ROBOT PLANNING SYSTEM BASED ON PROBLEM SOLVERS

Tadashi Nagata, Masato Yamazaki and Michiharu Tsukamoto
Electrotechnical Laboratory
Tokyo, Japan

ABSTRACT

This paper is concerned with a planning system for a robot with a hand and an eye which can manipulate blocks. Two different kinds of problem solvers are used. One is mainly composed of a theorem prover based on the resolution principle, and the other consists of a theorem prover based on pattern matching. These problem solvers are called GOAL-FINDER and JOB-SCHEDULER corresponding to their functions. GOAL-FINDER decides a goal state that is suitable for an order of an operator. JOB-SCHEDULER produces a job sequence for a robot to perform the goal state given the constraints of the block world.

Description terms

robot planning, problem solving, theorem proving and data structure.

1. Introduction

Papers ' ' ' have been presented hitherto concerning robot planning systems that understand instructions from an operator and which make plans for attaining goals.

This paper describes a robot planning system developed for use in a robot. This system is concerned with a robot with a hand and an eye which manipulates building-blocks in response to the orders of an operator.

Two different kinds of theorem provers are used to demonstrate the effectiveness of this planning system. One is based on the resolution principle, the other on pattern matching. The former is used in a problem solver which understands the meaning of a statement and which decides a goal state corresponding to the statement. The latter is applied to a problem solver which effects a concrete job sequence to realize the goal state described above. These problem solvers are called GOAL-FINDER and JOB-SCHEDULER respectively.

The reason why two theorem provers are used in a robot planning system is that it is not necessarily profitable to process two operations, for example that of representing a state and of obtaining a job sequence, with only one method.

A further feature of this planning system is that the procedural data structure is used for increasing efficiency in problem solving processes. An outline of the robot planning system, descriptions of each of the problem solvers and future considerations are given in the following chapters.

The problem solvers and the robot planning system discussed in this paper would undoubtedly be applicable in many fields.

2. The outlines of the robot planning system

The following problem is presented as an example of the working of a robot planning system. There are many blocks, *of* various shapes and colours, and working desks upon which a robot manipulates the blocks. The robot is made to build structures, such as a tower, a house, a bridge, etc., in response to task instruction given by an operator to the robot in a resticted form as follows;

"BUILD A TOWER"

"BUILD A HOUSE WITH A RED ROOF", etc.

The robot attempts to determine the meanings of words or phrases, such as "a tower", "a house" and "with a red roof", from its knowledge. If it fails, it asks the operator to explain the meanings. The meanings of the concepts, "tower", "house" and "roof", are given to the robot as axioms in the form of first-order predicate calculus wffs. Interpreting the statement, the robot decides on a suitable state-representation for a block building. This state-representation is called a goal state and is shown in the form of "(ON A (ON C D)V, where (ON X Y) means "X is on Y".

The theorem prover based on the resolution principle and on the functional data structure is used as a main tool in this part of the robot planning system. This part in the system is called "GOAL-FINDER". Finding the goal state suitable for the statement, the robot must define a job sequence for constructing the building. That is, a robot roust decide the procedure to realize the goal state under some constraints. A theorem prover based on pattern matching is applied to this process. The theorem prover that constitutes this part of the system is called "JOB-SCHEDULER".

Taking the goal state and the present state, JOB-SCHEDULER decides a job sequence given the constraints of the block world.

A diagram of the robot planning system is shown in Fig. 1, The details of GOAL-FINDER and JOB-SCHEDULER are described in chapters 3 and 4.

The relationships between the blocks manipulated by a robot and the characteristics of the blocks (ex. shape, colour, position, etc.) are represented by FDS*. FDS forms a network and blocks with similar characteristics are connected. Therefore, the suitable problem domain may be effectively established by the problem solver. For example, the problem domain composed of only red blocks can be extracted from the block world in response to the statement.

3. An input interpreter and GOAL-FIKDER

In this chapter, we shall describe an interpreter which construes input statements given by an operator and a state-oriented problem solver, GOAL-FINDER, which finds a goal state to fulfill the statements.

GOAL-FINDER is made up of three major parts, a world model, a storage, which

* FDS is an abbreviation of functional data structure.

memorizes knowledge of concepts of block buildings, and the resolution theorem prover. This theorem prover is similar to QA3[5].

