Planning under uncertainty Some key issues

Gregg Collins
The Institute for the Learning Sciences
Northwestern University
1890 Maple Avenue
Evanston, Illinois 60201
U S A
collins@ils nwu edu

Louise Pryor
Department of Artificial Intelligence
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
Scotland
louisep@aisb ed ac uk

Abstract

A planner in the real world must be able to handle uncertainty It must be able to reason about the effect of uncertainty on its plans, select plans that avoid uncertain outcomes when possible, and make contingency plans against different possible outcomes when uncertainty cannot be avoided We have constructed such a planner, Cassandra, which has these properties Using Cassandra, we have produced the Ant general solution to the keys and boxes challenge problem proposed by Michie over twenty years ago

1 Introduction

A planner ID the real world must face the challenge posed by uncertainty To be successful, it must, among other things, be able to reason about the effect of uncertainty on its plans, select plans that avoid uncertain outcomes when possible, and make contingency plans against different possible outcomes when avoidance is impossible Classical planning has largely ignored the issue of uncertainty,' assuming complete and accurate knowledge of the planning context and of the results of available actions Recently, however, several researchers have begun to address the issues involved in extending the classical approach to handle uncertainty [Etziom *et al*, 1992, Peot and Smith, 1992, Pryor and Collins, 1993, Draper *et ol* , 1994, Pryor, 1995] This paper explores the issues in the context of the keys and boxes challenge problem [Michie, 1974] We describe Cassandra,[2] a program that constructs plans that allow for uncertainly, show how it solves the keys and boxes problem, and discuss open issues

1 1 The keys and boxes problem

The keys and boxes problem, presented originally by Michie [1974] and discussed by Tate [1975] and Sacerdoti [1977], addresses many of the issues raised by the presence of uncertainty in the world It involves a robot locked inside a room containing an empty table, two boxes, and a pile of red things sitting by the door A key to the door is in one or the other of the boxes The robot can leave the room only if the key is by the door It can pick up and put down small objects (the key and the red things), but has no sensing capabilities that would allow it to determine what it is about to pick up or what it has in fact picked up The problem is to construct a plan that achieves the goal of having the robot take a red thing out of the room The simplest workable plan is for the robot to move a red thing to the table, move the key to the door, fetch the red thing from the table, and take it out of the room

The problem illustrates two important general issues that arise in planning under uncertainty The first concerns the acquisition of the key because the robot cannot resolve the uncertainty about the key's initial location, it must construct a plan that will work regardless of which possibility turns out to hold [3] The second arises because of the uncertain result of picking a single item out of a heterogeneous pile If the key is put in the pile by the door before the robot tried to pick up a red thing from that pile, then the key might be picked up instead, resulting in an unrecoverable plan failure (because the robot cannot determine what it has picked up) The only way to ensure a sound plan is therefore to avoid the uncertainty by ensuring that the red thing will be picked out of a homogeneous pile of red things Because the planner must (due to other constraints) get the key next to the door before picking up the red thing, the only solution is to move a red thing to another site before the key is moved, creating a new homogenous pile of red things

Although the keys and boxes problem was invented over twenty years ago, classical planning has failed to produce a convincing solution to the problem, largely because work in this area had not until recently addressed the problem of uncertainty (Tale's and Sacerdoti's approaches are discussed below) To our knowledge, no adequate solution has previously been presented

2 Cassandra

Cassandra is a member of the SNLP family of partial order planners [McAllester and Rosenblitt, 1991] It is

'With the notable exception of [Warren, 1976]

2A Trojan prophet fated to be disbelieved when she accurately predicted future disasters

[3] This problem is essentially isomorphic to Moore's "bomb in the toilet" problem [McDermott, 1987]

based on UCPOP [Penberthy and Weld, 1992], which extends the basic algorithm to handle context-dependent effects and a limited form of universal quantification Actions are represented by modified STRIPS operators [Fikes and Nilsson, 1971], each defined by the preconditions for executing an action and the effects that occur as a result of executing it Each possible effect has an associated set of *secondary preconditions* [Pednault, 1988], specifying the conditions under which the action will have that effect

The use of secondary preconditions is crucial to Cassandra's ability to represent uncertain effects, which it does by assigning them *unknowable* secondary preconditions, constructed using the pseudo-predicate unknown Unknowable preconditions designate both a source of uncertainty (e *g* , an instance of a coin toss) and a possible outcome of that uncertainty ("heads") We assume that each outcome of a given source of uncertainty has a unique came, and that the set of named outcomes is exhaustive and mutually exclusive These assumptions license two critical judgments that Cassandra must be able to make (1) that two actions or effects may not co-occur because they depend upon different outcomes of the same uncertainty, and (2) that a goal will necessarily be achieved by a plan, because it will be achieved for every possible outcome of every relevant uncertainty

