Selective Learning of Macro-operators with Perfect Causality

Seiji Yamada and Saburo Tsuji

Department of Control Engineering
Faculty of Engineering Science
Osaka University
Toyonaka 560, JAPAN
E-mail : yamada%ouceai.ce.osaka-u.junet@RELAY.CS.NET

ABSTRACT

A macro-operator is an integrated operator consisting of plural primitive operators and enables a problem solver to solve more efficiently. However, if a learning system generates and saves all macro-operators extracted from worked examples, they will increase explosively and eventually its problem solving will be less efficient than even a non-learning system. Thus, it is very important for macro-operator learning to select only the effective macro-operators. To cope with this problem, we propose a new method to select macro-operators by Perfect Causality, a new heuristic, and generalization of them with EBG. Both in classical robot planning and solving algebraic equations, we made the experiments using a selective macro-learning system with Perfect Causality, a non-selectively macro-learning system and a non-learning system. The experimental results verify much higher efficiency of the selective learning system than the other two systems over a lot of various problems. Finally, we discuss Perfect Causality as an operationality criterion in EBL perspective.

1. Introduction

In general, a problem solver cannot solve problems efficiently, and various methods for solving more efficiently have been proposed. One of the significant methods is a macro-operator learning. A macro-operator is an integrated operator consisting of plural primitive operators. Since the macro-operators can reduce a search space, a problem-solver can solve more efficiently with them. MACROPS in STRIPS [Fikes et al., 1971,1972] are the first learned macro-operators and the experiments were made in a classical robot planning. However, STRIPS saves all of many macro-operators generated from worked examples, the processes of solving in past. The saved macro-operators hence explosively increase and the cost of searching for applicable ones will eventually make the problem solver less efficient than a non-learning one [Minton, 1985]. Since most candidates for macro-operators are actually useless, it is very important for the macro-operator learning to select only effective macro-operators from them. Some methods to select macro-operators have been proposed (Minton, 1985][Iba, 1985). We propose a new method with *Perfect Causality* ; a new heuristic to select only useful macro-operators. We built the frame work system; PiL2 which selectively extracts macro-operators with Perfect Causality and generalizes them with EBG method [Mitchell et al., 1986]. Both in a classical robot planning and a solving various equations, we made the experiments using the PiL2, a non-selectively macro-learning system and a non-learning system. As a result, we found PiL2 could keep the more efficiency than other two systems over a lot of various problems both in two domains.

In this paper, we first explain PiL2 system. Next, we show the definition of Perfect Causality, the algorithm for extracting macro-operators and the generalization of them with EBG. Finally, the experimental results are shown.

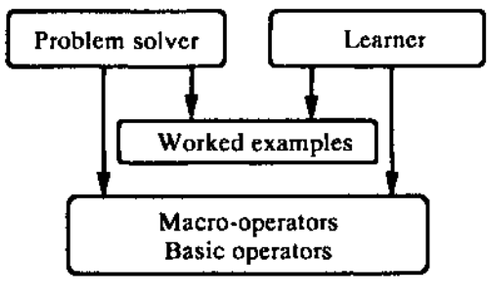


Fig.1 The structure of PiL2

2. PiL2 ; a frame work for the selectively macro-operator learning

Fig.1 shows the structure of PiL2, which consists of two

gotob(BX,RX)
{Go to object BX in room RX}
cond:[type(BX,object),
inroom(BX,RX),
inroom(robot,RX)]
delete:[nextto(robot,_)]
add:[nextto(robot,BX)]
main-effect:[nextto(robot,BX)]

gotod(DX,RX)
{Go to door DX in room RX}
[type(DX,door),
inroom(robot,RX),
connects(DX,RX,RY)]
[nextto(robot,_)]
[nextto(robot,DX)]
[nextto(robot,DX)]

pushb(BX,BY,RX)
{Push BX to object BY in room RX}
[type(BX,object),
pushable(BX),
nextto(robot,BX),
inroom(BX,RX),
inroom(BY,RX)]
[nextto(BX,_)]
[nextto(BX,BY)]
[nextto(BX,BY)]

gothrudr(DX,RX,RY)
{Go through door DX into RX from RY}
[type(DX,door),
status(DX,open),
type(RX,room),
nextto(robot,DX),
inroom(robot,RY),
connects(DX,RY,RX)]
[inroom(robot,_)]
[inroom(robot,RX)]
[inroom(robot,RX)]