3.1 An input interpreter and a world model

The main functions of this interpreter are to translate statements into list forms and to deliver them to GOAL-FINDER. In the present stage, task statements given by an operator are restricted to the following two types for the sake of simplicity.

BUILD adjective noun with adjective noun

MOVE adjective noun to P.

(where each adjective word or clause may be missing, and P represents the place where objects should be moved.)
For example,

"BUILD A TALL TOVER VITH A RED PILLAR"

is transformed into

BUILD (TOWER TALL (PILLAR RED)),

and such a statement as

"MOVE BLUE BLOCK TO PLACE-A"

yi elds

MOVE PLACE-A (BLOCK BLUE).

In addition to imperative statements, this interpreter *construes two* types of declarative sentences written in the form of well formed formulas (wffs) in the predicate calculus. One type is a statement for changing the world model and the other is for changing knowledge. The interpreter also prints out the message when the planning system fails to perform its function, and requires further instructions.

FDS is employed to describe such world models as those described by shape, name, colour, position and spatial relationships of the blocks. One feature of this program is its abili ty to make up an associative network of current information in the world.

Because of this structure, all features of the block can be seen easily, and blocks having specific features can be selected.

3.2 Algorithms used in a problem solver which finds goal states

Here, ve shall discuss the algorithms used in this problem solver, and explain how it finds a goal state which is suitable for task statement. An outline of the problem solving process is illustrated in Fig. 2. In explanation, let us briefly outline its processes.

First, the problem solver generates theorems to be proved corresponding to the statement, and

second it chooses the axioms which are necessary to prove the theorems and sets them up in each problem domain.

Thirdly it selects appropriate blocks from a world model and *sets them* up in each problem domain .

In conclusion, it proves theorems and produces a solution, i.e. a goal state.

These algorithms have the following three notable characteristics.

(1) Because axioms are constructed in a tree structure, the meanings of statements are reflected systematically in the theorem proving process.

(2) Some adjective words in a statement are reflected in the process of selecting axioms. For example, an additional axiom which can produce tall building will be selected to build a "TALL TOVER" rather than an ordinary TOWER.

(3) Other adjective words are reflected in the processes by *which* blocks are selected. For example, only blue blocks are selected from a world model to build a "BLUE HOUSE".

Let us briefly describe the knowledge shown in Fig.. 2. The knowledge, we discuss here, involves the axioms, which the theorem prover is to use, and the generalized solutions, where the generalized solution means a wffs which has been obtained by replacing constants in a solution by the corresponding predicate letters.

For example, suppose a solution is "(ON A B)" and the following descriptions are given;

PRISM (A) and BLOCK (B).

"(ON PRISM BLOCK)" is stored as a generalized solution and utilized in subsequent processes. Each piece of knowledge concerning an axiom consists of the name, the level, the corresponding adjective and the axiom itself and is stored in the form of list structures.

The axioms are given to the problem solver (GOAL-FINDER) in the form of predicate calculus wffs. Each axiom may or may not have leveling marks to represent the concepts, for example, "HOUSE", "TOVER" etc., in the tree structures.

Using the axioms with the leveling marks, efficiency of procedure in GOAL-PINDER can be greatly increased.

In this case, the theorem prover interpretes thie tree structure of a concept and introduces a solution containing the real obj ects.

As further explanations, ve shall *use* a simple example.

[Example 1]
Suppose the task statement i s

"BUILD A TALL TOWER VITH A BLUE PILLAR."

and the world model is as shown in Fig. 3-(2). In this example, the following two theorems to be proved are induced;

(3X) TOVER (X)

(3X) PILLAR (X).

The axioms are gathered in the following manner;

TOVER; level 0 (VxVY) (((ROOF X)(PILLAR Y))

;> (TOVER (ON X Y)))*,

ROOF; level] (^{V}X)((PRISM X) 2> (ROOFX)),

PILLAR; level 1 (^{V}X)((BRICK X)2(PILLAR X)),

and

PILLAR: level 1 tall $(\forall X \forall Y)$ (((PILLAR X) (PILLAR Y)) $\supset$ (PILLAR (ON X Y)))*

The last axiom is selected because there exists an adjective word TALL in the instruction.

