Learning General Completable Reactive Plans

Melinda T. Gervasio*

Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign
405 N. Mathews Ave., Urbana, Illinois 61801
gervasio@cs.uiuc.edu

Abstract

This paper presents an explanation-based learning strategy for learning general plans for use in an integrated approach to planning. The integrated approach augments a classical planner with the ability to defer achievable goals, thus preserving the construction of provably-correct plans while gaining the ability to utilize runtime information in planning. Proving achievability is shown to be possible without having to determine the actions to achieve the associated goals. A learning strategy called contingent explanation-based learning uses conjectured variables to represent the eventual values of plan parameters with unknown values a priori, and completors to determine these values during execution. An implemented system demonstrates the use of contingent EBL in learning a general completable reactive plan for spaceship acceleration.

Introduction

The planning problem may be characterized as the problem of determining an ordered sequence of actions which when executed from a given initial state will achieve a given goal. In classical planning [Chapman87, Fikes71, Stefik81], plans are determined completely prior to execution by using inference to predict the effects of actions and essentially construct proofs of goal achievement. Provided a classical planner has perfect a priori knowledge, its plans are guaranteed to achieve the given goals. Unfortunately, real world domains can rarely be characterized perfectly. Reactive planning [Agre87, Firby87, Schoppers87, Suchman87] is an alternative approach which makes no predictions about the future and instead repeats a cycle of evaluating the environment and determining an appropriate action. Reactive planning thus solves the extended prediction problem faced by classical planning [Shoham86] simply by eliminating it. Because reactive planning is essentially a hill-climbing approach, however, one wrong reaction may delay or prevent a reactive planner from achieving its goals. Furthermore, reactive planners must also be hand-tailored to achieve the desired behavior. Thus, while machine learning strategies have been successfully applied to classical planners in various domains [Chien89, Fikes72, Hammond86, Minton85], only preliminary work has been done in learning reactive rules [Blythe89, Schoppers87].

* This research was supported by the Office of Naval Research under grant N-00014-86-0309.

The planning approach presented in this paper integrates classical planning and reactive planning to solve some of their limitations while retaining their merits. In this integration, a classical planner is augmented with the ability to defer goals guaranteed to be achievable, thus enabling the use of runtime information without sacrificing the provably-correct nature of plans. The integrated approach also retains the learning abilities of classical planning through contingent explanation-based learning, a strategy which enables learning general plans for use in the integrated approach.

An Integrated Approach To Planning

Given an initial state description I and a goal state description G, the planning problem involves the determination of a plan P consisting of a sequence of actions which when executed from the initial state will achieve a goal state.[1] Here we consider the case wherein a planner has access only to partial state descriptions and thus needs to reason about sets of states rather than individual states.

Determining a plan in the integrated approach takes place in two stages. Prior to execution, a *completable* partial plan is constructed. That is, the plan may be incomplete, but it is guaranteed that the missing components can be determined during execution. The plan is then completed during execution, possibly with the use of information which becomes available then. Since the plan is completable, any deferred goals are guaranteed to be satisfied during execution.

Let states(S) be the set of states satisfying the partial state description S, and PR(p) and EF(p) the precondition and effect state descriptions of an action or action sequence p. Given initial and goal state descriptions I and G, the first stage consists of determining a sequence of subplans $Q = \{q_1, q_2, \ldots, q_n\}$ such that:

$$\text{states}(I) \subseteq \text{states}(q_1) \text{ AND } \text{states}(q_n) \subseteq \text{states}(G) \text{ AND } \text{achievable}(EF(q_i), PR(q_{i+1}),) \text{ for all } q_i \in \{q_1, \ldots, q_{n-1}\}$$

where *achievability* is defined as follows:

$$\text{achievable}(S_1, S_2) \text{ iff } \forall\ s \in \text{states}(S_1)\ \exists \text{ a plan p which when executed from s will result in a state in states}(S_2).$$

Note that proving achievability does not require determining a precise plan but rather proving only the existence of a plan.

