CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

NEWWATCH: Learning Interrupted Strategies by Observing Actions

Andrew Gans and Barbara Hayes-Roth

TECHNICAL REPORT
Number 25

April 1990

Stanford University

If you would like to contact the authors please write to:

c/o CIFE, Civil Engineering,
Stanford University,
Terman Engineering Center
Mail Code: 4020
Stanford, CA 95305-4020

Knowledge Systems Laboratory
Report No. KSL 89-44

March 1990

NEWWATCH: Learning Interrupted Strategies by Observing Actions

by

Andrew Gans, and Barbara Hayes-Roth

KNOWLEDGE SYSTEMS LABORATORY
Department of Computer Science
Stanford University
Stanford, California 94305

Abstract

An expert can normally demonstrate a strategy by solving individual problems. However, it is more difficult to articulate the set of general rules used to select actions. The WATCH project was designed to "watch over the shoulder" of an expert, to automatically learn strategic knowledge by observing the sequence of actions that were performed.

The original WATCH program was limited by the assumption that the entire sequence of problem-solving actions could be induced into a single hierarchical strategy tree. Any interruptions in this single strategy, whether a simple opportunistic sequence or a second parallel strategy, were simply treated as noise that confused the system. The NEWWATCH extension has been implemented to recognize various kinds of interruptions, thereby greatly increasing the range of strategies that the WATCH system can learn.

Acknowledgements

The NEWWATCH project was supervised by Barbara Hayes-Roth, and was done in conjunction with Tony Confrey's work on the META-WATCH system. The original WATCH program, which inspired and provided the platform for this project, was written by Jeff Harvey. A number of members of the BB1 group at the Knowledge Systems Lab, most notably Mike Hewett, provided assistance with implementing parts of the WATCH system as a BB1 application. Tom Gruber gave helpful advice about related work on learning.

Craig Cornelius implemented a revised version of the PROTEAN application, to obtain the initial data for WATCH. Iris Tommelein designed the SIGHTPLAN system, and provided the problems on which the WATCH system was tested.

The research was supported by CIFE, the Center for Integrated Facility Engineering at Stanford. Ray Leavitt motivated the WATCH project's goal of learning strategies for such Civil Engineering tasks as SIGHTPLAN.

Contents

1. Introduction

1.1 Overview

An expert's strategies are one of the most difficult to elicit types of information in an expert system knowledge base. Strategies can often be demonstrated for individual cases, yet they may not be as easily articulated as general rules. Therefore, a tool that could automatically learn the intended strategy by "watching over the expert's shoulder" would be a great aid in building expert systems. The WATCH system derives its name from the fact that it uses this style of apprenticeship learning.

WATCH learns expert strategies for application problems solved within the BB1 blackboard framework. It incrementally examines sequences of problem-solving actions, to induce the strategy used by the expert to select both the actions and their order. The strategies learned are saved in the format of BB1 control knowledge, so that they can be used to solve other problems.

The original WATCH program was limited by the assumption that the entire sequence of problem-solving actions could be induced into a single hierarchical strategy tree. Any interruptions in this single strategy, whether a simple opportunistic sequence or a second parallel strategy, were simply treated as noise that confused the system. The NEWWATCH extension has been implemented to recognize various kinds of interruptions, thereby greatly increasing the range of strategies that the WATCH system can learn.

1.2 Outline

This paper explains the NEWWATCH extension to the WATCH inductive learning program. The first part introduces the BB1 blackboard expert system framework and the ACCORD language, to show the context in which WATCH can learn. Then an explanation of the WATCH program clarifies the goals of the learning project and the power of the original system. Finally a discussion of NEWWATCH shows the usefulness of a system to automatically learn interrupted hierarchical strategies.

2. Background

2.1 The BB1 blackboard environment

BB1 is a framework for building expert systems based on the "blackboard" model. A blackboard allows a number of individuals to collectively solve a problem by each writing contributions in their area of expertise. In BB1, multiple independent knowledge sources cooperate by recording information on a global blackboard data structure. Each knowledge source (KS) is similar to an expert in a particular field. BB1's scheduler acts as the moderator to select one of the available KSes to work at each step of the problem-solving.

All information in the blackboard is globally accessible to the knowledge sources as they are performed. The variety of information from the application domain, the specific problem, and the language framework can all be easily examined by the KSes due to a consistent representation within the entire blackboard. The blackboard contains only objects, all of which may have both attributes with values and links to other objects. Domain objects and KSes are just two examples of information stored as objects within the blackboard.

Control of the problem-solving process is flexible in the BB1 environment. Therefore, building an application involves writing "control" type knowledge sources in addition to KSes relevant to the application "domain". Domain knowledge sources are executed to solve some part of the current problem, while control KSes create strategic decisions to help select later actions. The lack of a single fixed control strategy allows a large variety of problem-solving methods to be used within BB1. The cost of this flexibility is the need to write twice the knowledge sources, since the expert must explicitly specify both domain and control information.

BB1 dynamically selects a single knowledge source to execute at each cycle of its operation, based on the current strategy and state of the system. KSes may be executed if they have been triggered by some prior event, creating Knowledge Source Activation Records (KSARs), and also had their preconditions satisfied. The BB1 Agenda keeps track of the four kinds of KSARs: *triggered* by some event, *executable* since their preconditions have been satisfied, *executed* at an

earlier BB1 cycle, and *obviated* by satisfying conditions that demonstrate that the KSAR is no longer relevant.

Selection of a single executable KSAR at each BB1 cycle is done by rating the alternatives against the currently active strategic decisions. These decisions, in the Control Plan part of the blackboard, are created by the control knowledge sources that have already been executed. Since the control knowledge sources are selected dynamically and each posts its individual preferences, the active Control Plan decisions may reflect a number of interacting strategies created by different control KSes. An expert may write a variety of strategy types in the different control knowledge sources including hierarchical strategies, goal-directed strategies, and simple opportunistic decisions.

BB1 provides generic knowledge sources to aid in creating certain typical types of strategies. For example, in hierarchical strategies the expert defines a set of knowledge sources and their relationships within a tree-like hierarchy. BB1's generic KSes achieve the top-level abstract super-strategy by unfolding the tree. Each KS is accomplished by performing the sequential parts of its plan, its children, one at a time. The current KS remains an active decision on the Control Plan until its goals are met by the state of the BB1 system. The leaves of this hierarchical tree are the most specific strategic decisions, and whichever is active controls the problem-solving process by selecting which actions to perform next.

2.2 The ACCORD language

The BB1 framework includes languages that ease the task of writing certain types of applications. One such language is ACCORD, which is used to express arrangement-assembly type problems. ACCORD has been applied to such sample problem domains as PROTEAN, for the problem of protein structure identification, and SIGHTPLAN, for construction site layout.

ACCORD and other BB1 languages provide templates for expressing the actions of knowledge sources. This aids in the creation of KSes since the expert does not need to give all the details of their operation; instantiating an action based on a standard language verb specifies what its behavior will be. The

language also allows easier reasoning about actions, by allowing control knowledge sources to express strategic decisions in the same standard form.

The ACCORD language can be used by applications that solve arrangement problems. It defines standard actions that are based on the assembly method, the iterative application of constraints to find possible locations for the objects being arranged. ACCORD contains general background concepts that can be linked to specific objects from the problem domain. For example, a "partial arrangement" is a subset of the objects with their currently hypothesized locations. An "anchor" is a fixed object that provides a partial arrangement with a coordinate system.

ACCORD also provides templates for a set of standard action verbs. A problem-solving session will normally begin with a Create action, which will instantiate a partial-arrangement for the arrangement task. The template for the Include action allows the expert to specify an object and a partial arrangement into which it is placed.

Include object *in* partial-arrangement.

When an individual domain object is placed into a partial-arrangement, a special instance of the object is created. The instance is a dynamic entity used during the problem-solving, while the object it is based on is statically defined within the application's knowledge base. For example, if the SIGHTPLAN object power-plant is included in partial-arrangement-1, the instance power-plant-1 is used to represent it while the arrangement is being done.

ACCORD also defines the action Position, which locates an instantiated object within the arrangement by applying constraints between it and other objects. A list of templates for other frequently used ACCORD actions is contained in Appendix 1. The action verbs in a BB1 language such as ACCORD are related by a class hierarchy (Appendix 2). The Position action has a number of children which express more specific types of positioning, depending on the relationships of the constrained objects to the fixed anchor of our arrangement. Two frequently used Position verbs are the Anchor and Yoke actions. Anchor applies constraints between an object and the anchor; its template provides the standard syntax in which these parameters can be specified.

Anchor object *to* anchor *in* partial-arrangement
with constraints.

An expert can instantiate an action template with appropriate objects from the problem domain, such as SIGHTPLAN, to easily write an action sentence.

Anchor fabrication-yard *to* power-plant *in*
partial-arrangement-1 *with* constraint-closer-than-1mile.

The template may also be instantiated more generally, to create a strategic decision. This decision would express the type of action that would be preferred at a specific point in the problem-solving process. Whichever executable action matched the decision most closely would be the best alternative. For example, the strategy might wish to anchor any temporary object that was constrained to be close to the plant.

Anchor temporary-object *to* power-plant *in*
partial-arrangement *with* closer-than-constraint.

The sample instantiated action

Anchor fabrication-yard *to* power-plant *in*
partial-arrangement-1 *with* constraint-closer-than-1mile

would most likely be performed before other executable actions, because it matches the strategic decision perfectly. It is simply a more specific instantiation of this decision. A Yoke action, or an Anchor that specified different objects, would be inferior matches. An Include action would rate still lower, since its ACCORD verb is not a type of Position but rather is farther removed within the class hierarchy.

The task of matching pairs of actions requires a method of comparing actions that have different verbs and templates. It is here that languages such as ACCORD become particularly useful. Various ACCORD action verbs have different parameters in different orders, so background knowledge of how to appropriately compare these actions is needed. Each language framework within BB1 has an associated partial match table which defines the correspondence between parameters of different types of actions. For example, the ACCORD Anchor and Yoke actions have templates such as

Anchor object *to* anchor *in* partial-arrangement
with constraints

and

Yoke object *and* object *in* partial-arrangement
with constraints.

When comparing an Anchor action with a Yoke, the partial match table indicates that the object of the Anchor is matched against both objects of the Yoke action, the two partial arrangements are compared, and the two constraints are examined for similarity.

The partial match table and class hierarchy of standard verbs also define a method for generalizing two dissimilar actions together. An Anchor and a Yoke can be abstracted into a more general Position action, because both verbs are specifications of positioning within the action verb hierarchy.

Position generalized-object *in* generalized-partial-arrangement
with generalized-constraint-set.

The generalized terms are abstracted from the more specific parameters mentioned in the Anchor and Yoke sentences. The partial match table tells which parameters of the two actions to combine, and the appropriate class hierarchies provide a place to search for the maximally specific generalization.

3. WATCH

3.1 Introduction

WATCH learns control strategies by "watching over the shoulder" of an expert. The expert has some problem-solving strategy in mind, which he may not be able to articulate as BB1 control knowledge. Instead, he takes the place of the control strategy for some problem, by dynamically selecting a sequence of actions to perform. WATCH then tries to discover the intended strategy by incrementally examining the action sequence.

3.2 Assumptions and bias

WATCH assumes that the expert is behaving consistently according to some strategy as the actions are chosen one at a time. If the expert is acting randomly or changing the strategy as he goes, WATCH's attempt to learn the strategy will yield only spurious results.

WATCH was originally written with the assumption that a single hierarchical strategy was used by the expert to select the entire action sequence. Therefore, WATCH attempts to explain all the actions with only one hierarchy. This limitation on the kinds of strategies that can be learned motivated the NEWWATCH project. The NEWWATCH extension relaxes this limiting assumption by allowing the hierarchy to be interrupted, thereby increasing the range of strategies that can be learned.

The most important requirement for WATCH's learning is that the actions in the problem-solving sequence be expressed in ACCORD or a similar language. This language defines a method for generalizing among actions when doing inductive learning. ACCORD's action templates and partial match table provide a means for comparing dissimilar actions, while also defining a limited space in which to search for the maximally specific generalization. Without the bias on the search provided by ACCORD, WATCH could not induce abstractions of sub-sequences; there would be no way of making intelligent guesses about which features the actions had in common. Although WATCH does not have

potentially useful information about ACCORD, such as knowledge about the results or dependencies of the standard actions, the lack of a deeper understanding does not seriously limit the ability to learn. The bias provided by the ACCORD language is enough to allow WATCH to do useful inductive generalization.

WATCH gains power to learn strategies for BB1 application systems from the fact that it is also written as a BB1 application. WATCH is able to take advantage of the rich knowledge available in the BB1 environment. It can examine information in the BB1 agenda, such as the sequence of actions performed to solve the application problem. It also has access to general information on the blackboards, such as the objects and class hierarchies within the application domain, the problem, and the ACCORD language. This background knowledge, combined with the abilities of ACCORD, allows WATCH to inductively generalize domain-specific actions.