The tree structure of axioms and theorems is shown in Pig, 4.

In this figure, the adjective TALL belongs to the TOWER but it actually has an effect on the axiom of PILLAR with level 1. In addition, the adjective BLUE has an *effect* on the process of setting up the problem domain. That is, GOAL-FINDER sets up the world of only blue blocks as the solution thus -

PILLAR (ON A C).

This is substituted into the axiom of TOWER using only PRISMs.

The solution obtained is as follows;

TOWER (ON E (ON A C)).

GOAL-FINDER delivers the following expression;

{ON E (ON A C)),

to JOB-SCHEDULER.(Fig- 3-(3))

4. JOB-SCHEDULER

Suppose we supply the robot with the world model, the initial state and the goal state, and ask it to decide a sequence of actions to achieve the goal state. There are, of course, many constraints in the world of the robot. These constraints can be classified into the following two categories;

(1) actions——restrictions which are invariably accompanied by an action, and

(2) states——limitations which are provided by the world.

In this chapter a problem solver, JOB-SCHEDULER, is used to produce a sequence of actions to attain the goal state.

4.1 Theorem prover used in JOB-SCHEDULER

A PLANNNER-like theorem prover[6] is used to solve a job sheduling problem in order to obtain an action-sequence which will arrive at a goal state from a current state whilst under some constraints. Because of the flexibility of describing algorithms, two kinds of theorems are used in this theorem prover. They are as follows:

(I) A theorem of which the result is described as a state when the action has a clear effect on the world. For example, such an action as "Put X on Y" creates a state "X is on *J"*——j the pair of the action and the state becomes a general knowledge whereby the action corresponding to a state is directly obtained; and

(2) A theorem which describes procedures; this theorem gives the procedure of execution, so that it is suitable for the description of an algorithm. As far as the control of execution is concerned, the constraint is described (1) in a theorem, as a precondition peculiar to the robot system. Or the constraint is described (2) as a variable condition to be referred according to the situations.

When a conditional expression in a theorem is not satisfied, the control of the program may be changed according to the indication for "F", If the indication is the name of a label used in the same theorem, the control becomes a simple jump like the "GO TO" statement of the FORTRAN, On the other hand, if it is not, it will be interpreted as a branch of a new theorem and the following behaviours processed. The directed process is peformed and then the control returns to the branching point. If the conditional expression is still not satisfied, the control returns to last decision point and a new decision is made.

4.2 Basic algorithm of JOB-SCHEDULER

The motivation for job scheduling is the difference between the current state (S_c) and the goal state (Sg). Abstracting a new state, JOB-SCHEDULER sets up a subgoal. That is, within the basic framework JOB-SCHEDULER operates in a GPS-like manner[7]. Let us outline the algorithm of JOB-SCHEDULER:

(1) Transform the goal state (*Sg*) produced in GOAL-FINDER and the world model (Sp) in FDS into prefix notations.

(2) Set up restrictions to be duly considered in the process of the problem solving (job scheduling) and give effective means of settling trouble, for example, such a constraint as "more blocks than two cannot be put on the block A" is given in this theorem prover as follows;

```
(ADVICE (RANGE (?X ?Y)
        (ALL ?Y)
        (COND (AND ((ON ?X A) :F SUCCESS)
        ((ON ?Y ?X) :T FAIL F SUCCESS))) )),
```

where "SUCCESS" and "FAIL" mean that this theorem ends in success and in failure respectively.

(3) If the current state (S_c) coincides with the goal state (Sg), the algorithm terminates.

(4) If there is any state, not previously selected, in the differences between S_c and Sg, select a new subgoal (S_s) out of the differences. Otherwise, if it is impossible to select a new subgoal, backtrack to the preceding subgoal and reselect the other subgoal. If it is impossible, the algorithm terminates in unsolvability.

(5) Try to get to S_a• If this fails, return to step (4). Otherwise, return to step (3).

* In these expressions existential quantifiers had already been eliminated and the Skolem functions which represent the meanings of these quantifiers introduced.