1. To simplify the presentation, the planning problem is characterized as determining completely-ordered action sequences.

Q may thus be provably correct even while being only partially–determined.

The second stage consists of determining a set of subplans $R = \{r_1, r_2, ..., r_{n-1}\}$ to complete Q, that is:

- $\forall q_i \in \{q_1,...,q_{n-1}\}$ executing r_i in the effect state of q_i results in a precondition state for the subplan q_{i+1}.

Because achievability proofs for R were constructed in the first stage, it is guaranteed that subplans will be found for achieving the deferred goals. The final plan P is thus $\{q_1,r_1,q_2,r_2,...,q_{n-1},r_{n-1},q_n\}$.[2]

Proving Achievability

The integrated approach is naturally limited by the achievability constraint on deferred goals. However, it provides a solution to an interesting group of planning problems. There are many problems in relatively well–behaved domains where enough a priori information is available to enable the construction of almost complete plans. There are also certain kinds of information which are difficult to predict a priori but can trivially be gathered during execution. In these cases, a planner which constructs plans completely a priori faces an extremely difficult task, while a planner which leaves everything to be dynamically determined by reacting to the execution environment loses the goal–directed behavior provided by a priori planning. A planner in the integrated approach faces neither problem, with its ability to defer achievable goals and utilize runtime information in its planning.

An important criterion for the success of the integrated approach is proving achievability without having to determine actions to achieve the associated deferred goal. Isolating the issue of proving achievability provides two important advantages: 1) the option of deferring planning decisions simplifies a priori planning, and 2) the use of execution–time information lessens a planner's reliance on perfect knowledge prior to execution. Consider the problem of whipping cream. An a priori computation of the precise time interval over which to whip the cream in order to achieve soft peaks would have to account for a whole host of factors, such as the changing temperature of the cream, the speed of the beaters, and the humidity in the kitchen. In contrast, monitoring for the desired consistency at execution time, as would be done in the integrated approach, is a fairly simple task. Three classes of problems have been identified wherein achievability proofs can be constructed without determining the actions to achieve the associated goals.

Repeated Actions and Terminal Goal Values. The first class of problems involves repeated actions towards a terminal goal value, such as hammering a nail all the way into a piece of wood or completely unloading a clothes–dryer. The achievability proof for this class of problems lies in the notion of incremental progress, with every execution of an action resulting in a state nearer to the goal state until the goal is reached, in which case the action has no effect. For example, every action of pulling clothes out of the dryer will result in less clothes being in the dryer until the goal of the dryer being completely empty is reached. Instead of being precomputed, the precise number of repetitions needed can thus be determined at execution time by repeatedly performing the unloading action until the dryer is empty.

Continuously–Changing Quantities and Intermediate Goal Values The second class of problems involves continuously–changing quantities and intermediate goal values, such as whipping cream until soft peaks form or accelerating to some higher velocity. Proving achievability for this class of problems involves reasoning about the achievability of the limits on the value of a continuously–changing quantity, which guarantees the achievability of all the values within those limits. For example, whipping cream until stiff peaks form can be achieved by running the whipping process over some time interval, thus the intermediate soft–peaks stage is achievable by running the whipping process over some smaller time interval. During execution, this smaller interval can be dynamically determined by monitoring the cream for the desired consistency.

Multiple Opportunities The third class of problems involves multiple opportunities, such as choosing a paper cup for coffee or deciding which gas station to stop at on a long trip. The achievability proof for these problems depends upon the existence of several objects of some type needed to achieve the goal. For example, the state of having a clean, graspable cup at the time of getting coffee is achievable if there is a stack of cups in the cabinet from which one may be chosen. Thus, complicated reasoning about particular cups can be avoided by deferring the cup choice until the cups are in sight and one can be chosen trivially.

