

Constructing Simplified Plans via Truth Criteria Approximation

Steve Chien

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive, M/S 525-3660
Pasadena, CA 91109-8099
chien@aig.jpl.nasa.gov

Gerald F. DeJong

Beckman Institute
University of Illinois
405 North Mathews Avenue
Urbana, IL 61801
dcjong@cs.uiuc.edu

Abstract

This paper has presented an approach to dealing with the complexity of explanation-based learning plans in complex domains. This approach uses a simplified algorithm to construct plans, and employs later refinements to repair bugs in constructed plans. This algorithm has the theoretical properties of completeness and convergence upon soundness. This incremental reasoning planning and learning algorithm has been implemented using a partial-order constraint posting planner and empirically compared to a conventional exhaustive reasoning partial-order constraint-posting planner and learning algorithm. This comparison showed that: 1) incremental reasoning significantly reduced learning costs compared to exhaustive reasoning; 2) Explanation-based Learning (EBL) reduced failures from incremental reasoning; and 3) EBL with incremental reasoning required less search to solve problems than EBL with exhaustive reasoning.

Introduction

Explanation-based learning and incremental reasoning offer a powerful combination in dealing with complexity in planning. Incremental reasoning allows the planner to reduce the computational expense of plan construction by using simplifications. Our approach uses a general class of simplifications determined by a simplified truth criterion. A refinement algorithm based upon this simplified truth criterion allows simplifications to be retracted in response to failures. The end intended result is a set of simplified plans which fail infrequently and are learned at a reduced cost due to simplifications. We call our approach incremental reasoning because refinements converge upon the exhaustive reasoning approach (e.g., retracting all simplifications). Explanation-based learning (EBL) offers failure reduction in the long term by allowing learned plans to be used in place of failure-prone from scratch problem-solving. Additionally, as when used with exhaustive reasoning, EBL can also lead to reduced search to solve problems.

⁰This paper describes research conducted in Part by the Jet Propulsion Laboratory, under contract with the National Aeronautics and Space Administration and in part conducted at the Beckman Institute of the University of Illinois supported by an IBM Graduate Fellowship and The National Science Foundation under grant NSF-IRI-87-19766.

The combination of incremental reasoning and EBL holds both intuitive and computational appeal. From the intuitive standpoint, people seem to often make assumptions and simplifications to make reasoning more tractable. With time, increased expertise leads to better knowledge about which simplifications will still allow adequate reasoning. All of these characteristics are consistent with explanation-based learning and incremental reasoning. From a computational standpoint, incremental reasoning can be shown to reduce computation under certain conditions. Explanation-based learning can also be shown to reduce failures in certain cases.

This paper focusses upon a particular type of incremental reasoning, incremental reasoning about operator effects (Chien 1990) which is an extension of techniques described in (Chien89). In this approach the planner constructs initial plans ignoring a type of negative subgoal interaction. When failures occur, the planner expands its consideration of subgoal interactions to construct a viable plan. This process continues until a plan is found. Because the expansion of consideration of subgoal interactions converges upon exhaustive reasoning our incremental reasoning algorithm converges upon soundness. Because the simplifications are over-general, our incremental reasoning algorithm retains completeness.

Our incremental reasoning approach is applicable in cases where:

1. A sound and complete (although intractable) domain theory exists.
2. A strong diagnostic capability exists to construct explanations for failures.
3. The cost of failures is low.
4. The system is allowed multiple attempts to solve a problem.

While many systems have been constructed which use incremental reasoning in planning (Simmons 1988) and incremental reasoning in combination with explanation-based learning (Hammond 1989; Collins et al. 1989), there has been little effort devoted to empirical analyses of reduced computation from the use of simplifications and reduced failures due to learning. Many empirical studies of exhaustive reasoning systems combined with EBL have examined reduced search to solve problems due to learning (e.g., (Minton 1988)).

