Practical Temporal Projection

Steve Hanks*
Department of Computer Science and Engineering, FR-35
University of Washington
hanks@cs.washington.edu

Abstract

Temporal projection—predicting future states of a changing world—has been studied mainly as a formal problem. Researchers have been concerned with getting the concepts of causality and change right, and have ignored the practical issues surrounding projection. In planning, for example, when the effects of a plan's actions depend on the prevailing state of the world and that state of the world is not known with certainty, projecting the plan may generate an exponential number of possible outcomes. This problem has traditionally been eliminated by (1) restricting the domain so the world state is always known, and (2) by restricting the action representation so that either the action's *intended effect* is realized or the action cannot be projected at all. We argue against these restrictions and instead present a system that (1) represents and reasons about an uncertain world, (2) supports a representation that allows context-sensitive action effects, and (3) generates projections that reflect only the *significant* or *relevant* outcomes of the plans, where relevance is determined by the planner's queries about the resulting world state.

Introduction

Temporal projection consists of taking (1) a model of some world, and (2) the description of a particular series of events that happen in the world, and trying to predict the world's state after the events occur. In the planning community the events generally comprise a plan to be executed and the world model is a set of rules describing the effects of the actions that make up the plan.[1]

Temporal projection has been studied extensively in the literature on planning and acting,[2] but mainly as a formal problem: one starts with a logic that purports to capture notions involving time, action, change and causality, and argues that the inferences the logic licenses are the intuitively correct ones.

This paper takes a somewhat different view, arguing that temporal projection is an interesting *practical* problem. We argue that computing the possible outcomes of a plan, even if formally well understood, is computationally intractable, and thus one must restrict one's attention to the "important" or "significant" outcomes. This is especially true in domains for which the agent lacks perfect knowledge, and in which forces not under the agent's control can change the world—in other words, any interesting domain.[3]

We present an implemented framework for plan projection, which is actually part of a system that maintains the agent's *world model*—a network of beliefs that are both dynamic and tentative. This paper will focus on how commitments to act (potential plans) change the model; the planning process, conversely, is concerned with how the model guides the agent's choice of action.

We first discuss briefly a subsystem that performs probabilistic temporal reasoning—computing the likelihood that a proposition will be true at some point in time. We next motivate and present a representation for action and demonstrate the exact nature of the projection problem, then proceed to sketch our algorithm. We end by discussing the system's performance.

Probabilistic Temporal Reasoning

Central to the problem of predicting a plan's outcomes is determining the truth of various propositions (*e.g.* an action's preconditions) at some point in time (*e.g.* the time at which the action is to be executed). Since the

*This paper describes part of the author's thesis work at Yale University, advised by Drew McDermott and supported in part by DARPA grant DAAA15-87-K-0001. Thanks to Dan Weld and Tony Barrett for comments on this paper.

[1] Projection is a central problem in the area of qualitative physics as well. We will be concentrating on the area of planning and acting, but believe the problems we raise, and perhaps the solutions as well, are equally valid applied to QP situations. See, for example, [Kuipers and Chiu 1987]

[2] There are simply too many references to cite here, given the space limitations. An interested but uninformed reader might start with the relevant papers in [Ginsberg 1987] and work forward from there.

[3] See [Chapman 1987] for some discouraging results, even for artificially restricted domains.

agent will typically lack perfect information about the world (past, present, or future) we need some mechanism to express its uncertainty regarding the states of these propositions. We are using probabilities to represent this uncertainty, thus are concerned with computing the quantity P("φ true at time t"), where φ is some proposition and t is a time point. We will abbreviate this notation to $P(\varphi_t)$.

Three sorts of evidence get used in the computation: (1) reports from the sensors about φ's state, (2) symbolic causal rules of the form "if event E occurs while some fact P is true, then φ will become true at the next instant in time," and (3) background information (prior probabilities) about φ, E, P, and so on.

Uncertainty can come from a number of sources: (1) one can doubt whether the sensor reported correctly on φ's state, (2) one can be unsure as to whether a relevant event E actually occurred at some point in time, and (3) one can lack confidence in the causal rules: perhaps the rules mentioning φ aren't *really* necessary and sufficient predictors of φ's state changes. Our representation for propositions, rules, and the like takes into account all of these factors.

