May 1988 UILU-ENG-88-2219

COORDINATED SCIENCE LABORATORY
College of Engineering

SUBSTRUCTURE DISCOVERY OF MACRO-OPERATORS

Bradley L. Whitehall

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-88-2219	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION NSF, ONR, DARPA
6c. ADDRESS (City, State, and ZIP Code) 1101 W. Springfield Avenue Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code) 1800 G. Street, Washington D.C. 20552 800 N. Quincy, Arlington, VA 22202 1400 Wilson Boulevard, Arlington, VA22209-230
8a. NAME OF FUNDING / SPONSORING ORGANIZATION NSF, ONR, DARPA 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER NSF IST-85-11170, N00014-82-K-0186 N00014-87-K-0874
8c. ADDRESS (City, State, and ZIP Code) 1800 G. Street, Washington, D.C. 20552 800 N. Quincy, Arlington, VA 22202 1400 Wilson Boulevard, Arlington VA 22209-2308	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Substructure Discovery of Macro-Operators

12. PERSONAL AUTHOR(S)

Whitehall, Bradley L.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	88, May	15

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Substructure Discovery, Macro Operators, Background knowledge similarity-differenced based learning, conceptual clustering

19. ABSTRACT (Co

This paper describes an implemented system, PLAND, for discovering substructures in observed action sequences. The goal is to show how a system can learn useful macro-operators by observing a task being performed. An intelligent robot using this system could learn how to perform new tasks by watching tasks being performed by someone else, even if the robot does not possess a complete understanding of the actions being observed.

Macro-operators are discovered within a specific context that provides the types of generalizations allowed in the discovery process and uses the previously proposed macro-operators to build new ones. Background knowledge is used to determine which generalizations are appropriate and to control search. The system can discover syntactic structures (grammars) without background knowledge, but more meaningful and useful structures are discovered when background knowledge is incorporated into the process.

The foundations of PLAND are in similarity-difference-based (SDBL) learning systems that perform conceptual clustering; however unlike most SDBL systems, a large amount of background knowledge can be incorporated to improve learning effectiveness.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

Substructure Discovery of Macro-Operators*

Bradley L. Whitehall

Artificial Intelligence Research Group
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1101 West Springfield Avenue
Urbana, IL 61801

Telephone: (217) 244-1947
Arpanet: whitehal%uicsl@uxc.cso.uiuc.edu

March 1988

ABSTRACT

This paper describes an implemented system, PLAND, for discovering substructures in observed action sequences. The goal is to show how a system can learn useful macro-operators by observing a task being performed. An intelligent robot using this system could learn how to perform new tasks by watching tasks being performed by someone else, even if the robot does not possess a complete understanding of the actions being observed.

Macro-operators are discovered within a specific context that provides the types of generalizations allowed in the discovery process and uses the previously proposed macro-operators to build new ones. Background knowledge is used to determine which generalizations are appropriate and to control search. The system can discover syntactic structures (grammars) without background knowledge, but more meaningful and useful structures are discovered when background knowledge is incorporated into the process.

The foundations of PLAND are in similarity-difference-based (SDBL) learning systems that perform conceptual clustering; however unlike most SDBL systems, a large amount of background knowledge can be incorporated to improve learning effectiveness.

* This research was partially supported by the National Science Foundation under grant NSF IST-85-11170, the Office of Naval Research under grant N00014-82-K-0186, by the Defense Advanced Research Projects Agency under grant N00014-87-K-0874, and by a gift from Texas Instruments, Inc.

1. INTRODUCTION

The goal of the research presented in this paper is to discover (plausible) structures in observed action sequences. Specifically, the PLAND (PLAN Discovery) system discovers macro-operators (macrops) of action subsequences by searching for interesting substructures in observed action traces. By constructing a hierarchical representation from the flat (linear) structure of actions observed, the system gains insight into the interconnections of the actions and how these sequences could be used in future problem solving. This system learns in the same method an apprentice learns from a master craftsman — by observing skill in action.

The work described in this paper differs from previous work in macro-operator construction in two important areas. Unlike [Andreae84] and [Minton85], PLAND *discovers* macrops from observation and does not use examples to learn the new structures . The another difference is that PLAND does not use a problem solver to determine what the macrops for a task should be (as is done in STRIPS [Fikes72]) or have a complete theory of the task domain that may be used to explain the observation as done in EBL systems [DeJong86, Mitchell86]. PLAND does take advantage of domain specific background knowledge when possible, but it is not required to understand the observed actions. PLAND does not have a complete description of the problem goal or (necessarily) an understanding of the individual actions given as observations. This system is not a problem solver that stores the solutions to achieved goals. Rather, PLAND is a passive observer that induces the structure of the events it sees.