Planning under uncertainty requires the planner to build plan segments that may or may not be executed, depending on the contingencies that are encountered Keeping track of which elements of the plan are relevant under various contingencies requires effective bookkeeping Cassandra therefore propagates labels over actions effects, and goals in the plan to indicate both the contingencies in which these elements play a role and the contingencies with which they are compatible

Until Cassandra encounters an uncertainty, it proceeds in the same manner as other planners in the SNLP family, using a means-ends analysis involving the alternation of two processes planning for open subgoals and avoiding bad interactions An uncertainty is introduced into a plan when a subgoal is achieved by an effect with an unknowable precondition, and is noticed by Cassandra when its current plan becomes dependent upon a particular outcome of that uncertainty In effect the plan that has been built so far becomes a contingency plan for that outcome To build a plan that is guaranteed to succeed, Cassandra must also construct contingency plans for all other possible outcomes of the uncertainty it splits the plan into a set of branches, one for each possible outcome Cassandra labels the components in the existing plan to show that they depend on a particular outcome of the uncertainty, and introduces a new set, of goals for each other possible outcome The plan is thus not complete until all the goals are achieved in every possible outcome This procedure is repeated whenever an uncertainty is encountered

Once an uncertainty has been encountered and plan branches have been introduced, every new component (action or causal link) that is introduced into the plan is relevant only in some branches Each component is therefore labeled with the contingencies in which it facilitates goal achievement (these labels propagate backwards from the goals) and the contingencies in which it hinders goal achievement (these labels propagate forwards towards the goals) Labels of the latter type are introduced when an *unsafe* link is detected a causal link that is threatened by another effect (the *clobberer)* in the plan There are two standard methods for resolving an unsafe link in planners in the SNLP family *separate* the clobberer and the unsafe link by adding codesignation constraints ensuring that they don't unify, or reorder the actions in the plan to ensure that the clobberer occurs outside the range of the link The first of these methods is extended in Cassandra another method of separating the clobberer from the unsafe link is to ensure that they only occur in different contingencies, resulting in plan components being labeled with contingencies in which they may not occur

2 1 Cassandra's decisions

When an agent executes a branching plan, it must at some point decide which branch to take Previous work has simply assumed that the agent will execute those steps that are consistent with the contingency that actually obtains [Warren, 1976, Peot and Smith, 1992] However, in order to know which contingency this is, an arbitrarily large amount of work may be necessary in order to gather the information on which the decision is based [Pryor. 1994, 1995] For a plan to be viable, the planner must be able to ensure both that it will be in a position to make the decision by acquiring the necessary information, and that the information-gathering steps do not conflict with the rest of the plan

Our approach to this problem is to treat decisions as explicit plan steps, along with the actions to acquire and evaluate information that support the decision making procedure Cassandra therefore adds an explicit decision step each time it encounters a new source of uncertainty It is added to the plan along with ordering constraints ensuring that it occurs after the step with which the uncertainty is associated and before any step with a subgoal the achievement of which depends upon a particular outcome of the uncertainty

In Cassandra, the action of deciding which contingency to execute is modeled as the evaluation of a set of condition-action rules Each decision step in a plan is annotated with the set of rules that will be used to make it To evaluate a decision rule, the executing agent must be able to determine whether the rule's antecedent holds Thus, the preconditions for the decision step must include goals to know the current status of each condition that appears as an antecedent of a rule in this condition The preconditions of a decision step become open conditions in the plan in the same way as do the preconditions of any other step

Since the intended effect of evaluating the decision rules is to choose the appropriate contingency given the outcome of a particular uncertainty, the conditions are intended to be diagnostic of particular outcomes of the uncertainty The agent cannot, of course, directly determine the outcome of an uncertainty, so it must infer it from the presence or absence of effects that depend upon

```
Initial when [KEYOS  80I2]
                (AND (AT KEY BDI2) (AT THING BOH)
                     (IDT (AT KEY BOH))
                     (OT (AT THING BDI2)))
          When [KEYOS  BOK1]
                (AID (AT KEY BOX1) (AT THING B0I2)
                     (OT (AT KEY B0I2))
                     (OT (AT THING B0I1)))
                  other initial condition*
St«p  1 (3)  (DECIDE KEYOS)
                  Parallel branches for KEYOS
Step  2 (B)  (PICK-UP BDX2)
Step  3 (4) (PUT-DOWE DOOR)
3ttp  4 (2)  (PICK-UP BOX1)
Step  B (1) (PUT-DOWS DOOR)
Goal          (AT KEY DOOR)
              3 -> (AT KEY DOOR)   10  [KEYOS  B0X1]
              5 -> (AT KEY DOOR)   10  [KEYOS BOX 2]
```

