Contingency Selection in Plan Generation

Nilufer Onder
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
nilufer@cs.pitt.edu

Martha E. Pollack
Department of Computer Science
and Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA 15260
pollack@cs.pitt.edu

Abstract

A key question in conditional planning is: how many, and which of the possible execution failures should be planned for? One cannot, in general, plan for all the possible failures because the search space is too large. One cannot ignore all the possible failures, or one will fail to produce sufficiently flexible plans. In this paper, we describe an approach to conditional planning that attempts to identify the contingencies that contribute the most to a plan's overall utility. Plan generation proceeds by handling the most important contingencies first, extending the plan to include actions that will be taken in case the contingency fails. We discuss the representational issues that must be addressed in order to implement such an algorithm, and present an example which illustrates our approach.

Introduction

Classical AI plan generation systems assume static environments and omniscient agents, and thus ignore the possibility that events may occur in unexpected ways—that contingencies might arise—during plan execution. The problem with classical planners is, of course, things do not always go "according to plan."

In contrast, universal planning systems and more recent MDP-based systems make no such assumptions. They produce "plans" or "policies" that are functions from states to actions. The problem with universal planning and MDP systems is that the state space is typically much too large for them to be effective.[1]

Conditional planners take the middle road. They allow for both conditional actions with multiple possible contingencies, and for sensing actions that allow agents to determine which contingency occurred (Draper, Hanks, & Weld 1994; Etzioni *et al.* 1992; Goldman & Boddy 1994; Peot & Smith 1992; Pryor & Collins 1993). A key question in conditional planning is: how many, and which contingencies should be selected so that the plan can be extended to include actions that will be taken in case the contingency fails?

In this paper, we present an iterative refinement planning algorithm that attempts to identify the influence each contingency would have on the outcome of plan execution, and then gives priority to the contingencies whose failure would have the greatest negative impact. The algorithm first finds a skeletal plan to achieve the goals, and then during each iteration selects a contingency whose failure will have a maximal *disutility.* It then extends the plan to include actions to take in case the selected contingency fails. Iterations proceed until the expected utility of the plan exceeds some specified threshold.

Weighing Contingencies

We start with a basic idea: a plan is composed of many steps that produce effects to support the goals and subgoals, but not all of the effects contribute equally to the overall success of the plan. Of course, if plan success is binary—plans either succeed or fail—then in some sense all the effects contribute equally, since the failure to achieve any one results in the failure of the whole plan. However, as has been noted in the literature on decision-theoretic planning, plan success can often be modeled using rich utility functions. In this paper, we focus on the fact that the goals that a plan is intended to achieve may be decomposable into subgoals, each of which has some value associated with it.

For example, consider a car in a service station and a plan involving two goals: installing the fallen front reflector and repairing the brakes. The value of achieving the former goal may be significantly less than the value of achieving the latter. Consequently, the effects that support only the former goal (e.g., getting the reflector from the supplier) contribute less to the overall success of the plan than the effects that support only the latter goal (e.g., getting the calipers from the supplier). Effects that support both goals (e.g., having the number

[1]Recognizing this problem, (Dean *et al.* 1995) describes an algorithm that starts with an initial policy for a restricted set of states, and constructs policies for increasingly large sets of states.

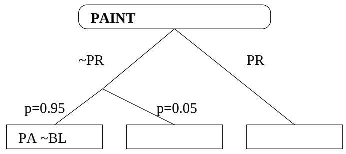
of the supplier dialed to place the orders for the parts) will have the greatest importance. We will say that an effect e *supports* a goal g if there is some path of causal links through the plan starting at e and ending at g.

We define *contingencies* to be alternative (subsets of) outcomes of probabilistic actions. Continuing with the current example, if we repair the brakes, we may or may not succeed in having the brakes functioning properly (perhaps because the new calipers are sometimes defective). Having the brakes functioning correctly is one contingent outcome of the repair action; not having them is another. A deterministic action is simply one with a single (contingent) outcome.