4.3 Algorithm to select a subgoal

Let us define the state (S_d) as the following equation;

$$S_d = S_g - (S_c \wedge S_g) . \quad (1)$$

This represents the differences between the current state (S_c) and the goal state (S_g). Suppose a subgoal (S_s) is defined according to a criterion, i.e.

$$S_s \in S_d . \quad (2)$$

An understanding of this algorithm is greatly aided by an simple example as follows: Suppose S_c and S_g are given as shown in Fig. 5-(1) and 5-(2) respectively. Representing S_c and S_g in the form of the prefix notation, we obtain the following;

S_c : (ON A 1), (ON B A), (ON C B) and (ON D 2)

S_g : (ON A 3), (ON B A), and (ON D B)

Moreover, from the equation (1) we get

S_d : (ON A 3) and (ON D B).

We give an order of priority for each element of S_d in selecting a subgoal. For example, blocks must be succesively laid upon one another from bottom to top, therefore, (ON A 3) is given the priority over (ON D B) in this example. (see Fig. 5-(4)). In fact, whether or not "?X" in (ON ?X ?Y) really exists at the suitable position in the goal state is checked, and then only elements suitable for this condition are selected as possibilities for the subgoal. Lastly, the order of priority of these possibilities is given to decide a subgoal.

In selection of a subgoal it sometimes happens that the same action is repeated and the goal state is not achieved. A special means of avoiding this inconvenience is devised in JOB-SCHEDULER as shown briefly in the next section.

4.4 Procedure to achieve a subgoal

In *JOB-SCHEDULER*, when such a subgoal as "(ON ?X ?Y)" is selected, the theorem "PUTON" is implicitly called to realize the state "(ON ?X ?Y)".

The remainder of this section is a brief description of the representative theorems used to attain subgoals.

PUTON [?X ; ?Y] : Putting ?X on ?Y, ?X must have no blocks on it, as it is picked up in the first place. Moreover, this applies to ?Y, too. After execution of this theorem, the world model varies.

TOPLESS [?X ; ?Y]* : If there is no blocks on ?X, it can be said that ?X is topless. If not, blocks on ?X must be removed to places other than ?Y.

REMOVE [?X ; ?Y]* : Removing ?X to any other place except ?Y, a space has to be found and then ?X is put there.

SEARCHPLACE 1 [?U ; ?X]* : This theorem finds out a block other than ?X on which other blocks can be put.

SEARCHPLACE 2 [?U ; ?X] * : This provides a new intermediate desk other than ?X.

By using these theorems, a subgoal can be selected in JOB-SCHEDULER without the repetition of the same action. Because a prohibitive position can be indicated in theorems.

As an example, we shall take up the same task as shown in [Example 1] of 3.2.

[Example 2]

Suppose the goal state is a block building on the desk W_0 shown in Fig. 3-(3), i.e. it is

(ON E A), (ON A C), and (ON C W_0),

the initial state is given in Fig. 3-(1), that is,

(ON E B), (ON B D), (ON D C), (ON A $W_{i,1}$), and (ON C $W_{i,2}$),

and the intermediate desks are

W_0, W_1, W_2, W_3, and W_4.

We give JOB-SCHEDULER such an advice that more blocks than three cannot be put on an intermediate desk;

```
(ADVICE (RANGE (?X ?Y ?Z ?U ?V)
        (ALL ?X ?Y ?Z)
        (<== ?V DESKS)
      A (<== (?U ?V) ?V)
        (COND ((EQUAL ?U NIL): S SUCCESS F B))
      B (COND (AND ((ON ?X ?U): F A)
                   ((ON ?Y ?X): F A)
                   ((ON ?Z ?Y): S FAIL F A))) )).
```

Using the preceding theorems as shown in Appendix, JOB-SCHEDULER produces a sequence of macro actions as follows;

(PUTON E W_1), (PUTON D W_2), (PUTON C W_0), (PUTON B D), (PUTON A C), and (PUTON E A).

The transitions of states are shown in Fig. 6.

Hereafter theorems of category 1 (actions) are intended to be partly learned by a robot itself in executing tasks, and it will be able to perform tasks more intelligently.