Achievability proofs are implemented as rule schemata, which are second–order predicate calculus rules which serve as templates from which to derive first–order predicate calculus rules for use in theorem–proving. Figure 1 shows an example of such a rule schema for the class of problems involving continuously–changing quantities and intermediate goal values, as well as a rule derived from that schema for increasing quantities. The reasoning embodied by this schema, also known as the Intermediate Value Theorem in calculus, is as follows. Let q be a continuous quantity having a value v_0 at some time t_0. Also let it be the case that certain conditions θ being true over some interval $(t_0\ t_2)$ will result in q having some other value v_2 at time t_2. Then for all values v_1 between v_0 and v_2 there exists some time t_1 within the interval $(t_0\ t_2)$ such that if θ holds over the interval $(t_0\ t_1)$, q will have the value v_1 at t_1. This reasoning as applied to an increasing quantity is depicted graphically in Figure 2 .

Learning Reactive Plans

In the integrated approach, a plan is partially constructed prior to execution and completed during execution. We call such a plan a *reactive plan*. The integrated approach requires that all deferred goals be achievable, hence the plans are *completable* reactive plans. The learning objective is to learn general completable reactive plans from observation.

Explanation–based learning (EBL) is a knowledge–intensive procedure by which general concepts may be learned

2. Cases with $\{r_0,q_1,...\}$ or $\{...,q_n,r_{n+1}\}$ can be treated similarly.

```
Intermediate Value Rule Schema
[∀ θ
   [∀ q v0 v2 t0 t2
      ((value q v0 t0) AND
       (continuous q) AND
       ((θ [t0 t2]) → (value q v2 t2)) )
   →
      [∀ v1
         (between v1 v0 v2)
       → [∃ t1 (within t1 (t0 t2)) AND
                ((θ (t0 t1)) → (value q v1 t1)) ] ] ] ]
```

```
Intermediate Value Rule for An Increasing Quantity
[∀ q v0 v2 t0 t2 v1
   (  (value q v0 t0) AND
      (continuous q) AND
      (  (qualitative_behavior q increasing (t0 t2))
      → (value q v2 t2))
      (between v1 v0 v2)  )
→
   [∃ t1 (within t1 [t0 t2]) AND
         (  (qualitative_behavior q increasing (t0 t1))
         → (value q v1 t1) ) ] ]
```

Figure 1. Sample Rule Schema and Derived Rule.

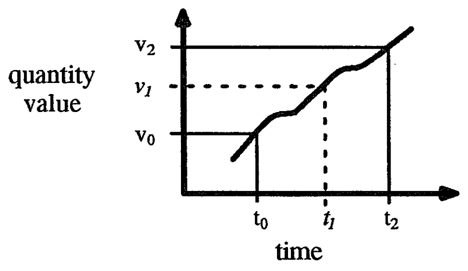


Figure 2. Reasoning about intermediate values for an increasing quantity.

from an example of the concept [DeJong86, Mitchell86]. EBL involves constructing an explanation for why a particular training example is an example of the goal concept, and then generalizing the explanation into a general functional definition of that concept or more general subconcepts. In planning, explanation and generalization may be carried out over situations and actions to yield macro–operators or general control rules. Here, we are interested in learning macro–operators or general plans.

Reactive plans present a problem for standard explanation–based learning [Mooney86]. Imagine the problem of learning how to cross. After the presentation of an example, an explanation for how the crosser got to the other side of the street may be that the crossing took place through some suitably–sized gap between two cars. Unfortunately, the generalization of this explanation would then include the precondition that there be such a suitably–sized gap between some two cars—a precondition which for some future street–crossing can only be satisfied by reasoning about the path of potentially every car in the world over the time interval of the expected crossing! The basic problem is that standard explanation–based learning does not distinguish between planning decisions made prior to execution and those made during execution. After execution, an explanation may thus be constructed using information which became available only during execution, yielding a generalization unlikely to be useful in future instances.

Contingent explanation–based learning makes two main modifications to standard explanation–based learning. The first is the introduction of *conjectured variables* into the knowledge representation to enable reasoning about deferred achievable goals. The second is the addition of a completion step which incorporates *completors* into the general plan for the execution–time completion of the partial plans derived from the general plan.

