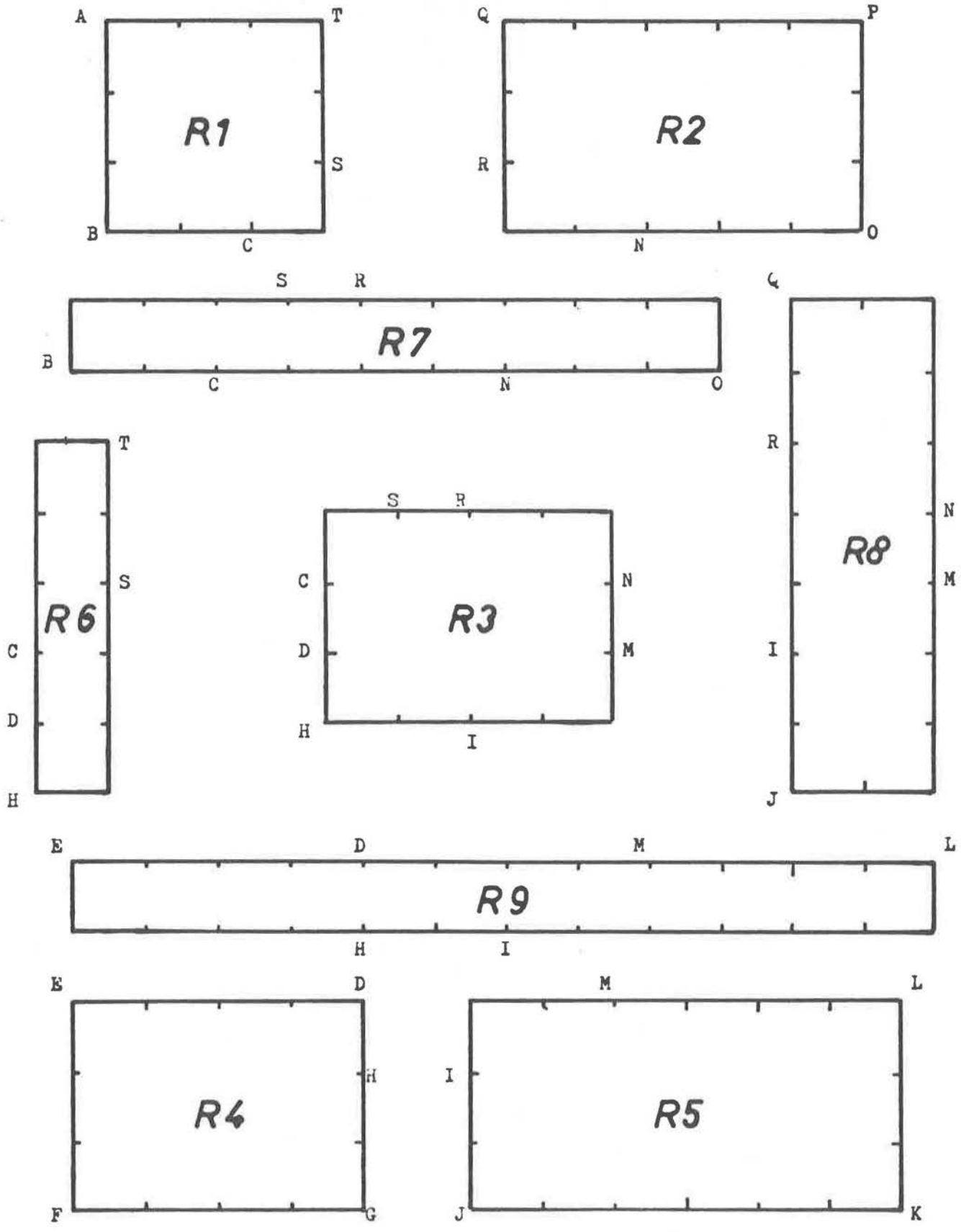
Decomposition of simple shapes into overlapping rectangles.

Figure 5.



It is not immediately obvious how to break the shape of OBJA into overlapping rectangles (but see figure 7).
The digits by each edge give the edge-number.

Figure 6.



Conceptual decomposition of the shape of CBJA (figure 6) into the MRTs R1, R2, ... , R9.

Figure 7.

DECOMP is a basic procedure in the system which decomposes the ring representation of an object into a list of all its MRTs. In a sense one can say that DECOMP "parses" an object into its constituent MRTs.

The algorithm DECOMP

DI. [Initialize]. Set up the empty MRT list.
DL1. Take the first left edge L and go to step DB1.
DL2. Take the next left edge L. If no more left edges, stop and return the MRT list.
DB1. Take the first bottom edge B after L in ring order which is accessible from L, that is, it would be possible to draw a rectangle with its left side defined by L and its bottom side defined by B. Go to step DR1.
DB2. Take the next bottom edge B accessible from L. If no such bottom edge exists, go back to step DL2.
DR1. Take the first right edge R after B which is accessible from L and B, that is, it would be possible to draw a rectangle with its left side defined by L, its bottom side defined by B, and its right side defined by R. Go to step DT.
DR2. Take the next right edge R which is accessible from L and B. If no such right edge exists, go back to step DB2.
DT. Check through those top edges which lie between the right edge R and the left edge L in ring order, and which overlap the horizontal interval defined by L and R. If one of these lies at or below the bottom ends of both L and R, then no inscribed rectangle exists whose left, bottom, and right sides are defined by L, B, and R respectively ; go back to step DR2. Otherwise, let T be the lowest of the top edges checked through. Then the rectangle whose left, bottom, right, and top sides are defined by L, B, R and T respectively is a maximal subrectangle : add it to the MRT list and go back to step DR2.

Figure 8 gives examples of shapes for which decomposition into MRTs is clearly not the best approach, but for the moment we ignore these complications.

2.3.3 Containment : the algorithm CONTAIN . Given two objects OBJA and OBJB, the question "Can OBJA be moved to fit inside OBJB ?" may now be answered. First, the "super-rectangle" of each object must be found.

> Definition : the super-rectangle of an object is the smallest rectangle which contains that object.

The partial ordering given by the relation of containment between rectangles is naturally represented as a lattice. The MRTs of each object are classified according to their dimensions and arranged in the lattice given by the containment relation. Since several MRTs in different parts of an object may have the same dimensions the lattice structure is actually imposed on equivalence classes of MRTs rather than on individual MRTs. The lattice of an object is invariant under rotations. For example,

four rectangles of dimensions 3-by-4, 3-by-9, 7-by-5, and 7-by-9 would be arranged in a diamond shaped lattice with the { equivalence class consisting of the } 7-by-9 rectangle at the top covering the two incomparable rectangles 3-by-9 and 7-by-5, while the 3-by-4 rectangle would be at the bottom, covered by the 3-by-9 and 7-by-5 rectangles. The top positions of the lattice correspond to {equivalence classes of } MRTs into one of which every other MRT could fit. As an example, the MRTs of figure 6 form the lattice shown in figure 9. The top positions in this lattice correspond to MRTs of dimensions 3-by-6, 1-by-12, 7-by-2.