The next section of the paper describes how macro-operators can be viewed as substructures along with some background on substructure discovery. The third section presents an overview of the PLAND system with the aid of an example. The fourth section describes the discovery process used for detecting loops and conditionals. This is followed by a demonstration of the system.

2. SUBSTRUCTURE DISCOVERY

A brief description of substructures and substructure discovery is presented in this section; for more detail see [Whitehall87] and [Holder88]. A substructure is defined as a collection of relations and the nodes associated with those relations. The nodes and relations constitute a connected portion of the structure within a complete event. All the nodes are connected by relations in the graph theoretic sense where the nodes are vertices and the relations are edges.

Discovering structure within given input events requires the use of *part-to-whole generalization* [Dietterich86]. In other words, from the pieces the system sees it explores how the complete event is best described. There are two types of structure being found by the PLAND system. The linear structure of actions is composed into macrops. There is also the hierarchical structure of the macrops that can be used to break the task into manageable chunks. These chunks can be used to recognize the hierarchy of goals of the problem solver. The system works at discovering the first kind of structure, macrops, but the hierarchical structure is a byproduct of allowing macrops to internalize other macrops.

Structure in this system is the relationship between actions used to accomplish the plan. Logical groupings of actions that perform a definable unit of work are assembled into macro-operators. A single macro-operator is a sequence of actions that are structurally linked by the order in which they are performed, a totally ordered subset of plan steps. The nodes of this structure are the actions. The one type of relation between the input actions is *follows*.

3. OVERVIEW OF PLAND SYSTEM

The goal of the PLAND system is to discover macrops from a given trace of primitive actions. The system uses background knowledge to help guide the quest for macrops. Unlike the macro-operators of other systems [Fikes72, Minton85], macrops for this system are not generalized by changing constants to variables and determining all of the preconditions for the execution of the macrop. The PLAND system is concerned with discovering possible macrops that are known to accomplish some task in the given execution trace, but whose general applicability is as yet unknown. Macro-operators discovered by the system could be passed to an explanation-based learning (EBL) system [DeJong86, Mitchell86] to determine an explanation based evaluation of the macrop's usefulness and to be further generalized. Having PLAND propose discovered macrops to an EBL system means that the system is deriving proofs only for macrops that have some empirical support.

PLAND is based upon similarity-difference-based learning (SDBL[1]) systems [Hayes-Roth78, Hoff83, Stepp84, Vere78]. As with all similarity-difference-based systems it does not try to prove the validity of the discovered macrops. There is a

[1] Similarity-difference-based learning is the same as what was previously referred to as similarity-based learning (SBL). However, these systems do not learn by observing only similarities. The differences between events also play an important role.

leap of faith in the generalization done in the discovery process. For example, it is possible for the system to classify an anomaly as a macrop. But in order for this to happen under typical heuristic biases, the anomaly would need to occur many times. In such a case one must question how irregular the observed actions really are.

The PLAND system takes a single sequence of observed actions as input. From this stream of actions it must discover logical units that can reduce the complexity of the trace. This is similar to the NODDY system of Andreae [Andreae84], but that system is given examples of a single iteration through the body of a loop from which it can learn conditionals and the loop structure. PLAND must itself break the action sequence into examples and work with those chunks to discover macrops. This complicates the problem because one is never sure that one is working with correct "examples". Wolff's SNPR system [Wolff82] for discovering grammars performs some of the functions of PLAND. The main differences being that PLAND discovers loops explicitly where SNPR does not, and that SNPR does not incorporate knowledge to allow actions to have associated attributes, a feature that aids the discovery process.

3.1. Types of Macrops Discovered

Three types of macrops are discovered by the PLAND system: sequences, loops, and conditionals.

- Sequences

  The most basic macrop is a simple sequence of steps. A sequence is a block of actions that have been used in many places in the input trace. Actions in a sequence have not occurred consecutively enough times to be considered a loop.

