Cybernetics and Systems: An International Journal, 12:345–362, 1981

DETERMINISTIC AND NONDETERMINISTIC PROGRAMMING IN ROBOT SYSTEMS

GIUSEPPINA GINI
MARIA GINI
MARCO SOMALVICO

Institute of Electrotechnics and Electronics, Milan

The modern results of artificial intelligence research have provided new techniques useful for the design of more sophisticated and advanced industrial robots. The use of computer integrated assembly systems is becoming more and more important in industrial applications.

The purpose of this paper is to illustrate how with the use of an automatic problem solver it is possible to achieve the automatic emergency recovery from a failure occurring during the assembly process. An experimental computer program implementing the required problem solving activity has been written in MICROPLANNER and tested on a UNIVAC 1108 computer.

1. INTRODUCTION

The recent development of new capabilities and performances of computers, and in particular the research results achieved within artificial intelligence (7, 11) have already shown their great utility in the solution of hard problems which man has to face in different aspects of his technological progress.

Industrial robotics represents an important fact within this modern trend. An industrial robot is considered as an artificial system, capable of interacting with the external world. Its intelligent behavior is obtained by the controlling activity of an interconnected computer which has the task of continuously monitoring its functions and operations (1, 6, 8, 10).

In this paper we are concerned with the application of programmable automation to assembly operations for batch-produced, discrete-part manufacturing. These operations, which are highly labor-intensive, have been estimated to represent an important part of product manufacturing costs.

0196-9722/81/030345-18$3.25

The industrial robots for such applications are devoted to the automatic execution of a fixed sequence of elementary assembly operations, which makes up the completely assembled system. Although quite sophisticated as artificial systems, mainly in their mechanical aspect, such industrial robots have no ability in overcoming the sudden difficulties which arise when an emergency situation occurs.

During the continuous repetition of the same assembly process, sometimes a defective component part is encountered and, as a consequence, the execution of one elementary assembly operation fails. The solution of such occurrences is available only to the man who has to find out how to recover from the emergency situation, in order to start again, afterwards, the deterministic and automatic assembly process.

This paper deals with the investigation of the techniques which allow the computer to provide, automatically and independently, the solution of the above outlined emergency problem. It is shown that this new capability requires that the computer is provided with an automatic problem solver, i.e., a program which is able to automatically construct the solution of a problem from a given representation (5, 7).

In this way it is illustrated how the interaction between the theoretical efforts pursued within artificial intelligence research and the practical exigences encountered in industrial robotics can lead to interesting results. The results presented here illustrate how an intelligent robot interacts with the computer monitoring the execution of the standard assembly algorithm (deterministic programming aspect of the computer), while the automatic construction of a solution to an emergency recovery problem is provided by an automatic problem solver. The solution of an emergency recovery problem is obtained by means of a search process operating on the representation of such a problem (nondeterministic programming aspect of the computer).

The research results have been simulated on a UNIVAC 1108 computer by utilizing the MICROPLANNER interpreter (9) as a problem solver.

Whenever a defective component part is encountered during the execution of the standard assembly algorithm, the execution has to be temporarily suspended and an emergency recovery problem has to be solved.

The MICROPLANNER interpreter is then called to automatically find the solution of the emergency recovery problem, to monitor its execution, and, afterwards, to give back the control to the execution of the standard assembly program.

This work represents a part of the research activity which is being developed at the Milan Polytechnic Artificial Intelligence Project (MP-AI Project),

and which is focused in the direction of the theory of problem solving, in the experimentation and design of representation languages, and in industrial robotics.

In Section II we will present the characteristics of the mechanical assembly problem and the occurrence of emergency situations. In Section III we will discuss problem solving at the light of the interaction between deterministic and nondeterministic programming as related with industrial robotics. In Section IV we illustrate the case study of assembly process by an industrial robot, which has been proposed by Olivetti Co., and we propose its solution. In Section V we describe some characteristics of MICRO-PLANNER programming in constructing the solution of the presented case study. In the Conclusions we summarize the results which have been obtained, and we outline the directions for future research work.