Fig.2 Basic operators

modules; a problem solver and a learner. The PiL2's problem solver is STRIPS |Fikes et al., 1971]. The problem state is represented by a set of well formed formulas(wffs) in the predicate calculus and rules, and operators can transform a problem state. Given an initial state, a goal state and operators as input, the STRIPS generates an operator sequence, which can transform the initial state into the goal state. To distinguish from macro-operators, we call the operators given as input the basic operators. Fig.2 shows the basic operators, which consist of cond-lists, delete-lists, add-lists and a main- effect-lists [Fikes et al., 1971,1972). When all wffs in the cond-list are satisfied, the wffs in the delete-list will be eliminated from the current problem state and those in the add-list will be added. A main-effect-list is used for the efficient searching for relevant operators [Fikes et al., 1971,1972], which have the different wffs between the current and goal states in their add-lists.

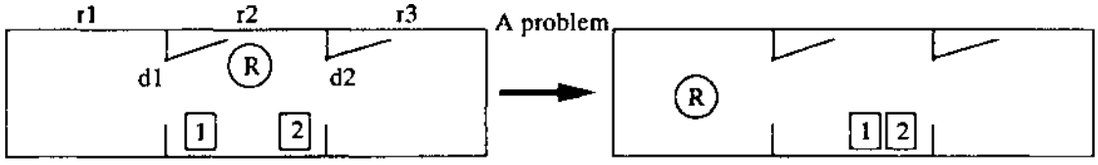
The PiL2's knowledge base consists of basic operators and macro-operators. The problem solver first searches for the relevant macro-operators and only when no relevant macro-operator is found, it searches relevant basic operators. The STRIPS uses the depth-first search and selects only one expanding node by the same heuristic to Fikes' one [Fikes et al., 1971], which evaluates the difference satisfied after the expansion. As a result, the STRIPS generates the operator sequence which can transform an initial problem state into a goal state. We call it a worked example for a learner. From a worked example, the learner extracts sub-sequences for macro-operators, generalizes, integrates and saves them.

3. Selecting macro-operators with Perfect Causality

The number of sub-sequences from a worked example with n steps is described as $M(n)=\Sigma_{k=2}^{n}nCk$. The worked example with 10 step generates M(10)=1013 sub-sequences. Therefore, if macro-operators are generated from all sub-sequences, their amount will increases explosively and the cost for searching applicable macro-operators makes the problem solving inefficient. S.Minton has reported that STRIPS which learns MACROPS non-selectively becomes less efficient than

```
INPUT(IS,OPS) {IS is initial problem state,
               OPS=[OP1•••OPn]}
Let IOP be OPS's operators applicable in IS
MOPS ← [ ]
i ← 1
WHILE i ≠ n DO BEGIN
   Let RESULT be the problem state after OPi
   was applied to IS without checking its cond-list
   MOP ← [OPi]
   j ← i+1
   WHILE j< n DO BEGIN
      IF OPj ∉ IOP
         THEN IF OPj is applicable in RESULT
                 THEN • RESULT ← RESULT OPj
                        applied
                      • assert OPj into MOP
      j ← j+1
   END
   IF MOP≠[OPi] THEN assert MOP into MOPS
   i ← i+1
END
OUTPUT:MOPS is macro-operator sequences
```

Fig.3 The algorithm of extracting macro-operators



GOAL STATE : [nextto(box1,box2),inroom(robot,r1)]