WATCH uses only the incomplete information of the action sequence and background knowledge, because it cannot know the complete state of the BB1 system at every instant of the problem-solving. Although WATCH learns by observing the action sequence incrementally over the shoulder of the working expert, it cannot examine the state at intermediate points in the non-interruptible BB1 cycle. It would also be impractical to store the entire system state for all previous actions in the sequence, to have this information available when doing induction on later actions. However, the BB1 agenda contains a trace of the problem-solving, such as which actions were executed, and this limited knowledge is a good substitute for the impractically large amount of state information.

3.3 The WATCH program

WATCH divides the problem of learning a strategy into six phases. Much of the work of learning falls into the first task, inductively generalizing the action sequence. It is here that WATCH needs the most heuristic guidance. Once the generalization has been done, strategies can be collected to find the best few. Finally, WATCH concludes by providing the results of its learning as BB1 strategies.

1. Generalize the action sequence.
   Incrementally examine the sequence of actions, and build a tree of possible generalizations of various sub-sequences.
2. Choose possible strategies.
   Identify all possible strategies by finding the sets of sub-sequence generalizations in the tree that explain the whole action sequence.
3. Refine the strategies.
   Simplify strategies by removing unnecessary levels of abstraction.
4. Choose the best strategies.
   Rank the possible strategies using a set of heuristic preferences.
5. Hypothesize modifiers.
   Add appropriate modifiers that explain the order of actions within each sub-sequence of the strategy.
6. Write out control strategies.
   Output the strategies in two formats useable by BB1, as a set of control knowledge sources and as a skeletal plan.

3.3.1 Inductive generalization

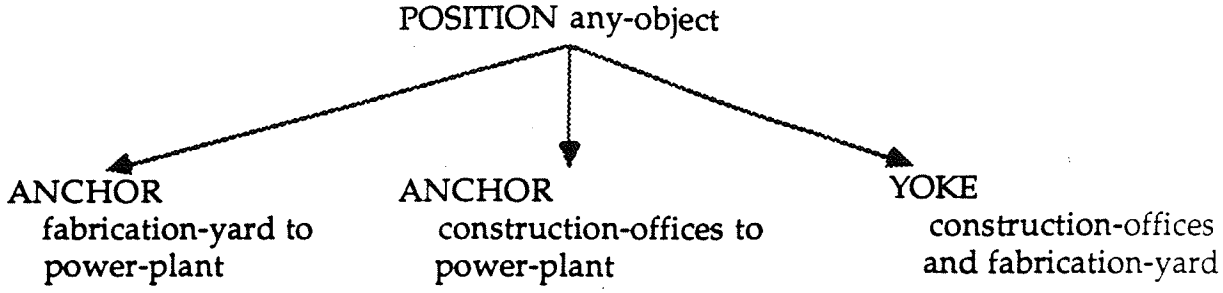
WATCH's most difficult task is inductively generalizing the action sequence to find a reasonable hierarchical explanation. The sequence can be broken down into an exponential number of sub-sequences. WATCH could conceivably try to learn any combination from treating the whole sequence as one very general strategic decision to treating each individual action as a specific strategic focus. The ideal strategy is the one which the expert had in mind when choosing the actions, but there may be a number of reasonable strategies that the expert might have intended.

The combinatorial problem of generalizing over the action sequence can be shown even in a simple example of a three-action sequence. The expert may have specified a sequence of actions to perform that included the following three actions in the order specified.

(1) *Anchor* fabrication-yard *to* power-plant
(2) *Anchor* construction-offices *to* power-plant
(3) *Yoke* construction-offices *and* fabrication-yard

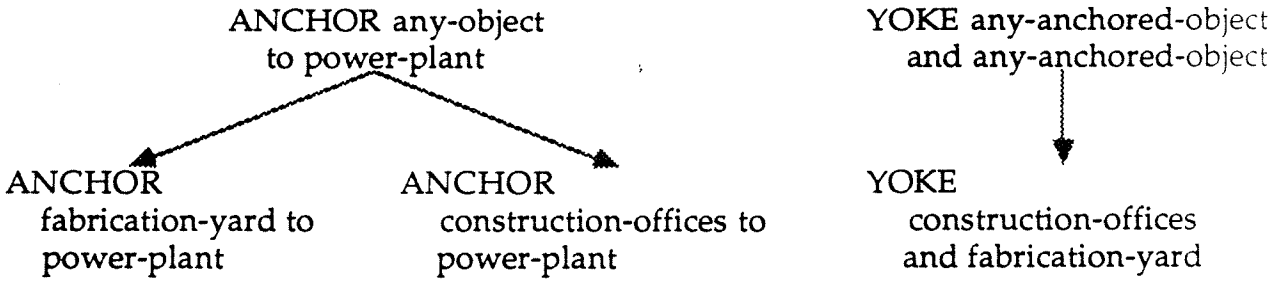
The expert may not care about the order of these positioning actions, and so may have merely intended a very general control strategy which could have chosen these three actions in any order.

STRATEGY 1:



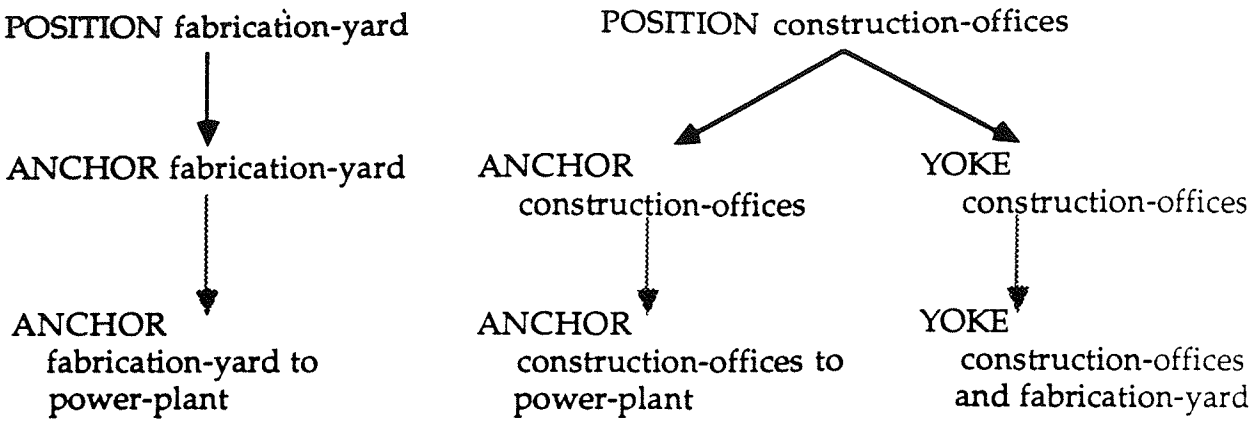
However, there may be some important reason for the order. If it were important that Anchor actions precede Yokes, the strategy would have to be structured to explicitly cause this ordering.

STRATEGY 2:



The expert could also have intended an alternative strategy in which the ordering of actions was based on the objects involved, rather than the action verbs. For example, this particular sequence may have been selected because a fabrication yard requires more space than offices, and so should be positioned earlier. In this case, the hierarchy must specify positioning one type of object first, and then acting on the other.

STRATEGY 3:



3.3.2 Heuristic search

Even with a simple sequence of three actions, there are a number of different strategies that can be inductively generalized. Therefore, WATCH must use heuristic guidance to intelligently search through the possible strategies.

Strategy 2 may appear to be the one that the expert most likely intended, based on a preference that actions should be grouped by their ACCORD action verbs. Such a heuristic would be logical, since each language verb has a unique operation and therefore is likely to be contained in a distinct phase of the problem-solving. However, the third strategy also could have been what the expert had in mind when selecting the three actions, since domain-specific knowledge and heuristics make that strategy seem natural. Both strategies are reasonable, since different experts use their own personal problem-solving methods. For this reason, the heuristics must merely be flexible guides for the generalization process, rather than fixed rules.

WATCH uses five heuristics to guide the generalization process. Some, like the "avoid generalizing different actions together" heuristic that would prefer Strategy 2 above, are based on knowledge of the ACCORD language and its use. Others, such as "prefer longer sequences of actions" and "prefer generalizing over fewer levels of abstraction" are general to induction over any sequence. It is interesting to note that these latter two are actually contradictory, and so they

compete in selecting which generalization to perform. If more actions are added to a sequence, generalization over more levels of abstraction will usually be necessary to find one common representation.

Because WATCH is written as a BB1 application, all control knowledge, including the five generalization heuristics, is flexible and can be changed to reflect different learning preferences. The complete list of WATCH's current learning heuristics is contained in Appendix 3.

3.3.3 Incremental learning

WATCH incrementally examines the sequence of problem-solving actions to build a plausible tree of generalizations. As each new action is seen, BB1 stores it on the LEARN blackboard, at the lowest level of the generalization hierarchy. KSARs are triggered for all of the possible generalizations of the new action with the existing action sequence and generalization tree. BB1 rates these possibilities based on WATCH's set of five generalization heuristics; only the heuristically best few generalizations are performed, to limit the combinatorial explosion.

WATCH performs two different types of generalization. The knowledge source GENERALIZE-STATEMENT-SEQUENCE can generalize the new action with some existing object in the generalization tree, and it can abstract generalizations in the tree to higher levels of the hierarchy. The WATCH KS called POSTULATE-STATEMENT-SEQUENCE creates new generalizations in the tree by postulating that a sub-sequence that includes the new action is a discreet strategic decision.

Generalizing a new action with either an existing generalization tree object or other actions in the sequence involves the task of comparing multiple actions to find similarities. It is here that ACCORD provides WATCH with its bias for learning. The standard action templates and partial match tables allow WATCH to compare dissimilar actions. This matching process can then use the domain-specific class hierarchies to find the maximally specific generalization of two or more actions.

3.3.4 Modifiers

Once WATCH has generalized a sequence of actions together, it must give an explanation for the ordering within the sequence. WATCH's fifth learning phase does this, by hypothesizing the appropriate modifiers.

Modifiers are domain or language specific concepts that are used to rank objects. For example, the SIGHTPLAN application might define a modifier called Large which would rate a fabrication-yard higher than construction-offices. An expert could then express the strategy that larger objects should be positioned on a construction site first, with a control strategy decision explicitly mentioning this modifier.

Position Large object *in* partial-arrangement *with* constraints.

If the BB1 scheduler were trying to choose between a Position action mentioning a fabrication-yard and an equivalent Position for the construction-offices, this strategy would cause it to select the former first.

WATCH hypothesizes modifiers for a sequence by adding to the strategic decision all of the modifiers that individually would explain the ordering of actions. In other words, any modifier that would rank the actions in monotonically decreasing order is assumed to be present. While this method risks much spurious learning, it will certainly discover any modifiers that could have been specified by the expert. The resulting strategy learned by WATCH may cause actions to conform to some rational ordering scheme, when a more random sequence was in fact intended; however, such over-specification does not limit the usefulness of the learned strategy.

4. NEWWATCH

4.1 Introduction

The original WATCH program was designed to learn only purely hierarchical strategies. It assumed that the entire action sequence could be generalized into a single hierarchical explanation. However, the flexibility of control reasoning in BB1 allows actions to be chosen by a number of other strategy types as well as multiple concurrent strategies. If an expert selects actions with some more complex strategy in mind than a single hierarchy, WATCH's limiting assumption hinders its ability to correctly learn the intended strategy.

The NEWWATCH extension increases the range of strategies that WATCH can learn. NEWWATCH still assumes that the expert is behaving primarily according to a hierarchical strategy, yet the possibility of other strategies working concurrently is allowed. NEWWATCH's task is to recognize actions in the sequence that might not have been caused by the main hierarchy. Since these interruptions are no longer treated as noise when doing the inductive generalization, NEWWATCH can do a better job of learning the hierarchical strategy. In addition, the secondary strategies that the expert used to select actions in the interruptions can also be discovered at the same time.

4.2 Interruptions

NEWWATCH attempts to discover interruptions in the action sequence. *An interruption is defined as any sub-sequence of actions within the observed action sequence that is not explained by the primary hierarchical strategy.* Interruptions may reflect different types of strategies intended by the expert. The action sequence may have been selected by multiple parallel strategies working concurrently. In this case, NEWWATCH could recognize one as the main hierarchical strategy and treat the others as interruptions to it. The expert may also have intended one primary hierarchy, with a special set of *opportunistic* actions to perform in certain situations. Regardless of the strategy type, recognizing the appropriate actions as interruptions eases the task of learning the main hierarchical strategy.

NEWWATCH modifies the original WATCH program so that it searches for possible interruptions in the action sequence while doing the inductive generalization. Since interruptions are defined as anomalies in the primary hierarchical strategy, the hypothesized interruptions must be consistent with the hierarchical strategies induced at any point of the incremental learning. For this reason, a sub-sequence is not hypothesized as an interruption until the inductive generalization has been completed for its predecessors in the action sequence. The possible strategies generalized from the predecessors are the basis for hypothesizing that the interruption is not explained by the current strategy.

The number of possible interruptions in an action sequence is exponential, since any sub-sequence of one or more actions might be an interruption. Every action could be hypothesized to be an interruption by itself, yet this would leave no hierarchical strategy to learn. One the other hand, if NEWWATCH does not notice that the expert chose certain actions using a secondary strategy, the hierarchy learned would be adversely affected by the "noise" actions. Just as WATCH's process of inductive generalization needs heuristic guidance to control its search for a plausible strategy, so does NEWWATCH's task. NEWWATCH must heuristically guess when an interruption has occurred, to find a reasonable strategy intended by the expert.