Conjectured Variables. Reactive plans involve deferred goals and hence plan parameters whose values are determined during execution. For example, in a reactive plan for hammering a nail all the way into a wooden plank, the precise number of pounding actions needed is unknown prior to execution. However, an achievability proof can be constructed, as discussed in the section on proving achievability, which guarantees that there exists some number of pounding actions which will achieve the goal. A planner must be able to recognize such plan parameters which have undetermined values prior to execution and whose values must be determined during execution. Contingent EBL uses conjectured variables for this purpose. A *conjectured variable* is a planner–posed existential used in place of a precise parameter value prior to execution, thus acting as a placeholder for the eventual value of a plan parameter. The integrity of an explanation containing conjectured variables hinges upon whether or not values can be found for the conjectured variables—hence the term *contingent* explanation–based learning.

In the integrated approach, a planner is restricted to introducing conjectured variables only if achievability proofs can be constructed for the associated deferred goals. In the class of problems involving continuously–changing processes and intermediate goal values, for example, the achievability proof warrants the introduction of a conjectured variable regarding the end of the time interval at which the goal value for a particular continuously–changing quantity will be reached. Such a conjectured variable, supported by an achievability proof, is called a *valid* conjectured variable. Only valid conjectured variables are allowed into the domain knowledge of a system in the integrated approach. Thus, a system may reason with conjectured variables only by reasoning about their achievability as well. For example, in attempting to find a binding for the cup in a get–coffee problem during explanation, a system may explain the achievability of determining a precise cup during execution by reasoning about multiple opportunities. It may thus unify the cup with a valid conjectured variable which would have no a priori value but whose achievability is guaranteed by information available to the system before execution. By disallowing a system from unrestrainedly using conjectured variables to finesse the problem of requiring an explanation grounded in initial data, the provably–correct nature of explanations and their generalizations is preserved.

Completors. The second modification made by contingent explanation–based learning involves the incorporation of additional operators called *completors* into the general plan. Completors are responsible for determining a completion to a reactive plan by finding appropriate values for conjectured variables during execution. Since only valid conjectured variables are allowed in the integrated approach, every conjectured variable in a general plan will have an accompanying achievability proof in the generalized explanation. This proof provides the conditions supporting the introduction of the conjectured variable, which are used in constructing an appropriate completor.

There are currently three types of completors, one for each of the three types of achievability proofs discussed earlier. *Iterators* perform a particular action repeatedly until some goal is achieved. *Monitors* observe a continuously–changing quantity to determine when a particular goal value for that quantity has been reached. *Filters* look for an object of a particular type. The contingent explanation–based learning algorithm is summarized in Figure 3.

```
Input training example and goal concept
Construct an explanation for why the example is an
  example of the goal concept
If an explanation is successfully constructed
 Then
  Generalize and construct a general plan using:
   the goal (root)
   the preconditions (leaves) determining applicability
   the sequence of operators achieving the goal
  Identify the conjectured variables in the generalized
    explanation.
  If there are conjectured variables
   Then
    For every conjectured variable
     Identify the conditions supporting the introduction
       of the variable in the generalized explanation.
     Construct an appropriate completor using these
       conditions and including the plan components with
       this variable.
     Add completor to the operators of the general plan.
    Output general completable reactive plan.
   Else
    Output general non–reactive plan.
 Else
  Signal FAILURE.
```