The importance of planning for the failure of any particular contingency can then be seen to be a factor of two things: the probability that the contingency will fail, and the degree to which the failure of the contingency will effect the overall success of the plan. To compute the former, one needs to know both the probability that the action generating the particular contingency will be executed, and the conditional probability of the contingency given the action's occurrence. Computing the degree to which the failure of a contingency will effect the overall success of the plan is, in the general case, very difficult. For the current paper, however, we make two strong simplifying assumptions. First, we assume that the top-level goals for any planning problem can be decomposed into subgoals, each of which can be assigned a fixed scalar value. Moreover, they are *utility independent* (Haddawy & Hanks 1993), i.e., the utility of one subgoal does not depend on the degree to which any other goals are satisfied. Second, we assume that all failures are equally difficult to recover from at execution time. In practice though, even the failure of some low probability contingencies that support highly valuable goals may not be worth planning for, because they are relatively easy to handle "on the fly" (for example, if the supplier does not have the necessary parts, a plan to call another supplier can easily be constructed).

With these two assumptions, we can compute the importance of planning for the failure of a given contingency—what we term the *expected disutility* of a contingency's failure: it is simply the probability of its failure, multiplied by the sum of the values of all the top-level goals it supports. Planning for the failure of a contingency means constructing a plan that does not depend on the effects produced when the contingency occurs, i.e., a plan that will succeed even if the contingency fails.

In what follows, we formalize these notions. We adapt the representation for probabilistic actions that was developed for BURIDAN (Kushmerick, Hanks, & Weld



PAINT : {<{~PR},0.95,{PA,~BL}>,<{~PR},0.05,{}>,<{PR},1,{}>}

Figure 1: The `PAINT` action.

1995, p. 247). Throughout this paper, we will use the term *state* to refer to a complete description of the world at any instant. The probability of an expression e with respect to a state st is defined as:

$$P[e|st] = \begin{cases} 1, & \text{if } e \subseteq st, \\ 0, & \text{otherwise.} \end{cases}$$

Action Representation

We use the term *causal action* for the actions that alter the state of the world by producing effects. For instance, the `PAINT` action depicted in graphical and textual formats in Fig. 1 has the effects of having the part painted (`PA`), and removing the blemishes on the part (~`BL`) 95% of the time when the part has not been processed (~`PR`).

Definition 1 (*Causal action*). A *causal action* `N` is defined by a set of *causal consequences*

$$\texttt{N}: \quad \{< t_1, \rho_{1,1}, e_{1,1} >, \ldots, < t_1, \rho_{1,l}, e_{1,l} >, \ldots, \\ < t_n, \rho_{n,1}, e_{n,1} >, \ldots, < t_n, \rho_{n,m}, e_{n,m} >\}.$$

For each i, j: t_i is an expression called the consequence's *trigger*, and $e_{i,j}$ is a set of literals called the *effects*. $\rho_{i,j}$ denotes the probability that the effects in $e_{i,j}$ will be produced given that the literals in t_i hold. The triggers are mutually exclusive and exhaustive.

Following the BURIDAN model, the probability distribution after an action is executed depends both on the state of the world before the action is executed and on the changes caused by the action. The effects the execution of an action has on the state of the world is defined in the usual manner, i.e., $RESULT(e, st)$ is a new state that has the negated propositions in e deleted from and the non-negated propositions added to st.

In addition to causal actions, conditional plans require *observation actions* so that the executing agent will know which contingencies have occurred, and hence, which actions to perform. Observational consequences are different from causal consequences in that they record two additional pieces of information; namely, the *label* (shown in //), and the *subject* (shown in \\). The label shows *which proposition's* value is being reported on. In some situations, when an aspect

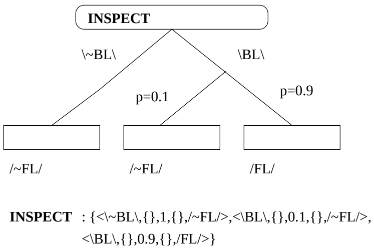


Figure 2: The `INSPECT` action.

of the world is not directly observable, another (correlated) aspect of the world is used by the sensor to produce the label. The subject shows what the sensor looks at. For instance, a robot inspecting a part might look at whether it is blemished (`BL`) or not, to report whether it is flawed (`FL`) or not (Fig. 2).The names inside the subject and label brackets are not arbitrary strings, but they refer to propositions that can be true or false in the world.