Important computational problems arise in implementing this approach: a tremendous amount of evidence must be brought to bear in computing the probability. Sensory observations of φ might extend arbitrarily far back into the past, as do the relevant causal rules (since they are implicitly quantified over all time points). Most of this evidence, however, will not affect φ's probability significantly.

Although computing φ's exact probability requires considering a potentially infinite amount of information, we might expect that under the right circumstances—sensors that are reasonably reliable and changes that occur reasonably infrequently—we can compute a good approximation of the probability using only a few pieces of evidence. The question is how good need a "good" approximation be?

The application program (planner) provides this information in the form of a *probability threshold* τ, indicating that it doesn't care what φ's probability is, but only to what side of τ the probability falls. A "good" approximation is therefore one that reports correctly with respect to the threshold. [Hanks 1988] presents a heuristic algorithm for limiting the search for evidence, the limit depending on how close the current estimate is to the threshold. The program also computes probabilities for joint events (logical conjunction), and monitors the database for new information that might invalidate the current approximation. Details of this subsystem appear in [Hanks 1988] and [Hanks 1990]. The projector invokes the probability-calculation subsystem by posing a *probabilistic query*—a question of the form "$P(\varphi_t) > \tau$?" The answer is returned in a data structure, called a *belief*, which is an assertion about which side of the threshold the probability lies (given current evidence).

Action representation

The typical action representation in the literature on action and planning (*e.g.* [Fikes and Nilsson 1971], [Lifschitz 1987], [Ginsberg and Smith 1988]) describes an action as a mapping from *preconditions* into *effects*. A good example is the "start the car" scenario, usually used to motivate the qualification problem: if there is fuel in the tank, the starter is working, ..., and there is no potato obstructing the tailpipe, then turning the key causes the engine to start.[4] A planner or projector faced with the hypothetical execution of such an action would first try to infer the action's preconditions; if they turn out to be true the effects would be noted in the resulting world state; if false, the action is said to be infeasible, and the planner could not reason about subsequent execution of the plan.

What's troublesome here is the planner's inability to reason about the action in cases where the precondition is false: in no sense is the action of turning the key of a car with, say, an obstructed tailpipe *impossible* or *meaningless*, yet that is what is implied by these systems' failure to project further.

In fact the action will have predictable effects even if the tailpipe is obstructed: it will take some time, wear down the battery, make some noise, perhaps generate a spark. It turns out that one particular effect, the engine running, will not be realized, and it may be the case that in some situations that's the only effect the planner is interested in, but it does not mean that executing the action is meaningless. What these systems have done is confused the notion of an action being *meaningful* or *conceivable* with the notion of an action's achieving its intended effect.[5] The former is rightfully a property of the action's definition, but the latter depends on the situation at hand. Actions will typically have a variety of effects depending on the circumstances, and it may be impossible to predict ahead of time which of those effects will turn out to be important.

If our representation for actions and plans is to support clever detection of planning bugs (*e.g.* trying to start the engine too many times thus running down the battery, or generating a spark in a gas-filled room) or innovative planning or plan recognition (running the car to warm the garage or to light the headlights without taxing the battery) we must associate intention or relevance with the *situation* rather than with the action's *definition*.

Consider as an alternative the code in Figure 1.[6] It is intended to represent the action "drive the truck

[4] The point is that there are myriad preconditions to any action—too many to verify explicitly.

[5] Pednault's ADL [1988a] is an exception, since it explicitly allows for context-dependent effects. In [Pednault 1988b] he notes various computational problems associated with using the representation for planning or projection.