Now we outline the algorithm CONTAIN for answering the question of containment. The input to CONTAIN consists of two objects OBJA and OBJB where each input object has been decomposed into a list of MRTs, and the component MRTs organised into a lattice structure by an algorithm which is a considerably modified version of Donald E. Knuth's topological sort[11]. The output is either a straight "no", or "yes" together with the rotation and translation required to move OBJA into OBJB (ignoring the complications of possible obstructions such as other objects, walls, etc.). For instance, if one of the input arguments to CONTAIN were the OBJA of figure 6, it would be accompanied by the list of 9 MRTs indicated in figure 7 and the lattice shown in figure 9.

The algorithm CONTAIN

CT1. Can the super-rectangle of OBJA fit inside the super-rectangle of OBJB, or in other words are the x- and y-dimensions of OBJA both less than or equal to the x- and y-dimensions of OBJB ?
If not, answer "no" and stop.

CT2. If OBJB is a single rectangle, answer "yes" and stop.

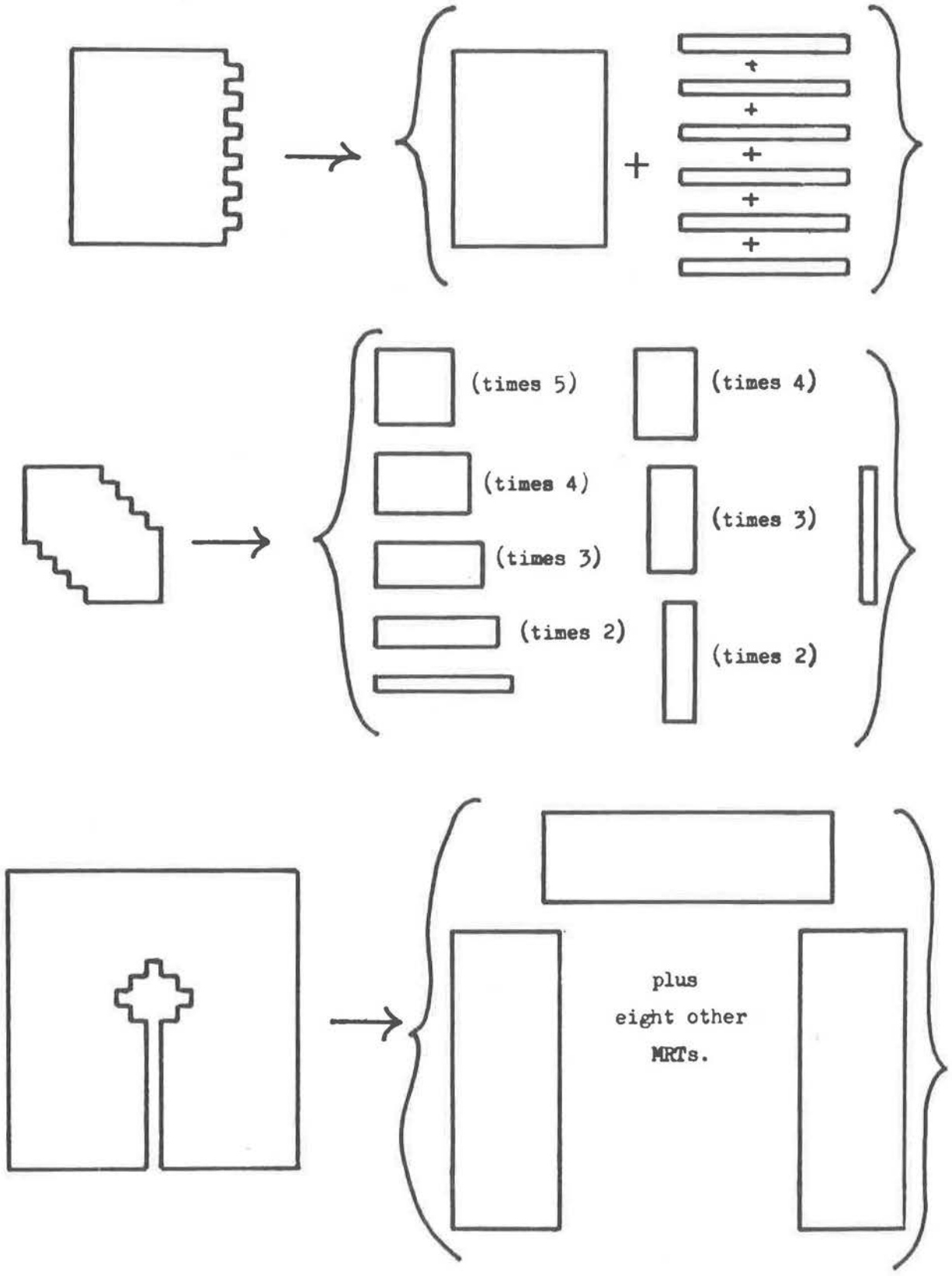
CT3. Can the super-rectangle of OBJA fit into one of the top MRTs of OBJB ? If so, answer "yes" and stop. If not, and OBJA is a single rectangle, answer "no" and stop ; otherwise proceed.

CT4. Can each of the top MRTs of OBJA fit into one of the top MRTs of OBJB ? If not, answer "no" and stop.
< Now we know that, disregarding the relative positions of the MRTs of OBJA, every MRT of OBJA can fit into OBJB somewhere. >

CT5. Take each of the top MRTs of OBJA in turn and count how many different ways there are to fit it into MRTs of OBJB, then pick an MRT A* of OBJA for which the number of different ways is a minimum.

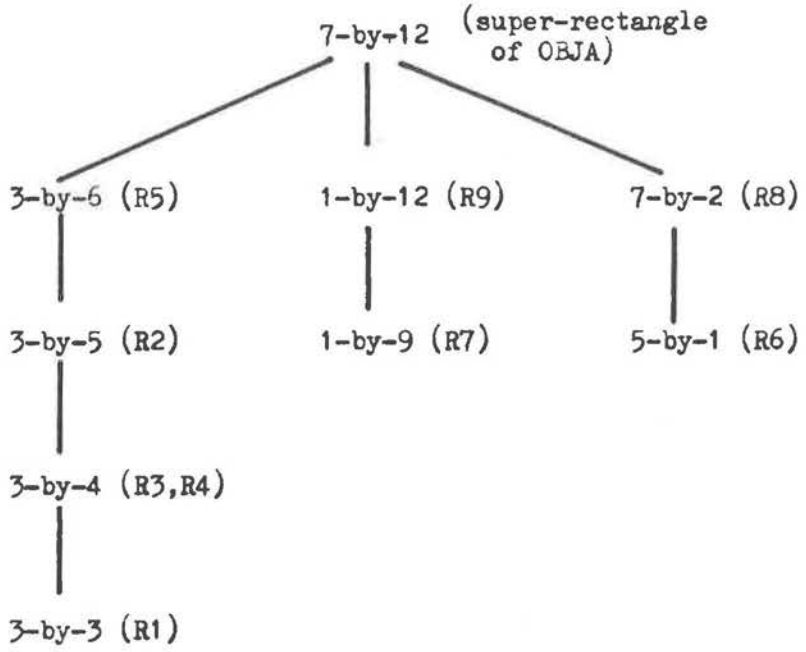
CT6. For each of the different ways in which A* can fit into OBJB, take the translation of OBJA required and check if all the remaining MRTs of OBJA are indeed inside an MRT of OBJB. If a suitable translation is found, answer "yes" and stop ; otherwise, answer "no" and stop.