Definition 2 (*Observation action*). An *observation action* `N` is a set of *observational consequences*

$$\begin{aligned}\texttt{N}: \quad & \{< sbj_1, t_1, \rho_{1,1}, e_{1,1}, l_{1,1} >, \\ & \ldots, < sbj_1, t_1, \rho_{1,l}, e_{1,l}, l_{1,l} >, \ldots, \\ & < sbj_n, t_n, \rho_{n,1}, e_{n,1}, l_{n,1} >, \\ & \ldots, < sbj_n, t_n, \rho_{n,m}, e_{n,m}, l_{n,m} >\}.\end{aligned}$$

For each i, j: sbj_i is the *subject* of the sensor reading, t_i is the *trigger*, $e_{i,j}$ is the set of *effects* (an observation action can alter the state of the world), and $l_{i,j}$ is the label that shows the *sensor report*. $\rho_{i,j}$ is the probability of obtaining the effects and the sensor report. The subjects and triggers are mutually exclusive and exhaustive. The proposition referred to in the subject of an observation action cannot occur in a trigger expression of the same action. But the subject and the label of a consequence can refer to the same proposition.

This observe action representation is similar to C-BURIDAN's (Draper, Hanks, & Weld 1994). However, C-BURIDAN labels are arbitrary strings that do not relate to propositions, and the subject of the sensor reading is not distinguished.

Having defined both causal and observational consequences, we will use the term "consequence" without a qualifier when it is clear from the context which is being meant, or when it makes no difference.

Contingencies

An important issue in the generation of probabilistic plans is the identification of equivalent consequences. Suppose that an action is inserted to support a proposition `p`. If all the outcomes of the action produce `p` as an effect, then all the outcomes are equivalent, leaving no need to consider planning for the alternatives.

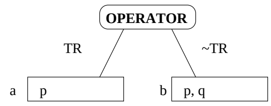


Figure 3: Multiple outcomes.

We keep track of the equivalent outcomes by clustering them into "contingencies" when an action is inserted into a plan, or when an existing step is used to support a proposition. If the operator in Fig. 3 is used to support `p`, then {a,b} is the single contingency, but if it is used to support `q`, then {a} and {b} are alternative contingencies. The contingencies of a step are relative to the step's intended use. Therefore, contingencies of a step are not defined once, but they are rather formed every time the step is used to support a proposition. In short, a *contingency* is a subset of the consequences of an action, containing all those outcomes relevant to the action's purpose.

When the outcomes of an action are clustered into contingencies, the planner needs to factor the trigger formulae to determine which triggers actually influence the proposition supported by the contingency. If a trigger can be identified as not influencing the result, then the planner can avoid (over-)committing to adopting that trigger. In general, logical manipulation of the trigger formulae is needed to keep track of which triggers contribute to the truth value of the formula. The resulting expression is called a *condition*, and it tells when the contingency holds. A condition for contingency i is of the form: $< \{t_{i,1}, \rho_{i,1}\}, \ldots, \{t_{i,j}, \rho_{i,j}\} >$ (implicit disjunction).

The Planning Problem

A planning problem is defined by initial conditions encoded in the usual way as a start step (`START`: $\{< \emptyset, \rho_1, e_1 >, \ldots, < \emptyset, \rho_n, e_n >\}$), the goal (`GOAL`: $\{< \{g_1, \ldots, g_n\}, 1, \emptyset >\}$), a required utility threshold, and a set of available actions. We assume that the values associated with the top-level goals are kept separately, and use the function $val(g_i)$ to retrieve the value associated with the *top-level* goal g_i. Throughout this paper, we will use the term "action" to refer to action types, whereas "step" will refer to instantiated actions. We will use T_c to refer to the set of causal steps, and T_o to refer to the set of observational steps in a plan.