Figure 1 Cassandra's plan to geT the key to the door

that outcome

The most straightforward approach to constructing decision rules would be to make the antecedent for a given contingency be the conjunction of all the effects that could be expected to result from the relevant outcome of the uncertainty However, this turns out to be overkill In fact, it need only be verified that the contingency plan can succeed The decision rule antecedent for a contingency u thus the conjunction of all the direct effects of the relevant outcome that are used to establish subgoals in that branch of the plan

3 Cassandra's solution

Cassandra solves the keys and boxes problem by explicitly representing and reasoning about the underlying uncertainty of the domain It also performs sensibly on reasonable van at ions of the original problem For instance, a planner that would produce the same plan for the problem despite being given the ability to see and distinguish objectfl clearly could not be considered to be competent in this domain Cassandra produces the correct plan for this case as well We consider Cassandra's approach to the two major parts of the problem separately

3 1 Retrieving the key

The retrieval of the key presents two interesting issues the planner must be able to recognize how the uncertainty about the key's location affects its plan to acquire the key, and the planner must be able to recognize that, in the absence of a method for determining the key's location in advance, it must try picking the key up from all the possible locations In general, uncertainty affects a plan when the achievement of a subgoal depends on an outcome of the uncertainty For example, the uncertainty about the key's location affects the keys and boxes plan just because the operator for acquiring the key (pickup) depends on the location of the object to be picked up Were this not the case, the uncertainty would be irrelevant For example, if the planner were given a goal to trigger a radio-controlled bomb that could be in either of the two boxes, the uncertainty about

```
St«p  1 (4)  (SCAN  BOX1)
Slip  2 (7)  (SCAN  B0I2)
Step  3 (3)  (DECIDE KEYOS)
              (and (HOT (AT THING BOX1)
                   (AT KEY BOX1)
                   T) -> [KEYOS  BOX1]
              (and (I0T (AT THUG B0I2))
                   (AT KEY BOX2)
                   T) -> [KEYOS  B0X2]
Step  4 (6)  (PICK-UP B0X2)  YES  [KEYOS  B0I2]
                             ■0   [KEYOS  BOX1]
   steps labeled with the appropriate contingencies
```

Figure 2 Part of another plan to get the key to the door

that location would not affect the resulting plan Cassandra recognizes when any given uncertainty affects its plans, irrelevant uncertainties are simply ignored during the planning process

A plan is sound only if it achieves its goal for every possible outcome of each relevant uncertainty There are two strategies for accomplishing this the planner can make contingency plans for each outcome, and decide during execution which should be executed, or it can make contingency plans for each outcome and execute them all "in parallel", having determined that they do not interfere with each other Cassandra pursues both these strategies, thus, when it cannot find a plan for making an explicit decision, it can find one involving parallel execution

Cassandra's flexibility is a result of its explicit representation of decisions as plan steps As well as the type ofdecision described above, in which rules are used to determine which plan branch should be executed, Cassandra can also use a type of decision in which all branches are executed When uncertainty is encountered during the planning process, either type may be used The decision type affects the remainder of the planning process If the decision uses rules, the necessary knowledge preconditions must be added and threats can be resolved by ensuring that the ciobberer and the threatened link occur only in different contingencies If the decision is to execute all plan branches, no knowledge preconditions are added, and all steps and effects must be compatible with all others, regardless of which branch they are in

Figure 1 shows the plan Cassandra constructs when given the goal of getting the key to the door The initial conditions state that either the key is in box1 and something else is in box2, or *vice versa* The plan's decision step says to execute both branches in parallel The goal is achieved separately in each contingency—for instance, step 3 will not achieve the goal if the key was onginally in boxI, but will do so if it was in box2