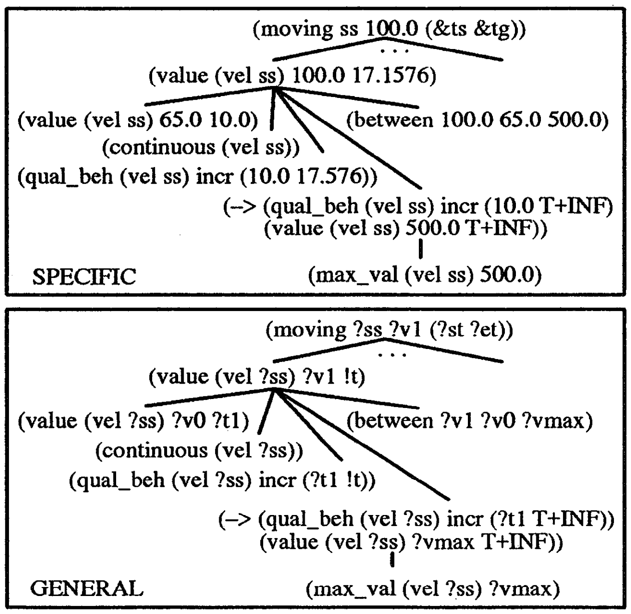
Figure 3. Contingent EBL Algorithm.

Example

A system written in Common LISP and running on an IBM RT Model 125 implements the integrated approach to planning and learning reactive operators. The system uses a simple interval–based representation and borrows simple qualitative reasoning concepts from Qualitative Process Theory [Forbus84]. The system is thus able to reason about quantity values at time points as well as quantity behaviors over time intervals. For example, (value (velocity spaceship) 65 10) represents the fact that the spaceship is traveling at 65 m/s at time 10), and (behavior (velocity spaceship) increasing (10 17)) represents the fact that the spaceship's velocity was increasing from time 10 to 17). The system also uses a modified EGGS algorithm [Mooney86] in constructing and generalizing contingent explanations.

The system is given the task of learning how to achieve a particular goal velocity higher than some initial velocity—i.e. acceleration. The example presented to the system involves the acceleration of a spaceship from an initial velocity of 65 m/s at time 10 to the goal velocity of 100 m/s at time 17.1576, with a fire–rockets action executed at time 10 and a stop–fire–rockets action executed at time 17.1576. In explaining the example, the system uses an intermediate value rules for an increasing quantity (see Figure 1) to prove the achievability of the goal velocity. It determines that the following conditions hold: 1) velocity increases continuously while the rockets are on, 2) if the rockets are on long enough, the maximum velocity of 500 m/s will be reached, and 3) the goal velocity of 100 m/s is between the initial velocity of 65 m/s and 500 m/s. There is thus some time interval over which the spaceship can be accelerated so as to achieve the goal. In this particular example, that time interval was (10 17.1576).

The part of the explanation the system constructs regarding the achievement of the goal velocity at time 17.1576 is shown in Figure 4, together with its corresponding part in the generalized explanation. The general explanation yields a