4.3 A sample interrupted action sequence

Any observed sequence of actions can be explained in two different ways. All of the actions may have been chosen by a single decision in the hierarchical strategy; this would allow the uninterrupted sequence to be generalized into a single explanation. Alternatively, some sub-sequence of the actions may have been caused by a secondary strategy. In this case, the primary strategy can only be properly learned if this interruption is omitted from the observed sequence.

For example, consider a simple sequence of four actions.

(1) *Anchor* fabrication-yard *to* power-plant
(2) *Anchor* parking-lot *to* power-plant
(3) *Yoke* parking-lot *and* fabrication-yard
(4) *Anchor* construction-offices *to* power-plant

This sequence could be generalized into a single abstraction, without hypothesizing the existence of any interruptions. The non-interrupted strategy specifying

Position Large object

would choose the actions in the observed order, assuming that the modifier Large ranked the objects fabrication-yard, parking-lot, and construction-offices in decreasing order. No distinction would be made between the Anchor and Yoke type actions, since they are both specifications of Position.

There is also an alternate strategy that could have been intended by the expert in selecting this action sequence. The Yoke sequence may be an anomaly in an otherwise coherent set of actions. Omitting this interruption would allow the remaining three Anchor actions to be generalized into a more specific decision specifying

Anchor Large object *to* power-plant.

In other words, the observed sequence of four actions would be explained by two strategies working concurrently. The primary decision would select Anchor operations, while some alternate strategy would cause Yoke interruption to be suddenly performed.

4.4 The NEWWATCH program

NEWWATCH is written as a modification to the WATCH application within BB1. It is intended to have all the functionality of the original WATCH program, with the additional ability to recognize and classify interruptions in the action sequence.

NEWWATCH discovers interruptions using a three step process. It first uses a set of fixed rules to hypothesize when an action or sequence of actions might be an interruption. Secondly it integrates these hypotheses into WATCH's normal inductive generalization, by rating them against the regular uninterrupted sequences. Finally it classifies the type of strategy that might have caused the interruption to occur. In other words, NEWWATCH's first phase generates the set of possible interruptions, its second phase uses heuristics to guide the search through the feasible interrupted strategies, and the final phase classifies the strategies that might have caused the reasonable interruptions.

4.4.1 Hypothesizing an interruption

A sequence of one or more actions is hypothesized to be an interruption if it satisfies a set of three necessary conditions.

1. There is some current strategy generalized from the sequence's predecessors which does not explain the actions in the sequence.
2. This current strategy does explain the sequence's immediate successors.
3. The first successor action was already executable when the interruption sequence occurred.

The criteria are based on the definition of an interruption as a subsequence of actions within the observed action sequence that is not explained by the primary hierarchical strategy. Since the hierarchical strategy is the goal of WATCH's learning rather than a known fact, the hypothesis of an interruption must be based on the strategies learned by the inductive generalization. NEWWATCH must wait for the incremental learning to be completed for the sequence's immediate predecessors, before it uses these results to hypothesize the interruption. NEWWATCH can then collect all the generalizations of the predecessor actions found by the inductive generalization up to this point. These generalizations are effectively just the last phases of all the possible strategies at this stage of the incremental learning.

Any generalization of the predecessor actions which does not explain the sequence might be an interrupted strategy, since it satisfies the first condition for interruptions. However, this strategic decision is only interrupted if it is still active after the interruption has occurred. The second and third criteria together guarantee that some later actions could have been selected by this strategy, had the interruption not been performed instead. The second condition insures that the successor action could have been specified by the interrupted strategy, since the strategy is one possible explanation of it. In addition, the third criterion checks that the first successor was executable when the interruption occurred, and so was a valid alternative that could have been executed had the interruption not interfered.

The interruption sequence is the set of actions that falls between predecessor and successors which are explained by a current strategy.

NEWWATCH begins the process of finding such a sequence by finding the first action not explained by a possibly interrupted strategy. The interruption sequence is completed by the longest continuous sequence of actions immediately following the first that are also not explained by the strategy. The second criteria insures that there is some successor action that is explained by the strategic decision; this action provides the end boundary for the interruption sequence. If there were no such successor action, the second criteria would fail. This would be logical, since there would be no way to discriminate whether these actions are an interruption or simply a new phase of the hierarchical strategy.

The three criteria are necessary conditions for the existence of an interruption. However, they limit NEWWATCH to discover only those interruptions which break up a distinct phase of the strategy. An interruption can only be discovered if it comes between predecessor and successor actions that were both caused by a single strategic decision. An interruption that occurs between two different strategic decisions can be recognized only if it interrupts a single abstract parent of these decisions in the hierarchical strategy tree.

4.4.2 Integrating the interruption

Every sequence of actions that satisfies NEWWATCH's criteria for interruptions is hypothesized to be a possible interruption. Most were not chosen by the expert using some alternative strategy to the primary hierarchy, but rather are just spurious. Therefore, NEWWATCH must intelligently choose among the hypotheses. Heuristic rules are used to search through the set of possible interruptions, just as WATCH prunes its space of possible generalizations of sub-sequences. These rules are applied by NEWWATCH's second phase of finding an interruption, the task of integrating the interruption into the inductive generalization process.

NEWWATCH's first phase creates a set of special action sequences that can be integrated into the generalization process. After finding a strategy that might have been interrupted by a sequence of actions, NEWWATCH builds a sequence that omits the interruption. This sequence is simply the longest contiguous set of statements preceding and following the interruption that would be explained by the interrupted strategy. The interrupted strategy and interruption-omitted

sequence are found by NEWWATCH's first phase after the incremental learning has been done for the actions preceding the interruption. However, the hypothesis is not integrated by NEWWATCH's second phase until the incremental learning has progressed further.

Once the incremental learning has progressed to the last action in the sequence that omitted the interruption, NEWWATCH tries to integrate the hypothesis into WATCH's normal generalization. This is possible because the special interruption-omitted sequence is similar to a normal sequence postulated by WATCH; it simply has a gap where the actions belonging to the interruption have been removed. Therefore, the interruption-omitted sequence is just treated as an extra alternative generalization of the new action.

NEWWATCH does not have its own special set of heuristics for rating sequences that omit possible interruptions. Instead it uses existing preferences in WATCH that rate generalizations of sub-sequences. For example, a hypothesized interruption-omitted sequence and a normal uninterrupted sequence both are ranked by the learning heuristic to "avoid generalizing different actions together." Either could be found preferable, depending on the specific actions in the sequence.

In the sample interrupted sequence, a single Yoke action might have been an interruption to a strategy specifying Anchor operations. A special sequence that hypothesized that the Yoke was an interruption and omitted it would be rated fairly highly by the heuristic to "avoid generalizing different actions together"; all the remaining actions in the interruption-omitted sequence would have the same Anchor verb. In contrast, the normal WATCH generalization would treat all the actions as a single uninterrupted sequence of positioning actions, and would not appear as favorable since Anchor and Yoke would have to be abstracted together to the Position verb. Therefore, if this were the only heuristic active at the time, NEWWATCH would determine that an interruption had taken place, and would use the interruption-omitted sequence in the inductive generalization process.

The consistency of having only a single set of learning heuristics has obvious advantages. NEWWATCH can directly compare uninterrupted sequences with interruption-omitted sequences, and so it can integrate only

those hypothesized interruptions that are judged better than the normal alternatives. In addition, since NEWWATCH uses a single set of learning heuristics for sequences with or without interruptions, its preferences are easier to examine and modify. However, NEWWATCH cannot take advantage of any extra knowledge about interruptions, since it does not have special preferences for this unique case.

4.4.3 Categorizing the interruption

NEWWATCH tries to recognize interruptions to improve its ability to learn the primary hierarchical strategy intended by the expert. At the same time, it can learn about the alternate strategies that selected the interrupting actions. Since these secondary strategies are part of the overall plan that the expert had in mind, NEWWATCH is learning the expert's intentions more closely if it also identifies these concurrent strategies.

An expert may use two different combinations of strategies to select an interrupted action sequence. In the first case, the observed action sequence may have been chosen by two or more *parallel* strategies, active concurrently during the problem-solving process. In other words, the action sequence contains a mix of actions selected by different strategies. NEWWATCH must learn by considering one strategy the primary hierarchy, and treating actions selected by other strategies as interruptions, regardless of their strategy type.

The interrupted sequence may also have been chosen in a second fashion. The expert may have been primarily selecting actions based on a single hierarchical strategy, but occasionally choosing *opportunistic* actions to perform. In other words, the control strategy contains an opportunistic knowledge source, in addition to the hierarchical plan. Opportunistic KSes operate by causing a special control decision to become active when the BB1 system state satisfies certain requirements. This decision on the Control Plan dramatically improves the ratings for certain preferred actions, perhaps overriding other active strategies. The actions usually have preconditions similar to those in the control KS, so that they become executable at the same time. As a result, opportunistic actions are performed quite suddenly, usually just as they become executable within the BB1.

NEWWATCH classifies its hypothesized interruptions into two categories: parallel and opportunistic. The interruption sequences are differentiated by examining when their actions became executable, and when they were selected to be executed by BB1. Opportunistic control decisions usually override other strategies to immediately select special actions, so interruption sequences that are performed immediately after becoming executable are likely opportunistic. On the other hand, if the actions in the interruption sequence were executable for a while but were only performed after other actions, then they were probably chosen by a parallel strategy. This concurrent strategy either was not active or did not rate its preferences high enough at earlier points in the problem-solving to perform these executable actions sooner.

4.5 Limitations

The NEWWATCH extension to WATCH has a number of limitations. Most importantly, it still makes strong assumptions about the types of strategies that might appear. While is not limited like WATCH to only a single hierarchical explanation, NEWWATCH cannot recognize such strategies as goal-directed reasoning. However, the range of strategies that can be discovered is fairly broad.

NEWWATCH often has difficulty finding the correct interruption, since the range of possibilities is quite large. For an interruption of a strategic decision to be discovered by NEWWATCH, WATCH must first inductively generalize some similar strategy. If the needed generalization is not one the the possibilities chosen heuristically by WATCH, the interruption will not be found. In the case of multiple interruptions caused by a parallel strategy, NEWWATCH may have trouble selecting the main hierarchy to exclude the noise actions.

4.5.1 Dependence on inductive generalization

The hypothesis of interruptions is based on possible hierarchical strategies found during the incremental learning process. Therefore, if WATCH does not find a certain strategic decision as one of the possible inductive generalizations, NEWWATCH cannot notice an interruption to this strategy. Such a problem

usually occurs because the generalizations are at the wrong level of abstraction to be helpful in discovering interruptions.

NEWWATCH will only discover an interruption when WATCH learns the needed generalizations of the preceding actions. For example, the following sequence may represent three Anchor actions chosen by a single strategic decision, with an interrupting Yoke action.

(1) *Anchor* fabrication-yard-1 *to* power-plant-1 *in* pa1
(2) *Anchor* parking-lot-1 *to* power-plant-1 *in* pa1
(3) *Yoke* parking-lot-1 *and* fabrication-yard-1 *in* pa1
(4) *Anchor* construction-offices-1 *to* power-plant-1 *in* pa1.

The interruption will be correctly hypothesized if the first two actions are generalized to the intended interrupted strategy,

Anchor object *to* power-plant-1 in pa1.

This strategy explains the succeeding Anchor action but not the Yoke interruption. Therefore, if it is one of the generalizations found by WATCH, the interruption can be found.

WATCH might not learn the necessary generalization of the first two actions. For example, the two Anchors may each be abstracted to a separate strategic decision. In this case, the generalization of the single predecessor is

Anchor parking-lot-1 *to* power-plant-1 *in* pa1.

This decision is too specific to explain the interruption's successor action, so there is no continuing strategy that the Yoke might have interrupted. On the other hand, WATCH may only find generalizations that are not specific enough for the interruption to be properly discovered. For example, the first two Anchor actions may have been combined other preceding actions to induce a more general abstraction such as

Position object in pa1.

Since Yoke is a Position type action, the interruption is explained by this strategy, and therefore cannot be considered an anomaly.

WATCH always does the maximally specific abstraction, to avoid throwing away potentially useful information by over-generalizing. However, for the inductive generalizations to be useful in discovering interruptions, abstracting out of specific individual and instance objects is usually necessary. The expert could not have intended a strategy that mentions parking-lot-1, since

control decisions are normally written in terms of static objects, rather than dynamic problem-specific instances. In addition, while strategies may intentionally mention specific individual objects rather than classes of objects, this is not common since it would limit their applicability. Therefore, NEWWATCH assumes that it can abstract the strategy generalizations to object classes. This increases its chance of finding interruptions, since the generalizations are more likely to explain only the appropriate actions.

4.5.2 Use of look-ahead

WATCH was designed as an apprenticeship style learner that could induce a strategy as the expert solved a problem. Therefore, it performs its main task, the inductive generalization, in an incremental fashion one action at a time. NEWWATCH breaks with this model by looking ahead in the action sequence when hypothesizing interruptions.