II. THE MECHANICAL ASSEMBLY PROBLEM

The advent of the modern technologies of electronics, computer science, and automatic control, together with the advanced and recent results in digital systems and in mechanical trasducers and manipulators, has provided a great impact and potentiality for the automatization of industrial processes. In particular, an improved and sophisticated use of computers has enabled the design and construction of much more powerful artificial systems than the traditionally used numerical control machines.

Thus, the advent of robots and industrial robots has dramatically changed the environment and the technology in which to insert artificial systems considered as powerful tools for reducing labor and difficulty in human activities (6).

Since the beginning of the first studies and designs of intelligent robots, the problem of mechanical assembly has been considered as one of the most important, in which the potentiality of such artificial systems could be matched with the exigencies of real industrial problems.

The mechanical assembly problem arises from the exigence of assisting and possibly substituting human operators in the task of carrying on a sequence of elementary assembly operations which are necessary for making up a mechanical system composed by a given number (usually some decades) of component parts.

The solution of this exigence has been firstly achieved by means of mechanical trasducers, controlled by the man, which were able to perform only few elementary assembly operations. Thus, the whole assembly process

was organized with the use of assembly lines, capable of producing at their outputs the assemblage of a certain number of mechanical systems. Different types of simple mechanical trasducers constrained in continuously repeating the same kind of elementary assembly operation and distributed along the assembly line, were necessary to carry on the assembly process.

A recent solution of the same problem has been obtained within the use of assembly stations where a concentrated and sequential activity, based on different kinds of elementary assembly operations is performed by complex and sophisticated multipurpose mechanical trasducers.

The increasing automatization of such assembly stations requires more and more sophisticated multipurpose mechanical trasducers, namely, the programmable industrial robots. The main feature of these industrial robots, is characterized by their interaction with a computer, generally a minicomputer or even a microcomputer (possibly, a special purpose wired program computer).

The computer has the task of monitoring the execution of the sequence of elementary operations which has to be executed in order to achieve the assembly of a mechanical system.

This activity requires that the sequence of operations has been determined by the programmer, and given, in some way, to the control unit.

However, every time an emergency arises, the robot's automatic activity has to be stopped, and humans have to take over the responsibility of solving that emergency.

Because of the too many different reasons and situations in which emergencies might occur, it is not convenient to program completely all such events, and to provide the industrial robot with all the corresponding sequences of operations necessary to automatically recover from such situations.

Therefore, it seems useful to increase the intelligence of the robot by giving it the responsibility of finding with its own processing capabilities an appropriate solution whenever it might be confronted by an unpredictable emergency recovery problem.

This new ability implies that the computer which controls the industrial robot will have the capability of finding the solution to emergency recovery problems. This capability is connected with the availability of automatic problem solver programs (2).

The next section is devoted to the illustration of useful interactions

between industrial robotics and artificial intelligence for the solution of this problem.

III. DETERMINISTIC AND NONDETERMINISTIC PROGRAMMING

A robot device to be self-controlling will certainly require a problem-solving capability for planning its activity. The exigency to provide a robot with autonomous deductive capabilities is related to the availability, on the computer, of a suitable deduction language. By such a language it is possible to design a program which enables the robot to plan its actions for achieving a goal in a specified world (5).

Giving the description of the initial state, the final state and the available laws, the program generates the "plan" for the robot, i.e., finds a sequence of primitive actions which permit the transition from the initial to the final state. The sequence of physical actions which the robot must execute can correspond to this sequence.

Existing systems, such as STRIPS (4), are severely restricted in that they take a long time to produce even short and simple plans. Therefore the explicit description of the required assembly steps in their proper sequence seems a more practicable approach. In that case the robot performs the task by executing the programmed sequence of actions.

During execution the robot may encounter a variety of unforeseen circumstances, some of which will prevent successful completion of the task. Failures occur partly because sensory information is inaccurate, partly because effectors do not always perform as expected, partly because the domain in which the robot operates cannot be characterized exactly.