In order to empirically evaluate the computational properties of incremental reasoning in combination

with explanation-based learning we used two planning and learning systems. The first, the control system, was a partial-order constraint-posting exhaustive planner which used conventional explanation-based learning. The second, the experiment system, was also a partial-order constraint-posting planner which used explanation-based learning, but constructed plans using simplifications. In order to remove effects of search heuristics, both systems searched plans breadth-first in the number of operators in the plan, thus plans searched statistics report upon the size of the search spaces searched by the respective systems.

To test these systems, we constructed two domain theories. Ideally, the systems would be tested upon domain theories used in other empirical tests (e.g. constructed by other researchers), however, almost all domain theories are restricted to STRIPS operators because most planners and learners cannot deal with conditional effects. Our incremental reasoning approach is designed precisely to deal with the additional complexity of operators with conditional effects. Consequently, we constructed two new domain theories by extending domain theories used by Minton in his empirical studies (Minton 1988).

In our empirical tests, we did indeed observe the expected three characteristics predicted by advocates of incremental reasoning and explanation-based learning. The observed characteristics were: reduced overall learning cost due to incremental reasoning, EBL produced a reduced failure rate from incremental reasoning, and EBL+incremental reasoning offered reduced overall search in problem solving compared to exhaustive reasoning + EBL.

The principal contributions of this paper are:

1. It presents a novel simplification method applicable for explanation-based learning of plans. This method is unique in that it uses a simplification of a truth criterion to guide reasoning.
2. It supports this method with empirical results of a) reduced learning cost due to simplification; b) reduced failures due to EBL; and c) reduced search due to simplifications + EBL.
3. It presents some of the first work to apply more powerful partial-order and constraint-posting planning techniques in conjunction with EBL.

The remainder of this paper is organized as follows. Section 2 describes the incremental reasoning techniques evaluated in this paper. Section 3 sketches the proofs for why our incremental reasoning approach is complete and converges upon soundness. Section 4 describes the domains used in the empirical evaluation and uses an example to illustrate our refinement approach. Section 5 presents the empirical results derived from our study, Section 6 describes related work and section 7 summarizes the principal contributions of this paper.

An Overview of Incremental Reasoning

Our refinement approach presumes a representation based upon situation calculus in which situations are complete and consistent propositional world descriptions. In our representation, an action A can be executed in a situation S1 only if the preconditions for A

are true in S1. The situation resulting from this action execution is computed by first asserting all of the direct effects of A, asserting all of the conditional effects of A whose conditional preconditions are met in S1, and finally asserting all of those facts in S1 which are not contradicted by any of the director conditional effects of A. Actions are not allowed to have contradictory effects. Thus, a fact can be true in a situation via a direct effect, conditional effect, or persistence. Shown below is part of a sample state description from an extended STRIPS domain. In this example, the robot moving from room1 to room2 causes box1 to move to room2 but box2 does not move. This is because a conditional effect of the move-robot operator is that boxes being carried move with the robot and box1 is being carried but box2 is not. Note also that the assertion of the fact $\neg(\text{location box1 room1})$ is also conditional on the carrying state of box1. Thus, the deletion of the fact $(\text{location box1 room1})$ is also conditional on the carrying state of box1.

InitialState	NewState
$(\text{location box1 room1})$	$\neg(\text{location box1 room1})$
$\neg(\text{location box1 room2})$	$(\text{location box1 room2})$
$(\text{location box2 room1})$	$(\text{location box2 room1})$
$\neg(\text{location box2 room2})$	$\text{allocation box2 room2}$
$(\text{location robot room1})$	$\neg(\text{location robot room1})$
$\neg(\text{location robot room2})$	$(\text{location robot room2})$
$(\text{robot_carrying box1})$	(robot_carrying)
$\neg(\text{robot_carrying box2})$	$\neg(\text{robot_carrying box2})$

An important point is that this representation allows an Operator to have effects conditional upon the state in which they are executed. While this conditionality is finite, and can be represented in conventional STRIPS operators, it would require a number of STRIPS operator definitions exponential in the number of conditional effects (i.e., a non-conditional operator for each possible combination of occurring conditional effects). This representation does not allow direct representation of inferred effects. However, if there are a finite number of inferred effects, these can be represented as conditional effects by requiring a conditional effect for each unique derivation for an inferred effect.