Figure 10 shows two cases in which step CT6 must be invoked



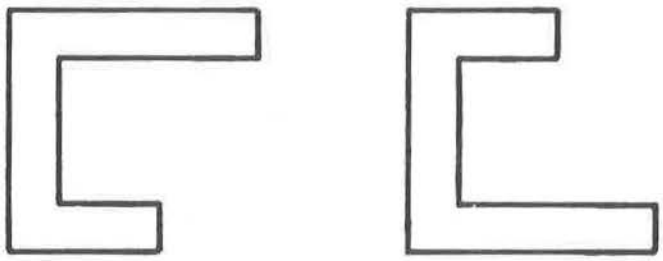
Examples of shapes for which representation by straight-forward decomposition into MRTs is clearly not the best approach.

Figure 8.



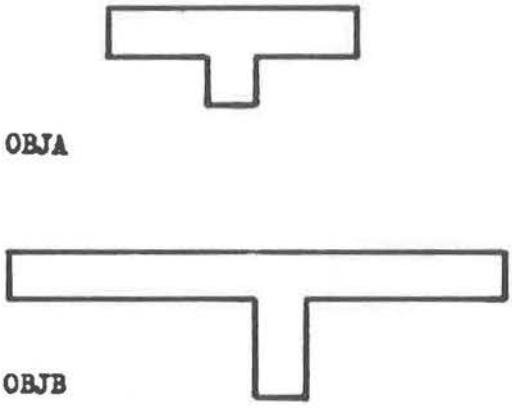
The lattice formed by the MRTs of OBJA of figure 7.

Figure 9.



OBJA OBJB

CONTAIN (OBJA,OBJB) doesn't answer "no" until CT6.



OBJA

OBJB

CONTAIN (OBJA,OBJB) doesn't answer "yes" until CT6.

The containment question : two pairs of arguments for the algorithm CONTAIN for which step CT6 must be invoked to answer correctly.

Figure 10.

to answer "yes" or "no" correctly.

The above algorithm works reasonably well for two objects of similar size and complexity of shape. However, improvements could be made. For instance, the searching required in CT5 could be considerably reduced by making use of connectedness when the objects being compared consist of long sequences of connected MRTs. This would be done by utilizing the following obvious fact about connectivity :

> If MRTs A1 and A2 are connected in OBJA, and A1 can fit into MRT B1 of OBJB, then A2 can only fit intc B1 or some MRT of OBJB that is directly connected tc B1.

2.3.4 Plans: their construction and execution. The plans dealt with in the system so far are merely those required for Robbie to move from place to place in a reasonably efficient manner within an environment where the only obstacles are small fixed objects. At the time of writing even these plans are limited, in that Robbie cannot yet deal satisfactorily with the unexpected occurrence of large fixed or movable objects while executing a plan.

The construction and execution of plans requires extensive use of Robbie's model of the world. We take the decomposition of the environment into MRTs and set up for every pair of overlapping MRTs an "overlap" link and insert between them an "intersection rectangle" (abbreviated "IRT") which specifies how they overlap. Now suppose Robbie is at position A in MRT1 in the environment, and he must reach position B in MRT5 if that is possible, as illustrated in figure 11. The environment, when decomposed into MRTs and with overlap links and intersection rectangles inserted, may be viewed as a graph whose vertices are MRTs and whose edges are the overlap links between MRTs. The program MAKPLAN uses a path-finding algorithm to find a chain of MRTs connected by overlap links from MRT1 to MRT5. If such a chain is found, it constitutes a plan of action for going from A to B ; otherwise, no such chain exists and it is impossible to reach B from A. Figure 12 illustrates the construction of a plan to reach position B.

The path finding algorithm

> Call the starting node in the graph HERE and the node to which a path must be found THERE. Colour HERE red and THERE blue. No other nodes are coloured initially. A wavefront of red nodes expands in steps from HERE and a wavefront of blue nodes expands in steps from THERE. At step n the red wavefront consists of all nodes whose shortest path back tc HERE is of length n , and the blue wavefront consists of all nodes whose shortest path back tc THERE is of length n. A node retains the colour first assigned to it. The red and blue wavefronts are expanded in alternate steps. When the wavefronts

meet, a path has been found from HERE to THERE, and this is the path produced by the algorithm. Note that this is a path of minimal length, and that it would be a simple matter to modify the algorithm to obtain all paths from HERE to THERE.

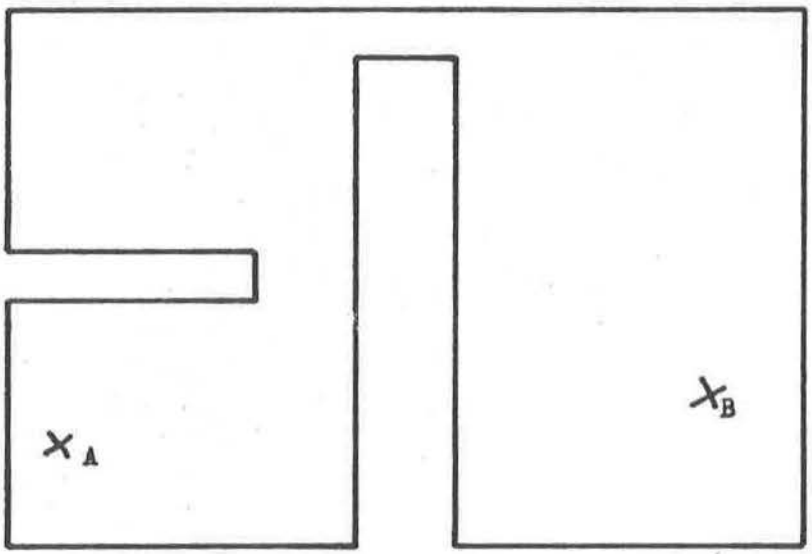
The algorithm is illustrated in figure 13. The algorithm was written quite independently of Pohl, who discusses path problems in depth and describes a similar path-finding algorithm in similar language[14].

A plan produced by MAKPLAN is executed by the program EXPLAN. The execution of a plan is hierarchically organised. EXPLAN calls on MOVE, MOVE calls on LINEAR, LINEAR calls on STEP and TURN, and the effects of STEP and TURN are defined by the procedures which simulate reality. MOVE is in charge of each leg of the plan, where a typical leg is "move through MRT3 into IRT4" ; it is also in charge of avoiding any unexpected obstacles. LINEAR is in charge of moving as linearly as possible from Robbie's current position to a specified destination position.