4. Conclusion and future considerations

The robot planning system developed by the robotics group in the Electrotechnical Laboratory has been described in this paper. The features of this system are as follows:

(1) A theorem prover based on the resolution principle is used to find the goal state

---

* We can also use theorems with fewer arguments than shown in the preceding descriptions. For example, "TOPLESS [?X]" means that an object on "?X" can be put on any other place in the world without restriction.

of the problem. Another theorem prover based on pattern matching is applied to obtain a job sequence with vhich the goal state is achieved. That is, the former is to interpret concepts (states), and the latter is concerned with actions. As a result, both of these theorem provers do a share of the work according to their features.

(2) The efficiency of robot planning is increased by the structuring of axioms into trees in GOAL-FINDER and the modifying the theorem prover in JOB-SCHEDULER after the suitable manner for this system.

(3) *FDS is* adopted to raise efficiency *of* the system by setting up the problem domain.

The robot planning system described here is being combined with a hardware system developed in our laboratory and integrated into a total robot system.

Acknowledgement

The authors would like to express their gratitudes to the members of the robotics group of the Electrotechnical Laboratory for their helpful discussions and criticisms.

References

1. J. H. Munson, "Robot planning, Execution, and Monitoring in an Uncertain Environment, "Proc. Int. Jt. Conf. Art. Intel., London, Sept. 1971, pp 338-349.

2. R. E. Fikes and N. J. Nilsson, "STRIPS: A New Approach to the Application of theorem Proving to Problem Solving, ibid., pp 608-620.

3. T. Vinograd, "Procedures as a Representation for Data in a Computer Program for Understanding," AI TR-17, MIT. 1971.

4. R. E. Pikes et al., "Learning and Executing Generalized Robot Plans," Artificial Intelligence 3, 1972; pp 251-288.

5. C. Green, "Theorem Proving by Resolution as a Basis for Question-Answering Systems," Machine Intelligence 4, American Elsvier, N.Y., 1969, pp 183-205.

6. C. Hewitt, "Description and Theoretical Analysis of PLANNER," AI TR-258, MIT, 1972.

7. G. ¥. Ernst and A. Newell," GPS : A Case Study in Generality and Problem Solving," ACM Monograph Series Academic Press, N.Y., 1969.