Plans

While constructing the plan, the planner records the relationship of the steps in the plan using *causal links*, which emanate from contingencies rather than individual outcomes. For instance, in Fig. 4, the first contingency of `SHIP` produces `PR` for `GOAL`.

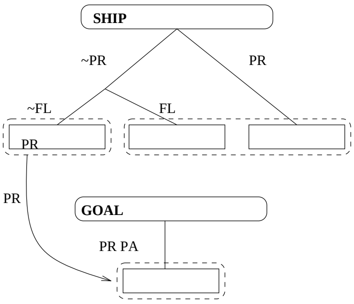


Figure 4: A causal link.

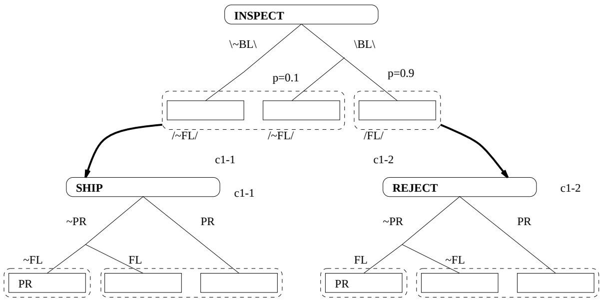


Figure 5: Observation links.

Definition 3 (*Causal link*). A *causal link* is a 5-tuple $< S_i, c, p, c_k, S_j >$. Contingency c of step $S_i \in T_c \cup T_o$ is the link's *producer*, contingency c_k of step $S_j \in T_c \cup T_o$ is the link's *consumer*, p is the proposition *supported* by the contingency.

An *observation link* is used to specify which steps will be taken depending on the report provided by an observation step. It also emanates from a contingency. For instance, in Fig. 5, `SHIP` will be executed if `INSPECT` reports /`FL`/, and `REJECT` will be executed otherwise.

Definition 4 (*Observation link*). An *observation link* is a 4-tuple $< S_i, c, l, S_j >$. Contingency c is the collection of consequences of $S_i \in T_o$ that report l. An observation link means that $S_j \in T_c \cup T_o$ will be executed only when the sensor report provided by S_i is l.

Definition 5 (*Partially ordered plan*). A *partially ordered plan* is a 6-tuple, $< T_c, T_o, O, L_c, L_o, S >$, where, T_c and T_o are the sets of causal and observation steps, O is a set of temporal ordering constraints, L_c is a set of causal links, L_o is a set of observation links, and S is a set of subgoals.

In a plan, each step has a (possibly null) *context label*. A step with a context label $C_{i,j}$ is executed when contingency j of observation step i is realized ((Peot & Smith 1992; Pryor & Collins 1993) describe how context labels are created and used).

A totally ordered plan is a total ordering consistent with the partially ordered plan.

Definition 6 (*Totally ordered plan*). A totally ordered plan is a sequence of steps each of which is either a *simple step*, or a *composite step*. A simple step is a single causal step. A composite step is an observe step and the branches that will be taken depending on the sensor report. It is of the following form:

$$\begin{array}{ll} O & < \{t_{1,1}, \rho_{1,1}\}, \ldots, \{t_{1,l}, \rho_{1,l}\} >< S_{1,1}, \ldots, S_{1,m} >, \\ \ldots, & < \{t_{p,1}, \rho_{p,1}\}, \ldots, \{t_{p,q}, \rho_{p,q}\} >< S_{p,1}, \ldots, S_{p,r} > . \end{array}$$

$< S_{i,1}, \ldots, S_{i,m} >$, called *branch* i, includes the steps that will be executed if contingency i of the observe step is realized. $< \{t_{i,1}, \rho_{i,1}\}, \ldots, \{t_{i,l}, \rho_{i,j}\} >$ is the condition for contingency i. For instance,

$$\begin{array}{ll} O & < \{BL, 1\}, \{BL, 0.1\} >< S_1, S_2 >, \\ & < \{BL, 0.9\} >< S_3 > \end{array}$$

means that steps $< S_1, S_2 >$ will be executed when `BL` is false or 10% of the time when `BL` is true. Step $< S_3 >$ will be executed 90% of the time when `BL` is true.