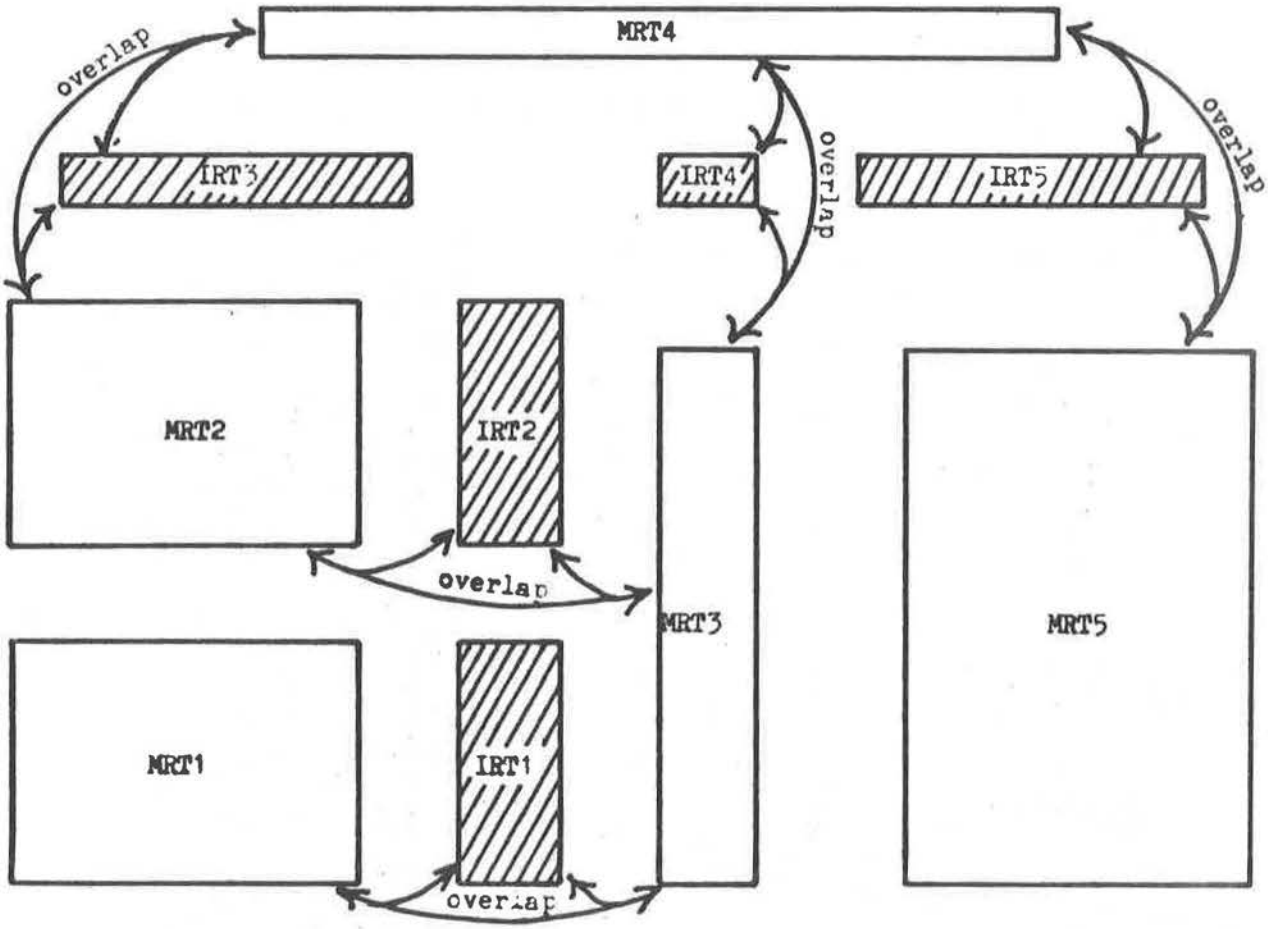
One of the difficulties inherent in any system for executing a plan is dealing with the unexpected. This takes different forms at different levels. At the lowest level in our system, STEP can fail because the square in front of Robbie is not vacant. TURN can fail only when Robbie is holding an object. LINEAR can fail if STEP fails, and reports this back to MOVE. MOVE fails when LINEAR fails, assumes the failure is due to an unexpected object, and takes avoiding action with calls to STEP and TURN. EXPLAN fails if MOVE persists in failing after several attempts at avoiding action, and reports failure back to the control program. In a more sophisticated system there would be, at this level, re-planning by MAKPLAN.

2.3.5 Exploration . We have not yet indicated how Robbie generates the ring representation of his environment in the first place, or how he first finds an isolated object and then generates it's ring representation. To generate the environment's ring representation the procedure FIND sends Robbie off in a straight line until a 'B' square is found. Then the procedure FOLLOW causes Robbie to follow the boundary through 360 degrees, using a set of procedures called RINGS to generate the ring-description as he goes.

To find isolated objects Robbie does the following for each MRT of his environment. First he goes to it, using PLANS, then he uses the procedure EXPLORE to explore it. EXPLORE is, at the time of writing, extremely crude: it merely sends Robbie to the centre of the MRT, and if by chance he encounters a non-blank square he uses the procedure FOLLOW to follow the boundary of the object of which this square was a part.

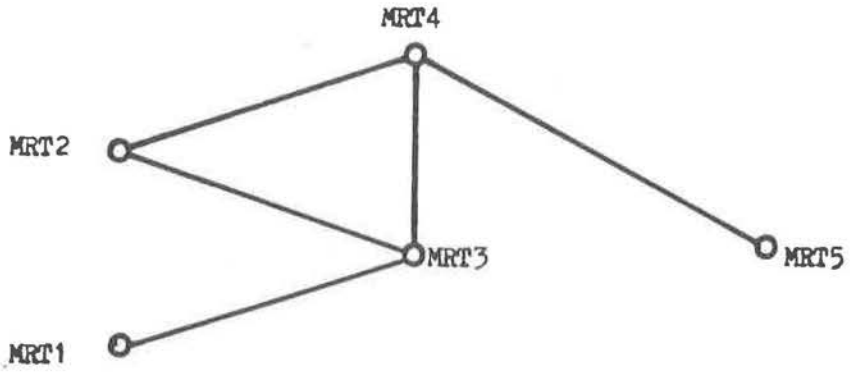


Environment E



Decomposition of E into MRTs plus overlap links and intervening intersection rectangles (shaded).

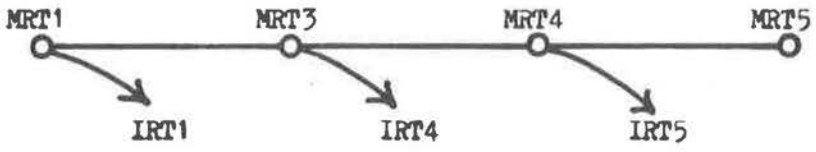
Figure 11.