Our goal is to design a robot control system in such a way that it can recover from emergency situations in an intelligent manner. Robot systems with automated planners have traditionally dealt with the problem of error recovery by replanning to achieve the desired goal.

The central idea developed here is that recovery from failures can be planned with special techniques. Those techniques are based on knowledge about failures, why they occur, what can be done about them and so on. Having detected a failure the system is confronted with the problem of dealing with the unexpected event. Recovery actions can be found by determining why the failure occurred and what the outcome of an action is.

We will explain the proposed method in the context of a case study.

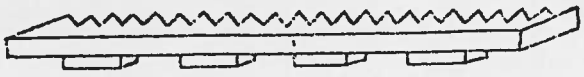


Figure 4.1 Scheme of the driver.

IV. A CASE STUDY OF ASSEMBLY PROCESS BY AN INDUSTRIAL ROBOT

In the previous sections we have illustrated the relevance in industrial robotics of mechanical assembly systems and the usefulness of the nondeterministic behavior of an industrial robot in the event of emergency situations.

In this section we are going to illustrate a real example which has oriented and motivated our research activity. The case study here illustrated has been proposed by Olivetti Company, which is beginning to insert multifunction industrial robots in its mechanical assembly lines (3).

In Figure 4.1 we show the schema of a real mechanical subsystem of a teletype which has to be assembled. The mechanical system is the driver of a teletype drum. The driver is composed of nine parts: four blocks and one bar on which the blocks must be fixed by four screws, in correspondence with four holes in the bar.

In Figure 4.2 we show the different component parts of the driver.

The assembly task is performed by a computer-controlled robot, which is made up by a mechanical arm whose hand is used to pick up a component part from its loader, and to fetch it in the appropriate position of the assembly platform. Moreover the hand is used to screw together one block with the bar, by means of one screw in the corresponding hole. In the sequel we wil simply mention the hand, without making any distinction on which of the two indicated actions we are referring to.

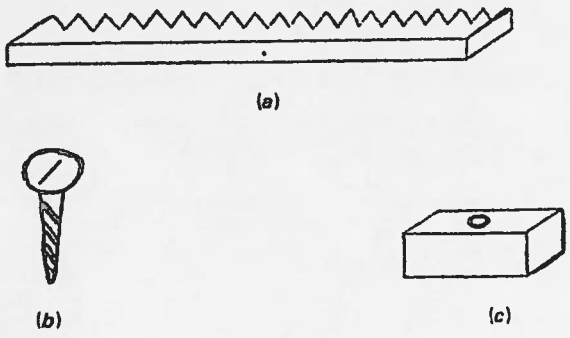


Figure 4.2 Component parts: (*a*) bar, (*b*) screw, (*c*) block.

Our investigation on the emergency situations arising in the manipulation behavior of such a robot, have been carried on a simulation of its activity on our UNIVAC 1108 computer. Thus the problems connected with a real time control of the mechanical arm have been considered only with respect to the characterization of the real environment from which arise the emergency problems to be solved automatically.

Each one of the three component parts of the driver, namely the bar, the block, and the screw, is positioned on a loader containing many instances of the same component part. The assembly process is performed on a platform from where the assembled driver is taken away.

The elementary assembly operators which belong to the robot activity are the following:

1. GRASP: Appropriate for picking up a component part or (partially or totally) assembled driver.
2. UNGRASP: Appropriate for leaving a component part or a (partially or totally) assembled driver.
3. TRANSLATE: Appropriate to the necessary translation movements among the loaders, the platform, the wastebasket, and the driver's basket.
4. SCREW: Appropriate to the mechanical connection between a block and a bar by means of a screw.
5. UNSCREW: Appropriate to the mechanical disconnection between a block and a bar previously connected by means of a screw.

In the sequel we will utilize special names for describing the activity of each operator.

We will indicate the following position names:

1. P1, P2, P3, P4, indicate the central position of each block on the assembly platform;
2. P5 = g (P1, P2, P3, P4) indicates the central position of the bar on the assembly platform;
3. L1, L2, L3 indicate the picking up position from the loaders of the blocks, bars, and screws.