IS={ f0:type(robot,robot), f1:type(box1,object), f2:type(box2,object), f3:type(d1,door), f4:type(d2,door), f5:type(r1,room), f6:type(r2,room), f7:type(r3,room), f8:pushable(box1), f9:pushable(box2), f10:inroom(robot,r2), f11:inroom(box1,r2), f12:inroom(box2,r2), f13:status(d1,open), f14:status(d2,open), f15:connects(d1,r1,r2), f16:connects(d2,r2,r3) }

A worked example

OP1 : gotob(box1,room2) {Go to box1 in room2}
cond : [f1,f10,f11] delete : [ ] add : [f17:nextto(robot,box1)]

OP2 : pushb(box1,box2,room2) {Push box1 to box2 in room2}
cond : [f1,f8,f11,f12,f17] delete : [f17]
add : [f18:nextto(box1,box2),f19:nextto(robot,box1)]

OP3 : gotod(door1,room2) {Go to door1 in room2}
cond : [f3,f10,f15] delete : [f19] add : [f20:nextto(robot,d1)]

OP4 : gothrudr(door1,room1,room2) {Go through door1 into room1 from room2 }
cond : [f3,f5,f10,f13,f15,f20] delete : [f10,f20] add : [f21:inroom(robot,r1)]

GOAL : [f18,f21]

Fig.4 A given problem and the basic operator sequence

non-learning STRIPS even in a small number of problems [Minton, 1985]. Since the sub-operator sequences include many useless ones, selecting only the effective ones enables a learning system to keep efficiency. We propose a method to select only useful macro-operators with Perfect Causality, a new heuristic. We assume that the macro-operators are generated from only the sub-sequences which satisfy Perfect Causality. The definition of Perfect Causality is as follows.

Let a worked example be OPS=[OP1...0Pn] and an initial problem state be IS. If an arbitrary operator:OPm (m≠i) in PCOPS=[OPi...OPj] (I<i<j<n) satisfies the following two preconditions, then PCOPS satisfies Perfect Causality.

1) OPm is not applicable to IS.
2) After [OPi...OPm-I] were applied to IS, OPm is applicable to the problem state.

These preconditions mean that the applications of [OPi ...OPm-1] guarantee the application of OPm which cannot be applied to an initial problem state. From every sub-operator sequence:[OPk...OPn](I<k<n), the longest operator sequences which includes OPk and satisfies Perfect Causality are extracted for macro-operators. The algorithm for extracting macro-operators with Perfect Causality is shown in Fig.3.

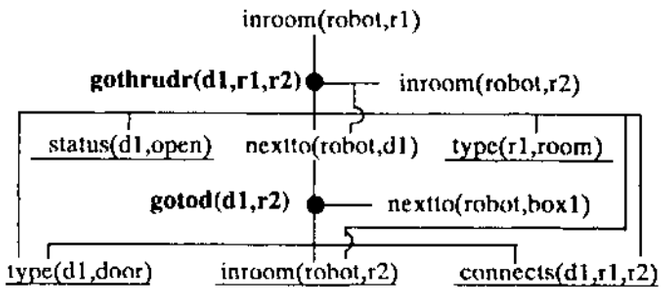
We explain how the algorithm concretely extracts the operator sequences from the worked example in Fig.4, which is cited from Fikes' paper [Fikes et al., 1972). The applied basic operators in Fig.4 are shown in Fig.2. Let OPS be [OPI,OP2, OP3.0P4]. First, [OPI,OP3] is substituted for IOP. Because all wffs in the cond-lists of OPI and OP3 are satisfied in IS. Next, OPI is applied to IS without checking its cond-list. Then fI7 is added and RESULT=[fO~f17], MOP=[OPI] are determined. Since next OP2 is not included in IOP and its cond-list, [fI,f8,fII,fI2,fI7], is satisfied in RESULT, OP2 is applied and RESULT is updated. Then MOP is also updated to [OPI,OP2]. OP3 is included in IOP and OP4 is not applicable to RESULT. Thus, this cycle with i=I is finished and MOP =IOPI,OP2] (≠ [OPI]) is extracted for a macro-operator.