The graph form of environment E.

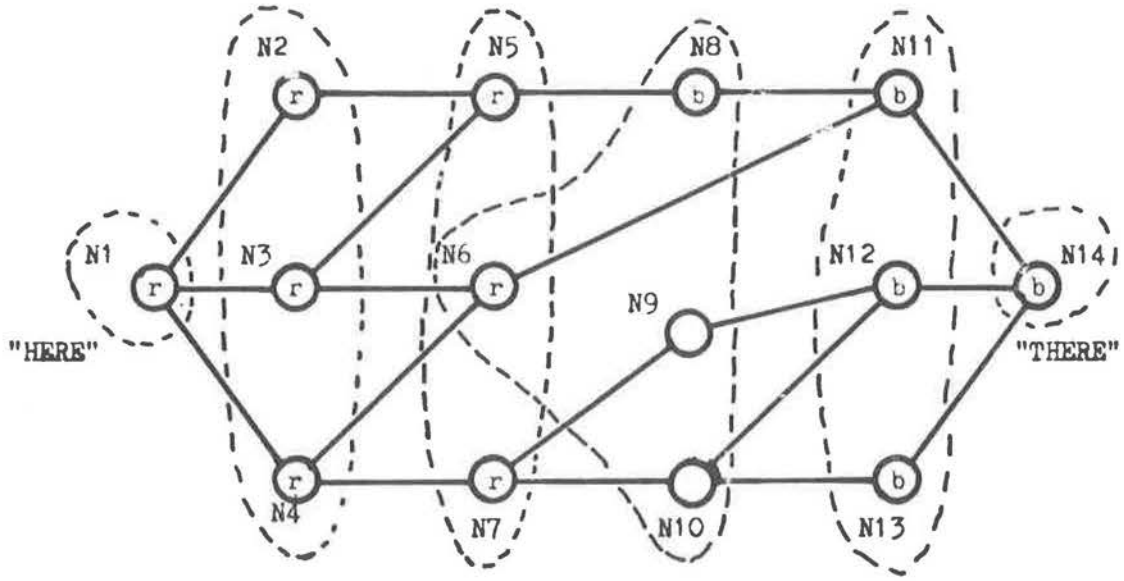
```
Start in MRT1
Move through MRT1 into IRT1
Move through MRT3 into IRT4
Move through MRT4 into IRT5
Ends in MRT5.
```

Printed output of the path-finding program MAKPLAN.



The chain of pointers produced by MAKPLAN.

Figure 12.



(r) is a node that has been coloured red.

(b) is a node that has been coloured blue.

The encircled groups of nodes of the graph are the successive wavefronts as found by the algorithm PF. The table below gives the order in which the wavefronts are found.

Successive RWAVEfronts	Successive BWAVEfronts
N1	
	N14
N2,N3,N4	
	N11,N12,N13
N5,N6,N7	
	N8, ...

The path N1, N3, N6, N11, N14 from HERE to THERE is found when advancing the BWAVEfront for the second time.

Example to illustrate the action of the algorithm PF on a graph.

Figure 13.

2.4 Summary of Robbie's world

On the level of direct contact with the outside world, Robbie knows his position and orientation and can "see" the eight squares surrounding him, and possesses a pickup arm which, when "active", "holds" a movable object in the outside world.

The data structures used in Robbie's computational nervous system are extremely simple. At the top level he has two pointers and four stacks of objects. One pointer, called "home", points to the header of the ring representation of his environment. The other pointer, called "currentmrt", points to the MRT of the environment in which he is currently located. The stacks are used for the four different kinds of objects that Robbie may find in his environment : fixed objects, movable objects, holes, and anything else that doesn't fit into one of the first three categories.

The header of the ring representation of the environment or of any object contains several pointers. Four point into the ring representation (having four instead of one merely speeds up questions of congruence, corner-congruence, and similarity), one points to the list of MRTs of the object, and one points to the lattice structure associated with the MRTs of that object.

At a lower level, the overlapping MRTs of the environment are linked together with overlap pointers, and each MRT of a pair of overlapping MRTs possesses a pointer to the intersection rectangle of that pair.

The programs which build and manipulate Robbie's model of the world will now be listed. These should be regarded as being part and parcel of his model : the programs and the representations on which they act are inextricably intertwined.

RINGS simply constructs a ring representation when Robbie is following the boundary of his environment or of an object. DECOMP produces a list of MRTs from a ring representation. SETOLAP constructs the overlap pointers. LATCONS constructs the lattice structure of an object from its list of MRTs. FIND and FOLLOW first find and then follow the boundary of the environment or of an object. PLANS incorporates MAKPLAN and EXPLAN , and is perhaps the most often used program. MAKPLAN uses the overlap pointers to construct a plan which is then executed in hierarchical fashion by EXPLAN . CONGRUENT , C_CONGRUENT , SIMILAR and CONTAIN are used to compare the shapes of two objects. EXPLORE finds new isolated objects.

3. ROSS: a RObot Simulation System

The preceeding design has been incorporated in the interactive program ROSS. The three most basic parts of ROSS are:

1. The simulated world, REALITY.

2. The simulated robot, Robbie, which is in, and interacts with, REALITY.

3. Robbie's computational nervous system which contains, inter alia , his model of the world.

To make it into a usable system two other components are provided.

4. A camera to provide snapshots on a display screen which show how REALITY changes as a result of Robbie's actions, and to show how Robbie uses his model of REALITY .

5. A command interpreter by means of which the user can issue commands to Robbie, or alter other parts of the system.

3.1 Commands

There are four groups of commands. The global commands administer the simulated world, and the display commands control the camera; no more will be said of these two groups. The action commands allow the user to request Robbie to carry out various actions, while the comparative commands cause Robbie to compare in various ways the shapes of objects that are known to him. These last two groups will be illustrated by examples.

The action commands occur at three conceptual levels. At the lowest level we can request Robbie to take a STEP, to TURN, to PICKUP a movable object, or to DROP a movable object. At this same level we can ask him to move as LINEARly as possible to a new position, which request he would execute by means of a sequence of STEP and TURN actions that aproximate his motion to a straight line. Note that all of these commands may fail, either because Robbie has encountered a fixed object or hole, or because he is holding a movable object and the requested action would cause the held object to collide with a wall or other object in the environment.

At a higher level we may ask Robbie to FIND the boundary of his environment or, having found it, to FOLLOW the boundary all the way round.

At what is, currently, the highest conceptual level we have the commands of most interest: HOME , WALK , and EXPLORE .

3.2 Examples of action and comparative commands .

Six snapshots are shown in figures 14,15 and 16. The first five are all taken from one episode in Robbie's life, and illustrate the action commands; we refer to this as episode A. The last snapshot is taken from another episode, episode B, and

suffices to illustrate the comparative commands.

Snap #4 shows the configuration after the commands "LINEAR :19,22", "PICKUP", have been issued.