- Loops

  Loops are defined as sequences that appear juxtaposed for at least a minimum number (a parameter) of iterations. In the normal meaning of the word, loops have test conditions to stop their execution. PLAND does not determine what those stopping criteria are but learns only the sequence of actions that compose the body of the loop. Learning the exit conditions for a looping construct requires a system like BAGGER [Shavlik87a, Shavlik87b] because exit conditions are reflected in the state of the system but not in the actions performed. As an example, consider an input string of *ABCDCDCDEFCDCD*, from which the loop macrop, using formal

grammar syntax, *(CD)** is discovered. The string can now be described as being generated by *AB(CD)*EF(CD)**.

- Conditionals

  The third type of macrop found is conditionals. A conditional allows a choice of actions for some particular point in time. PLAND defines conditionals as macrops that have more than one choice point within them. A particular choice point is not limited to just two alternatives. For example, if given *ABCADCAECABCADC* the system discovers the macrop *(A(B+D+E)C)**. In a manner analogous to loops, the situations that cause a specific branch of a choice point to be performed are not learned.

These are the three types of macro-operators that the system can discover. By nesting previously discovered macrops within other macro-operators the system is able to discover complex relationships between different macrops and build up a hierarchical structure of an observed sequence of actions. The code to discover these constructs is a major portion of the knowledge built into the system. Additional heuristics and generalizations done by the system are supplied by background knowledge.

3.2. Top Level Organization

The system works on multiple levels of generalization called *contexts*. A context contains all the information needed to process a set of actions for a given level of abstraction. This includes the input sequence of actions, the previously discovered macrops, agendas, and information about how macrops overlap and subsume each other. The system can proceed to a more abstract level by creating a new context in which the actions are generalized. The actions can be generalized by replacing groups of actions with macrops or by the use of a fuzzy matching algorithm.

The system can change the level it is working on by replacing the current context. This flexibility is useful if the system is unsure which level of generalization is appropriate for the problem. The system can work on one context for a specified amount of time, then swap contexts and work on a different one.

A second top level data structure is an *agenda*. An agenda indicates where to look for new macrops in the given example. There are many agendas competing for processor time, and a simple agenda control system manages their priorities. An agenda contains information on where in the action sequence the search for a macrop is to begin. The previously found macrops that can be used in building up

the current macrop are specified in the agenda. The type of macrop wanted is indicated, either a loop or a conditional. Searching directly for sequence macrops is not done, but information regarding a possible sequence macrop is updated whenever a new macrop is discovered. Agendas compete only with other agendas of the same context.

3.3. Use of Background Knowledge

The AI community has recognized that in order to learn substantive concepts a system must possess knowledge about the domain [Schank86, Winston84]. Some SDBL systems, e.g., [Fisher87, Hoff83, Langley86, Stepp84], have not used background knowledge in a flexible manner to guide the learning process, but rather have used knowledge to control the types of generalizations allowed. Recently, researchers have incorporated more background knowledge into the systems to help with the discovery process [Lebowitz86, Mogensen87, Stepp86]. PLAND continues this trend. This section describes the multiple ways in which domain specific knowledge is used by the system.

Observed trace of actions working with:
(WAKE-UP) (EAT_1) (GOTO GYM_1) (GOTO $WORK_1$) (GOTO $HOME_1$)
(EAT_2) (GOTO BED) (WAKE-UP) (EAT_1) (GOTO $WORK_2$) (GOTO GYM_2)
(GOTO $HOME_2$) (EAT_3) (GOTO BED) (WAKE-UP) (EAT_1) (GOTO GYM_1)
(GOTO $WORK_3$) (GOTO $HOME_1$) (EAT_4) (GOTO BED) (WAKE-UP) (GET-SNACK)
(GOTO BED) (WAKE-UP) (EAT_1) (GOTO $WORK_2$) (GOTO GYM_2)
(GOTO $HOME_3$) (EAT_2) (GOTO BED) (WAKE-UP) (EAT_1) (GOTO GYM_1)
(GOTO $WORK_1$) (GOTO $HOME_1$) (EAT_5) (GOTO BED) (SLEEP WALK)
Ready to start another cycle. The result of the last context was:

CONTEXT 1

cogsav	name	macrops
28.0	M1	<WAKE-UP EAT (GOTO-GYM + ()) GOTO-WORK (GOTO-GYM + ()) GOTO-HOME EAT GOTO-BED>

Observed trace of actions working with:
[M1] (WAKE-UP) (GET-SNACK) (GOTO BED) [M1] (SLEEP WALK)
All interesting macrops were discovered. This example is finished.