[6] This is a simplified version of an action based on the robot-truck world of [Firby and Hanks 1987].

```
(action (travel R L1 L2)
  (if (not (loc TRUCK L1))
(A)   (outcomes INFEASIBLE)
      (if (< FUEL-LEVEL 10)
(B)       (outcomes  (loc TRUCK ON-ROAD)
                     (status TRUCK OUT-OF-GAS)
                     (consume FUEL 10)
                     (duration 0 60))
          (if (not (muddy R))
(C)           (outcomes (loc TRUCK L2)
                        (consume FUEL 10 15)
                        (duration 45 60))
              (if (chance 0.25)
(D)               (outcomes (loc TRUCK ON-ROAD)
                            (status TRUCK STUCK)
                            (muddy TRUCK)
                            (consume FUEL 0 30)
                            (duration 0 75))
(E)               (outcomes (loc TRUCK L2)
                            (muddy TRUCK)
                            (consume FUEL 25 30)
                            (duration 50 75]
```

Figure 1: Traveling on a dirt road

down road R from location L1 to location L2. The code describes a mapping from state descriptions to sets of *outcomes*.[7] Note that actions can still be infeasible, as outcome set (A) indicates, but the idea is that it is truly meaningless to contemplate a trip starting at location L1 if you are not at location L1. Running out of gas or getting stuck in the mud may be inconvenient, may even be impossible (probability 0) under some circumstances, but it is not inconceivable.

The feasible outcome set (D) indicates that as a result of executing the **travel** action the truck is somewhere on the road, stuck in the mud, has used up some quantity of fuel less than 30 gallons, and somewhat less than 70 time units will have passed.

An outcome set's *label*, the conjunction of all the **if** propositions leading to it, describes states of the world in which the outcome set will be realized. Set (D), for example, will be realized if the action is executed when the world satisfies the the following condition:

(loc TRUCK L1)∧ (not (< FUEL-LEVEL 10))∧
(muddy R)∧ (chance 0.25)

The action's **if-then-else** form ensures that the labels for different outcome sets will be mutually exclusive and that the labels for *all* an action's outcome sets will be exhaustive. Therefore exactly one set's label will be true at any point in time, so no matter when the action is executed exactly one set of outcomes will be realized. Of course we may not know what that state of the world is, even after the action is executed.[8]

[7] Outcomes are essentially the same as the "effects" above, but extended to allow reasoning about sets, continuous quantities, passing time, and so on.

[8] We account for an action's having indeterminate effects by introducing the chance predicate, *e.g.* Figure 1 outcome set (D), whose real truth value is never known, but whose probability can easily be computed.

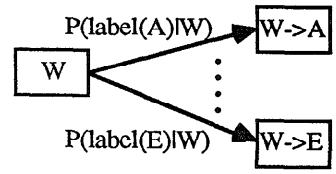

Figure 2: Projecting a single action

An outcome set's label is a formula that can be posed as a probabilistic query (given a time point representing the proposed time of execution). We can therefore compute a probability distribution over the outcome sets relative to a time point, asking, for each outcome set (X), "what is the probability that (X)'s label will be true?"

We can then view the process of projecting a single action as building a tree as pictured in Figure 2. Nodes are states of the worlds, and each arc represents the possibility that a particular outcome set will be realized, given that the world is in a state like that of its parent node. The probability associated with an arc, *e.g.* P(label(A)|W) is the probability that outcome set A's label will be true, and thus its outcomes will be realized, given that the world is in state W at execution time. W→A is then the state of the world resulting from the action's outcome set A being realized in world W.[9]

Scenarios and Projection

To project a sequence of actions $A_1, A_2, \ldots$, we just iterate the process of single-action projection: we project action A_{i+1}, in *each* of the world states resulting from the execution of A_i. Projecting a plan therefore results in a directed tree which branches forward in time. We call this tree a *scenario* structure, and each path through the tree a *chronicle*. Each chronicle has an associated probability, and the probabilities of all chronicles in a scenario must sum to 1.

Figure 4(a) shows the structure generated by projecting the sequence (load O1); (load O2); (travel BR L3 L4) whose action descriptions appear in Figure 3. The load actions—load an object into the cargo bay—have a 0.1 chance of failing, but take 1 time unit in either case; BR is a bridge that will collapse if the truck is carrying two or more items when it crosses.