NEWWATCH's look-ahead is possible because the WATCH system is currently used in batch mode. First the application problem is solved by the expert, and the entire action sequence is stored. Then WATCH is run to learn a strategy based on the trace of the actions. The batch style operation is done simply for practical reasons, such as speed of operation. Therefore, WATCH is written so that it can be easily modified to be an true incremental learner.

NEWWATCH could also be modified to conform to the incremental model. Instead of hypothesizing an interruption when it first appears by using look-ahead, it could wait until later in the action sequence to make the hypothesis retrospectively. NEWWATCH is not currently implemented in this fashion, because its three step operation is more intuitive. Interruptions are hypothesized when they first appear, but are not integrated and classified until the appropriate later time when more of action sequence has been generalized.

4.5.3 Lack of special heuristics

NEWWATCH does not contain a set of heuristic preferences to guide the discovery of interruptions. Instead it merely uses the standard WATCH heuristics to integrate interruptions into the inductive generalization process.

While this single set of learning preferences is conceptually clean and easy to modify, it prevents NEWWATCH from using any specialized knowledge about interruptions.

NEWWATCH hypothesizes interruptions using a set of fixed criteria. However, this could be augmented by a set of heuristics to rate the quality of the interruption sequence. Some of these could be similar to those used in WATCH, although preferences such as "prefer longer sequences of actions" do not necessarily apply here. New heuristics based on additional information could also be added. NEWWATCH could take into account when the interruption's actions became executable, how many interruptions were hypothesized for the same actions, and how different the interruptions were from the strategies during which they occurred. The hypotheses would then have different ratings, which would determine whether to integrate them into the generalization.

4.5.4 Learning of alternate strategies

NEWWATCH currently categories all interruptions into one of two types, parallel and opportunistic. However, it does not attempt to learn more about these alternate strategies. Therefore, even if the primary hierarchical strategy closely resembles that intended by the expert, the usefulness of the strategies learned by NEWWATCH is limited. Without a better description of the full set of concurrent strategies, the performance of the expert cannot be attained.

NEWWATCH could perform a number of tasks to learn more about secondary strategies. Each interruption could be individually given as input to WATCH, to discover whether the actions comprise a secondary hierarchical strategy. If no reasonable explanation was found, the interruptions would have to be assumed to be chosen randomly or by some new strategy type, such as goal-directed reasoning.

Additional learning could be based on the classified type of the interruption. NEWWATCH's recognition of an opportunistic strategy would be improved if it knew what state of the BB1 system caused it to occur. While this could not be completely inferred without knowing the BB1 system state at the moment the strategy began, deeper knowledge about the language actions might

provide some possibilities. For example, the Restrict action in ACCORD is usually chosen by an opportunistic decision because it is used to limit the possible locations of objects when the alternatives are too numerous. Knowing this background about the action would indicate that the strategy is triggered by the part of the system state involving the hypothesized locations of an object.

If a single action sequence has multiple interruptions classified as parallel strategies, NEWWATCH could attempt to combine their actions to learn the secondary strategies. Multiple interruptions may occur during a single strategy generalization, or may have similar actions. By assuming that a single parallel strategy explained all of these anomalous actions, they could be treated as a continuous sequence to allow a possible hierarchical explanation to be learned by WATCH.

5. Evaluation

5.1 Introduction

The NEWWATCH extension was motivated by the Lac-repressor headpiece problem in PROTEAN. The action sequence was selected by a straightforward hierarchical strategy, except for a single action chosen opportunistically. This one action seriously limited WATCH's ability to correctly learn the intended strategy. NEWWATCH was designed to recognize such interruptions, to improve the range of strategies that could be learned.

Testing of the NEWWATCH program has been done in a second application domain, the SIGHTPLAN problem. To insure that NEWWATCH was learning real strategies rather than invented ones, it has been tested on action sequences chosen by existing control strategies for SIGHTPLAN. The gold-standard for WATCH's output is to rediscover the actual strategy. However, this is merely necessary for the validation of NEWWATCH. The system's intended use is as a knowledge acquisition tool that will automatically learn unknown strategies by observing an expert at work.

NEWWATCH has also been tested against human experts, to decide what learning was ideally possible. The subjects were all individuals quite familiar with using the BB1 architecture. For each problem the experts were provided with the action sequence, a type hierarchy for the language and application, a list of ACCORD action templates, and a description of the available modifiers and their ratings. The goal was to find an explanation for the actions and their ordering, although no prior prejudice about types of strategies was defined.

5.2 Test cases

Three sample test cases are presented here. The first is the original problem from PROTEAN, while the latter two are from the SIGHTPLAN domain. The interesting features of each are described in a short introduction. Then the full observed action sequence is listed, followed by the actual control strategy that selected those actions.

The results of the strategy learning by WATCH and NEWWATCH are shown for each test problem, to demonstrate the advantage of recognizing interruptions. First, the strategies learned by the original WATCH program for each action sequence are presented. Then, the new strategies learned with the NEWWATCH extension are listed. Various intermediate results of NEWWATCH, such as the interruptions that are hypothesized, are also shown.

The strategies are all represented as hierarchical trees, with the most abstract parent decisions on the left-hand side. The trees all unfold from the top down, so the uppermost leaf strategies are the first to control the selection of actions. Each focus leaf is linked to the actions in the sequence that it selected, to make it clear which decision caused each of the actions. The strategies in the tree are represented by ACCORD sentences with the action verbs capitalized, if such sentences were used. Strategic decisions that were not expressed in ACCORD are simply shown as abstract descriptions written in small letters.

5.2.1 Sample problem 1

Application domain: PROTEAN

Problem: LAC
Lac-repressor headpiece protein

Strategies: Assemble one partial-arrangement:
hierarchical strategy
actions 1-8, 10-15

Restrict locations:
simple opportunistic decision
action 9

Problem features:

- All actions are selected by a single hierarchical strategy, except for a special Restrict action. This opportunistic Restrict is selected when BB1 is in a state where one of the objects, in this case Helix2-1, has too many possible locations.

- The opportunistic Restrict action is the simplest possible type of interruption. It consists of only a single action, and this action is based on the Restrict verb that is unique within the action sequence.

Results of WATCH:

- WATCH's generalization is confused by the presence of the Restrict action. It incorrectly groups the positioning actions, because it does not notice the Anchor-Yoke-Anchor-Yoke pattern for the two different types of objects, helices and randomcoils.

Results of NEWWATCH:

- NEWWATCH hypothesizes a number of spurious interruptions when it examines the Include actions. All including actions are chosen by a single strategic decision, which does not specify any particular ordering among them. However, NEWWATCH notices

alternate strategies that divide the inclusion steps into those mentioning helix or randomcoil type objects.
NEWWATCH hypothesizes that the inclusion of randomcoil2 by action 4 is an interruption to a strategy indicating helices for actions 2, 3, and 5. The same is done in reverse for the spuriously generalized strategy to include randomcoils (actions 4 and 6) that is believed to be interrupted by a helix (action 5).
Fortunately, the incorrectly hypothesized sequences are rated lower than the uninterrupted sequence that includes all the objects. This is logical, since the longer sequence with all the objects is not much more abstract that the sub-sequences. Therefore, the spurious interruptions are not integrated into the generalization.

- Because NEWWATCH notices that Restrict (action 9) interrupts two Anchor actions (actions 8 and 10), it correctly groups the Anchors together. This phase can then be combined with the Yoke operation (action 11), to discover the Position helices strategy. As a result, the learned hierarchical strategy properly distinguishes the separate phases of positioning for helices and randomcoils. NEWWATCH also appropriately labels the unique Restrict action as opportunistic when it integrates the interruption hypothesis into the generalization.

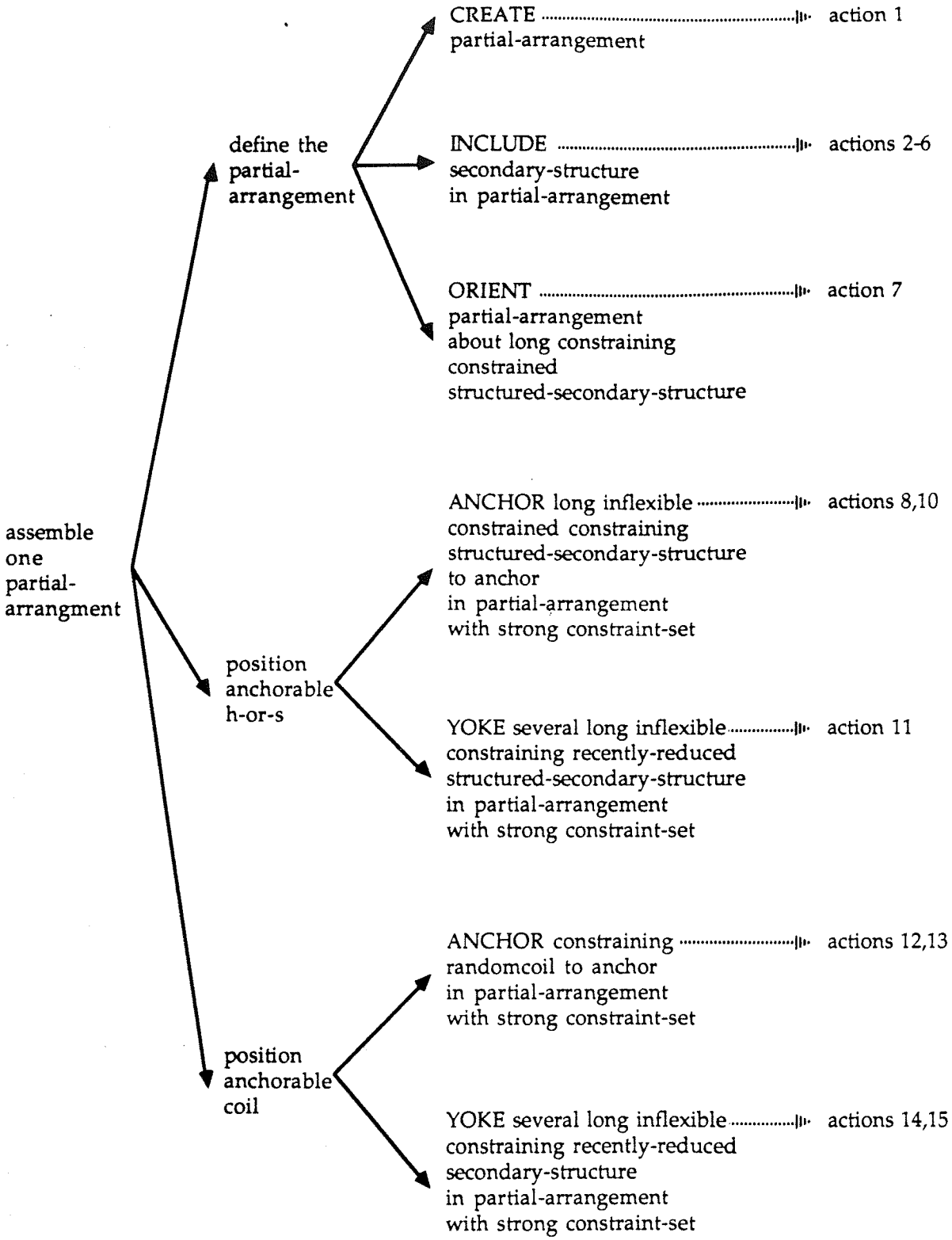
Results of human subjects:

- None of the human subjects was hindered in the task of finding the main strategy by the existence of the anomalous Restrict action. One individual was not familiar with ACCORD Restrict verb, but assumed simply from its name that it was doing something opportunistic in "restricting the problem-solving". Another ignored the action simply because it was unfamiliar and confusing, and went on to learn from the remaining actions; at the end he merely wondered why only this single Restrict action occurred. None of the subjects felt that there was enough information to find out more about this action.