Then the next cycle with i=2 begins. MOP=[OP2] is determined and OP2 is applied to IS without checking its cond-list. Since fI7 in the delete-list is not included in IS, fI7 is not removed. The fI8 and fI9 are added to IS and RESULT is updated to tfI~fI6,fI8,fI9]. Next, OP3 included in IOP is skipped and the applicability of OP4 is investigated. The f20 in the cond-list of OP4 is not in the problem state because of non-application of OP3. Thus, OP4 is not applied and this cycle finishes as MOP=[OP2]. This MOP=[OP2] cannot satisfy the precondition: MOP≠[OPi], thus this cycle dose not yield any macro-operator.

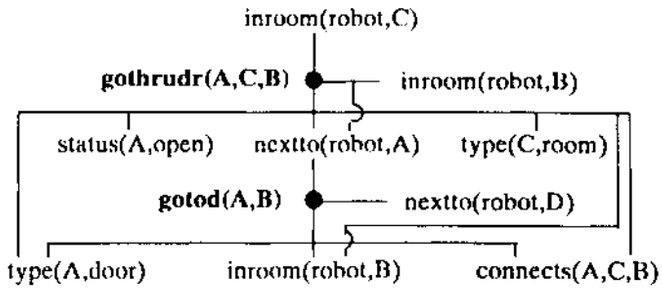
Finally, the output: MOP=[[OPI:gotob,OP2:pushb],[OP3 :gotod,OP4:gothrudr]] is obtained. As seeing from this result, Perfect Causality can extract the operator sequences which are executive independently. Fig.5 shows all suboperator sequences from the worked example in Fig.4. The M3,M6 are not executive in any problem state and M4,M9 are nonsense. Therefore, most these candidates are useless and only useful

M1:[gotob(OB1,R1), pushb(OB1,OB2,R1)]
M2:[gotod(D1,R1), gothrudr(D1,R2,R1)]
M3:[gotob(OB1,R1), gothrudr(D1,R2,R1)]
M4:[gotob(OB1,R1), gotod(D1,R1)]
M5:[pushb(OB1,OB2,R1), gotod(D1,R1)]
M6:[pushb(OB1,OB2,R1), gothrudr(D1,R2,R1)]
M7:[gotob(OB1,R1), pushb(OB1,OB2,R1), gotod(D1,R1)]
M8:[gotob(OB1,R1), pushb(OB1,OB2,R1), gothrudr(D1,R2,R1)]
M9:[gotob(OB1,R1), gotod(D1,R1), gothrudr(D1,R2,R1)]
M10:[pushb(OB1,OB2,R1), gotod(D1,R1), gothrudr(D1,R1,R2)]
M11:[gotob(OB1,R1), pushb(OB1,OB2,R1), gotod(D1,R1), gothrudr(D1,R2,R1)]

Fig.5 All sub-operators sequences



(a) An explanation tree



(b) A generalized explanation tree

Fig.6 Generalization of a macro-operator

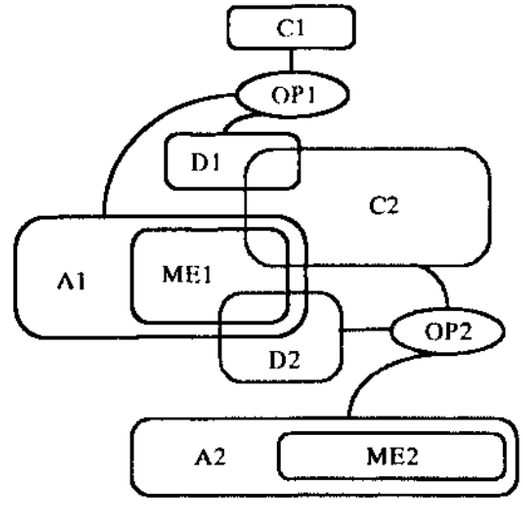


Fig.7 Generation of macro-operators

M1,M2 are extracted with Perfect Causality. Futhermore, this algorithm can extract even discontinuous operator sequences for macro-operators and is available to the worked examples including macro-operators.