The projection problem manifests itself in the proliferation of chronicles: a plan with n actions each containing an average of m outcome sets will gener-

[9] The projector can also account for the possibility that other relevant events occur while the action is executing, but that process is beyond the scope of this paper.

```
(action (load ?x)
  (if (not (reachable ?x))
(F)   (outcomes INFEASIBLE)
      (if (chance 0.9)
(G)       (outcomes (holding ?x)
                    (duration 1))
(H)       (outcomes (duration 1)))))

(action (travel BR L3 L4)
  (if (not (loc TRUCK L3)
(I)       (outcomes INFEASIBLE)
      (if (>= (card CARGO) 2)
(J)       (outcomes (loc TRUCK RAVINE)
                    (status TRUCK MANGLED)
                    (duration 0 7))
(K)       (outcomes (loc TRUCK L4)
                    (duration 3 7))))
```

Figure 3: More action code

ate roughly m^n chronicles. Although we can obviously do some pruning of the tree, by eliminating zero-probability and infeasible chronicles for example, an impractical number of feasible and possible chronicles remain.[10]

Figure 4(a) shows an exhaustive projection of the plan above. Every arc in the tree has a single outcome set associated with it, so the tree branches with every possible outcome for every possible action. We can reduce proliferation of the tree by instead associating a *set* of outcome sets with each arc. The intuition is that the distinctions noted by outcome sets *within* each set are irrelevant, while the distinctions between outcomes in *different* sets are significant.

Referring back to the `travel` action in Figure 1, suppose that we care only about whether the truck reaches its destination L2. In that case we don't care about the distinctions implied by outcomes sets (A) (B) and (D), nor about the distinctions between (C) and (E). So we form two sets and produce two branches in the tree instead of five. We call each of these groupings a *bundle*. Each arc in the scenario tree has an associated bundle, which contains one or more outcome sets.

Bundling outcome sets speeds the process of computing label probabilities as well, in that the label associated with a bundle (the disjunction of its member's labels) may be much simpler than the individual labels. As an extreme case suppose we are only interested in whether Figure 1's action is feasible or not. In that case we put (A) in one bundle and the rest of the outcome sets in another. Computing the probability of the second bundle (which has four members) is simply a matter of evaluating (loc TRUCK L1). Note that as long as we assign each outcome set to exactly one

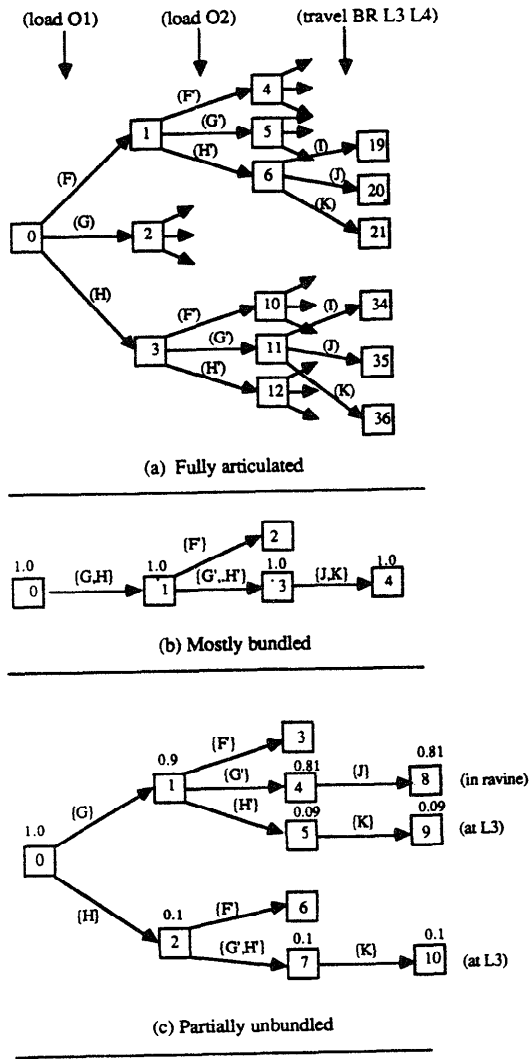

Figure 4: Bundled and unbundled scenario trees

[10] We have been testing the program with examples in which $n = 10$ and $m = 6$, which gives rise to some 6×10^7 chronicles in the worst case. Pruning infeasible and impossible chronicles gets rid of some $\frac{3}{4}$ of them, but that still leaves more than ten million.

bundle, the bundles still partition the set of possible execution-time situations, and thus their probabilities sum to 1.