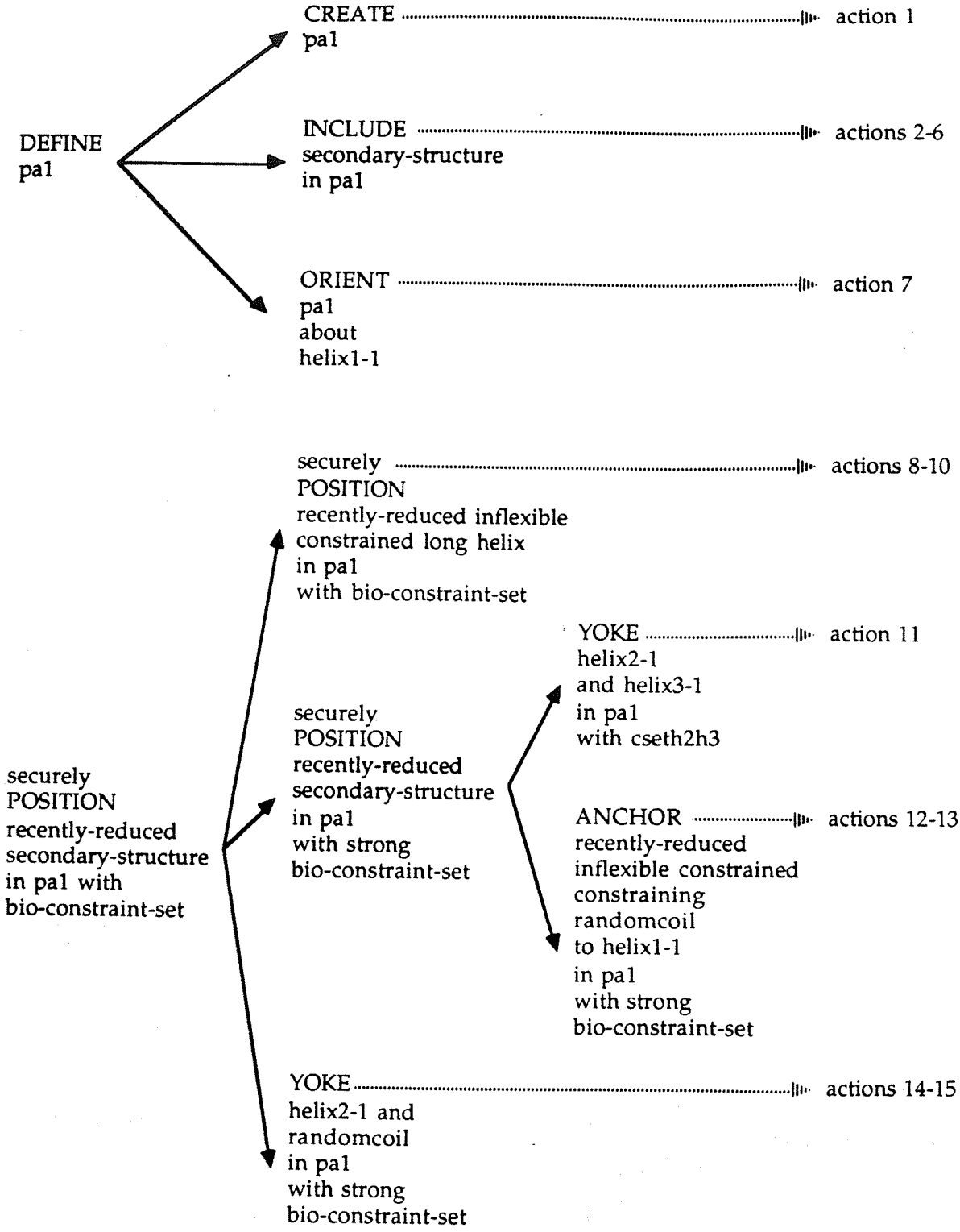
Sample problem 1: Observed action sequence

1. Create pa1
2. Include helix3 in pa1
3. Include helix2 in pa1
4. Include randomcoil2 in pa1
5. Include helix1 in pa1
6. Include randomcoil1 in pa1
7. Orient pa1 about helix1-1
8. Anchor helix2-1 to helix1-1 in pa1 with cseth1h2
9. Restrict helix2-1 in pa1 with cseth2
10. Anchor helix3-1 to helix1-1 in pa1 with cseth1h3
11. Yoke helix2-1 and helix3-1 in pa1 with cseth1r2
12. Anchor randomcoil2-1 to helix1-1 in pa1 with cseth1r2
13. Anchor randomcoil1-1 to helix1-1 in pa1 with csetr1h1
14. Yoke randomcoil2-1 and helix2-1 in pa1 with csetr2h2
15. Yoke randomcoil1-1 and helix2-1 in pa1 with csetr1h2

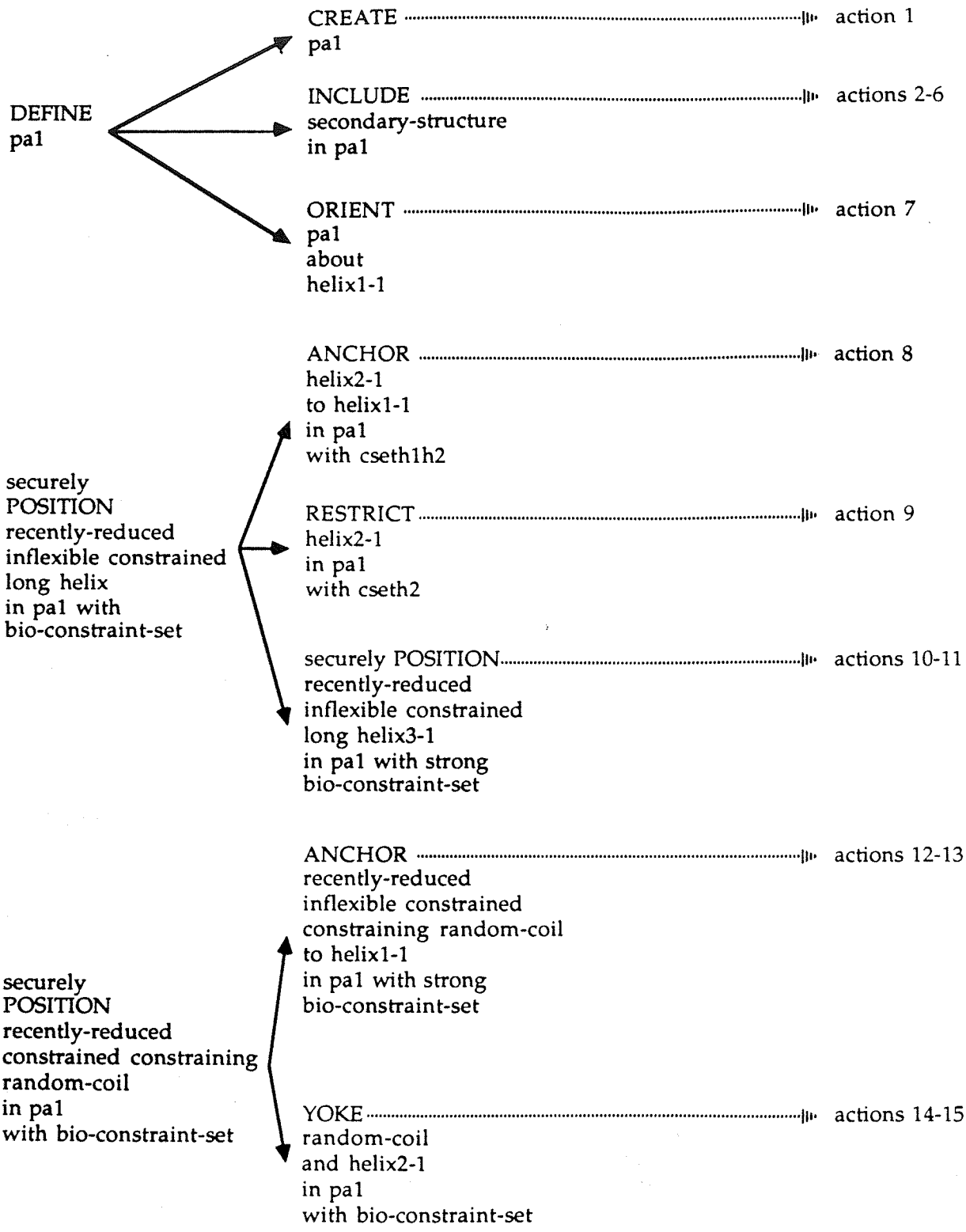
Sample problem 1: Actual control strategy



Sample problem 1: Strategy learned by WATCH



Sample problem 1: Strategy learned by NEWWATCH



Sample problem 1: Spurious hypothesis of an interruption

Statement 4

INCLUDE randomcoil2 in pa1

is the first and only action in a parallel-type interruption of strategy

INCLUDE helix in pa1

which explains the sequence of actions

2, 3, 5.

```
INTERRUPTION1 is at the LEARN.INTERRUPTION level.

        Attributes:
             ACTION-STMT:  (ACCORD.EVENT.DID-INCLUDE PROBLEM.SOLID.RANDOMCOIL2 IN
                           SOLUTION.PARTIAL-ARRANGEMENT.PA1)
  INTERRUPTED-SEQUENCES:  (((((LEARN.LEV0-S.LEV0-S2 . 1)) ((LEARN.LEV0-S.LEV0-S3 . 1))
                           ((LEARN.LEV0-S.LEV0-S5 . 1))) PARALLEL))
INTERRUPTED-STRATEGIES:  (((ACCORD.EVENT.DID-INCLUDE) (BIO.CONCEPT.HELIX)
                           (SOLUTION.PARTIAL-ARRANGEMENT.PA1)))
      LAST-CHANGE-TYPE:  HYPOTHESIZE-INTERRUPTION
            PARSED-VAL:  ((ACCORD.EVENT.DID-INCLUDE) (PROBLEM.SOLID.RANDOMCOIL2)
                           (SOLUTION.PARTIAL-ARRANGEMENT.PA1))
          STMT-NUMBER:  4

           Links: <none>

          $Links: <none>
```

Sample problem 1: Correct hypothesis of an interruption

Statement 9

RESTRICT helix2-1 in pa1 with cseth2h3

is the only action in an opportunistic-type interruption of strategy

ANCHOR helix to helix in pa1 with bio-constraint-set

which explains the sequence of actions

8 and 10.

```
INTERRUPTION3 is at the LEARN.INTERRUPTION level.

          Attributes:
               ACTION-STMT:  (ACCORD.EVENT.DID-RESTRICT SOLUTION.SOLID.HELIX2-1 IN
                             SOLUTION.PARTIAL-ARRANGEMENT.PA1 WITH PROBLEM.CONSTRAINT-SET.CSETH2R3)
     INTERRUPTED-SEQUENCES:  (((((LEARN.LEV0-S.LEV0-S8 . 1)) ((LEARN.LEV0-S.LEV0-S10 . 1))) OPPORTUNISTIC))
    INTERRUPTED-STRATEGIES:  (((ACCORD.EVENT.DID-ANCHOR) (BIO.CONCEPT.HELIX) (BIO.CONCEPT.HELIX)
                             (SOLUTION.PARTIAL-ARRANGEMENT.PA1) (BIO.CONCEPT.BIO-CONSTRAINT-SET)))
          LAST-CHANGE-TYPE:  HYPOTHESIZE-INTERRUPTION
                PARSED-VAL:  ((ACCORD.EVENT.DID-RESTRICT) (SOLUTION.SOLID.HELIX2-1)
                             (SOLUTION.PARTIAL-ARRANGEMENT.PA1) (PROBLEM.CONSTRAINT-SET.CSETH2R3))
               STMT-NUMBER:  9

             Links: <none>

            $Links: <none>
```

5.2.2 Sample problem 2

Application domain: SIGHTPLAN

Problem: AM1-SIMPLE
American1 power plant, simple version

Strategies: Architect-engineer layout:
hierarchical plan, with one pair of strategic decisions operating in parallel

Problem features:

- While all of the lowest level strategic foci of the hierarchical plan are expressed as ACCORD sentences, the more abstract parent decisions are not. They are simply general goals, which are refined into the ACCORD decisions that will actually select the actions to perform. Therefore, any abstract strategies that WATCH learns will be spurious. While they might be ACCORD equivalents of the general goals, they are not the actual decisions in the existing strategy.

- SIGHTPLAN uses the ACCORD positioning verbs such as Anchor, Yoke, and Append in an unusual fashion. Rather than basing the strategy on the specific features of these different actions, it simply mixes them up randomly. This is because most of the constraints do not mention the fixed anchor of the construction site; therefore, the possible locations of objects are not propagated from the anchor, through the anchorees, to the appendage objects. Instead, the order of positioning actions is determined by the objects and constraints involved. The fact that different ACCORD verbs appears is not significant. It merely confuses WATCH and the human subjects.

- Most of the positioning actions (actions 15-28) are selected by two strategic foci operating in parallel. One focus specifies that objects should be positioned on the site; the other updates the record of

which objects have been fixed and now occupy space. The actions of the two are very hard to differentiate. Both use the same verbs, such as Yoke; there is also much overlap in the objects mentioned. In addition, the two foci don't alternate in any recognizable pattern, such as one action each, since they are performed dynamically based on the state of the problem.

Results of WATCH:

- WATCH incorrectly groups all the Include actions into a single strategic decision, since its heuristics lead it to generalize actions with common verbs together. The various sub-sequences of Include actions that mention specific types of objects are less preferable since they are shorter sequences, and yet not much less abstract.

- Because the actions selected by the parallel strategies do not fall into any regular pattern of alternation, WATCH cannot differentiate them when finding the hierarchical explanation. Any similarity between the strategy learned by WATCH for those actions and the intended plan is merely coincidental.

Results of NEWWATCH:

- NEWWATCH does not hypothesize any interruptions for this problem, so it obtains the same results as WATCH. While there are a number of interrupting actions caused by the parallel foci, they are not found by NEWWATCH. The interruptions are quite similar to the main actions, since they use the same Yoke verb and mention the same objects. Therefore, they are difficult to recognize as interruptions since they are explained by the strategies learned in the inductive generalization.

Results of human subjects:

- All of the subjects were extremely confused by the sequence of positioning actions in this problem. All were familiar with ACCORD, and so had a strong prejudice that actions with the same verb should be grouped together. Also, there was an expectation

that Anchors would precede Yokes, rather than just be randomly intermingled with them.