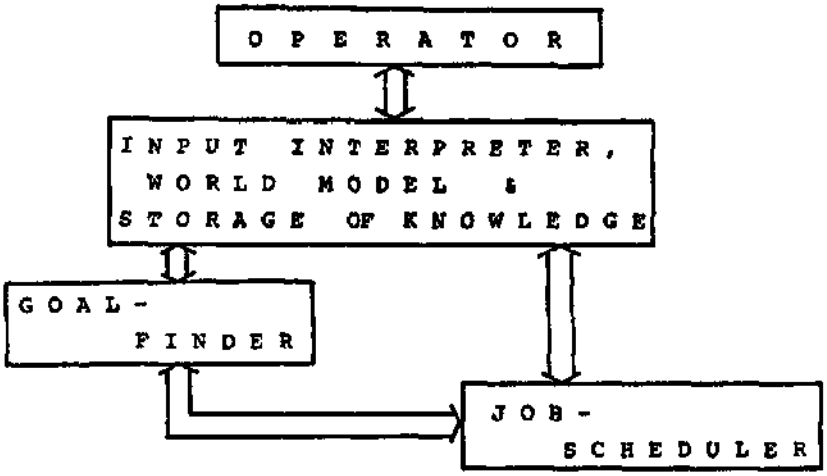

Fig.1 Flow diagram of robot planning system

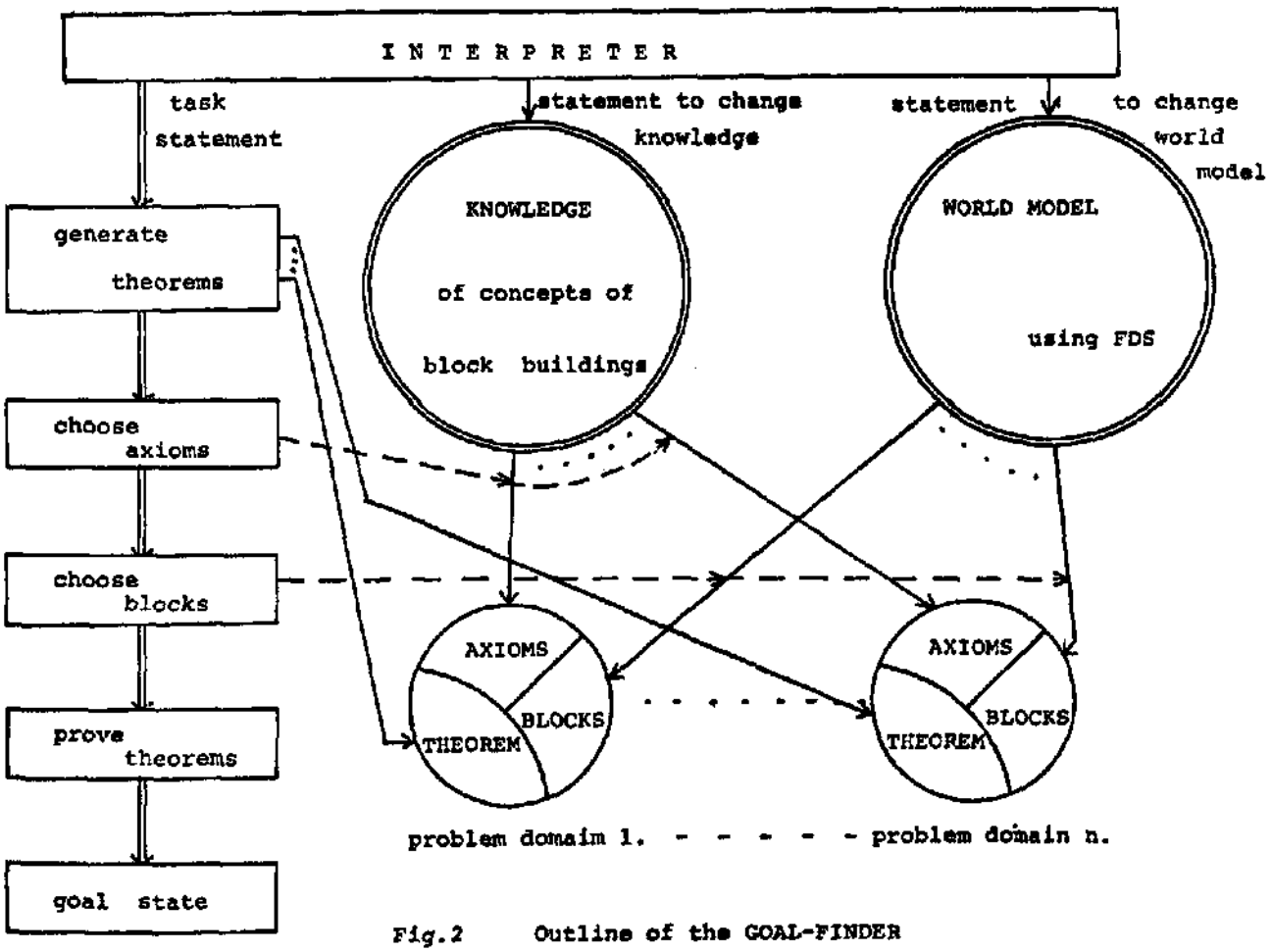