4. Generalizing macro-operators with EBG and integrating them

The extracted operator sequences are instances, therefore PiL2 generalizes them with EBG method [Mitchell et al., 1986]. The extracted operator sequence corresponds to an explanation tree in EBG. Fig.6(a) shows the explanation tree constructed with the operator sequence, [gotod, gothrudr) obtained in a last section. In Fig.6(a), the black circles stand for the basic operators and the nodes over, right and under them indicate wffs in a add-list, a delete-list and a cond-list, respectively.

In general, EBG method needs four inputs: Goal Concept, Training Example, Domain Theory and Operationality Criterion. The underlined leaf nodes in Fig.6(a) arc training examples for learning the precondition of the macro-operator and the two operators correspond to the domain theory. A goal concept is the precondition of the macro-operator. However, what corresponds to an operationality criterion? This problem is discussed in section 8. The generalized explanation tree with EBG method |Mitchell et al., 1986] is shown in Fig.6(b), where the upper-case letters indicate variables.

Next, we explain how to integrate the explanation tree into a macro-operator. Fig.7 shows an operator sequence consisting of two operators, OPI,OP2. The Cn,Dn,An and MEn indicate a cond-list, a delete-list, a add-list and a main-effect-list, respectively. Every list is a set of wffs. Therefore, a macro-operator is generated by the recursive applications of the set operations in the following.

$MC=C1 \cup (C2 \cap \overline{D1} \cap \overline{A1})$
$MD=(D1 \cup D2) \cap \overline{A1}$
$MA=A2 \cup (A1 \cap \overline{D2})$
$MME=ME2 \cup (ME1 \cap \overline{D2} \cap \overline{C2})$

The MC,MD,MA and MME indicate a cons-list, a delete-list, a add-list and a main-effect-list of the generated macro-operator, respectively. Fig.8 shows the macro-operator generated from Fig.6(b). In PiL2, all operators including macro-operators are represented in the same structure.

5. The experiment in a classical robot planning

In a domain of a classical robot planning, we made the experiment using the three systems ; (a) a non-learning system, STRIPS,(b)a non-selective macro-learning system, M-STRIPS, (c) a selective macro-learning system, PiL2. The STRIPS is the problem solver of PiL2 and the M-STRIPS generates macro-operators from all sub-operator sequences of worked examples. Note that these three systems use the same problem solver and the only difference between M-STRIPS and PiL2 is in the

```
macro(gotod(A,B), gothrudr(A,C,B))
    MC : [type(A,door), status(A,open),
          type(C,room), inroom(robot,B),
          connects(A,C,B)]
    MD : [nextto(robot,_), inroom(robot,B)]
    MA : [inroom(robot,C)]
    MME : [inroom(robot,C)]
```

Fig.8 The macro-operator from Fig.6

```
M1 : [gotod(D,R1), open(D), gothrudr(D,R2,R1)]
M2 : [gotod(D,R1), gothrudr(D,R2,R1)]
M3 : [gotob(A,R1), pushb(A,B,R1)]
M4 : [gotob(A,R1), pushd(A,D,R1), pushthrudr(A,D,R2,R1)]
M5 : [pushd(A,D,R1), pushthrudr(A,D,R2,R1)]
```