After having presented an intuitive description for each operator of the robot, we shall now introduce a more formal and rigorous characterization for each one of them.

Each operator is characterized by two principal components: the preconditions, which must hold before that the operator may be applied, and the effects produced by the operator (4). The effects are represented by a set of conditions to be added to the description of the situation, and a set of conditions to be removed, because they will be no more valid.

We shall now illustrate the formal description of each one of the previously introduced operators. If the first operator is described in such a theoretical formulation, we have the following description:

1. Name: GRASP (hand grasps the object a in the position p)
   parameters: a,p
   Preconditions: IN (Hand,p), CLEAR (Hand), OBJECT (a), IN (a,p)
   effects:
   add list: CARRYING (a)
   delete list: CLEAR (Hand), IN (a,p)

The operator may be applied if the preconditions are satisfied, i.e., if an object a can be found in the position p, and if the robot's hand is clear and positioned in p.

When the operator is applied, the relationships CLEAR (hand) and IN (a,p), are removed, and CARRYING (a) is added to the description of the actual state.

In the same way we will describe the remaining operators.

2. name: UNGRASP (object a is left in p)
   parameters: a,p
   preconditions: OBJECT (a), IN (Hand,p), CARRYING (a)
   effects:
   add list: CLEAR (Hand), IN (a,p)
   delete list: CARRYING (a)
   This operator is just the reverse of the operator GRASP.
3. Name: TRANSLATE (there is a translation from p to q)
   parameters: p,q
   preconditions: IN (Hand,p)
   effects:
   add list: IN (Hand, q)
   delete list: IN (Hand, p)
4. Name: SCREW (a screw a is screwed in the position p)
   parameters: a,p

preconditions: IN (Hand, p), CLEAR (Hand), SCREW (a), IN (a,p)
effects:
add list: SCREWED (a,p)
delete list: IN (a,p)

5. Name: UNSCREW (a screw a is unscrewed in position p)
parameters: a,p
preconditions: IN (Hand,p), CLEAR (Hand), SCREWED (a,p)
effects:
add list: IN (a,p)
delete list: SCREWED (a,p)

A state description for this world is a set of predicates which expresses the relations between objects at a certain instant.

The initial state is described by the following assertions

A1: IN (Hand, Posin)
A2: CLEAR (Hand)
A3: IN (Block 1, L1)
A4: IN (Bar 1, L2)
A5: IN (Screw 1, L3)
$A6_1$: BLOCK (Block1) . . . $A6_m$: BLOCK (Blockm)
$A7_1$: BAR (Bar1) . . . $A7_n$: BAR (Barn)
$A8_1$: SCREW (Screw1) . . . $A8_p$: SCREW (Screwp)

The assertions A1.-A5. describe the initial position and situation of the hand and of the component parts. The state can change by means of a modification of its defining assertions during the execution of the program. The assertions $A6_1$-$A8_p$ are always true; they define the characteristics of the objects.

The precondition OBJECT (a) is deduced from the implication: (BLOCK (x) OR BAR(x) OR SCREW (x)) IMPLIES OBJECT (x). Every time, Blocki, or Barj, or Screwk is picked up from its loader, the assertion

$A3_{i+1}$. IN (Blocki+1, L1) or

$A4_{j+1}$. IN (Barj+1, L2) or

$A5_{k+1}$. IN (Screwk+1, L3)

is added by a loader operator, here not defined.

Of course we have described only that part of the world which is strictly related to the robot activity and to the possible source of emergency situations. We have in fact described an open world which interacts with other parts of the world, such as the loaders, the wastebasket, and the driver's basket. We are not considering in detail the operators whose activity is not strictly internal to the specific world of that assembly.

Our representation of the world is focused on the actions that the robot can perform in its world, and not on the external world in which other independent systems can be present. The problem of correlating such systems is not considered in our exposition, because it is not relevant to our main concern, i.e., the arising of emergency problems related with the presence of defective component parts.