Fig.2 Outline of the GOAL-FINDER

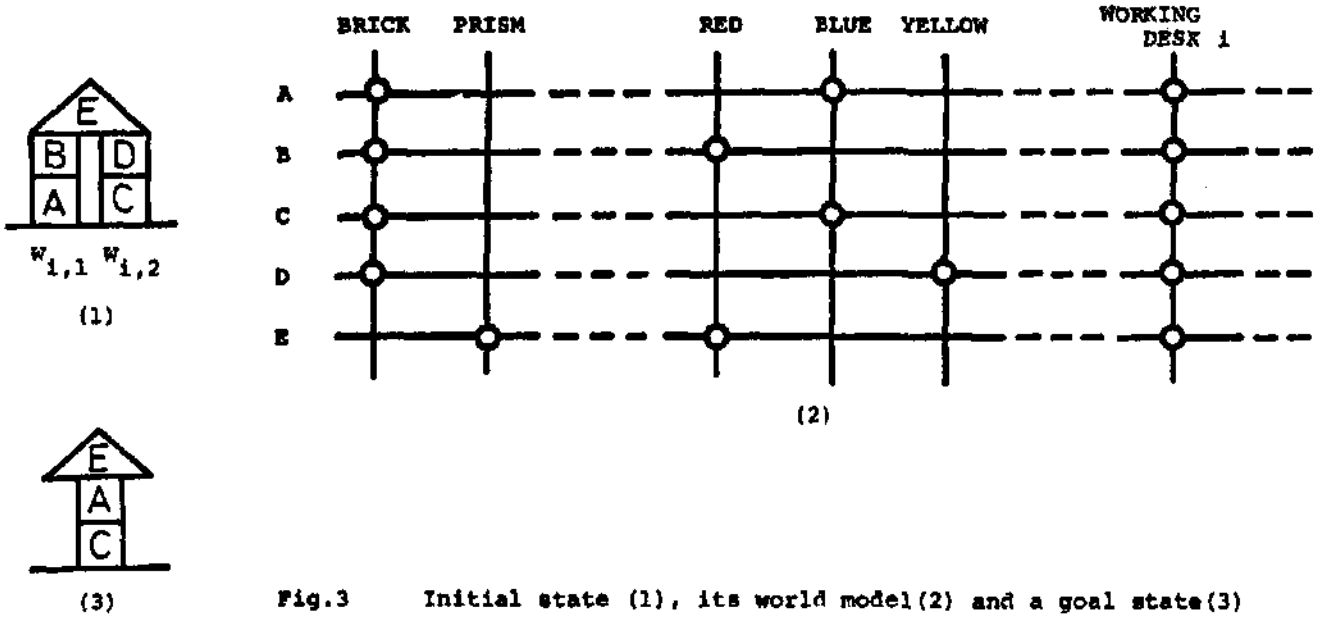

Fig.3 Initial state (1), its world model(2) and a goal state(3)