Several things should be noticed here. The 3-by-3 array called SENSE is all that Robbie can "see" at one time. The 3-by-3 array MSENSE is only defined when Robbie is holding a movable object, and then allows him to "see" under the held object so that he can avoid falling into holes. For instance, he must avoid the "L"-shaped hole in this configuration. Note that the SENSE and MSENSE arrays are printed as they would appear relative to Robbie himself. In the fifth line from the bottom, "HOLDING : MOBJ 3", the name "MOBJ 3" is known only to the world simulation procedures, not to Robbie. All he knows is that he is holding something. Finally, remember that although we can view the environment as a whole, all that Robbie is aware of is summarised beneath the horizontal "curtain" of dots drawn below the environment.

Snap #30 shows the configuration after an extended sequence of commands at the lowest level. The "T" has been inverted, and the small "L"-shaped object at centre-left has been moved from one room (MRT) into another.

The next three snaps, #32,#33,#34, illustrate the action commands at the highest conceptual level. Snap #32 shows the result of issuing the command "HOME". Robbie first found the boundary by going horizontally right and then, after finding a corner of the boundary, he followed it all the way round. As a result he now knows the ring representation of his environment and how it decomposes into overlapping MRTs. Notice the fourth line from the bottom, "CURRENT MRT: MRT 1": he is aware of what room he is in. At this point the dimensions and positions of the six MRTs of this environment are printed out for the user.

Snap #33 shows the effect of the command "WALK TO MRT:5". By means of the PLANS procedures, a plan was created and successfully executed; note that the "CURRENT MRT" is now MRT 5.

Snap #34 shows the effect of the command "EXPLORE MRT :3". First of all Robbie created and executed a plan to reach MRT 3, just as for a WALK command. Note how he bumped into and then sidestepped the isolated fixed object at position (4,11). Then he found the "L"-shaped object in MRT 3 and followed its boundary in the same way that he followed the boundary of his environment in snap #32. Consequently he is now aware of one movable object in his environment, as printed in the second line from the bottom.

Snap #8 is taken from episode B, and shows the result of a sequence of "EXPLORE" commands. (The hole has been slightly enlarged.) Robbie now knows of four movable objects and one hole, by the names "OBJECT 2", ..., "OBJECT 6". Several examples of comparative commands follow, where "MOBILE" is to be understood as "MOVABLE OBJECT". To the command "IS? MOBILE:2

CONGRUENT TO MOBILE:3", Robbie replied "NO".

To the command "IS? MOBILE:2 SIMILAR TO HOLE:6", Robbie answered "YES","MOVE 10 STEPS UP , AND 17 STEPS RIGHT". The moving information produced ignores the fact that there is a wall in the way!

To the command "IS? MOBILE:5 CONTAINABLE IN HOLE:6", Robbie replied, after a certain amount of computation in which he constructed the (extremely simple) lattices associated with objects 5 and 6 , "NO".

3.3 Implementation

ROSS is implemented on an IBM Model 360/67 at the University of British Columbia. It consists of over 25 PL/1 external procedures, amounting to about 4,700 PL/1 statements, and was compiled by an IBM PL/1 F compiler, version 5.0, running under MTS. The system occupies 50 pages of core prior to any list processing, but by the end of episode A an extra 16 pages had been used. This last extravagant figure could be reduced by using PL/1's AREA variables to keep all the space allocations in one place, and by more careful garbage collection. Episode A, which involved about 25 action commands intermixed with global and display commands, took only 2.19 seconds of CPU time, so the execution time of ROSS is negligible.

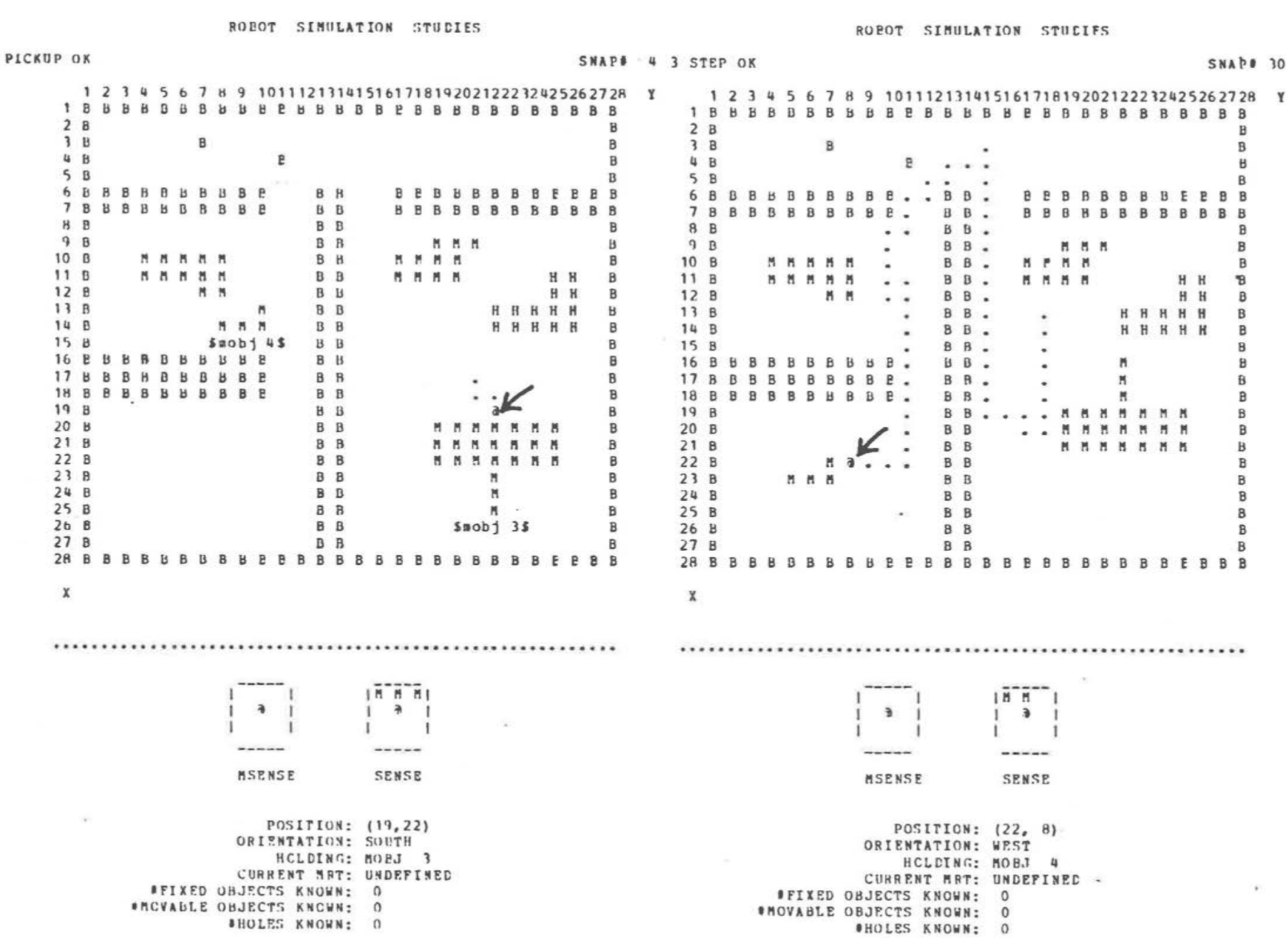
4. Conclusion

4.1 Summary and problems

We have designed and implemented a simple robot simulation system. The robot can explore its environment in a simple fashion, and can make elementary plans to move from place to place in a manner which, though not novel, is at least appropriate. The robot uses an elementary model of his world to move about, and can add new information to this model in an unstructured way as he explores his world. The basic problems of concept representation and the creation and execution of plans are dealt with in a simple way, but the handling of the exploration problem needs improvement. In attempting to make the robot more intelligent some important problems must be faced, as detailed below.