Figure 1: Student Example

To help explain the use of background knowledge in the system, a simple example is presented. The system is capable of dealing with far more complex examples. Background knowledge is expressed by rules and the system does backward chaining through the rules to obtain an answer to a query about the current processing. For this simple example the input (as shown in figure 1) consists of a week's worth of actions performed by a hypothetical graduate student. The goal of the system is to discover a macrop that will define a typical day in the life of this student. The background knowledge specifies that going to the gym is optional and that a day must start by waking up in the morning, end by going to bed, and a student must also get some work done during the day. A conceptual version of the rule for the constraints on the desired macrop is given in figure 2, where MATCH is a function that makes use of a fuzzy matching algorithm to determine whether two actions are equivalent.

MATCH does not require that two actions be "EQ" equal. The fuzzy matcher will enable predefined patterns to determine how a match for particular items may occur. Simple things such as allowing numeric values to fall within a range, ignoring parts of an action, and forcing strict equivalence are built in to the matching routines. More complex matching, such as traversing defined ISA links for actions, can be specified by defining LISP code. SOME-ACTION just checks that the first argument MATCHes at least one action in the second argument (a macrop).

Although simple names or letters are used for actions here, it should be clear that exact syntactic matches are not required. Thus a loop indicated by X* could actually represent a loop where an occurrence is a list of actions from the set $\{X_1, X_2, X_3, \ldots, X_n\}$ where each X_i is a known way to achieve X. X_i need not be a primitive action, but could be a macrop that describes very complex actions whose

```
IF (AND (MACROP-STRUCTURE ?x)
        (MATCH (FIRST-ACTION ?x) '(WAKE-UP))
        (MATCH (LAST-ACTION ?x) '(GOTO ?y))
        (MATCH ?y 'BED)
        (SOME-ACTION '(WORK) ?x))
THEN
  (VALID-MACROP ?x)
```

Figure 2: Background Knowledge Rule

result is known. For example, in figure 1 there are many ways a student can eat: fix a bowl of cereal, grab some fast food, fix a meal at home, etc., thus the reason for different subscripts.

In the example, PLAND finds a macrop for the student which is *wake up*, optionally *go to the gym*, *work*, optionally *go to the gym*, *head home*, *eat*, and *go to bed*. Although the output of the system in figure 1 makes the problem look simple, it was not solvable without the domain knowledge in figure 2[2].

PLAND uses background knowledge in three distinct ways. At the highest level, the background knowledge acts as meta-knowledge. The system works on levels of generalization called contexts. The system queries the knowledge base to determine what should be the current working context. The current context is retained if there are good opportunities for discovering new macrops within it. If a different context is suggested, the knowledge base returns the new context to be used. At this level, the knowledge is used to control the level of generalization of the action steps. Thus the system can process the input at a higher level of abstraction after some macrops have been found. If the search at the higher level is fruitless then the system can return to the more detailed level. This is a powerful problem solving mechanism because it allows the system to pursue many possible goals. If the system discovers a macrop that indicates a certain environment is present, then it can create a context that generalizes some of these actions to help confirm that notion. This type of hypothesis formation is rare in SDBL systems.

When the top level decides to change contexts, background knowledge is consulted as to what type of generalizations should be performed on the actions (via the fuzzy matcher) of the current context. Consultation with the background knowledge in this fashion allows the system to discover macrops that would be impossible to discover otherwise. A system without knowledge specific to the domain could not make logical guesses at which of the many possible generalizations has meaning to the problem at hand.

For the example given earlier, no such knowledge exists. But for more complex problems the actions could be interpreted in numerous ways. Knowledge of these different interpretations would allow the creation of multiple contexts

[2] Due to the amount of regularity in the example, a large number of macrops were created. The handling of these macrops slows the discovery process. The simple condition of forcing days to start by waking up and end by going to bed is enough to prune the number of possible macrops so that the system can function. This example shows that the addition of simple knowledge can greatly improve the prospects of discovering the correct solution.

where the type of matches allowed for the actions was different. The system could then let each context run a few selected agendas and continue with the context that had found the best macrops (as measured by *cognitive savings* or other heuristic criteria).