The problem with bundling outcome sets is that we can make less precise predictions. When we project an action across a bundle of outcome sets we can infer only the weakest conclusion allowed by all members of that bundle. If we construct the bundle {(C),(E)} from Figure 1, for example, we can conclude that the truck will be at L2, but we can only make vague predictions about the fuel consumption (between 10 and 30 gallons), and the action's duration, and we cannot predict whether or not the truck will be muddy. But then again we may not care.

Practical projection, then, is a process of balancing the need for parsimony in the scenario tree and speed in computing chronicle probabilities against the need to make precise predictions about what the world will be like after the plan is executed. The former argues for keeping much of the tree's structure implicit by forming a few large bundles; the latter argues for representing the tree explicitly. Obviously we want to make explicit the "important" or "significant" or "relevant" distinctions in the tree and leave the rest implicit. But how do we determine which distinctions these are?

The answer is that we can use the probabilistic temporal queries, which may be posed by the planner or generated as the projector computes a chronicle's probabilities: the projector wants to articulate exactly enough of the tree to give the best possible answers to the queries it receives. Of course the nature of these queries will not be known in advance, so the projector must be able to articulate the tree more fully—to "unbundle" outcome sets—on demand.

To see how the process works, consider again the sequence `(load O1); (load O2); (travel BR L3 L4)`. The planner additionally supplies the projector with a time point at which to begin, initial assumptions (the truck is at L3, objects O1 and O2 are reachable, the cargo bay is empty, and the fuel tank is full), and a probability threshold. Chronicles whose probabilities fall below this threshold will be abandoned, at least initially.

The projector proceeds to project each action in sequence, assuming that the only distinction the planner is interested in is between feasible and infeasible outcomes. The initial projection appears in Figure 4(b).[11] The projector returns the (single) chronicle representing a feasible plan completion: the chronicle ending in node 4. At this point we can say little about where the truck is or what it is carrying.

Now suppose the planner poses the query "is the truck at L3?" The projector notes that the current tree yields an ambiguous answer to this question, but that splitting the bundle {(J),(K)} would result in a more precise answer. In the process of splitting that bundle it needs to compute label probabilities for (J) and (K), leading it to ask "(>= (card CARGO) 2)?" So the projector poses that query with respect to node 3. Once again the answer is ambiguous: currently we can predict only that the cargo bay will contain between 0 and 2 items, but two bundles, {(G), (H)} and {(G'), (H')} contribute to the ambiguity, so the projector considers the possibility of splitting them. Details of when and how the projector splits a bundle of outcome sets appear in [Hanks 1990, Chapter 4].

When all the splitting completes, queries get answered, and projection finishes, the tree appears as in Figure 4(c). Now there are three feasible chronicles, each of which answers unambiguously the question "is the truck at L3?"[12] The probability for a query Q is the sum $\sum_c P(Q \mid c)P(c)$ where c varies over all chronicles. The probability for "is the truck at L3" is therefore roughly 0.19. The probability of "is the truck at L3 and carrying both O1 and O2" would be 0, and could be computed without further splitting.

Maintaining the world model

Our system performs three main functions: answering probabilistic queries, projecting plans, and monitoring the database for information that would change existing beliefs. It is interesting to note how closely interrelated the three processes are: adding plans causes label probabilities to be computed, which gives rise to queries. A query may demand that the scenario tree be split, thus causing more projection, and so on. A query may also cause previously abandoned chronicles, like the ones ending in Nodes 3 and 6, Figure 4(c), to be projected further. New information, like relevant sensor reports, may cause beliefs to change, plan commitments to be rethought, new queries to be posed, and new plans to be projected.

The system can thus be regarded as maintaining the planner's world model, one in which evidence takes the form of plan commitments and sensor reports, and aspects of the model are revealed through the *belief* data structures.

Performance

It is hard to make precise general statements about the projector's performance. The worst case, involving a complete articulation of the tree, is exponential both in the number of actions and in the average number of outcome sets per action. In general the tree proliferates to the extent the planner asks questions about plan outcomes that depend on propositions whose state the projector cannot predict definitively (with probabilities near 0 or 1). So performance depends on the

[11] We have pruned away 0-probability outcomes, like one in which O1 instantaneously becomes unreachable, or the truck inexplicably gets moved away from L3. Node 2 represents a situation in which somebody removes object O2 during the time that O1 was being loaded, but it has a small enough probability that it is not projected further.