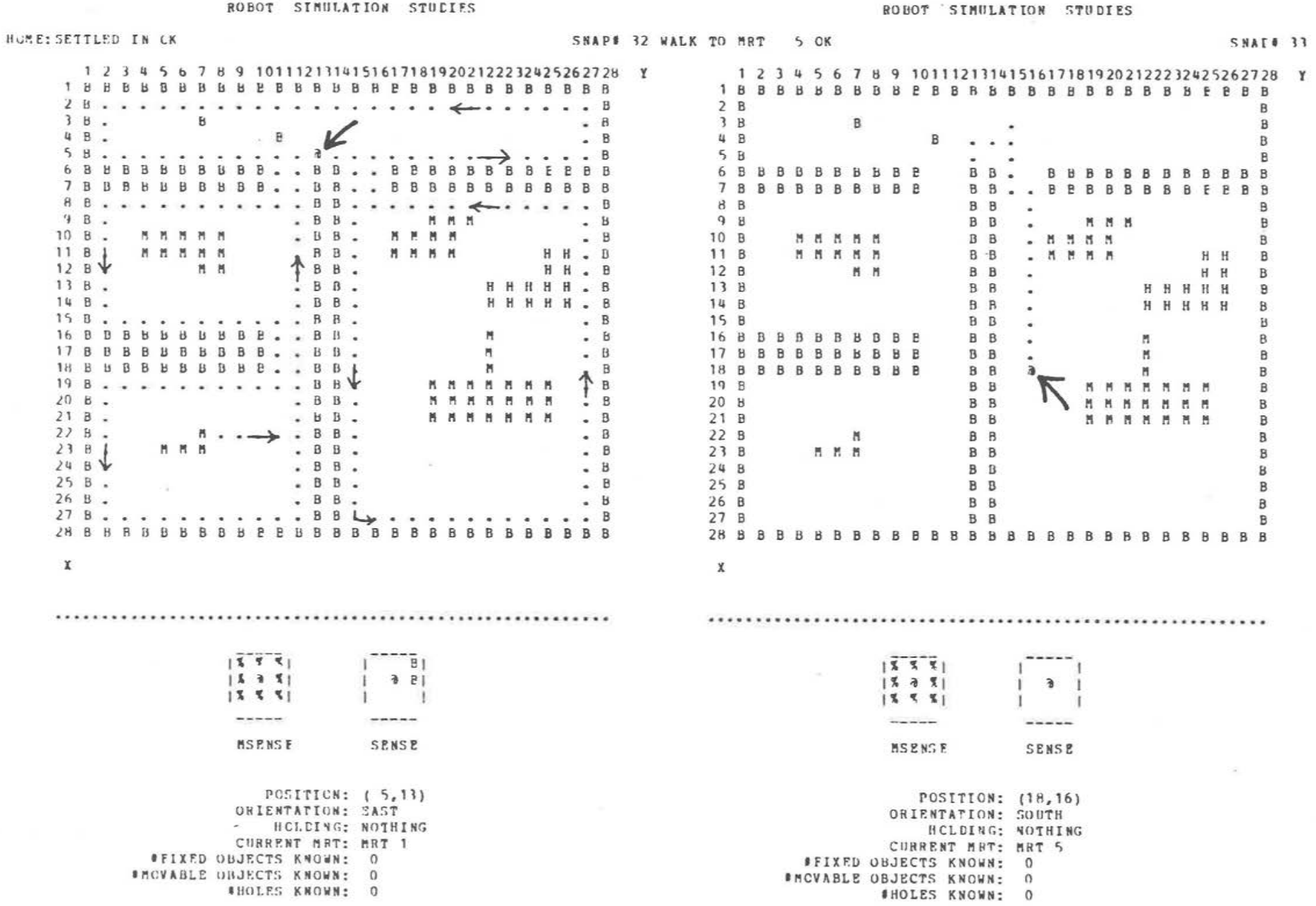
Exploring problems . There are two of these. The first is: What "expectations" or "hypotheses" should the robot have, and how should the robot behave as a consequence of these hypotheses, when first introduced to a new environment? The second is: When the robot has discovered a new object in its surroundings, how should it utilize this information to improve its planning abilities?

Moving problems . Again, there are two of these. The first is: Suppose the robot wants to move a simple rectangular object from