Fig.10 Macro-operators obtained in PiL2

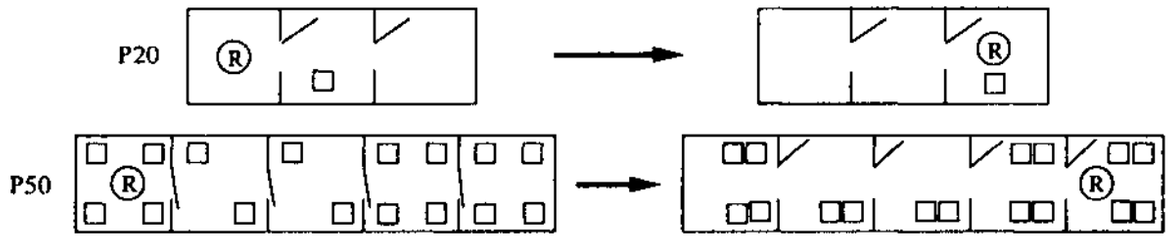


Fig.9 The examples of training problems

problemID	P10	P20	P30	P40	P50
	STRIPS : M-STRIPS : PiL2	STRIPS : PiL2	STRIPS : PiL2	STRIPS : PiL2	STRIPS : PiL2
Branches evaluated	180 : 140 : 9	72 : 10	315 : 20	486 : 20	756 : 60
Nodes expanded	31 : 9 : 5	16 : 5	51 : 7	77 : 9	155 : 54
Solution length	8 : 2 : 3	5 : 2	10 : 4	12 : 4	28 : 12
Macro-operators	0 : 70 : 3	0 : 5	0 : 5	0 : 5	0 : 5
CPU time(sec)	47 : 104 : 10	20 : 12	114 : 19	244 : 27	1819 : 197

Table1 Experimental results

numbers of the macro-operators saved.

We gave each of them 7 basic operators and 50 problems as input. The basic operators include ones in Fig.2 and are the same to ones in [Fikes et al., 1972]. We did not select the operators and the problems for PiL2's good performance. Fig.9 shows samples of given problems and the number of steps for solving the most difficult problem is 28. The experimental results for five problems sampled from a series of 50 are shown in Tablel. The cpu time does not include the time taken to generate macro-operators, only the time necessary to find a solution. Typically, the learning time is considerably less than the search time. As seeing from this table, M-STR1PS has the result only for P10, because a stack-overflow occurred when M-STRIPS was generating the macro-operators after PI3 was solved and we could not continue the experiment. M-STR1PS generated 202 macro-operators on P1-P12 and it took 200sec cpu time to search all of them. Therefore, If M-STRIPS could continue to solve the problems after PI3, the cpu time would be more than 200sec and this time is much longer than PiL2's one.

For P10, the efficiency of M-STRIPS is already worst. Though the branches evaluated of M-STRIPS are less than STRIPS's ones, M-STRIPS spends longer cpu time than STRIPS. Because the time for searching the relevant macro-operators is, in general, longer than that of basic operators. Note that M-STRIPS has already generated 70 macro-operators. Comparing with other two systems, PiL2's branches evaluated, generated macro-operators and cpu time are very small. Therefore, PiL2 is most efficient.

For P20-P50, Tablel shows the results for only STRIPS and PiL2. Note that PiL2 has only five macro-operators even for P50. These five operators are shown in Fig. 10. If PiL2 learned non-selectively, the number of macro-operators would be more than 1000. We can find that Perfect Causality extremely prevented PiL2 from generating a lot of redundant macro-operators. For P20-P50, PiL2 is constantly more efficient than STRIPS.

Table2 shows the averages of cpu time. As is evident from it, PiL2 could solve far more efficiently than STRIPS and M-STRIPS both over PI~P13 and P1-P50. This means that PiL2 can learn only useful macro-operators over many various problems in a classical robot planning.

	P1~P13	P1~P50
PiL2	7	28
STRIPS	16	218
M-STRIPS	104	?