Expected Value

Definition 7 *The probability of a state after a (possibly null) sequence of steps is executed*:

$$P[st'|st, <>] = \begin{cases} 1, & \text{if } st' = st, \\ 0, & \text{otherwise.} \end{cases}$$

$$P[st'|st, < S >] = \sum_{< t_i, \rho_{i,j}, e_{i,j} > \in S} \rho_{i,j} P[t_i|st] P[st'|RESULT(e_{i,j}, st)],$$

where S is a simple step.

$$P[st'|st, < S_1, \ldots, S_n >] = \begin{cases} \sum_u P[u|st, S_1] P[st'|u, < S_2, \ldots, S_n >], \\ \qquad \text{if } S_1 \text{ is a simple step,} \\ \sum_i \sum_j \rho_{i,j} P[t_{i,j}|st] P[st'|st < S_{i,1}, \ldots, S_{i,m} >], \\ \qquad \text{if } S_1 \text{ is a composite step.} \end{cases}$$

In a conditional plan, the steps that are part of a branch are executed only when the condition for taking the branch holds. The steps that are not part of a composite step are always executed, i.e., $P[S \text{ is executed}] = 1$, if S is not part of a composite step.

Definition 8 *Probability that a step which is part of a composite step is executed*: Suppose that S is a step in the sequence $< S_1, \ldots, S_n >$, and S_j is the innermost composite step that contains S. If S is in branch i which has the condition $< \{t_{i,1}, \rho_{i,1}\}, \ldots, \{t_{i,m}, \rho_{i,m}\} >$, then,

$$P[S \text{ is executed}] = P[S_j \text{ is executed}] \sum_{k=1}^{m} \rho_{i,k} P[t_{i,k}| < S_1, \ldots, S_{j-1} >].$$

Definition 9 (*Expected value of a totally ordered plan*). The expected value of a plan $< S_1, \ldots, S_n >$ is defined in terms of the probability that each of the z goal propositions will be true after S_n is executed, and the value of the goal.

$$\sum_{i=1}^{z} \sum_{u} P[g_i|u] \times P[u| < S_1, \ldots, S_n >] \times val(g_i).$$

The expected value of a contingency includes the value of all the top-level goals it directly supports and the expected value of the contingencies supported by the contingency. The main idea is to find out which top-level goal(s) will fail if the step fails to produce this contingency. For example, suppose that a contingency c supports top-level goal g_i with probability 0.7, and supports the same goal with probability 0.8 via another path of causal or observation links. If c fails, the most that will be lost is the support for g_i with probability 0.8, and thus the expected value of c is $0.8 \times val(g_i)$. Therefore, while propagating the values to a contingency, we take the maximum for each top-level goal.

Definition 10 *Expected value of a contingency*: Suppose that contingency c has outgoing causal links to consumer contingencies $c_1, \ldots, c_n$ and possibly an outgoing observation link to step S. Assume that k_i is the probability that c_i supports goal g, and k_{n+1} is the probability that S supports g. Then, the expected value of c with respect to g is:

$$evg(c, g) = \begin{cases} val(g), \text{if } c \text{ directly supports } g, \\ max(k_1, \ldots, k_{n+1}) \times val(g), \text{otherwise.} \end{cases}$$

For z top-level goals, the expected value of contingency c is

$$EV(c) = \sum_{i=1}^{z} evg(c, g_i).$$

Finally, the expected disutility of a contingency is defined as the probability that it fails multiplied by the value of the contingency. The probability of failure is a factor of the probability that the action generating the contingency will be executed, and the conditional probability of the contingency given the action's occurrence.

Definition 11 *Expected disutility of the failure of a contingency*: Let S be a step and c be the ith contingency of S. If the condition for c is $< \{t_{i,1}, \rho_{i,1}\}, \ldots, \{t_{i,j}, \rho_{i,j}\} >$, then the expected disutility of c is defined as

$$P[S \text{ is executed}] \times (1 - (\textstyle\sum_{k=1}^{m} \rho_{i,k} P[t_{i,k}| < S_1, \ldots, S_{i-1} >]) \times EV(c).$$

Note that this definition gets more complicated depending on the structure of the steps $< S_1, \ldots, S_{i-1} >$. We omit the complementary definitions from this paper.