Plan construction and generalization occurs according to a truth criterion, which states the cases in which facts are true in situations in a plan. Our truth criterion is similar to that stated in (Chapman87), except for two changes. First, it handles conditional effects of actions. Second, it does not allow white knights (a form of disjunctive support). Basically, this truth criterion states that a fact will be true if it is established and then protected. Establishing a fact means ensuring it is true at some point before it is needed, and protecting a fact means ensuring that it will remain true until needed. A fact F is established at a situation S_{est} iff:

1. S_{est} is SO (the given initial state) and $F \in I$ (the initial state description).
2. S_{est} is the output state of some action A which has the direct effect F.
3. S_{est} is the output state of some action A which has the conditional effect F, whose conditional preconditions are true in the input situation for A.

A fact F is protected from Set to S_{des} if there is no action A_{clob} such that:

1. A_{clob} is ordered after S_{est} and before S_{des} and

2. A_{glob} has either:

- (a) a direct effect D which contradicts F or
- (b) a conditional effect C which contradicts F and whose conditional preconditions are true in the input situation to A_{glob} .

The truth criterion is interpreted procedurally to construct plans. Thus, the planner might use a conditional effect of an action already in the plan to achieve a goal, or add an action to the plan which had a conditional effect which achieves the goal. Correspondingly, the planner would also have to protect the newly achieved goal from each action effect already in the plan and protect already achieved goals from action effects being introduced into the plan by addition of operators to the plan. For each action effect and protection, either: 1) the action must occur outside the protection interval, 2) the effect must not contradict the protected fact; or 3) the effect is a conditional effect and a conditional precondition of the conditional effect must not be true in the input state to the action.

Within this framework, planning can be viewed as search in the plan space (the space of all possible plans). Search operators are modifications to the plan (e.g., add an operator, constrain an operator) and may introduce subgoals. Search begins with a null plan, and terminates when either: 1) a resource bound is exceeded, 2) no alternatives exist, or 3) a plan which achieves all of the problem goals is found.

The protection aspect of planning is computationally expensive because it requires reasoning about the effects of all potentially interfering operators. Because operators may have conditional effects, reasoning about the effects of an operator requires reasoning about the situation in which that operator is executed. Consider attempting to achieve a fact P at time T_1 using an action A . Establishment requires reasoning to ensure that A is before T_1 and requiring that $S_{in}(A)$ is sufficient to allow A to assert P . Reasoning about protecting P from the execution of A until T_1 requires: 1) determining all actions that might occur after A and before T_1 ; 2) determining whether any possible effects of these actions possibly contradicts P ; and 3) determining all of the potential situations in which actions satisfying 1 & 2 might be executed and determining which of these situations is sufficient to produce the contradicting effect(s). Unfortunately, task 3 involves repeatedly determining the truth value of a fact in a partially ordered plan with conditional effects which is a known NP-hard problem (Chapman87).

An important point is that this problem of expensive reasoning about protections is representation independent. The complexity arises from the expressive power of conditional effects, not the exact representation of operators and persistence that we employ. In (Chapman87), Chapman shows that for any representation sufficient to represent conditional effects, determining the truth value of a fact in a partial-order plan is an NP-hard problem. For example, using frame axioms to prove persistence would require many explanations for multiple orderings. To guarantee soundness any frame axiom persistence proof would need to check the same potentially relevant facts as our approach of checking against operator effects.

Not only is it computationally expensive to perform exhaustive protection from conditional effects, conditional interference is more likely to be preventable (i.e., via refinement) than would a direct effect. Interference. This is because for a direct effect, the interference can be removed only by ordering or by ensuring code-signations do not allow interference. For a conditional effect, there are these two options plus the additional option of planning to invalidate a conditional precondition of the conditional effect.

The motivation for this inference limitation is not that this is the exact bound which should necessarily be placed upon inference, but rather that inference typically is limited by computational resources and an intelligent planning system must have the capability to reason about these limitations and extend its inference as directed by world feedback. The focus of this research is upon developing methods for making and retracting simplifications, not deciding when to make simplifications.