The background knowledge used at the intermediate level helps direct the searching process for the macrops through agenda control. Before any agenda is given control, the background knowledge is queried to approve the agenda's applicability. If the knowledge indicates that macrops are not to be searched beyond a certain point in the input sequence, then those agendas can be pruned. Information contained in the agenda could signify that the agenda should not be performed. The knowledge used in this method can save substantial amounts of processing and significantly prune the search tree.

If the system had a large number of actions it observed while the student was sleeping, such as snoring, rolling over, etc., then knowledge could eliminate searching this area. For example, a rule might reject an agenda entry for finding loops that start in a position between an action of going to bed and waking up.

At the lowest level, background knowledge is used to control the macrops allowed by the system. After finding the sequence of actions for a macro-operator, a query to background knowledge determines if the new sequence meets any simple criteria expressed for macrops. With simple rules for checking macrop sequences, like the rule presented in figure 2, the system is able to eliminate the generation of useless macrops. This saves storage and time. Knowledge at the lowest level can rid the system of macrops having low utility. Although this seems trivial, this level of control determines whether a system finds a solution or runs out of space and/or time. Control like this is missing in many SDBL systems. Those systems can only make guesses at what is useful, much as this system does when no background knowledge is present. The implication that nothing valid can be done without background knowledge is not intended. In fact this system can find useful results even when no knowledge is given. In these cases the system acts like a finite state machine builder. The input is like a string where the actions are the letters of the string. Then the discovered macrops act as formal grammars defining portions of the input string and together constitute a generalized regular grammar that can generate the input and other "similar" strings.

3.4. Cognitive Savings

The utility of a discovered macro-operator is measured by *cognitive savings.* PLAND uses the cognitive savings values to determine which of two substructures has the best potential for extension and applicability. The intent of this value is to capture the mental savings one gains by working with the substructure instead of the primitive actions that compose it. A simple formula for cognitive savings is

(number of structure occurrences - 1) * size of structure

where size of structure can be defined as number of nodes, number of relations, or some other formula using the components of the structure. This formula incorporates components of Wolff's compression principles [Wolff82]. A macrop that can chunk a large number of primitive actions and occurs many times is very useful. There is a trade off when a macrop expansion causes some prior occurrences to cease to be covered. The exact threshold for this cut off point is domain dependent.

4. MACROP DISCOVERY DETAILS

There is only room here to describe at the highest level how loops and conditionals are discovered. Detail of the processes is explained in [Whitehall87].

The most basic concept underlying loop macrop discovery is *if a sequence occurs many times, with one occurrence following the other, then reduce the sequence.* This is a simple concept that has been used before to reduce given sequences [Restle70, Simon63]. But even this simple concept is difficult to implement in practice. Finding answers to simple questions can explode in exponential time when the examples are not explicitly given. Such questions as "where does the loop body begin?," "how long is the sequence of the body of the loop?," and "does the action that begins the loop also occur within the loop body (thus not always indicating a new iteration)?" are difficult questions to answer. The loop discovery module in PLAND expects parameters in the agenda to guide the search for answers to these questions.

This section describes in a schematic way the algorithm used for loop discovery. A list is created that has the positions in the input sequence for a particular action type. Consecutive occurrences of the action indicate the (possible) start of a new loop iteration[3]. The actions between two possible iterations are added incrementally. As long as the sequence generated thus far

[3] Action occurrences may be skipped to allow the first action of a loop to be used various times within the loop body.

does not completely describe the actions between two starting positions of the proposed loop then the sequence is extended (grown). The sequence is grown by all possible macrops and single actions. This does not explode, as there is a fixed number of macrops which are usable (as defined by the agenda), and each iteration of the proposed loop body acts as a constraint on what is allowed.

The discovery of conditionals is more complex than the discovery of loops. The problem with discovering conditionals is that anything could be made optional. In the extreme case a sequence could be described by a loop of length one with the body consisting of a single conditional for all possible actions. The algorithm used by PLAND avoids this pitfall by requiring that all conditionals have a base or key point that cannot be part of a choice set. The basic principle in discovering conditionals is *find actions that occur on fixed intervals, then make conditionals out of what lies between these key points.*

The first step in the discovery of conditionals is to build a difference array. The difference computed is the number of actions between two sequential occurrences of an action type. Next an array is built which indicates the number of consecutive differences of equal spacing for each difference array. This is done so that in the next step the action with the largest number of iterations (key action) can be found. From the explanation thus far, it should be clear that the conditionals discovered must be part of a loop. It is the repetition of actions at a fixed distance from each other in the sequence that allows the conditionals to be discovered. The actions do not have to be a fixed distance from each other in the primitive version of the observed trace. But they must be a fixed "action" distance apart which means a variable length macrop (such as a loop) could be used in the conditional. Filling in the actions around the key is the fourth step. This is where the actual choice sets get constructed. The last step of the algorithm is to convert the best descriptions of the conditional into macrop structures and find the other occurrences of the conditional. If there are ties for the best conditional then all of them are returned.