If we give Cassandra a variant of the problem in which it has an operation, scan, that can be used to spot the location of the key in advance, it constructs a plan in which the robot first scans the area, then picks up the key from the correct location Figure 2 shows part of this plan The decision has two rules, one for each outcome of the uncertainty The antecedents of the rules become knowledge preconditions for the decision step, and are achieved by the scan actions In the original

```
effects (( when ( oneof (??grab ?x) (moveable ?x)
                        (at ?x ?loc))
          effect ( and (holding ?x)
                       ( not (empty-hand))
                       ( not (at ?x ?loc))))))
```

Figure 3 The oneof construct

problem, the lack of such a sensing operation makes it impossible to determine which contingency holds, which is why in that case Cassandra produces a plan calling for ail branches to be pursued at once

3 2 Acquiring the red thing

A plan that will fail for some outcome of an uncertainty is unsound One approach to constructing a correct plan in the face of such an uncertainty is to ensure that the problematic outcome does not arise For example, consider the classic example of drawing a ball from an urn from the planner's point of view, the outcome of the action is uncertain, since any of the available balls might be drawn If a plan cannot be elaborated to deal with the case in which, say, a black ball is drawn, then the planner's only alternative is to ensure that no black ball is present in the first place

For a non-sensing robot, the action of picking an object from a pile is identical to the problem of drawing a ball from an urn Since the robot cannot distinguish one object from another, it is unable to construct sound plans for cases in which it is attempting to pick up a needed object from a heterogeneous pile (unless, of course, any of the objects in the pile will suffice for its purposes) The planner's only alternative in these *cases* is thus to enforce the condition that no object of the wrong type is present when it does a pickup Cassandra uses a special construct, the one of construct, to represent the type of uncertainty found in this sort of situation One of the effects of the pickup action is shown in figure 3, representing the fact that the action will result in the robot holding one of the moveable objects that are present at the location at which the action is executed

When Cassandra encounters a one of condition in a subgoal, it enumerates the objects that meet the type specification, in this case, all noveable objects These designate all possible outcomes of the uncertainty attached to this construct (?? grab in the example) Since individual branches may be made impossible if the conditions for their occurrence are not met, Cassandra maintains completeness by trying each possible combination of outcomes, adding sub goals to ensure that only the appropriate branches will occur In the example, these subgoals concern the presence or absence of the relevant objects at the specified location Of course, the combination with only one possible outcome effectively removes the uncertainty, and Cassandra recognizes that there is no need to add a decision into the plan in this case

When Cassandra schedules an action to acquire a red thing by picking it up, it will try to find contingency plans for outcomes in which it picks up some object other than that a red thing, but will fail [4] The only alternative

[4] Cassandra can also construct an appropriate plan when there is a sensing action available

```
Step  1 (4)   (PICK-UP DOOR)
                   0 -> (AT RED-THING DOOR)
                   0 -> (NOT (AT THING DOOR))
                   0 -> (NOT (AT KEY DOOR))
Step  2 (3)   (PUT-DOWN TABLE)
Step  3 (6)   (PICK-UP BOX1)
                   0 -> (NOT (AT RED-THING BOX1))
                   0 -> (NOT (AT THING BOX1))
                   0 -> (AT KEY BOX1)
Step  4 (5)   (PUT-DOWN DOOR)
Step  5 (2)   (PICK-UP TABLE)
                   2 -> (AT RED-THING TABLE)
                   0 -> (NOT (AT THING TABLE))
                   0 -> (NOT (AT KEY TABLE))
Step  6 (1)   (TAKE-OUT)
                   4 -> (AT KEY DOOR)
Goal          (AT RED-THING OUTSIDE)
                   6 -> (AT RED-THING OUTSIDE)
```

Figure 4 Ensuring that no unwanted objects are present

available is to make sure that no other objects are in the pile Thus, for every instance in which the robot must pick something up, Cassandra will explicitly ensure that no unwanted objects are present Figure 4 shows the relevant steps from the plan for the situation in which the key is in boil

The construction of the correct plan for the keys and boxes problem follows directly from this Since it is impossible to construct a sound plan for the case in which the red thing is picked up from beside the door,[6] the red thing must be picked up from another location, i e , from one of the boxes or from the table If a box is chosen, the constraint that the key not be at the pickup location will again conflict, in a different way, with the schedul ing of the movement of the key, and the obvious plan of moving a red thing to the box, then moving the key to the door cannot be completed [6] In the plan involving the table, the constraint that the key is not on the table is unproblematic However, Cassandra must ensure that there is a red thing on the table, meaning that it must put one there To do this, it must possess a red thing, which it can get by picking it up from the pile by the door Since this operation can only succeed if the key is not by the door, it must be scheduled before the key is moved Thus, we get the plan red thing to table, key to door, red thing out