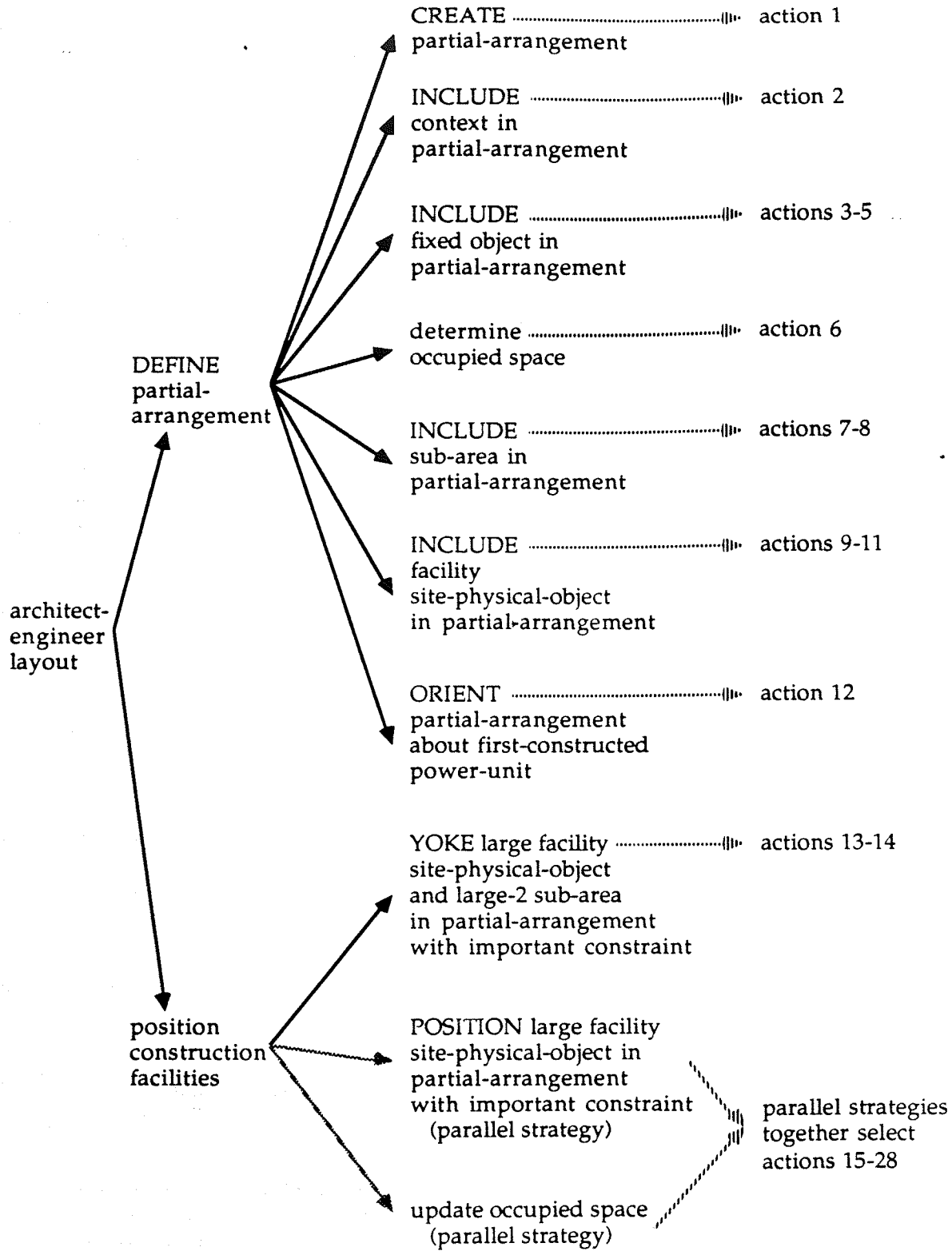
- There was some limited success in recognizing the first part of the "position construction facilities" strategy which Yoked objects to the site's sub-areas. One subject noticed that the first few Yokes contained objects modified by Large-2, the modifier exclusively for sub-areas. Another assumed an overall positioning strategy that preferred Yokes that constrained a space with a physical object over actions that mentioned two physical objects.

- None of the subjects had any success in recognizing the parallel foci. There was no obvious pattern of alternation to be found. Therefore, since the actions selected by the two foci were quite similar, each of the subjects tried to group all of the positioning into a single explanation. Much exasperation was felt since none of the hypothesized strategies really seemed to fit; the general solution was simply to choose some seemingly reasonable possibility and ignore the actions that were not explained by it.

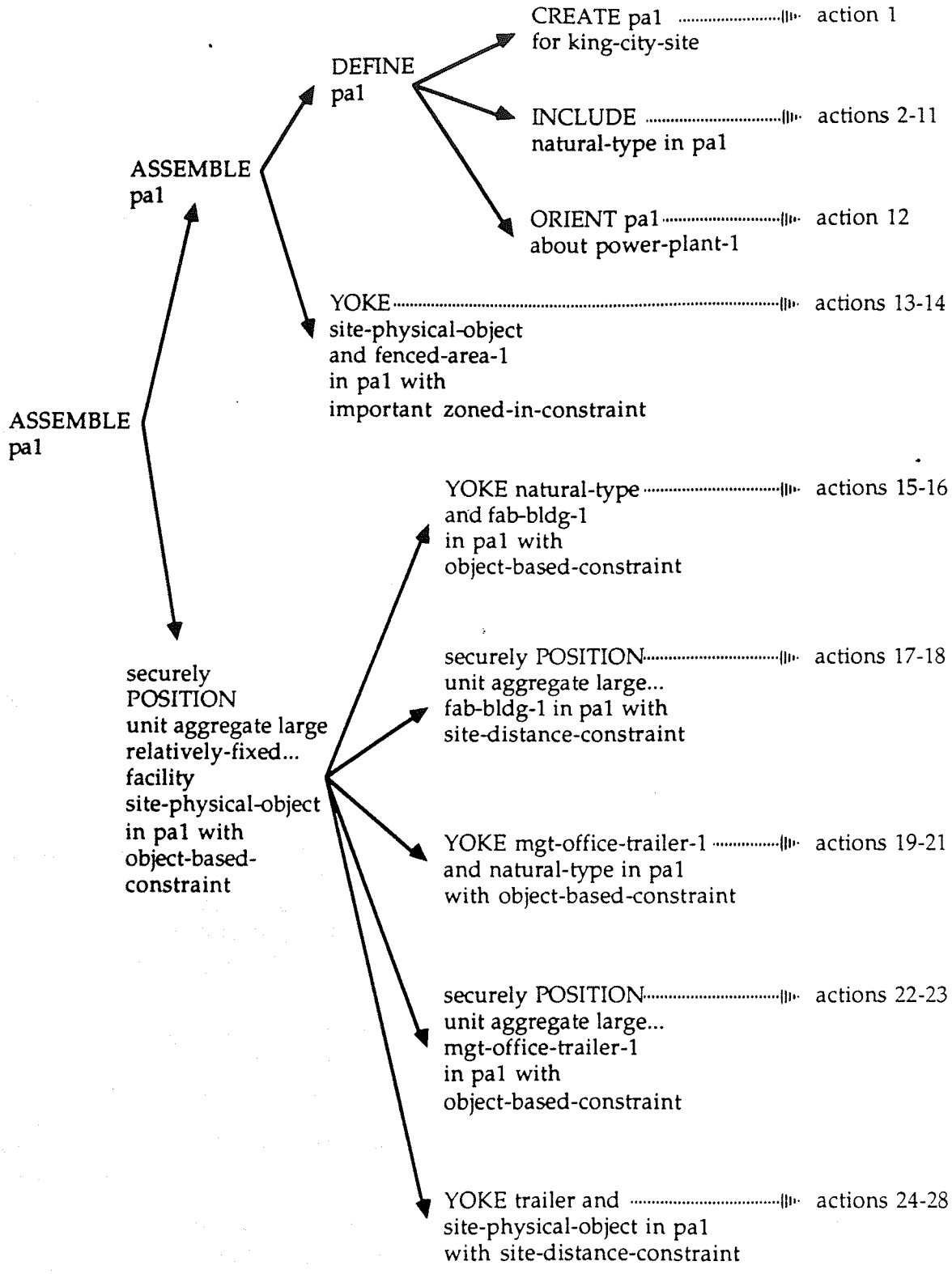
Sample problem 2: Observed action sequence

1. Create pa1 for king-city-site
2. Include king-city-site in pa1
3. Include power-plant in pa1
4. Include gas-turbine in pa1
5. Include plant-entrance-road in pa1
6. Include am1-occupied-space in pa1
7. Include laydown-area in pa1
8. Include fenced-area in pa1
9. Include fab-civil in pa1
10. Include mgt-office-trailer in pa1
11. Include fab-bldg in pa1
12. Orient pa1 about power-plant-1
13. Yoke fenced-area-1 and fab-bldg-1 in pa1 with zoned-in-103-fenced
14. Yoke fenced-area-1 and mgt-office-trailer-1 in pa1 with zoned-in-105-fenced
15. Yoke fab-bldg-1 and am1-occupied-space-1 in pa1 with non-overlap-set-103-occupied
16. Yoke mgt-office-trailer-1 and fab-bldg-1 in pa1 with closer-than-103-105
17. Yoke gas-turbine-1 and fab-bldg-1 in pa1 with closer-than-103-gt
18. Anchor fab-bldg-1 to power-plant-1 in pa1 with as-close-as-103-plant
19. Yoke mgt-office-trailer-1 and am1-occupied-space-1 in pa1 with non-overlap-set-105-occupied
20. Yoke mgt-office-trailer-1 and fab-bldg-1 in pa1 with closer-than-103-105
21. Yoke plant-entrance-road-1 and mgt-office-trailer-1 in pa1 with adjacent-to-105-road
22. Yoke mgt-office-trailer-1 and fab-bldg-1 in pa1 with closer-than-103-105
23. Anchor mgt-office-trailer-1 to power-plant-1 in pa1 with parallel-1-105
24. Yoke mgt-office-trailer-1 and fab-bldg-1 in pa1 with closer-than-103-105
25. Yoke gas-turbine-1 and mgt-office-trailer-1 in pa1 with as-close-as-105-gt
26. Yoke mgt-office-trailer-1 and fab-bldg-1 in pa1 with closer-than-103-105
27. Yoke gas-turbine-1 and fab-civil-1 in pa1 with closer-than-121-gt
28. Yoke plant-entrance-road-1 and fab-civil-1 in pa1 with as-close-as-road

Sample problem 2: Actual control strategy



Sample problem 2: Strategy learned by WATCH and NEWWATCH



Sample problem 3:

Application domain: SIGHTPLAN

Problem: AM1-AGGR
American1 power plant, simple version
with aggregate objects

Strategies: Architect-engineer layout:
hierarchical plan, with one pair of strategic decisions operating in parallel
actions 1-10, 19-27

Aggregate-object layout:
secondary hierarchical plan, performed opportunistically during main architect-engineer strategy
actions 11-18

Problem features:

- The main strategy used in the AM1-AGGR problem is the same as that used by AM1-SIMPLE. However, in addition, there is an opportunistic strategy used to arrange the fabrication-shops aggregate object. The opportunistic strategy is triggered by the inclusion of an object (fab-shops) that has important closeness constraints with other objects; it arranges these collectively and then places them on the main site as an aggregate. The aggregate-object arrangement strategy is itself a hierarchical plan, although somewhat simpler than the main strategy.

Results of WATCH:

- The interrupting strategy that builds the aggregate is partly found by WATCH. The actions that Define the arrangement are correctly grouped. However, while the two Anchor actions are properly explained by a single strategy, they are not connected to the Define

phase. WATCH does not recognize that all the actions that mention the second partial-arrangement are related.

- The presence of the secondary strategy seriously hampers WATCH's learning of the main hierarchy. For example, the three phases of defining the main partial-arrangement are not connected, since the Create and Include actions precede the aggregate actions while the Orient follows it.

Results of NEWWATCH:

- NEWWATCH hypothesizes that the actions building the aggregate object are an interruption. This is noticed since one of the current possible strategies is defining partial-arrangement-1, which later is concluded by the Orient operation in action 19. This strategy does not explain any of the aggregate operations (actions 11-18), so they are hypothesized to be an interruption. The interruption is classified as opportunistic, since the aggregate object is built immediately after the fab-shops is included in the main arrangement.