The solution of the previous assembly task is constituted by the sequence of operators illustrated in Figure 4.3. This sequence can be given or can be obtained automatically by a problem-solving, from the description of the final state:

BLOCK(a1), IN (a1,P1), BLOCK (a2), IN (a2, P2),
BLOCK(a3), IN (a3,P3), BLOCK (a4), IN (a4, P4),
BAR (B1), IN (b1,P5), SCREW (s1),
SCREWED (s1,P1), SCREW(s2), SCREWED (s2, P2),
SCREW(s3), SCREWED (s3,P3), SCREW (s4),
SCREWED (s4, P4)

The operators which are used to make up the solution of the normal assembly do not make use of the UNSCREW operator, which is used only when an emergency arises.

Since the problem is deterministic we may use this sequence of operators directly programmed once for all.

The assembly process previously considered is based on the assumption that all the component parts have no defects. On the other hand, in a real situation, there might possibly be some component parts which are defective; in this case the assembly sequence of operators previously illustrated cannot be carried on.

Whenever such defective component parts are encountered during an assembly process an emergency recovery has to be done. In this example, we consider only the emergency determined by the following possible reasons:

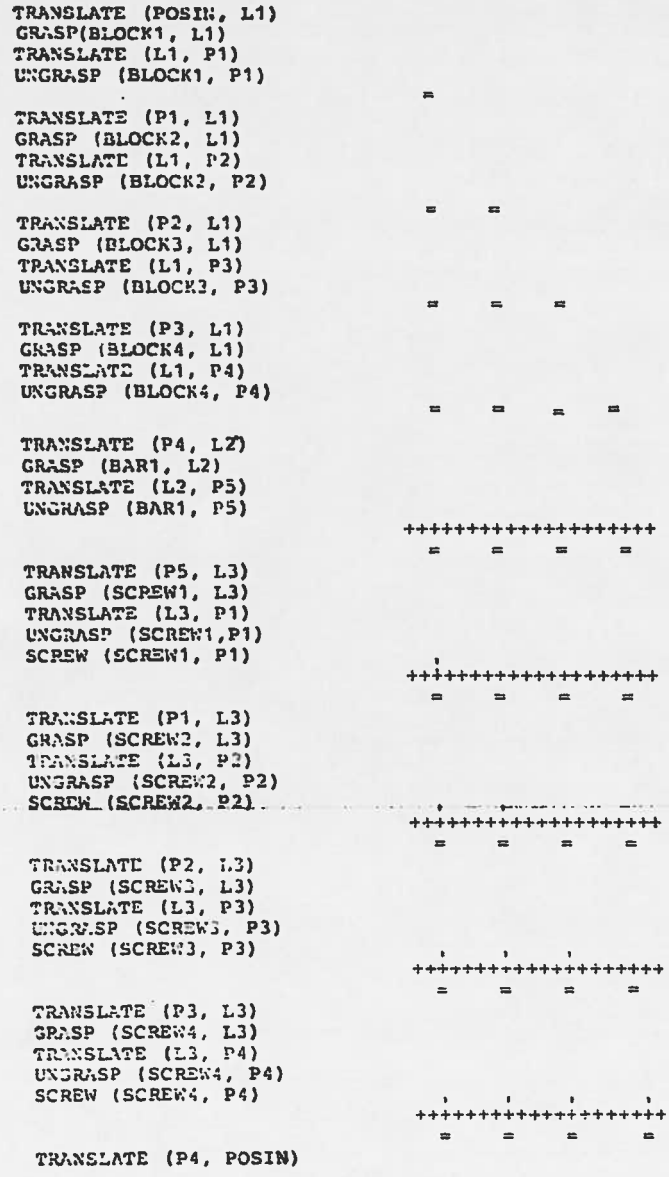


Figure 4.3 Deterministic assembly program.

1. The holes are defective (in width, filleting, or position) in the bar or in some blocks;
2. The screws are defective (in filleting, length, or width).

The arising of such an emergency problem forces the robot to interrupt the assembly during the application of the operator SCREW.