5. ANOTHER EXAMPLE

The example, whose run is shown in figure 3, demonstrates that conditionals may be found with embedded macrops. The system can discover conditionals and loops nested to an arbitrary depth. For brevity of presentation no background knowledge was incorporated in this example and the actions are letters. When no background knowledge is used the system finds regular expressions that define

```
Observed trace of actions working with:
(X) (X) (X) (X) (A) (X) (X) (X) (A) (B) (A) (X) (X) (A) (B) (A) (B) (A)
Ready to start another cycle. The result of the last context was:

          CONTEXT 1
cogsav  name    macrops
10.0    M1      <((X + B) A)*>
8.0     M2      <(X)*>
15.0    M3      <(([M2]* + B) A)*>
5.0     M6      <([M1]* X X)*>
5.5     M7      <(A [M2]* A B)*>
10.0    M10     <(X [M1]*)*>

Observed trace of actions working with:
[M3]
All interesting macrops were discovered.  This example is finished.
```

Figure 3: Example Output

chunks of the input string. The first macrop found by the system, M1, is a conditional but it does not yet have the embedded loop macro. This is because the loop macrop for X^* has not been discovered at this point; it is discovered next. Now the system finds the better conditional, M3, which expresses the complete string, $((X^* + B)\ A)^*$. The system continues to find other macrops which are less interesting as indicated by the cognitive savings value. Notice that the macrop numbers are not sequential. Nonsequential macrop numbers indicate the system has discovered macrops that are subsumed by previously defined macrops. The subsumed macrops are noted and not used further by the system.

6. CONCLUSION

The PLAND system demonstrates substructure discovery is useful for finding macro-operators. Substructure discovery allows a system to learn more complex relationships than are possible with attribute only systems. The hierarchical structure of an observed sequence of actions can be constructed without complete background knowledge or explicitly stated examples. Background knowledge can be incorporated to allow the system to discover more appropriate macro-operators with less effort.

PLAND is not a grammar induction system. It is able to induce grammars to define the observed sequences but it goes far beyond capabilities of such systems. It is more powerful because it discovers loops explicitly and uses background knowledge in combination with the fuzzy matcher to allow the system to discover more than just syntactic structures.

The system is novel in the area of macro-operator processing because it does not perform problem solving and store the results. The system discovers the macro-operators by passively observing the actions of another agent performing a task and inducing the structure of the macrops.

7. ACKNOWLEDGMENTS

This work benefited from discussions with Robert Stepp, Larry Holder, Bob Reinke, Bharat Rao, and Diane Cook.

REFERENCES

[Andreae84]

P. M. Andreae, "Justified Generalization: Acquiring Procedures from Examples," Ph. D. Thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, January 1984. (Also appears as Technical Report 834, MIT AI Laboratory.)

[DeJong86]

G. F. DeJong and R. J. Mooney, "Explanation-Based Learning: An Alternative View," *Machine Learning 1*, 2 (April 1986), pp. 145-176. (Also appears as Technical Report UILU-ENG-86-2208, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.)

[Dietterich86]

T. G. Dietterich and R. S. Michalski, "Learning to Predict Sequences," in *Machine Learning: An Artificial Intelligence Approach, Vol. II*, R. S. Michalski, J. G. Carbonell and T. M. Mitchell (ed.), Morgan Kaufmann, Los Altos, CA, 1986, pp. 63-106.

[Fikes72]

R. E. Fikes, P. E. Hart and N. J. Nilsson, "Learning and Executing Generalized Robot Plans," *Artificial Intelligence 3*, 4 (1972), pp. 251-288.

[Fisher87]

D. H. Fisher, "Knowledge Acquisition Via Incremental Conceptual Clustering," Ph.D. Thesis, Department of Information and Computer Science, University of California, Irvine, Irvine, California, 1987. (Also appears as Technical Report 87-22)

[Hayes-Roth78]

F. Hayes-Roth and J. McDermott, "An Interference Matching Technique for Inducing Abstractions," *Communications of the Association for Computing Machinery 21*, 5 (1978), pp. 401-410.