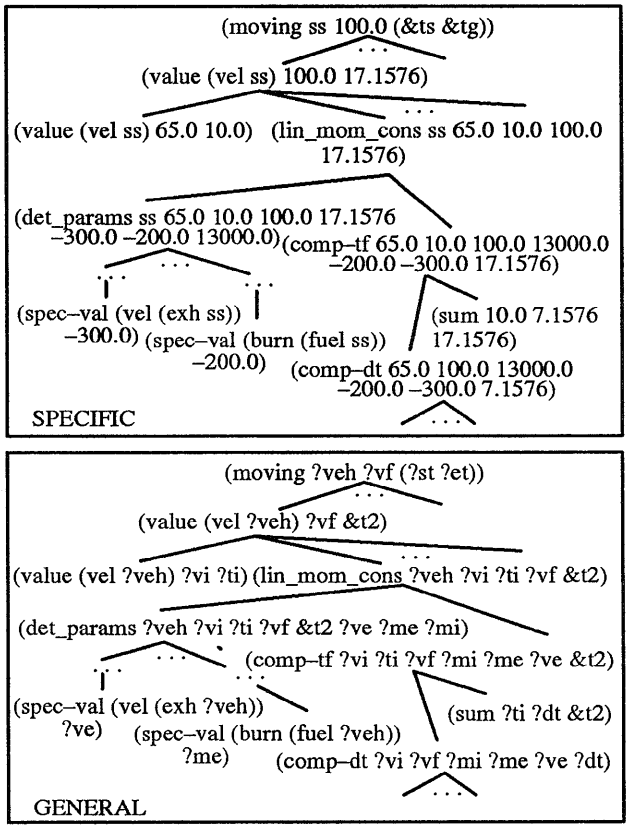


(moving ss 100.0 (&ts &tg))
(value (vel ss) 100.0 17.1576)
(value (vel ss) 65.0 10.0)
(between 100.0 65.0 500.0)
(continuous (vel ss))
(qual_beh (vel ss) incr (10.0 17.576))
(–> (qual_beh (vel ss) incr (10.0 T+INF)
(value (vel ss) 500.0 T+INF))
SPECIFIC
(max_val (vel ss) 500.0)
(moving ?ss ?v1 (?st ?et))
(value (vel ?ss) ?v1 !t)
(value (vel ?ss) ?v0 ?t1)
(between ?v1 ?v0 ?vmax)
(continuous (vel ?ss))
(qual_beh (vel ?ss) incr (?t1 !t))
(–> (qual_beh (vel ?ss) incr (?t1 T+INF))
(value (vel ?ss) ?vmax T+INF))
GENERAL
(max_val (vel ?ss) ?vmax)

Figure 4. Portions of reactive explanations.

two–operator—fire–rockets and stop–fire–rockets—general reactive plan. This plan contains the conjectured variable !t, which is the time the goal velocity is reached and the stop–fire–rockets action is performed. Using the conditions provided by the achievability proof, a monitor operator is created for observing the increasing velocity during the acceleration process and indicating when the goal velocity is reached to trigger the stop–fire–rockets operator.

The system is also run in a non–reactive (classical planning) mode on the same example. Here, the system uses equations derived from the principle of the conservation of linear momentum in order to explain the achievement of the goal velocity. This involves reasoning about various quantities, including the combustion rate of fuel and the velocity of the exhaust from the spaceship, in order to determine the acceleration rate. Corresponding portions regarding the achievement of the goal velocity at time 17.1576 in the specific and general explanations are shown in Figure 5. The general explanation yields a general non–reactive plan also involving a fire–rockets operator and a stop–fire rockets operator. However in this plan, the time at which the stop–fire–rockets action is performed is precomputed using some set of equations rather than determined during execution.



(moving ss 100.0 (&ts &tg))
(value (vel ss) 100.0 17.1576)
(value (vel ss) 65.0 10.0)
(lin_mom_cons ss 65.0 10.0 100.0 17.1576)
(det_params ss 65.0 10.0 100.0 17.1576 −300.0 −200.0 13000.0)
(comp–tf 65.0 10.0 100.0 13000.0 −200.0 −300.0 17.1576)
(spec–val (vel (exh ss)) −300.0)
(spec–val (burn (fuel ss)) −200.0)
(sum 10.0 7.1576 17.1576)
(comp–dt 65.0 100.0 13000.0 −200.0 −300.0 7.1576)
SPECIFIC
(moving ?veh ?vf (?st ?et))
(value (vel ?veh) ?vf &t2)
(value (vel ?veh) ?vi ?ti)
(lin_mom_cons ?veh ?vi ?ti ?vf &t2)
(det_params ?veh ?vi ?ti ?vf &t2 ?ve ?me ?mi)
(comp–tf ?vi ?ti ?vf ?mi ?me ?ve &t2)
(spec–val (vel (exh ?veh)) ?ve)
(spec–val (burn (fuel ?veh)) ?me)
(sum ?ti ?dt &t2)
(comp–dt ?vi ?vf ?mi ?me ?ve ?dt)
GENERAL

Figure 5. Portions of non–reactive explanations.

Reactive vs. Non–Reactive Acceleration Plan

Consider the performance of the system using the general reactive plan vs. using the general non–reactive plan. Given the problem of achieving a goal velocity of *vf* from the initial velocity of *vi* at time *ti*, the system may construct either a reactive plan from the general reactive plan or a non–reactive plan from the general non–reactive plan (Figure 6).