1. **Skeletal plan.**
   `Construct a skeletal plan.`
2. **Plan refinement.**
   `While the constructed plan is below the given utility threshold:`
   - `2a. Select the contingency with the highest expected disutility.`
   - `2b. Extend the plan to include actions that will be taken in case the contingency fails.`

Figure 6: The planning algorithm.

The Planning Algorithm

The high level specification of the planning algorithm is shown in Fig. 6. A *skeletal plan* is a plan in which every subgoal is supported by exactly one causal link or observation link. A single link constitutes the minimal support for a subgoal, and within the current scope of this paper, we are not concerned with increasing the likelihood that certain contingencies will occur by adding extra links of support. The skeletal plan can be constructed by using a probabilistic planner such as BURIDAN with the restriction that the planner is allowed to have exactly one link that supports a subgoal.

During the construction of the skeletal plan, when a contingency is used to support a subgoal, the triggers of the contingency are adopted as subgoals. The skeletal plan is considered to be "complete" when all the subgoals are supported by exactly one link, and all the threats in the plan are resolved. At this point, the expected disutility for all the contingencies is calculated. Suppose that c is the contingency with the maximal disutility, and the causal link emanating from c is $< S_i, c, p, c_k, S_j >$. Because the proposition supported by c is p, an observe action (say S_o) that reports the status of p is selected and inserted into the plan. S_o ordered to come after S_i so that the agent can look to see whether p is true after executing S_i. The contingencies of S_o are formed in such a way that c_1 contains all the consequences that produce the label $/p/$, and c_2 contains all the other consequences. The causal link $< S_i, c, p, c_k, S_j >$ is removed and an observation link $< S_o, c_1, /p/, S_j >$ is inserted to denote that S_j will be executed when the sensor report is $/p/$ (Fig. 7).

In addition, a causal link, $< S_i, c, sbj, c_1, S_o >$ is inserted. This link prevents the insertion of an action that would perturb the correlation between the propositions coded in the subject and the label.

Conditional planning then proceeds in the CNLP style, i.e., the goal is duplicated and labeled so that it cannot receive support from the steps that depend on contingency c_1 of the observe step. Note that with CNLP, the branches do not rejoin, whereas with C-

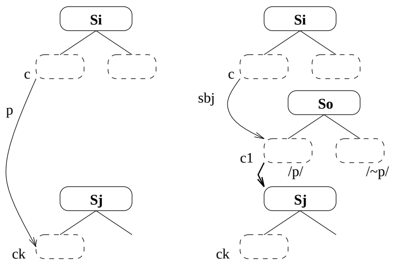


Figure 7: Inserting a new observe action.

BURIDAN they can. To the best of our knowledge, methods for rejoining branches has not been explored in the context of CNLP style algorithms.

Example

Consider the following example solved by C-BURIDAN. The goal is to process (`PR`) and paint (`PA`) parts. Initially, 30% of the parts are flawed (`FL`) and blemished (`BL`), the rest are in good condition. The "correct" way of processing a part is to reject it if it is flawed, and to ship it if it is not flawed. We assign 100 to the value of processing the part, and 560 to the value of painting the part[2]. The action library contains the `SHIP`, `REJECT`, `INSPECT` and `PAINT` actions.

Suppose that the planner starts by constructing a skeletal plan, and (nondeterministically) chooses the `SHIP` action to process the part (Fig. 8). Two alternative contingencies of `SHIP` are constructed (shown in dashed boxes): the first one produces `PR`, and the second does not. A causal link that emanates from the first contingency is established for `PR`.

The triggers for the first contingency are adopted as subgoals (˜`PR`,˜`FL`), and both are supported by `START`. For `START`, two different sets of contingencies are constructed: one with respect to ˜`PR`, and one with respect to ˜`FL`. When the `PAINT` action is inserted to support the `PA` goal, and the ˜`PR` trigger of the first contingency is supported by `START`, the skeletal plan is complete.