[Hoff83]

W. A. Hoff, R. S. Michalski and R. E. Stepp, "INDUCE 3: A Program for Learning Structural Descriptions from Examples," Technical Report UIUCDCS-F-83-904, Department of Computer Science, University of Illinois, Urbana, IL, 1983.

[Holder88]

L. B. Holder, "Discovering Substructure in Examples," M.S. Thesis, Department of Computer Science, University of Illinois, Urbana, IL, 1988.

[Langley86]

P. Langley, J. M. Zytkow, H. A. Simon and G. L. Bradshaw, "The Search for Regularity: Four Aspects of Scientific Discovery," in *Machine Learning: An Artificial Intelligence Approach, Vol. II*, R. S. Michalski, J. G. Carbonell and T. M. Mitchell (ed.), Morgan Kaufmann, Los Altos, CA, 1986, pp. 425-469.

[Lebowitz86]

M. Lebowitz, "Concept Learning in a Rich Input Domain: Generalization-Based Memory," in

Machine Learning: An Artificial Intelligence Approach, Vol. II, R. S. Michalski, J. G. Carbonell and T. M. Mitchell (ed.), Morgan Kaufmann, Los Altos, CA, 1986, pp. 193-214.

[Minton85]

S. N. Minton, "Selectively Generalizing Plans for Problem-Solving," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 1985, pp. 596-599.

[Mitchell86]

T. M. Mitchell, R. Keller and S. Kedar-Cabelli, "Explanation-Based Generalization: A Unifying View," *Machine Learning 1*, 1 (January 1986), pp. 47-80.

[Mogensen87]

B. N. Mogensen, "Goal-Oriented Conceptual Clustering: The Classifying Attribute Approach," M.S. Thesis, Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL, 1987. (Also appears as Technical Report UILU-ENG-87-2257)

[Restle70]

F. Restle, "Theory of Serial Pattern Learning: Structural Trees," *Psychological Review 77*, 6 (November 1970), pp. 481-495.

[Schank86]

R. C. Schank, G. C. Collins and L. E. Hunter, "Transcending Inductive Category Formation in Learning," *Behavioral and Brain Sciences 9*, 2 (1986), pp. 639-686.

[Shavlik87a]

J. W. Shavlik and G. F. DeJong, "BAGGER: An EBL System that Extends and Generalizes Explanations," *Proceedings of the National Conference on Artificial Intelligence*, Seattle, WA, July 1987, pp. 516-520. (Also appears as Technical Report UILU-ENG-87-2223, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.)

[Shavlik87b]

J. W. Shavlik and G. F. DeJong, "An Explanation-Based Approach to Generalizing Number," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987, pp. 236-238. (Also appears as Technical Report UILU-ENG-87-2220, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.)

[Simon63]

H. A. Simon and K. Kotovsky, "Human Acquisition of Concepts for Sequential Patterns," *Psychological Review 70*, 6 (1963), pp. 534-546.

[Stepp84]

R. E. Stepp, "Conjunctive Conceptual Clustering: A Methodology and Experimentation," Ph.D. Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1984.

[Stepp86]

R. E. Stepp and R. S. Michalski, "Conceptual Clustering: Inventing Goal-Oriented Classifications of Structured Objects," in *Machine Learning: An Artificial Intelligence Approach, Vol. II*, R. S. Michalski, J. G. Carbonell and T. M. Mitchell (ed.), Morgan Kaufmann, Los Altos, CA, 1986, pp. 471-498.

[Vere78]

S. A. Vere, "Inductive Learning of Relational Productions," in *Pattern Directed Inference Systems*, D. A. Waterman and F. Hayes-Roth (ed.), Academic Press, New York, 1978.

[Whitehall87]

B. L. Whitehall, "Substructure Discovery in Executed Action Sequences," M.S. Thesis, Department of Computer Science, University of Illinois, Urbana, IL, 1987. (Also appears as Technical Report UILU-ENG-87-2256)

[Winston84]

P. H. Winston, *Artificial Intelligence (Second Edition)*, Addison-Wesley, Reading, MA, 1984.

[Wolff82]

J. G. Wolff, "Language Acquisition, Data Compression and Generalization," *Language and Communication* 2, 1 (1982), pp. 57-89.