In computing the time at which to stop the rocket–firing, the non–reactive plan assumes a constant exhaust velocity and burn rate. Provided the expected values are accurate, it will achieve the goal velocity. However, if the actual values differ, the spaceship may not reach or may surpass the goal velocity. Even small deviations from the expected values could have devastating effects if a plan involved many such a priori computations, through which errors could get propagated and amplified. In contrast, the reactive plan makes no assumptions regarding the exhaust velocity and burn rate, and instead uses execution–time information to determine when to stop firing the rockets. It is thus more likely to achieve the goal velocity regardless of such variations.

Reactive Plan
[**fire–rockets** at time *ti*
monitor increasing velocity for the goal value of *vf*, binding *!t* to the time this value is reached
stop–fire–rockets at time *!t*]

Non–Reactive Plan
[**fire–rockets** at time *ti*
wait for time $t = \frac{e^{\frac{vf - vi}{ve}} M - M}{me}$
given vi = velocity at time *ti*
vf = goal velocity
ve = relative exhaust velocity
me = burn rate
M = initial mass of spaceship
stop–fire–rockets at time $tf = ti + t$]

Figure 6. Reactive vs. Non–Reactive Acceleration Plans.

For a classical planner to correctly compute when to stop the rockets, it would have to completely model the rocket–firing process—including the fuel–to–oxygen ratio, combustion chamber dimensions, nozzle geometry, material characteristics, and so on. This intractability is avoided in the integrated approach through the deferment of planning decisions and the utilization of execution–time information in addressing deferred decisions.

The integrated approach does have its own limitations. A planner in this approach incurs the additional cost of proving achievability as well as completing plans during execution. There may thus be cases in which this cost will be higher than the cost incurred by a classical planner, in which case complete a priori planning may provide the better solution. However, there are many planning problems which involve particular goals difficult to plan for prior to execution but easy to address during execution. Furthermore, these problems have simple achievability proofs which do not require the kind of real–world modeling likely to lead to intractability. In these problems, the integrated approach provides an attractive alternative to complete a priori reasoning.

Discussion And Conclusions

The integrated approach to planning presented in this paper is an attempt to solve the individual limitations of classical planning and reactive planning through an integration which augments a classical planner with the option of deferring achievable goals. Achievability proofs, which preserve the provably–correct nature of plans, are currently required to be absolute. However, the real world is rarely certain, and an important area for future research is that of probabilistic achievability proofs. The completors incorporated into learned general plans are responsible for gathering information for plan completion during execution. The system currently has simple completors with minimal runtime responsibilities. However, the identification of other classes of problems and the consideration of probabilistic achievability proofs will probably require more complicated completors with greater planning responsibilities. Another area for future work is in the addition of some quantitative knowledge to the system's reasoning abilitites. This is expected to extend the applicability of the system to more complicated problems and allow for a more thorough study of learning general reactive plans.

This work relates in different ways to other work in various research areas. The problems arising from imperfect a priori knowledge in classical planning was recognized as early as the STRIPS system, whose PLANEX component employed an execution algorithm which adapted predetermined plans to the execution environment [Fikes72]. Later work such as [Wilkins88] further addresses the problem of execution monitoring and failure recovery. The integrated approach to planning presented in this paper currently monitors execution only to complete the partial plans constructed prior to execution. However any monitoring and failure recovery capabilities applicable to a classical planner can also be incorporated into this approach. The idea of integrating a priori planning and reactivity has also been investigated in other work [Cohen89, Turney89]. The work presented in this paper differs primarily in that it focuses on the integration of planning and execution within a single plan rather than the integration of the planning and execution of multiple plans. The contingent explanation–based learning algorithm presented in this paper was developed to allow for the learning of general reactive plans for use in the integrated approach. Other work on learning to be reactive [Blythe89] has been on learning stimulus–response rules such as that used in reactive planning.