Thus, with regard to the truth criterion described above, the difference between the exhaustive and incremental approaches can be summarized as follows. The exhaustive approach ensures that all conditional effects of operators do not invalidate protections in the plan. The incremental approach checks only those conditional effects which are used for establishment purposes in the plan. This simplification corresponds to not considering general class of negative goal interactions.

However, incomplete reasoning about protections allows the possibility that plans will fail unexpectedly. Because the incomplete reasoning about protections is the only source of unsoundness used in constructing the initial plan, any future plan failures must be due to unconsidered conditional effects of operators violating protections in the plan. Consequently, when the system encounters a failure it uses a full execution trace of the failure to find a previously overlooked relevant negative effect-protection interaction. At this point, the incremental system performs the same interaction analysis for this effect protection pair that the exhaustive system does during initial plan construction as described above.

Incremental Reasoning is Complete and Converges Upon Soundness

The incremental reasoning approach we have described has the theoretical properties of convergence upon soundness and completeness. Because of space constraints, we do not present full proofs of these properties, but outline the proofs of these properties. For further details see (Chien 1990).

Incremental reasoning about protections converges upon soundness. This means that after a finite number of failures (and corresponding refinements) an incrementally constructed explanation converges upon soundness. The proof goes as follows. A bug in a plan is a case where a conditional effect interferes with a protection. Because there are a finite number of operators in the plan, and there are a finite number of conditional effects per operator, the number of possible bugs is finite. Because refinement enforces the same protection constraints used in exhaustive reasoning, each refinement

removes a bug. Thus, after a finite number of failures and refinements all possible bugs will be removed and the refined plan will be sound.

Incremental reasoning about protections is complete. This is because the incremental reasoning set of conditions is a relaxation of the constraints from the exhaustive reasoning truth criterion. As refinements are made, they force the set of protection checks to converge upon those required by the exhaustive approach, but still are a subset (or possibly equal) to those required by the exhaustive approach. Thus, the incremental approach will still consider a set of plans a superset (although possibly equal) of the exhaustive set and thus the incremental approach is complete.

Note that while the incremental approach allows a superset of the plans considered by exhaustive approach, this does not mean that the incremental approach is performing more search. This is because the search is conducted in the space of constraints upon the plans, thus the incremental approach is searching a smaller space by enforcing and checking less constraints (because enforcing less constraints means considering more plans).

Domains and Example

We now describe the domain theories used to test the incremental reasoning approach. Ideally our incremental reasoning approach would be tested upon domain theories constructed by other researchers. However, there are very few domain theories used by planning or machine learning researchers with conditional effects. Notable exceptions are (Schoppers 1989; Pednault 1991). However their representations are very different from ours, which prevented use of their domain theories.

Consequently, we constructed two domain theories by modifying domain theories used by Minton (Minton 1988) in his learning research. The empirical evaluation compared the performance of the incremental and exhaustive approaches in two domains - a workshop domain and a modified STRIPS domain.

The STRIPS domain consisted of five operators. A robot could pickup, putdown, align (for pushing), and unalign blocks that were carryable or pushable (a robot could carry one block plus push another at any one time). Changing rooms caused aligned or carried blocks to change rooms with the robot. In order to align or pickup a block, the robot had to be next to the block, which was defined as being in the same room. This domain theory could be formulated with the change-room operator having four conditional effects, and no other operators having conditional effects. Problems in the STRIPS domain were specifications of goal locations for blocks and robots. The most complex problem set, problem set C, involved a three room world with two block location goals and a robot location goal (the longest solution required would thus be 12 operators). The STRIPS domain is highly recursive and requires deep subgoaling. Additionally, there are many protection choices because there are many potentially relevant conditional effects each with several conditional preconditions.