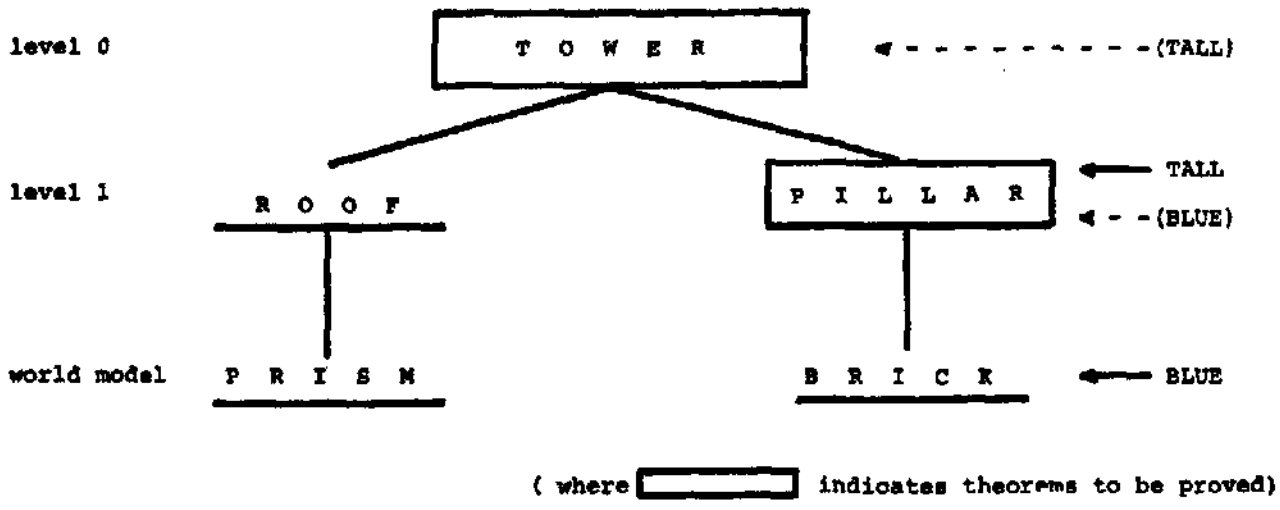

Fig.4 Tree structure of axioms and theorems

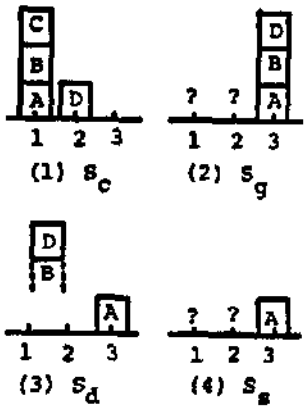

Fig.5 Example for selecting a subgoal

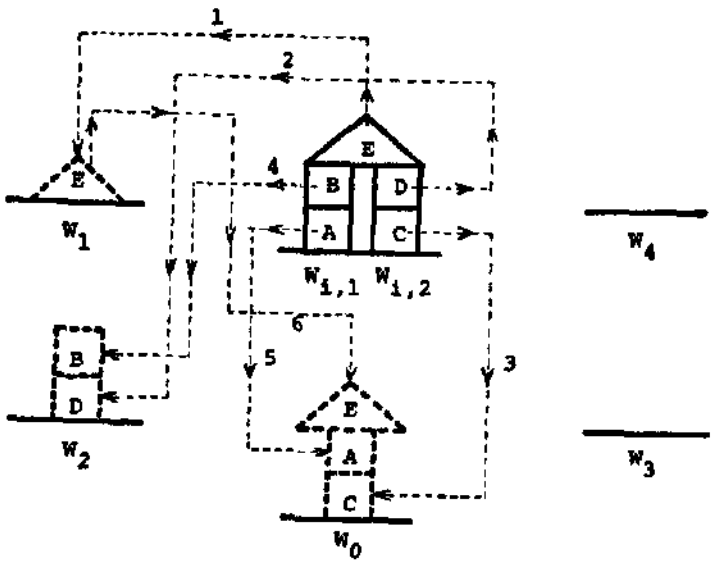

Fig.6 Transitions of states

(job sequence : 1,2,---,6)

Appendix

```
(PUTON (RANGE (?X ?Y ?U ?V ?W ?YU)
       (ALL ?V ?W)
       (<== ?YU &?Y)
       (IF   ((TOPLESS &?X) : F (TOPLESS &?X &?YU)))
REPEAT (<== (?U ?Y) &?Y)
       (IF   (AND ((TOPLESS &?U) : F (TOPLESS &?U (&?X &?YU)))
                  ((EQUAL &?Y NIL) : F REPEAT) ))
       (@ (IF   ((ON &?X ?W) : ))
          (ADD (TOPLESS &?W)))
       (@ (DELETE (ON &?X ?V)))
       (DELETE (TOPLESS &?YU))
       (ADD (ON &?X &?YU))
       (DO (ON &?X &?YU)) ))

(TOPLESS (RANGE (?X ?Y ?U ?V)
       (@ (IF   (NOT ((ON ?V &?X) : F SUC)))
          (: (REMOVE &?V &?Y)))
SUC    (ADD (TOPLESS &?X))
       (DO (TOPLESS &?X)) ))

(REMOVE (RANGE (?X ?Y ?U)
       (IF   (OR ((SEARCHPLACE1 &?U) : F (SEARCHPLACE1  ?U &?Y))
                 ((SEARCHPLACE2 &?U) : F (SEARCHPLACE2  ?U &?Y)) ))
       (: (PUTON &?X &?U)) ))

(SEARCHPLACE1 (RANGE (?U ?X ?V)
       (<== ?V ?BLOCKS)
REPEAT (<== (?U ?V) &?V)
       (IF   ((EQUAL &?U NIL) : S RETURN F NEXT))
NEXT   (IF   (AND (NOT ((MEMBER &?U &?X) : S REPEAT))
                  (NOT ((PRISM &?U) : S REPEAT))
                  ((TOPLESS &?U) : F REPEAT) ))
       (DO (SEARCHPLACE1 &?U)) ))

(SEARCHPLACE2 (RANGE (?U ?X ?V)
       (<== ?V ?DESKS)
REPEAT (<== (?U ?V) &?V)
       (IF   ((EQUAL &?U NIL) : S RETURN F NEXT))
NEXT   (IF   (AND (NOT ((MEMBER &?U &?X) : S REPEAT))
                  ((TOPLESS &?U) : F REPEAT) ))
       (DO (SEARCHPLACE2 &?U)) ))
```