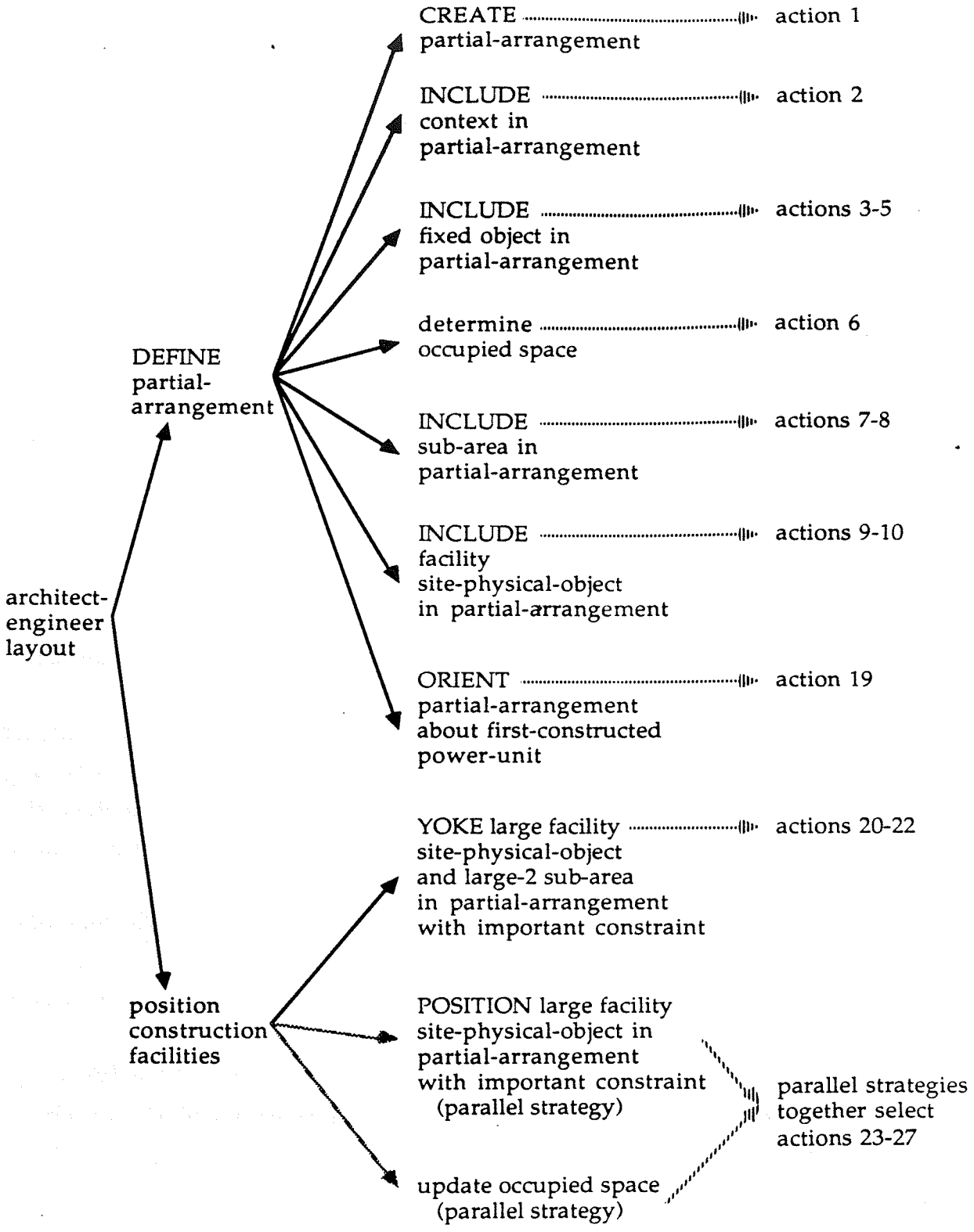
- NEWWATCH does not learn the secondary strategy, but merely recognizes and classifies it. The aggregate strategy is in fact hierarchical, and could be learned independently by WATCH.

- Since NEWWATCH recognizes that the interruption takes place within the high level Define strategy, it assumes that this is the correct generalization. Therefore, it does not refine this strategy to the appropriate Create, Include, and Orient phases for the main arrangement. In addition, placing this high level abstraction at the lowest level of the strategy confuses the final grouping of the hierarchy learned by NEWWATCH. Actions 20-22 (Yoke object to sub-area) are correctly placed in the first half of the strategy, and actions 23-27 (the parallel position actions) are separated into a latter phase. However, most of the higher-level generalizations and groupings within the hierarchical tree are spurious.

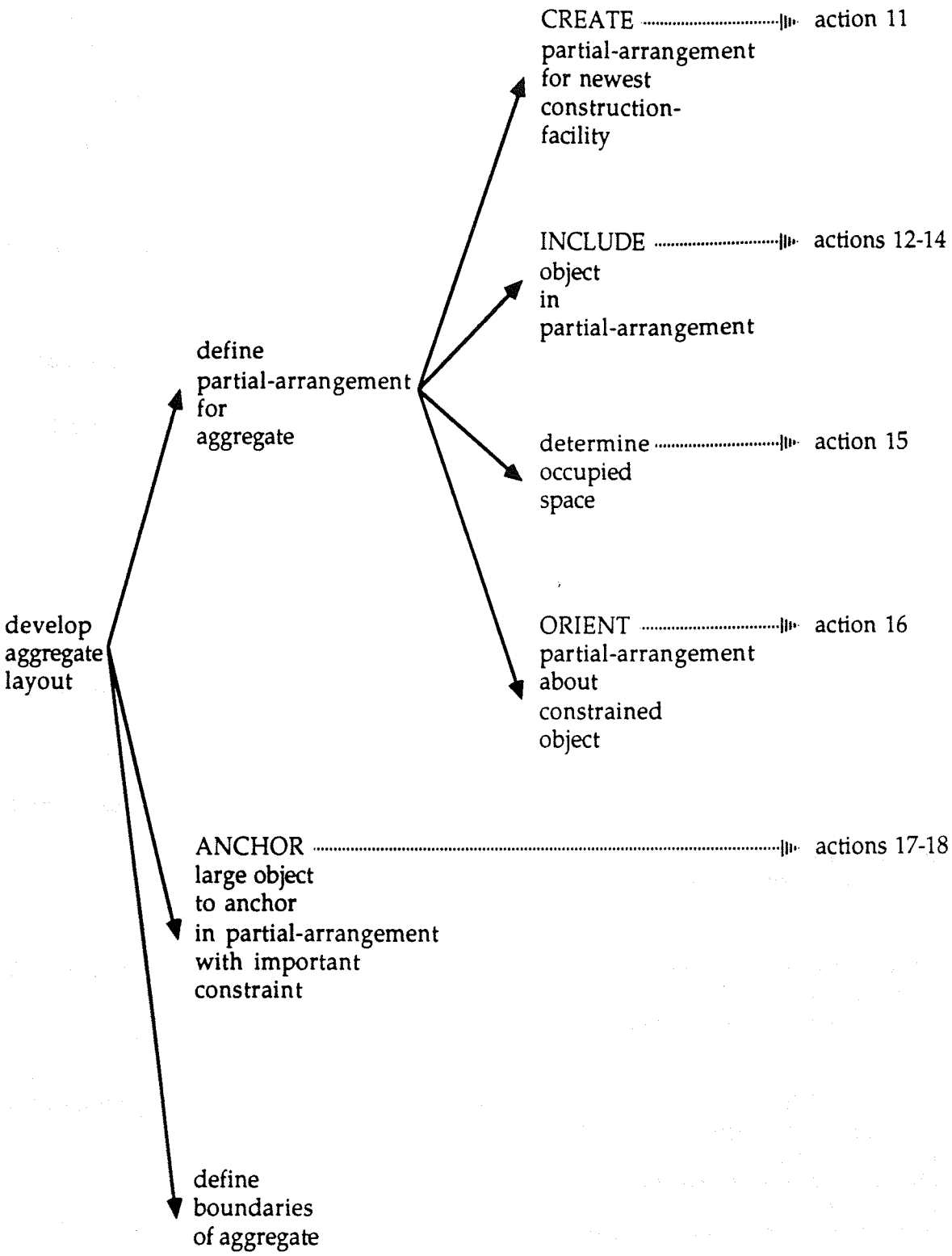
Sample problem 3: Observed action sequence

1. Create pa1 for king-city-site
2. Include king-city-site in pa1
3. Include gas-turbine in pa1
4. Include power-plant in pa1
5. Include plant-entrance-road in pa1
6. Include am1-occupied-space in pa1
7. Include laydown-area in pa1
8. Include fenced-area in pa1
9. Include parking in pa1
10. Include fab-shops in pa1
11. Create pa2 for fab-shops
12. Include fab-electrical in pa2
13. Include fab-civil in pa2
14. Include fab-pipefitters in pa2
15. Include am1-occupied-space in pa2
16. Orient pa2 about fab-civil-2
17. Anchor fab-pipefitters-2 to fab-civil-2 in pa2 with adjacent-to-120-121
18. Anchor fab-electrical-2 to fab-civil-2 in pa2 with adjacent-to-121-122
19. Orient pa1 about power-plant-1
20. Yoke laydown-area-1 and parking-1 in pa1 with zoned-in-104-laydown
21. Yoke fenced-area-1 and parking-1 in pa1 with zoned-out-104-fenced
22. Yoke fenced-area-1 and fab-shops-1 in pa1 with zoned-in-103-fenced
23. Yoke am1-occupied-space-1 and parking-1 in pa1 with non-overlap-set-104-occupied
24. Yoke plant-entrance-road-1 and parking-1 in pa1 with as-close-as-104-road
25. Yoke fab-shops-1 and am1-occupied-space-1 in pa1 with non-overlap-set-103-occupied
26. Yoke gas-turbine-1 and fab-shops-1 in pa1 with closer-than-103-gt
27. Anchor fab-shops-1 and power-plant-1 in pa1 with as-close-as-103-plant

Sample problem 3: Actual control strategy



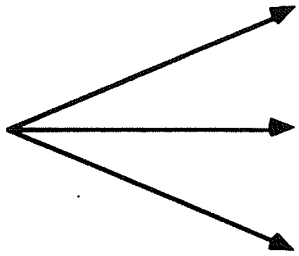
Sample problem 3: Secondary control strategy



Sample problem 3: Strategy learned by WATCH

CREATE pa1 ..... action 1
for king-city-site

INCLUDE ..... actions 2-10
natural-type
in pa1



CREATE pa2 ..... action 11
for fab-shops

INCLUDE ..... actions 12-15
natural-type
in pa2

ORIENT pa2 ..... action 16
about fab-civil-2

ANCHOR ..... actions 17-18
unit aggregate large
unpositioned...
facility trailer
to fab-civil-2
in pa2 with
important
adjacent-to-constraint

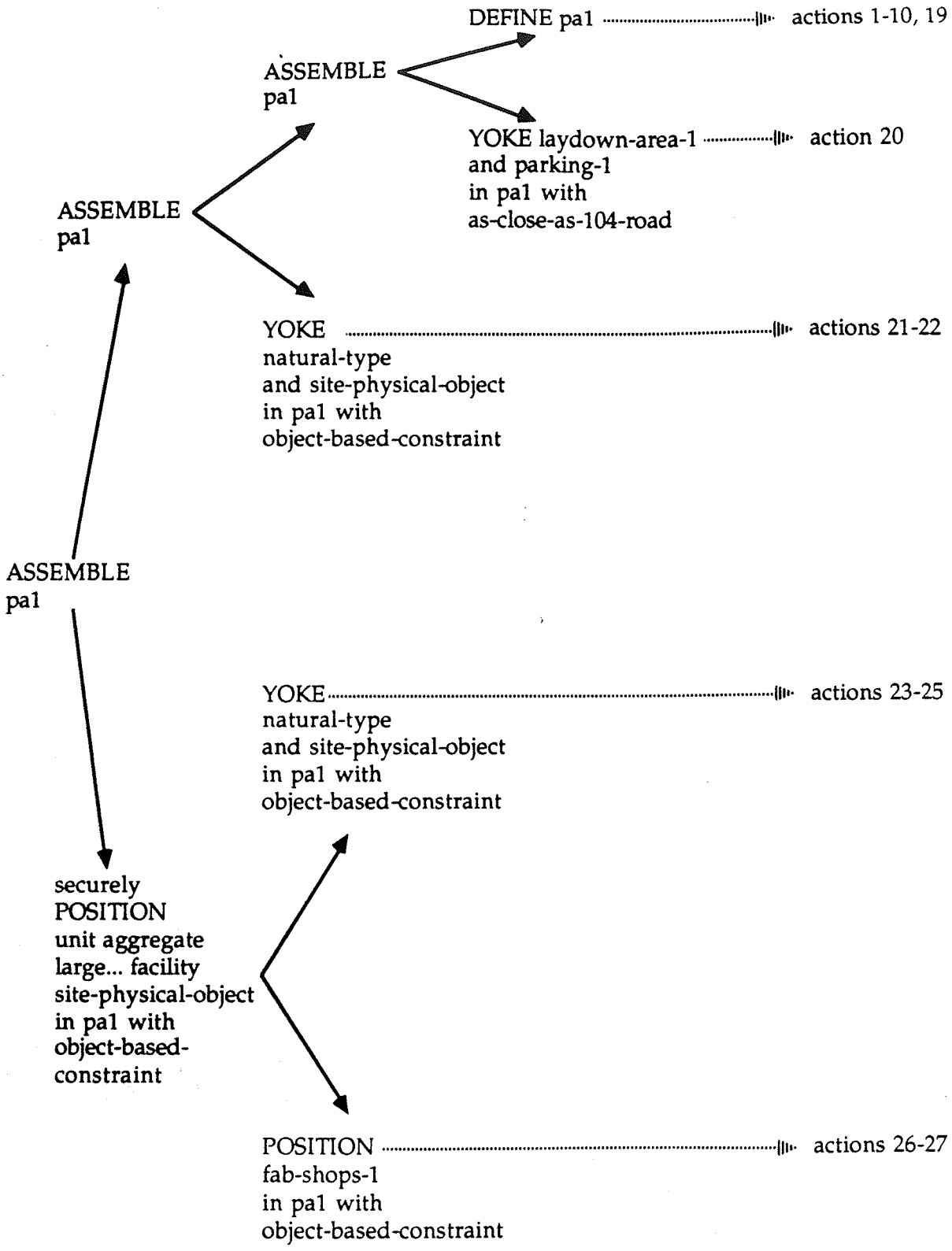
ORIENT pa1 ..... action 19
about power-plant-1