The workshop domain consists of eight operators with a total of thirteen conditional effects. In this domain the

goal of the system is to join pieces together in certain orientations and achieve certain attribute goals (such as length and smoothness). Assuming no interactions, join goals took roughly 3 operators to achieve and most attribute goals could be achieved in a single operator. Some of the workshop domain problems are unsolvable. The most difficult workshop domain problems took 10 operators to achieve. In the most difficult problem set, WKC, problems contained three objects, three attribute goals, and three join goals. In the workshop domain, the goal structure is very shallow (e.g., did not require deep subgoaling) and the planner attempts to avoid interactions. In this domain the incremental planner makes a large number of assumptions about attribute values for objects persisting through conditional effects. An English description of the cut operator from the workshop domain is shown below:

Operator:	cut <i>object</i> to <i>length</i>
Preconditions:	<i>saw</i> is not hot
Effects:	<i>object</i> length is <i>length</i>
Conditional Effects:	if <i>object</i> is large then saw now hot if <i>object</i> is metal then saw now hot if <i>object</i> is large then surface of the object now rough

We now describe an example of incremental refinement in a workshop domain problem. In this problem, the system is given the goal of producing an object of a certain length which also has a smooth surface. The planner decides to achieve the length goal by using a cut operator and to achieve the surface smooth goal by using a sand operator. The system then attempts this plan, executing the sand operator and then the cut operator on a large object.

This results in an end state where the length goal is achieved but not the smooth goal. The smooth goal has been invalidated by a conditional effect of the cut operator, which states that when a large object is cut, it is no longer smooth. This operator was introduced to achieve the length goal. This triggers a refinement of the plan, which involves enforcing the protection relationship between the cut operator and the protection of the smooth state from the execution of the sand operator to the end of the plan. Enforcing the protection produces three options: 1) move the cut operator before the sand operator or after the goal state; 2) try to sand a different object from the object that is cut; or 3) enforce that the object used not be large. Option 1 means cut before sand, and will result in a correct plan. Option 2 cannot be applied because the goal requires that a single object fulfill the length and smooth goals. Option 3 would also result in a correct plan. Note that this enforcement of the protection constraints and evaluation of protection options is identical to that which would be performed by the exhaustive approach. The difference is that the incremental approach enforces these constraints only in response to failures.

Empirical Evaluation

This section describes the empirical evaluation of incremental reasoning and explanation-based learning. First we describe the characteristics that we wished to measure. Second, we describe the experimental methodol-

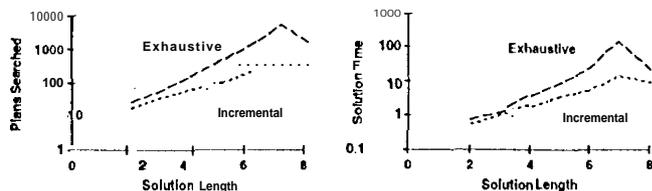


Figure 1: impact of Simplified Truth Criteria on Learning Cost as a Function of Problem Complexity

ogy used to measure these characteristics. Third, we describe the results of our empirical tests.

The three phenomena we wished to verify empirically are the following:

1. Incremental reasoning reduces the computational expense per example of applying explanation-based learning.
2. Explanation-based learning reduces the long-term failure rate for incremental reasoning.
3. Explanation-based learning reduces the long-term search requirements for incremental reasoning:

In order to measure these three phenomena we used an exhaustive control system and an incremental experiment system. Both systems were partial-order constraint-posting planners. Additionally, to ensure the results were not skewed by use of search heuristics, both systems used search breadth-first in the number of operators in the plan.

We now present results from the WKB problem set from the workshop domain. In order to measure the impact of incremental reasoning on the cost of applying EBL, we measured the number of plans searched to construct an explanation (plan) and the CPU time for the entire learning process (including search to construct an explanation (plan)). Because the search to find a solution is the majority of the learning cost, these statistics are very similar. In both these cases, the learning cost was measured over 700 problems and is shown in Figure 1 as a function of problem complexity as measured by the solution length. The cost of learning is exponential as a function of problem complexity, as is indicated by the roughly linear shape of the cost on the logarithmic scale.