Whenever we do not have any knowledge about the reason of the emergency, we can approach the emergency recovery according to three different strategies:

1. Changing the screw;
2. Changing the block and the related screw;
3. Unscrewing all the previously assembled blocks and changing only the bar, the block, and the screw on which there was an interruption.

Each one of these strategies can be viewed as a new problem, which has to be solved by the computer, by searching a new sequence of operators to apply in the present state. After the application of those operators the emergency situation has been solved and then the normal assembly will start again from an appropriate reentry point. The reentry point in the control flow of the assembly program (deterministic program)—after the solution of the emergency problem (nondeterministic program) is dependent on the following two factors:

i. The point of the deterministic program where the emergency has arisen;
ii. The way in which the nondeterministic problem has been solved.

The way in which the emergency problem can be solved may be selected as well according to a global strategy, which takes account of additional information which might easily be obtained from the working status of the robot.

This information can be related to the emergency causes (for example, the point of depth in which the screw has been blocked); moreover additional information such as the cost of the component parts and the cost of the operations can influence the selection of the appropriate strategy.

The construction of the appropriate solution path for the emergency recovery can be obtained in a completely automatic way on the basis of such additional information. The investigation of this aspect constitutes a possible direction for further research work.

V. PROGRAMMING THE EMERGENCY RECOVERY

In the previous section we have examined an example of mechanical assembly, and we have discussed the arising of emergency problems and the automatization of their solution. In this section we examine some characteristics of goal-oriented languages as suitable languages apt to describe to the computer both the deterministic and the nondeterministic part of the program.

We will make use as goal-oriented language of the MICROPLANNER (9) language, which runs on our UNIVAC 1108 computer and which has been used for control of intelligent robots (5). Moreover we will illustrate some preliminary experimental results.

We are going now to show how we can easily translate the descriptions given in the previous section for each operator into suitable procedures of the MICROPLANNER language. The activation and the execution of these procedures provides the construction of the solution of the problem.

So MICROPLANNER can be viewed as an excellent tool for experimenting the described emergency recovery method. It is important to notice that our attempts have been oriented toward demonstrating the feasibility of the proposed method, and little attention has been paid to questions of computational cost.

We are now going to illustrate, as an example, the description of the MICROPLANNER translation of the operator GRASP. Comments which illustrate such a translation are presented as well.

```
(PUT 'GRASP 'THEOREM '(THCONSE (OBJ POS)
    (CARRYING $-OBJ)
(THGOAL (OBJECT $-OBJ)
(THGOAL (IN $-OBJ $-POS))
(THGOAL (CLEAR HAND))
(THGOAL (IN HAND $-POS) $T)
(THERASE (IN $-OBJ $-POS))
(THERASE (CLEAR HAND))
(THASSERT (CARRYING $-OBJ)) ))
```

This type of theorem is characterized by a pattern, namely (CARRYING $-OBJ) which expresses the result of the application of the theorem, and by a sequence of instructions. The theorem, is written in a way such that "the body implies the pattern."

If we want to demonstrate the truth of the pattern we must demonstrate,

i.e., execute successfully, the body of the theorem. Every step is an instruction, whose evaluation gives "success" or "failure," as result.

The accomplishment of successive deductions is provided because it is possible with those instructions to call other theorems, and, thus, to set up a chain of deduction steps. When the problem to be solved is expressed for example by the goal:

```
(THGOAL (CARRYING BLOCK1) $T)
```

the system sets up a search among the assertions and successively, among the theorems, which extract, by pattern matching, the appropriate element of the problem base. If there are different assertions or theorems whose pattern matches the one of the goal, the system makes an arbitrary choice, backing up and trying another automatically, whenever the selected one leads to a failure.

We can make the following observations which compare the theoretical description illustrated in Section IV with the MICROPLANNER program.

1. The parameters of the operators become the variables of the theorem; the theorem can have other variables as well. The variables are indicated by the prefix $.