[12] Node 8, Figure 4(c) says "no," 9 and 10 say "yes."

queries, both their types and their thresholds, and on the underlying probabilistic model.

A more interesting question is whether the projector always produces the splits necessary, and *only* the splits necessary, to answer a particular query. If it fails to make a necessary split it will report an overly vague answer to a query; if it makes unnecessary splits it will report the correct answer but do so inefficiently. We address this question in [Hanks 1990], but the short answer is that the projector will never fail to make a relevant split in the tree, but will sometimes make unnecessary splits. The decision whether to split an arc is made locally at that arc, but we cannot always determine locally whether a split is really necessary given other splits that might be made. Superfluous splitting does not seem to be a problem in practice, however.

We have implemented the projector and tested it on fairly complex examples, involving 15 or so plan steps including conditionals, loops, and information-gathering actions. Each projection took about 4 minutes on average, but we have reason to believe that more powerful computing equipment and superficial code optimizations would cut this figure in half. We were surprised to learn that most of the time was spent in the temporal database manager module (which maintains the network of time points and constraints), so improving its efficiency will significantly improve performance as well.

Conclusion

Systems for action and planning have made stringent assumptions about the domains in which they operate and the nature of the actions they manipulate. They typically assume that no events occur apart from those the planner intends, and that an action either achieves its intended effect or cannot be meaningfully executed.

These assumptions together mask an important practical problem: when the effects of one's actions depend on the state of the world at execution time, and when one is uncertain about that world state, one faces an explosion of possible plan outcomes. The hope is that most of these outcomes can be dismissed as uninteresting, improbable, or both.

We have presented a system to manage the process of hypothetical reasoning about actions and plans. The system builds a "scenario tree," tracing the possible outcomes of a plan, but tries to keep as much of the tree's structure implicit as possible, thus avoiding the explosion of possible plan outcomes. It uses the planner's queries as a guide to what aspects of the world, thus what outcomes of the plans, are important and deserve explicit consideration.

References

[Chapman 1987] David Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32(3):333–378, 1987.

[Fikes and Nilsson 1971] Richard Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3):189–208, 1971.

[Firby and Hanks 1987] R. James Firby and Steven Hanks. The simulator manual. Technical Report 563, Yale University, Department of Computer Science, November 1987.

[Ginsberg and Smith 1988] Matthew L. Ginsberg and David E. Smith. Reasoning about action I: A possible worlds approach. *Artificial Intelligence*, 35(2):165–196, 1988.

[Ginsberg 1987] Matthew L. Ginsberg, editor. *Readings in Nonmonotonic Reasoning.* Morgan-Kaufmann, 1987.

[Hanks 1988] Steven Hanks. Representing and computing temporally scoped beliefs. In *Proceedings AAAI*, pages 501–505, 1988.

[Hanks 1990] Steven Hanks. Projecting plans for uncertain worlds. Technical Report 756, Yale University, Department of Computer Science, January 1990.

[Kuipers and Chiu 1987] Benjamin Kuipers and Charles Chiu. Taming intractible branching in qualitative simulation. In *Proceedings IJCAI*, pages 1079–1085, 1987. Also appears in [Weld and de Kleer 1989].

[Lifschitz 1987] Vladimir Lifschitz. Formal theories of action. In Frank Brown, editor, *The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Workshop.* Morgan-Kaufmann, 1987.

[Pednault 1988a] Edwin P.D. Pednault. Extending conventional planning techniques to handle actions with context-dependent effects. In *Proceedings AAAI*, pages 55–59, 1988.

[Pednault 1988b] Edwin P.D. Pednault. Synthesizing plans that contain actions with context-dependent effects. *Computational Intelligence*, 4(4):356–372, 1988.

[Weld and de Kleer 1989] Daniel S. Weld and Johan de Kleer, editors. *Readings in Qualitative Reasoning about Physical Systems.* Morgan-Kaufmann, 1989.