YOKE parking-1 ..... actions 20-21
and sub-area
in pa1 with
important site-zoning-constraint

YOKE site-physical-object ..... actions 22-23
and space in pa1
with object-based-constraint

securely POSITION ..... actions 24-27
unit aggregate large
relatively-fixed...
facility site-physical-object
in pa1 with
important object-based-constraint

Sample problem 3: Strategy learned by NEWWATCH



Sample problem 3: Correct hypothesis of an interruption

Statement 11

CREATE pa2 for fab-shops

is the first action in an opportunistic-type interruption of strategy

DEFINE pa1

which explains the sequence of actions

1-10 and 19.

```
INTERRUPTION1 is at the LEARN.INTERRUPTION level.

        Attributes:
              ACTION-STMT:  (ACCORD.EVENT.DID-CREATE SOLUTION.PARTIAL-ARRANGEMENT.PA2 FOR
                            AMERICAN1.CONSTRUCTION-FACILITIES.FAB-SHOPS)
        INTERRUPTED-SEQUENCES:  (((((LEARN.LEV0-S.LEV0-S1 . 1)) ((LEARN.LEV0-S.LEV0-S2 . 1))
                            ((LEARN.LEV0-S.LEV0-S3 . 1)) ((LEARN.LEV0-S.LEV0-S4 . 1)) ((LEARN.LEV0-S.LEV0-S5 .
                            1)) ((LEARN.LEV0-S.LEV0-S6 . 1)) ((LEARN.LEV0-S.LEV0-S7 . 1))
                            ((LEARN.LEV0-S.LEV0-S8 . 1)) ((LEARN.LEV0-S.LEV0-S9 . 1)) ((LEARN.LEV0-S.LEV0-S10
        INTERRUPTED-STRATEGIES:  . 1)) ((LEARN.LEV0-S.LEV0-S19 . 1))) OPPORTUNISTIC))
             LAST-CHANGE-TYPE:  (((ACCORD.EVENT.DID-DEFINE) (SOLUTION.PARTIAL-ARRANGEMENT.PA1)))
                  PARSED-VAL:  HYPOTHESIZE-INTERRUPTION
                            ((ACCORD.EVENT.DID-CREATE) (SOLUTIO! PARTIAL-ARRANGEMENT.PA2)
                            (AMERICAN1.CONSTRUCTION-FACILITIES.FAB-SHOPS))
                STMT-NUMBER:  11

              Links: <none>

             $Links: <none>
```

```
LEV1-S10 is at the LEARN.LEV1-S level.

        Attributes:
              ACTION-STMT:  (ACCORD.EVENT.DID-DEFINE SOLUTION.PARTIAL-ARRANGEMENT.PA1)
                CONFIDENCE:  2650
            FIRST-STMT-NUM:  1
       GENERALIZATION-LEVEL:  1
           GENERALIZED-VARS:  (1)
             GENERATOR-TYPE:  SEQUENCE-OF-STMTS
    INTERRUPTION-STRATEGY-TYPE:  OPPORTUNISTIC
           LAST-CHANGE-TYPE:  POSTULATE-SEQUENCE-WITHOUT-INTERRUPTION
              LAST-STMT-NUM:  19
          NUMS-OF-SEQUENCE:  (1 2 3 4 5 6 7 8 9 10 19)
                PARSED-VAL:  ((ACCORD.EVENT.DID-DEFINE) (SOLUTION.PARTIAL-ARRANGEMENT.PA1))
              STMT-NUMBER:  10

            Links:
          GENERALIZED-FROM:  LEARN.LEV0-S.LEV0-S1 LEARN.LEV0-S.LEV0-S2 LEARN.LEV0-S.LEV0-S3
                            LEARN.LEV0-S.LEV0-S4 LEARN.LEV0-S.LEV0-S5 LEARN.LEV0-S.LEV0-S6
                            LEARN.LEV0-S.LEV0-S7 LEARN.LEV0-S.LEV0-S8 LEARN.LEV0-S.LEV0-S9
                            LEARN.LEV0-S.LEV0-S10 LEARN.LEV0-S.LEV0-S19
```

6. Conclusions

The NEWWATCH program can learn hierarchical strategies for BB1 application programs, even when the observed action sequence contains interruptions. Rather than treating actions chosen by concurrent secondary strategies as noise, NEWWATCH discovers and classifies these alternate plans.

NEWWATCH's ability to correctly discover interruptions in many ways parallels the skills of human learners. Both are able to recognize that opportunistic sequences are anomalies, and therefore can set aside such actions to learn the main strategy. Like human subjects, NEWWATCH is confused when it tries to learn from action sequences chosen by parallel strategies, particularly if there is no obvious pattern of alternation or qualitative difference in the actions.

The ability to recognize interruptions increases the range of strategies that can be automatically discovered by the WATCH system. Without the limitation of assuming that a single hierarchical explanation is adequate, NEWWATCH has the power to learn real control strategies for such problems as the SIGHTPLAN application.

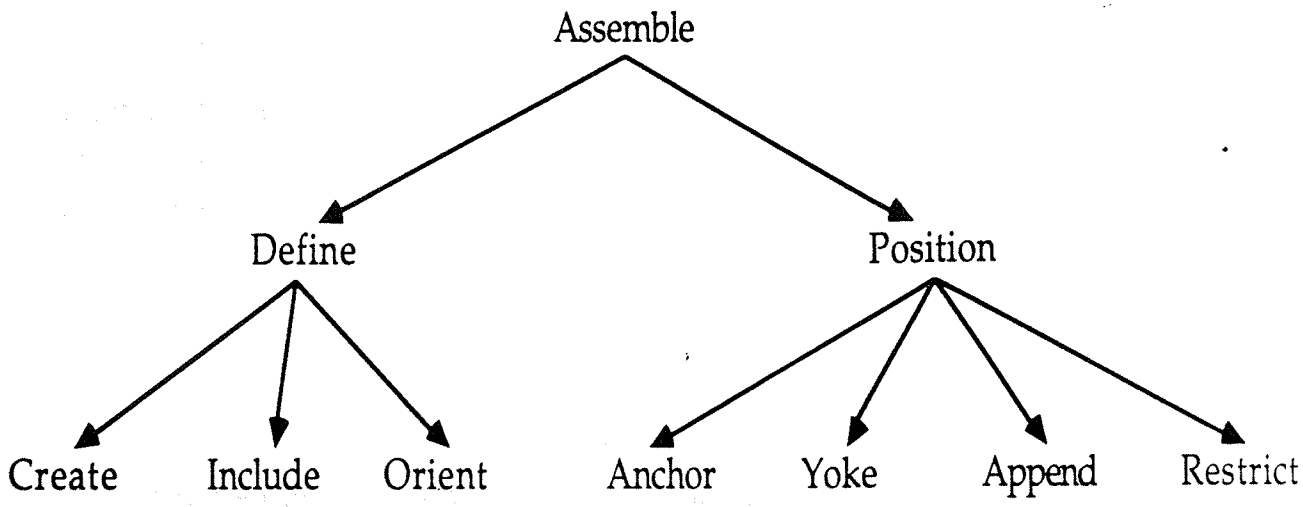
Appendix 1

ACCORD Action Templates

- *Assemble* partial-arrangement
  - *Define* partial-arrangement
    - *Create* partial-arrangement
    - *Include* object *in* partial-arrangement
    - *Orient* partial-arrangement *about* anchor
  - *Position* objects *in* partial-arrangement *with* constraints
    - *Anchor* object *to* anchor *in* partial-arrangement *with* constraints
    - *Yoke* object *and* object *in* partial-arrangement *with* constraints
    - *Append* appendage *and* object *in* partial-arrangement *with* constraints
    - *Restrict* object *in* partial-arrangement *with* constraints

Appendix 2

ACCORD Action Hierarchy



Appendix 3

WATCH generalization heuristics

Avoid generalizing different actions together

- AVOID-GENERALIZING-ACTIONS
- based on BB1 languages with standard action templates
- All actions based on a single standard verb have the same operation, and so are likely to have a common purpose. Therefore, group consecutive actions of a common type into a single phase of the strategy, without any different actions intervening.

Prefer longer sequences of actions

- PREFER-MORE-STATEMENTS
- general to induction over sequences
- A sequence containing more actions with something in common is more likely significant than a possibly coincidental shorter series. Also, a coherent sequence is preferable to any of its smaller sub-sequences.

Prefer generalizing over fewer levels of abstraction

- PREFER-GEN-FEWER-LEVELS
- based on access to problem, domain, and language class hierarchies within BB1
- The actions in the sequence are more closely related, and therefore more likely a common phase of the strategy, if less abstraction is required to find a common generalization.

Prefer generalizing fewer action parameters

- PREFER-GEN-FEWER-VARS
- based on BB1 languages with standard action templates

- The actions in the sequence are more closely related, and therefore more likely a common phase of the strategy, if they have more parameters in common.

Prefer existing abstraction

- PREFER-EXISTING-UPPER-LEVEL-STATEMENTS
- based on the incremental style of WATCH's generalization
- Prefer abstracting a new action with an generalization in the tree, rather than creating a new sub-sequence. It is better to extend existing hypothesized strategies with new actions than to create many independent strategies.

Bibliography

Billman, Dorrit and Evan Heit, "Observational Learning From Internal Feedback: A Simulation of an Adaptive Learning Method". *Cognitive Science* 12: 587-625, 1988.

Carbonell, Jaime G., Ryszard S. Michalski, and Tom M. Mitchell, "Machine Learning: A Historical and Methodological Analysis". *AI Magazine* 4: 69-79, 1983.

Cohen, Paul R. and Edward A. Feigenbaum, *The Handbook of Artificial Intelligence, Volume III*. Los Altos, Ca., Morgan Kaufmann Publishers, 1982. pp. 325-334.

Dietterich, Thomas G. and Ryszard S. Michalski, "Learning to Predict Sequences". In *Machine Learning: An Artificial Intelligence Approach,* Volume II, pp. 63-106.

Gruber, Thomas, "A Method for Acquiring Strategic Knowledge". 1988.

Harvey, Jeffrey M., "WATCH - Inductive Learning of Control Abstractions". 1987.

Hayes-Roth, Barbara, Bruce Buchanan, Olivier Lichtarge, Michael Hewett, Russ Altman, James Brinkley, Craig Cornelius, Bruce Duncan, and Oleg Jardetzky, "PROTEAN: Deriving Protein Structure from Constraints". Stanford University Report KSL 86-51, 1986.

Hayes-Roth, Barbara, Alan Garvey, M. Vaughan Johnson Jr., and Michael Hewett, "A Layered Environment for Reasoning about Action". Stanford University Report KSL 86-38, 1986.

Hayes-Roth, Barbara and Michael Hewett, "Learning Control Heuristics in BB1". Stanford University Report HPP 85-2, 1985.

Johnson, M. Vaughan and Barbara Hayes-Roth, "Integrating Diverse Reasoning Methods in the BB1 Blackboard Control Architecture". Stanford University Report KSL 86-76, 1986.

Kautz, Henry A., "A Formal Theory of Plan Recognition". Thesis. University of Rochester, 1987.

Michalski, Ryszard S., Jaime G. Carbonell, and Tom M. Mitchell, *Machine Learning: An Artificial Intelligence Approach.* Palo Alto, Ca., Tioga Publishing Company, 1983.

Michalski, Ryszard S., Jaime G. Carbonell, and Tom M. Mitchell, *Machine Learning: An Artificial Intelligence Approach, Volume II.* Los Altos, Ca., Morgan Kaufmann Publishers, 1986.

Mitchell, Tom M., Richard M. Keller, and Smadar T. Kedar-Cabelli, "Explanation-Based Generalization: A Unifying View". *Machine Learning* 1: 47-80, 1986.

Mitchell, Tom M., Paul E. Utgoff, and Ranan Banerji, "Learning by Experimentation: Acquiring and Refining Problem-Solving Heuristics". In *Machine Learning: An Artificial Intelligence Approach,* Volume I, pp. 163-190.

Schmidt, Charles F., "Plan Recognition and Revision: Understanding the Observed Actions of Another Actor". Rutgers University Report CBM-TR-115, 1980.

Schmidt, C.F., N.S. Sridharan, and J.L. Goodson, "The Plan Recognition Problem: An Intersection of Psychology and Artificial Intelligence". Rutgers University Report CBM-TR-86, 1978.

Simon, Herbert A., "Why Should Machines Learn?". In *Machine Learning: An Artificial Intelligence Approach,* Volume I, pp. 25-37.

Tommelein, Iris D., M. Vaughan Johnson Jr., Barbara Hayes-Roth, and Raymond E. Levitt, "SIGHTPLAN: A Blackboard Expert System for Construction Site Layout". *Expert Systems in Computer-Aided Design*, J.S. Gero, ed., Amsterdam, North-Holland Publishers, 1987. pp. 153-167.