2. The pattern indicates that the operator GRASP should be used only in order to achieve the goal (CARRYING $-OBJ). The preconditions of the operators are satisfied by the goals:

```
(OBJECT $-OBJ)      (IN $-OBJ $-POS)
(CLEAR HAND)        (IN HAND $-POS)
```

Every goal can be obtained by searching among the assertions or by calling another procedure (the form $T enables us to call a procedure whose pattern matches the assigned one).

3. The add list and delete list are translated in the Thassert and Therase form.

4. The order of the preconditions is a way of controlling the language and it is very important.

5. The set theoretical description of the operators does not include any indication about the strategy to be used in order to apply an operator. In the MICROPLANNER language the program includes the following components: the order in which the elements of the precondition set must be satisfied, the task for which the theorem can be used (i.e., the pattern), a

backtracking monitor which will consider the different choices available to the system.

The existence and the position of the blocks, bars, and screws in their loaders is simulated by MICROPLANNER assertions like

(BLOCK1 IN L1), (BLOCK BLOCK1)
(BAR3 IN L2), (BAR BAR3)
(SCREW7 IN L3), (SCREW SCREW7)

Every time an object is picked up from its loader an appropriate theorem is activated to generate a new assertion for a new object of the same type. In that way it is possible to simulate the continuous presence of a component part in the loader.

There is the possibility of avoiding the enumeration of the objects in the system. That is an advantage deriving from the use of MICROPLANNER instead of a formal language, like predicate calculus. The presence of apparently contradictory assertions, like

(BLOCK IN L1)

and

(BLOCK IN P1)

does not create any difficulties because it is controlled by the program itself.

Some suitable counters can be introduced. They are incremented every time a component part is taken from the loaders. When there are no more component parts the system stops the assembly process. Moreover in every moment it is possible to check the number of the component parts employed in the assembly or rejected.

Given a description of the final state, the solution normally proceeds in a *top-down* or goal-oriented way. It reduces problems to subproblems, with the objective of reducing the original problem to a set of solved subproblems, which are the assertions.

The possibility of *bottom-up* behavior is allowed. In that case new assertions are derived from the old ones with the objective of deriving a solution of the original problem.

The sequence of activations of theorems produces a *plan* for the robot. That plan consists of a list containing in their proper sequence all the required

movements of the objects. The assembly process is carried out by the MICROPLANNER program in a deterministic way and produces the deterministic assembly illustrated in the previous section, until no emergency situations arise.

In Figure 4.3 the example of this deterministic assembly is illustrated. The sequence of the applied operators is indicated in the listing and commented by some drawings. The names POSIN, WASTEBASKET, and STORE have an obvious meaning.

When an emergency arises, a particular emergency problem is activated on the basis of a prefixed strategy. In Figure 5.1 there is an example of the emergency problem arisen in screwing the screw in P4: the solution is illustrated from this point.

The example of emergency recovery problem, which has been automatically solved by the MICROPLANNER interpreter, shows that the solution of such problems is reasonably simple. Therefore such solutions can be obtained automatically by the machine, while it would have been very cumbersome for the man to go through the examination of all the possible emergency situations, and the subsequent programming of each solution.

VI. CONCLUSIONS

In this paper we have illustrated the utility of recent research results obtained within artificial intelligence investigation, mainly in problem solving and representation languages, toward the design of sophisticated intelligent robots which are capable of solving emergency recovery problems arising during their standard computer controlled activity.

The problem-solving techniques for emergency recovering are a contribution to the subject of robotics. Moreover the problem of error recovery is a member of a larger class of problems involving knowledge representation and common sense reasoning, both of which are research topics in artificial intelligence.

We have examined a case study related with emergency situations and recovery problems originating by the presence of defective component parts during an automatic mechanical assembly. The research results presented in the paper are part of the activity of the Milan Polytechnic Artificial Intelligence Project. An experimental computer program implementing the described emergency recovery method has been written in MICROPLANNER and tested on a UNIVAC 1108 computer.