The planning approach described in this paper presents an integration of classical planning and reactive planning which provides for the construction of completable reactive plans. By constraining the deferred goals to only those which can be proven achievable, the integrated approach preserves the provably–correct nature of plans. Also, by utilizing information gathered during execution in addressing the deferred planning decisions, the integrated approach provides for sensitivity to the runtime environment. Contingent explanation–based learning extends standard explanation–based learning to enable general reactive plans to be learned from observation, by allowing a distinction to be made between planning decisions made prior to execution and those made during execution. The use of completable reactive plans simplifies a priori planning as well as reduces the reliance on perfect a priori information, and enables the construction of plans guaranteed to have successful completions.

Acknowledgments. This work has benefited greatly from the comments and direction provided by my advisor, Gerald DeJong, and also from the many enlightening (and entertaining) discussions with Steve Chien, Scott Bennett, Jon Gratch, and Michael Barbehenn.

References

[Agre87] P. Agre and D. Chapman, "Pengi: An Implementation of a Theory of Activity," *Proceedings of the National Conference on Artificial Intelligence*, Seattle, WA, July 1987.

[Blythe89] J. Blythe and T. M. Mitchell, "On Becoming Reactive," *Proceedings of The Sixth International Workshop on Machine Learning*, June 1989.

[Chapman87] D. Chapman, "Planning for Conjunctive Goals," *Artificial Intelligence 32*, 3 (1987).

[Chien89] S. A. Chien, "Using and Refining Simplifications: Explanation–based Learning of Plans in Intractable Domains," *Proceedings of The Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, August 1989.

[Cohen89] P. R. Cohen, M. L. Greenberg, D. M. Hart and A. E. Howe, "Trial by Fire: Understanding the Design Requirements for Agents in Complex Environments," *Artificial Intelligence Magazine 10*, 3 (1989).

[DeJong86] G. F. DeJong and R. J. Mooney, "Explanation–Based Learning: An Alternative View," *Machine Learning 1*, 2 (April 1986).

[Fikes71] R. E. Fikes and N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence 2*, 3/4 (1971).

[Fikes72] R. E. Fikes, P. E. Hart and N. J. Nilsson, "Learning and Executing Generalized Robot Plans," *Artificial Intelligence 3*, 4 (1972).

[Firby87] R. J. Firby, "An Investigation into Reactive Planning in Complex Domains," *Proceedings of the National Conference on Artificial Intelligence*, Seattle, WA, July 1987.

[Forbus84] K. D. Forbus, "Qualitative Process Theory," *Artificial Intelligence 24*, (1984).

[Hammond86] K. Hammond, "Learning to Anticipate and Avoid Planning Failures through the Explanation of Failures," *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, August 1986.

[Minton85] S. Minton, "Selectively Generalizing Plans for Problem–Solving," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, August 1985.

[Mitchell86] T. M. Mitchell, R. Keller and S. Kedar–Cabelli, "Explanation–Based Generalization: A Unifying View," *Machine Learning 1*, 1 (January 1986).

[Mooney86] R. J. Mooney and S. W. Bennett, "A Domain Independent Explanation–Based Generalizer," *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, August 1986.

[Schoppers87] M. J. Schoppers, "Universal Plans for Reactive Robots in Unpredictable Environments," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987.

[Shoham86] Y. Shoham, "Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence," PhD. Thesis, Yale University, Dept. of Computer Science, New Haven, CT, 1986.

[Stefik81] M. Stefik, "Planning and Metaplanning (MOLGEN: Part 2)," *Artificial Intelligence 16*, 2 (1981).

[Suchman87] L. A. Suchman, *Plans and Situated Actions*, Cambridge University Press, Cambridge, 1987.

[Turney89] J. Turney and A. Segre, "SEPIA: An Experiment in Integrated Planning and Improvisation," *Proceedings of The American Association for Artificial Intelligence Spring Symposium on Planning and Search*, March 1989.

[Wilkins88] D. E. Wilkins, *Practical Planning: Extending the Classical Artificial Intelligence Planning Paradigm*, Morgan Kaufman, San Mateo, CA, 1988.