Table2 The averages of CPU times (sec)

r1 : LS=RS → LS+A=RS+A
r2 : LS=RS → LS/A=RS/A
r3 : A*B+A*C → A*(B+C)
r4 : 0+A → A
r5 : 0*A → 0
r6 : R1 − R1 → 0
r7 : A/A → 1
r8 : R1+R2 → R3
r9 : R1/R2 → R3
r10 : (A*B)*C → (A/C)*B

r1000 : 1*AL=R → solution

Fig.11 Basic-operators for solving equations

6. The experiment in solving various equations

Furthermore, we made the experiment in another domain, solving algebraic equations. In this experiment, we used PiL system instead of PiL2. The difference between PiL and PiL2 is only in their problem solvers. The PiL's problem solver uses the forward breadth-first search without any heuristic and its problem states are represented in list structures. In PiL, as well as PiL2, the macro-operators are selectively learned with Perfect Causality and generalized by an EBG method.

Fig. 11 shows a part of basic operators given to PiL. In this figure, A,B,C indicate arbitrary formulas. Rn, AL, NA and NRn stand for an arbitrary real number, a variable, an arbitrary formula but zero and an arbitrary real number but zero, respectively. The rlOOO is a operator for checking the solution state, 1*AL=R. When this operator is applied, a problem solving finishes. We gave a set of training problems consisting of 85 equations of the first degree, 211 equations of the second degree, 35 fractional equations, 78 logarithmic equations and 78 exponential equations. Only when PiL could not solve them by itself, we gave the worked examples and PiL learned macro-operators from them.

As a result, 13 macro-operators were generated for the equations of the first degree, 62 ones for the equations of the second degree, 58 ones for the fractional equations, 56 ones for the logarithmic equations and 57 ones for the exponential equations. All the macro-operators for the equations of the first degree are shown in Fig. 12. The number in the bracket indicates the number of basic operators in each macro-operator. The macro-operators marked with (DS) can directly solve the problems. Although PiL could not solve any given problem before learning, it was able to solve all of them after

[4] (R1*AL+R2)/NR1 → R3*AL+R4
[2] (R1*AL)/NR1 → R2*AL
[12] R1*AL+R2=R3*AL → 1*AL=R4 (DS)
[3] (R1*AL+R2)*R3 → NR1*AL+R4
[2] A*R1 − A*R1 → 0
[2] AL*R1+AL*R2 → NR1*AL
[15] NR1*AL+R1=R2*AL+R3 → 1*AL=R4 (DS)
[11] R1*AL=R2*AL+R3 → 1*AL=R4 (DS)
[9] R1=NR1*AL+R2 → 1*AL=R3 (DS)
[8] NR1*AL+R1=R2 → 1*AL=R3 (DS)
[5] R1=NR1*AL → 1*AL=R2 (DS)
[2] (NA*AL)/NA → 1*AL
[4] NR1*AL=R1 → 1*AL=R2 (DS)

Fig.12 All the macro-operators for equations of the first degree

learning. Solving the most difficult problem needs more than 20 applications of basic operators. If PiL generated all macro-operators, they would increase explosively. Therefore, it is evident that even in solving various equations, our selective macro-operator learner can solve many problems more efficiently than a non-selective learning system and a non-learning system.

7. Related works

STRIPS [Fikes et a!., 1972| saves all the sub-sequences from worked examples as MACROPS. However, our macro-learning method can selectively generate only the useful macro-operators from many candidates. Our system can thereby solve more efficiently than STRIPS. This is evident from the experimental results in a classical robot planning.

Both Minton's [Minton, 19851 and Iba's methods [Iba , 1985] for selecting macro-operators depend on the heuristic evaluation function for the problem solving. However, our method can select only useful macro-operators independently from the evaluation function. Furthermore, the generating Minton's S-MACRO |Minton, 1985], common sequences in worked examples, needs a lot of worked examples. Our method learns macro-operator from only a single worked example.

Korfs method is powerful to generate macro-operators in the domain that exhibits operator decomposability [Korf, 1985). However, our method's cost for generating the macro-operators is considered less than Korf's one. Furthermore, Korf's definition; a macro-operator achieves one of the subgoals of the problem without disturbing any subgoals that have been previously achieved, is considered more restricted than ours.