The skeletal plan contains three contingencies whose disutilities of failure must be calculated (the contingency for ˜`PR` of `START` has no alternatives). The expected disutilities for the first contingencies of `START`, `PAINT`, and `SHIP` are $0.3 \times 100 = 30$ (for ˜`FL`), $0.05 \times 560 = 28$ (for `PA`), and $0.3 \times 100 = 30$ (for `PR`), respectively. Note that the expected disutilities are quite close even though `PAINT` has a high probability of success.

Suppose that the planner chooses to handle the first contingency with the higher expected disutility, namely the first contingency of `START`. Because ˜`FL` is the proposition supported by the contingency, an action that

[2] C-BURIDAN does not represent the value information.

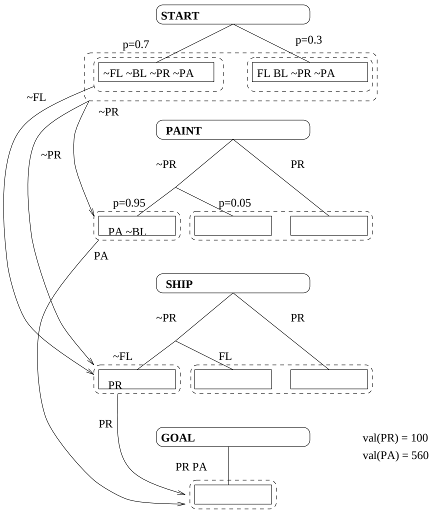


Figure 8: The skeletal plan.

inspects the value of `FL` is inserted before `SHIP` (Fig. 9). The causal link supporting ˜`FL` is removed and an observation link from the ˜`FL` report is inserted. The `INSPECT` step looks at `BL` to report about `FL`, therefore, the first contingency of `START` is linked to the first contingency of `INSPECT`. This causal link ensures that an action that alters the value of `BL` cannot be inserted between `START` and `INSPECT` (`BL` and `FL` are correlated).

The plan for the second contingency is constructed in the CNLP style. For the new goal, a new `REJECT` action is inserted to support `PR`, and the existing `PAINT` step is used to support `PA` yielding the final plan shown in Fig. 9.

Summary and Related Work

We have presented an approach to decision-theoretic plan refinement. We have defined contingencies, and identified two important criteria that can be used to select contingencies, namely probability of failure and the impact of failure. We have described an algorithm that orders contingencies in decreasing expected disutility and extends the plan with a subplan that covers the failure of the contingencies. We have also provided the formal foundations of our approach. We are currently implementing the described system and designing experiments to evaluate it.

In order to make directly reasoning about the value of planning for the failure of various contingencies feasible, we have adopted a different strategy towards conditional planning than that taken in some earlier systems, notably C-BURIDAN. C-BURIDAN constructs branches for alternative contingencies in a somewhat indirect fashion. When it discovers that there are two incompatible actions, say, $A1$ and $A2$, each of which