As indicated by these graphs, using the simplifications in learning significantly reduced the cost of learning. It is worth noting that there was only one problem of solution length 8, there were between 8 and 218 problems of the other solution lengths. because Figure 1 shows that the incremental reasoning approach offers greater than constant speedup on a logarithmic scale in problem solution (explanation construction) for explanation-based learning, this indicates that an exponential speedup has occurred. For unsolvable problems (e.g., those problems for which the planner was able to show that no solution exists within the depth bound), system performance is measured as the amount of search conducted before concluding no solution exists. In these test the incremental approach outperformed the exhaustive approach on the average in search 138.1 plans

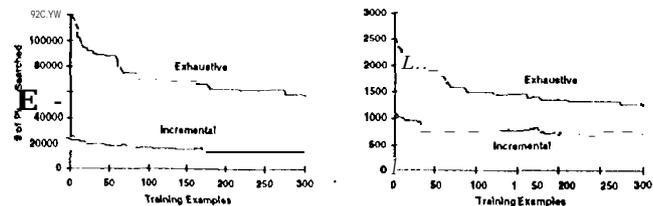


Figure 2: Impact of Learning on Problem-solving Performance

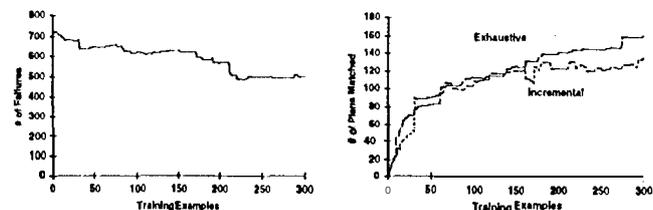


Figure 3: Impact of Learning on Failure Rate and Matching Rate

searched to 360.9 plans searched and in solution time 4.1 CPU seconds to 8.24 CPU seconds.

In order to measure the long-term learning effects on failure and search rates, a training and validation set each of 300 problems was generated. After each system was allowed to train upon a training set problem, the system was tested upon the entire validation set. Validation set performance was measured in three ways:

1. number of plans searched and computation time to solve all of the problems in the validation set;
2. number of failures encountered while solving all of the problems in the validation set; and
3. number of problems solved by matching learned plans from the plan library.

Figures 2 shows the number of plans searched and CPU time used by the incremental and exhaustive approaches to solve all of the problems in the validation set. In both cases the incremental approach performs better before learning and maintains its advantage throughout learning. For comparison, a non-learning incremental or exhaustive system performance would be a horizontal line from the Y intercept. Thus, the use of simplifications and learning outperforms both exhaustive without, exhaustive with learning, and simplifications without learning.

The graph at the left of Figure 3 shows the total number of failures encountered by the incremental approach in solving all of the problems in the validation set. This graph shows that the general trend is decreasing, indicating that learning is indeed reducing the number of failures made by the incremental approach. For comparison, performance of a non-learning incremental system would be indicated by a horizontal line from the Y intercept.

The graph at the right of Figure 3 shows the number of problems in the validation set solved by matching a

plan in the plan library. Note that the exhaustive approach match rate is nondecreasing because with the exhaustive approach the plan library never decreases in size. In contrast, with the incremental approach, failures in already learned plans causes them to be removed from the library so that the incremental match rate sometimes decreases.

Related Work and Conclusions

While there have been numerous failure-driven planning systems (such as (Simmons 1988; Hammond 1989; Collins et al. 1989; Bennett 1990; Sussman 1973)) our approach is unique in that it uses an approximation of an exhaustive truth criterion to construct plans. Additionally, while there have been numerous empirical evaluations of standard EBL systems (such as (Minton 1988)) there have been relatively few evaluations of incremental reasoning or failure-driven refinement plan learning systems. However, there have been numerous empirical evaluations for other abstraction-based learning systems applied to problem-solving such as (Tadepalli 1989), and (Unruh & Rosenbloom 1989). Other related work includes (Kambhampati & Kedar 1991) which describes the use of truth criteria to guide explanation-based generalization of plans but does not address issues of incremental reasoning or refinement.

The incremental reasoning approach described in this paper is substantially revised from the approach described in (Chien 89) and is described in more detail in (Chien 1990). The primary change in the approach is to make the simplifications more closely tied to truth criterion conditions. This general approach highlights how using the truth criterion as a guide to simplifications can be useful in attacking the computational expense of planning.