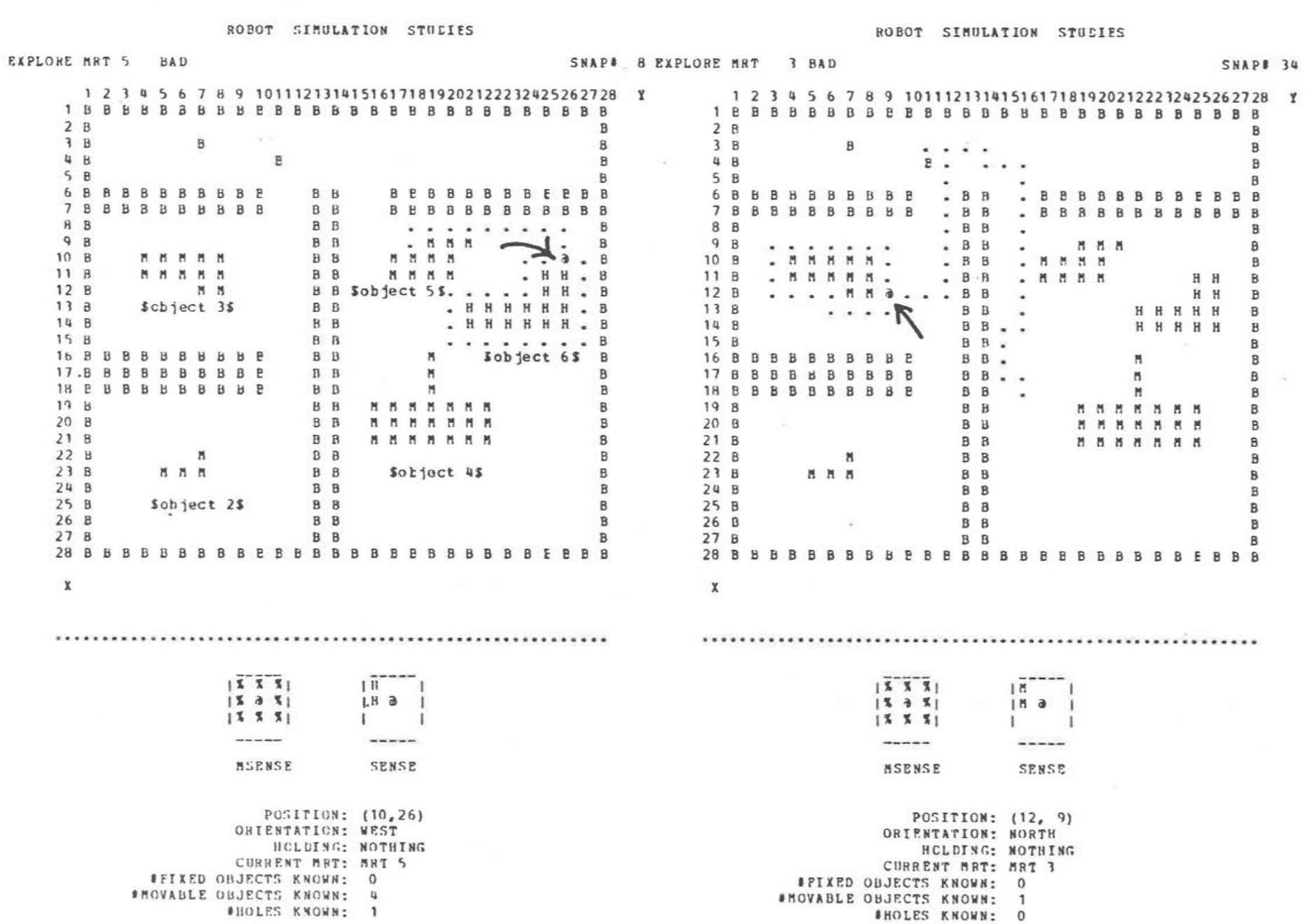
Two snapshots produced by ROSS in the course of episode A in the life of the simulated robot, Robbie. The arrow marks his current position, and the dots record his recent movements.

Figure 14.



Two snapshots produced by ROSS in the course of episode A in the life of the simulated robot, Robbie. The arrow marks his current position, and the dots record his recent movements.

Figure 15.



The right-hand snapshot was produced by ROSS in the course of episode A in the life of the simulated robot, Robbie ; the left-hand one in the course of episode B.

Figure 16.

one part of its environment to another. How should the robot plan its actions beforehand, and how should the robot represent its world when in the middle of executing such a plan of action? The second moving problem is the "furniture moving problem": like the first, except that instead of a simple rectangular object an object of complex shape is allowed. A solution of the second may consist of an easy extension to a solution of the first, by the use of standard problem-solving techniques.

Concept formation . To what extent could a robot be programmed to learn the concept of a rectangular space (MRT), the action procedures associated with it (for example, the procedure LINEAR), and how to relate rectangular spaces to one another for the purpose of moving from place to place? This is related to psychological questions concerning the development of the infant.

Analysis and comparison of rectanguloid shapes . Analyze shapes such as those in figure 8 in ways which will be useful to the robot in reasoning about its world. Prove or disprove that the algorithm DECOMP is efficient at parsing a rectanguloid shape into its maximal subrectangles. Can the algorithm CONTAIN , for comparing rectanguloid shapes for containment, be substantially improved, or else is it the case that containment is an inherently complex operation?

4.2 Future work

Work is proceeding on the exploring and moving problems. Beyond that, there are a whole host of ways in which we might generalize the system. For instance, by extending the rectangular world to three dimensions. In the more immediate future, we expect to incorporate a simple form of vision. Of course, in order to cope with any such generalization, the design of the robot's computational nervous system will have to be improved.

Bibliography

1. Aida, S., Cordella, L.,and Ivacevic, N. "Visual-tactile symbiotic system for stereometric pattern recognition". Second International Joint Conference on Artificial Intelligence , British Computer Society (1971), pp.365-369.

2. Barrow,H.G., and Salter, S.H. "Design of low-cost equipment for cognitive robot research". Machine Intelligence 5 , edited by B. Meltzer & D. Michie, American Elsevier Publishing Company, Inc., N.Y.(1970), pp.555-566.

3. Hjiri,M., Uno,T., Yoda,H., Goto,T., and Takeyasu,K. "An intelligent robot with cognition and decision-making ability". Second International Joint Conference on Artificial Intelligence , British Computer Society (1971) , pp. 350-353.

4. Ernst, George W., and Newell, Allen. GPS : A Case Study in Generality and Problem Solving . Academic Press, New York, 1969.

5. Feldman, J.A., et al. "The Stanford hand-eye project". Proceedings of the International Joint Conference on Artificial Intelligence , Washington (May 1969), pp. 350-353.

6. Feldman, J.A., et al. "The use of vision and manipulation to solve the 'instant insanity' puzzle". Second International Joint Conference on Artificial Intelligence , British Computer Society (1971) , pp.359-364.

7. Fikes, Richard E., and Nilsson, Nils J. "STRIPS : a new approach to the application of theorem proving to problem solving". Second International Joint Conference on Artificial Intelligence , British Computer Society (1971) , pp. 608-620.

8. Gregory,R.L. "The social implications of intelligent machines". Machine Intelligence 6 , edited by B. Meltzer & D. Michie, American Elsevier Publishing Company, Inc., N.Y.(1971), pp. 3-13.

9. Hayes, P.J. "A logic of actions". Machine Intelligence 6 , edited by B. Meltzer & D. Michie, American Elsevier Publishing Company, Inc., N.Y.(1971), pp. 495-520.

10. Hewitt, Carl. "Procedural embedding of knowledge in PLANNER". Second International Joint Conference on Artificial Intelligence , British Computer Society (1971), pp. 167-182.

11. Knuth, Donald E. The Art of Computer Programming , Vol. I. Addison-Wesley Publishing Company, Reading, Massachussetts, 1968, p. 262.

12. McCarthy, John. "Programs with common sense" (November 1958), and "Situations, actions, and causal laws" (1963). Reprinted together as Ch.7 of Semantic Information Processing, edited by M.L.Minsky, MIT Press(1968).

13. Nilsson, Nils J. "A mobile automaton : an application of artificial intelligence techniques". Proceedings of the International Joint Conference on Artificial Intelligence , Washington (May 1969) , pp. 509-520.

14. Pohl, Ira. "Bi-directional and heuristic search in path problems". Technical Report No. CS 136, Computer Science Department, Stanford University (May 1969).

15. Popplestone, R.J. "Freddy in toyland". Machine Intelligence 4 , edited by D. Michie & B. Meltzer, American Elsevier Publishing Company, Inc., New York (1969) , pp. 455-462.

16. Second International Joint Conference on Artificial Intelligence , British Computer Society (1971).