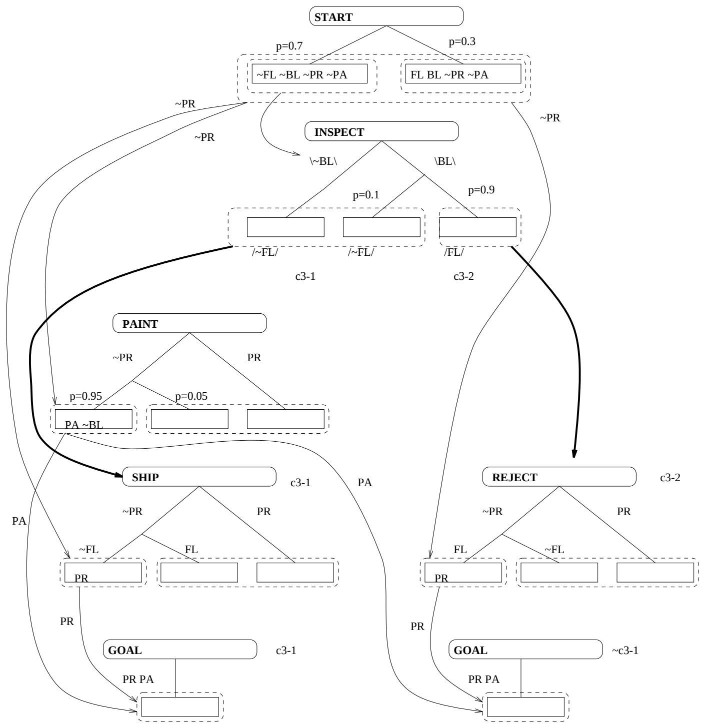


Figure 9: The complete plan.

can achieve some condition C, it introduces an observation action O with two contingent outcomes, and then "splits" the plan into two branches, associating each outcome with one of the actions. In the process of doing this, C-BURIDAN does not reason about whether both those contingent outcomes are worth planning for.

The PLINTH system skips certain contexts during the construction of probabilistic plans (Goldman & Boddy 1994). Our approach differs from theirs in two aspects. First, we are focusing on partial-ordering, where they use a total-order approach. Second, we consider the impact of failure in addition to the probability of failure.

Other relevant work includes the recent decision-theoretic planning literature, e.g., the PYRRHUS system (Williamson & Hanks 1994) which prunes the search space by using domain-specific heuristic knowledge, and the DRIPS system (Haddawy & Suwandi 1994) which uses an abstraction hierarchy. (Kambhampati 1994) describes planning algorithms with multi-contributor causal links. Our algorithm maintains multiple contributors at the action outcome level in a probabilistic setting.

Acknowledgments

This work has been supported by a scholarship from the Scientific and Technical Research Council of Turkey, by the Air Force Office of Scientific Research (Contract F49620-92-J-0422), by the Rome Laboratory (RL) of the Air Force Material Command and the Defense Advanced Research Projects Agency (Contract F30602-93-C-0038), the Office of Naval Research (Contract N00014-95-1-1161), by an NSF Young Investigator's Award (IRI-9258392).

References

Dean, T.; Kaelbling, L. P.; Kirman, J.; and Nicholson, A. 1995. Planning under time constraints in stochastic domains. *Artificial Intelligence* 76:35–74.

Draper, D.; Hanks, S.; and Weld, D. 1994. Probabilistic planning with information gathering and contingent execution. In *Proc. 2nd Int. Conf. on AI Planning Systems*, 31–36.

Etzioni, O.; Hanks, S.; Weld, D.; Draper, D.; Lesh, N.; and Williamson, M. 1992. An approach to planning with incomplete information. In *Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, 115–125.

Goldman, R. P., and Boddy, M. S. 1994. Epsilon-safe planning. In *Proc. 10th Conf. on Uncertainty in AI*, 253–261.

Haddawy, P., and Hanks, S. 1993. Utility models for goal-directed decision-theoretic planners. Technical Report 93-06-04, Department of Computer Science and Engineering, University of Washington.

Haddawy, P., and Suwandi, M. 1994. Decision-theoretic refinement planning using inheritance abstraction. In *Proc. 2nd Int. Conf. on AI Planning Systems*, 266–271.

Kambhampati, S. 1994. Multi-contributor causal structures for planning: a formalization and evaluation. *Artificial Intelligence* 69:235–278.

Kushmerick, N.; Hanks, S.; and Weld, D. S. 1995. An algorithm for probabilistic planning. *Artificial Intelligence* 76:239–286.

Peot, M. A., and Smith, D. E. 1992. Conditional nonlinear planning. In *Proc. 1st Int. Conf. on AI Planning Systems*, 189–197.

Pryor, L., and Collins, G. 1993. Cassandra: Planning for contingencies. Technical Report 41, The Institute for the Learning Sciences, Northwestern University.

Williamson, M., and Hanks, S. 1994. Optimal planning with a goal-directed utility model. In *Proc. 2nd Int. Conf. on AI Planning Systems*, 176–181.