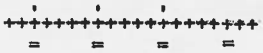
```
TRANSLATE (P3, L3)
GRASP (SCREW4, L3)
TRANSLATE (L3, P4)
UNGRASP (SCREW4, P4)
SCREW (SCREW4, P4)
     EMERCENGY PROBLEM
TRANSLATE (P4, WASTEBASKET)
UNGRASP (SCREW4, WASTEBASKET)
TRANSLATE (WASTEBASKET, L3)
GRASP (SCREW5, L3)
TRANSLATE (SCREW5, P4)
UNGRASP (SCREW5, P4)
SCREW (SCREW5, P4)
     EMERGENCY PROBLEM
TRANSLATE (P4, WASTEBASKET)
UNGRASP (SCREW5, WASTEBASKET)
TRANSLATE (WASTEBASKET, P5)
GRASP (BAR1, P5)
TRANSLATE (P5, STORE)
UNGRASP (BAR1, STORE)
```



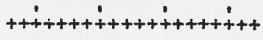
```
TRANSLATE (STORE, P4)
GRASP (BLOCK4, P4)
TRANSLATE (P4, WASTEBASKET)
UNGRASP (BLOCK4, WASTEBASKET)
TRANSLATE (WASTEBASKET, L1)
GRASP (BLOCK5, L1)
TRANSLATE (L1, P4)
UNGRASP (BLOCK5, P4)
```



```
TRANSLATE (P4, STORE)
GRASP (BAR1, STORE)
TRANSLATE (STORE, P5)
UNGRASP (BAR1, P5)
```



```
TRANSLATE (P5, L3)
GRASP (SCREW6, L3)
TRANSLATE (L3, P4)
UNGRASP (SCREW6, P4)
SCREW (SCREW6, P4)
```



```
TRANSLATE (P4, POSIN)
```

Figure 5.1 Solution of an emergency recovery problem.

The future research activity will be devoted to the study of the computational interaction between the deterministic and the nondeterministic programming, both on theoretical and on practical levels. The investigation of more complex emergency recovery problems in industrial robotics will be examined.

We would like to acknowledge the useful cooperation and interaction with Dr. D'Auria and Dr. Salmon of Olivetti Company, for having suggested

this research problem, and for having pointed out its relevance for industrial robotics.

REFERENCES

1. Barrow, H. G., and Crawford, C. F. 1972. The mark 1.5 Edinburgh robot facility. In Meltzer, B. and Michie, D. (eds.), *Machine intelligence 7*, pp. 465–480. Edinburgh: Edinburgh University Press.
2. Corti, P. L., Gini, G., Gini, M., and Somalvico, M. 1976. Problem solving and automatic emergency recovery: Towards the design of intelligent industrial robots. *Proc. 2nd CISM–IFTOMM symposium on theory and practice of robots and manipulators*, Warsaw, Poland, Sept.
3. D'Auria, A. 1977. SIGMA assembly robot application. *Proc. 7th international symposium on industrial robots*, Tokyo, Japan, Oct.
4. Fikes, R. E., and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artif. Intell.* 2:189–208.
5. Gini, G., and Gini, M. 1975. Control of intelligent robots and goal-oriented languages. *Ind. Rob.* 2(2).
6. International Fluidics 72. 1972. *Industrial robots—A survey*. International Fluidics Services Ltd., Bedford, England.
7. Nilsson, N. J. 1971. *Problem-solving methods in artificial intelligence*. New York: McGraw-Hill.
8. Nitzan, D., and Rosen, C. 1976. Programmable industrial automation. *IEEE Trans. Comput.*
9. Sussman, G. J., Winograd, T., and Charniak, E. 1970. *MICROPLANNER reference manual*. AI TR 203, M.I.T., Cambridge, Massachusetts.
10. Will, P. M., and Grossman, D. G. 1975. An experimental system for computer controlled mechanical assembly. *IEEE Trans. Comput.* C-24(9).
11. Winston, P. H. 1977. *Artificial intelligence*. Reading, Mass.: Addison-Wesley.

Received June 1981

Request reprints from Marco Somalvico, Institute of Electrotechnics and Electronics, Milan Polytechnic, Piazza Leonardo da Vinci, 32, I-20133, Milan, Italy.