The SOAR's generalization method of macro-operators is implicit and may leads the over/under generalization (Laird, 1986], Our macro-operators are explicitly generalized by an EBG method and the under/over generalization never occurs.

8. Perfect Causality as an Operationality Criterion in EBL

We discuss Perfect Causality in the EBL perspective. Our macro-learning method is considered one of EBL frame works. Therefore, we think our method has inputs corresponding to the four inputs of EBL. As already mentioned in section 4, our method has EBL's three inputs but an operationality criterion. What is an operationality criterion in macro-operator learning ?

The definition of operationality commonly cited in describing EBL system is the following; A concept description is operational if it can be used efficiently to recognize instances of the concept it denotes [Keller, 1987). The cond-lists of macro-operators, which are the concept descriptions in macro-learning, consist of wffs in the cond-list of basic-operators. Since a problem solver can easily recognize the wffs in basic operators, instances of the concept descriptions can be recognize efficiently. Thus, according to the definition of operationality mentioned before, all concept descriptions in macro-learning are operational. Is there no operationality criterion in macro-learning ?

Keller's research for an operationality gives the answer to this problem. He redefined an operationality more precisely [Keller, 1987]. His definition is the following; the concept description is considered operational if it satisfies the following two requirements: 1. usability: the description must be usable by the performance system, 2. utility: when the description is used by the performance system, the system's performance must improve in accordance with the specified objectives. The concept descriptions in macro-learning satisfy the first requirement and not the second one. Because most macro-operators can not actually make a performance system more efficient. Perfect Causality can select the effective macro-operators, whose cond-lists are the concept descriptions satisfying the second requirement. Thus, we consider Perfect Causality is an operationality criterion in macro-learning.

9. Conclusion

We proposed the method to selectively learn only useful macro-operators with Perfect Causality, a new heurictic, and to generalize them with an EBG method. The capability of our method was tested both in a classical robot planning and solving equations.

We verified the utility of Perfect Causality in two different domains. However, we do not know the utility in other domains and the analytical evaluation for the limitation of our method is necessary.

Acknowledgements

Many thanks to Dr. Norihiro Abe for his useful discussions about our research and to Tomohiro Ishikawa for providing very valuable comments on an earlier draft of this paper. Thanks also to all researchers of FAI group in Tusji Lab for helpful comments. Finally, I would like to thank the reviewers of AAAI-88 for inspiring us to this research.

References

[Fikes et al., 1971] Fikes,R.E. and Nilsson,N.J. STRIPS:A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence,* 2:189-208, 1971.

[Fikes et al., 1972] Fikes,R.E. Hars,P.E. and Nilson,N.J. Learning and Executing Generalized Robot Plans. *Artificial Intelligence,* 3:251-288, 1972.

[Iba, 1985] Iba,G.A. Learning by discovering macros in puzzle solvitg. *In Proceeding of the Ninth International Joint Conference on Artificial Intelligence,* page 640-642, Los Angeles, California, August 1985.

[Keller, 1987] Keller,R.M. Defining Operationality for Explanation-Based Learning. *In Proceeding of the Fourth National Conference on Artificial Intelligence,* pages 482-287, Seattle, Washington, July 1987.

[Korf, 1985] Korf,R.E. Macro-Operators: A Weak Method for Learning. *Artificial Intelligence,* 26(I):35-77, 1985.

[Laird et al., 1986] Laird,J.E. Rosenbloom,P.S. and Newell,A. Chunking in SoanThe Anatomy of a General Learning Mechanism. *Machine Learning,* 1(1): 11-46, 1986.

[Minton, 1985] Minton,S. Selectively Generalizing Plans for Problem-Solving. *In Proceeding of the Ninth International Joint Conference on Artificial Intelligence,* pages 595-599, Los Angeles, California, August 1985.

[Mitchell et al., 1986] Mitchell,T.M. Keller,R.M. and Kadar-Cabelli, ST. Explanation-Based Generalization : A Unifying *View. Machine Learning,* I(I):47-80, 1986.