An important area for future work would be to evaluate the incremental reasoning techniques described in this paper using generally available partial order planners such as UCPOP (Pemberthy & Weld 1992). This work (Chien 1990) was originally done before the existence of such planners but testing incremental reasoning on such planners (and the numerous domain associated domain theories) would be informative.

It is worth noting that in the two domains we used to test incremental reasoning *The Utility problem* (Minton 1988) did not arise. We expect that in general there will be interactions between incremental reasoning and the utility problem, as learning possibly faulty plans allows learning of more plans of potentially lower utility.

It is also worth noting that in both the workshop and modified STRIPS domain it was frequently possible to deny conditional effect interactions by planning to prevent their conditional preconditions. In general, as performing this type of fix becomes more difficult, the effectiveness of incremental reasoning about conditional effects will decrease. However, the general approach of simplifications based upon relaxing truth criteria is still valid, and might indicate other types of simplifications to apply.

This paper has presented an approach to dealing with the complexity of explanation-based learning plans in complex domains. This approach uses a simplified algorithm to construct plans, and employs later refinements

to repair bugs in constructed plans. This algorithm has the theoretical properties of completeness and convergence upon soundness. This incremental reasoning planning and learning algorithm has been implemented using a partial-order constraint posting planner and empirically compared to a conventional exhaustive reasoning partial-order constraint-posting planner and learning algorithm. This comparison showed that: 1) incremental reasoning significantly reduced learning costs compared to exhaustive reasoning; 2) EBL reduced failures from incremental reasoning; and 3) EBL with incremental reasoning required less search to solve problems than EBL with exhaustive reasoning.

References

- s. Bennett, 1990, "Reducing Real-world Failures of Approximate Explanation-based Rules," *Proc. Conf. on Machine Learning*, Austin, TX.
- D. Chapman, "Planning for Conjunctive Goals, 1987," *Artificial Intelligence* 32, 3.
- S. A. Chien, 1989, "Using and Refining Simplifications: Explanation-based Learning of Plans in Intractable Domains," *Proc. IJCAI89*, Detroit, MI.
- S. A. Chien, 1990, "An Explanation-based Learning Approach to Incremental Planning," Ph.D. Thesis, Dept. of Comp. Sci., Univ. of Illinois, Urbana, IL.
- G. Collins, L. Birnbaum, and B. Krulwich, 1989, "An Adaptive Model of Decision-making in Planning," *Proc. IJCAI89*, Detroit, MI.
- K. Hammond, *Case-Based Planning: Viewing Planning as a Memory Task*, Academic Press, 1989.
- S. Kambhampati and S. Kedar, 1991, "Explanation-based Generalization of Partially Ordered Plans," *Proceedings AA AI91*, Anaheim, CA.
- S. Minton, *Learning Effective Search Control Knowledge: An Explanation-based Approach*, Kluwer, Norwell, MA, 1988.
- E. Pednault, 1991, "Generalizing Nonlinear Planning to Handle Complex Goals and Actions with Context-Dependent Effects," *Proc. IJCAI91*, Sydney, Australia.
- J. Pemberthy and D. Weld, 1992, "UCPOP: A Sound, Complete, Partial-order Planner for ADL," *Proc. 3rd Int. Conf. on Knowledge Representation and Reasoning*.
- M. Schoppers, 1989, "Representation and Automatic Synthesis of Reaction Plans," Ph.D. Thesis, Dept. of Comp. Sci., Univ. of Illinois, Urbana, IL.
- R. Simmons, 1988, "Combining Associational and Causal Reasoning to Solve Interpretation and Planning Problems," Technical Report 1048, AI Lab, Massachusetts Institute of Technology, Cambridge, MA.
- G. Sussman, 1973, *A Computational Model of Skill Acquisition*, American Elsevier, New York.
- P. Tadepalli, 1989, "Lazy Explanation-based Learning: A Solution to the Intractable Theory Problem," *Proc. IJCAI89*, Detroit, MI.
- A. Unruh and P. Rosenbloom, 1989, "Abstraction in Problem-solving and Learning," *Proc. IJCAI89*, Detroit, MI.