4 Previous solutions

There have been two previous attempts at solving the keys and boxes problem by Tate [1975] and Sacerdoti [1977] We discuss them in turn

4 1 Tate's solution

Tate's planner, Interplan [1975], represents an early attempt to solve the keys and boxes problem Tate explicitly recognized three important issues (1) actions have

5 If the red thing is picked up first, there will be no way to get out of the door, if the key is moved first, it will be in the pile before the pickup

6 In fact, the less obvious plan of key to table, red thing to box, key to door, red thing to outnde works—and Cassandra can find it, but arrives at the simpler plan first

uncertain outcomes, (2) the robot cannoL tell which object IS a key, and (3) the planner cannot keep track of what objects are at each location

Although Tate recognized that uncertainty plays a key role in the problem, Interplan could not handle the general case of actions with uncertain effects, but only the special case required by the problem picking up one of a set of objects at a location It did this by using *sets* in the representation of the preconditions and effects of its actions For example, a precondition of its pickup action is that there is a subset of set X. of objects at the robot's location, the effect is that the robot is holding a subset of set I When the robot moves, the things that it is holding move with it Interplan's solution of the problem thus depended on a set-matching facility that can match patterns such as (subject X), (union X Y) and (setninus X Y) It also needed the ability to represent such knowledge as "If a set of objects has some property *P,* then a subset of that set has that property " These facilities were not implemented Tate presents a simulation of Interplan's solution to the problem Moreover, the implemented version of Interplan could not fully represent the initial conditions of the problem that the key is either in boil or in box2, and that all the objects at the door are red

Interplan solved the problem of getting the key to the door by reasoning that a subset of the things at boil *and* a subset of the things at box2 must be at the door This reasoning is accomplished by way of a special eitherof construct the set matching facility should transform (at (subset (either of A B)) door) into two conditions, both of which must be achieved Cat (subset A) door) and (at (subset B) door) Interplan could thus only handle uncertainty by executing plans for all contingencies at once, it could not build a plan with separate branches, only one of which should be executed

Next, Interplan detected that it would run into trouble if it tned to move the key to the door while holding the red thing It had several strategies that it could use in this situation it first tried to reorder the plan actions, but found this ineffective Its next strategy was to introduce another action into the plan that will achieve the goal of holding the red thing, in this case picking it up from the table It is not clear from Tate's description how the table is chosen as the intermediate staging point rather than, say, either of the two boxes

Interplan thus failed to provide a fully general solution to the keys and boxes problem It could handle only special cases of actions with uncertain effects, could not handle simple variants of the problem, and did not explicitly reason about removing sources of uncertainty

4 2 Sacerdoti's solution

Sacerdoti [1977] describes in detail how his planner NOAH, can be made to handle the keys and boxes problem Because NOAH did not reason explicitly about uncertainty, the approach necessarily involves neither recognition of the uncertainties that characterize the problem nor explicit reasoning about methods for coping with these uncertainties As such, the solution is suspect from the start, a close examination of the details confirms this judgment

NOAH'S solution to the key acquisition problem is somewhat difficult to interpret The initial state is represented as having "piles of keys" in both boxes (apparently a pile is to be interpreted as something that may have zero size) When NOAH expands the subgoal of getting a key to the location of the door, it simply constructs an iterative plan to move items from both boxes to the door This can be interpreted as "compiling in" a rather specific rule about how to acquire a single object from a set of possibly empty piles However, there is no obvious way in which NOAH would behave differently if it were known that the piles were all of non-zero size, in this case it would apparently still pick up an object from each pile Thus, the approach at best trivializes the problem, and at worst leads to incorrect solutions of related problems Furthermore, there does not appear to be any representation of the fact that the key must be on one pile or the other, it thus does not appear that NOAH has any way of knowing if its plan will succeed or an adequate rationale for planning to move the two piles

The approach to the acquisition of the red thing is more complex The process proceeds as follows

- Since the robot must possess a red thing in order to achieve the overall goal of the plan, NOAH schedules an action to pick one up,
- NOAH realizes that the necessary action of moving the key will interfere with the goal to keep holding the red tlung that has been picked up
- NOAH repairs this bug by scheduling an action to put the red thing down while it moves the key,
- NOAH elects to have the robot put the red thing on the table, this choice being governed by a heuristic that says that objects should always be put down in the least crowded available place,
- NOAH schedules an action to reacquire the red thing by picking it up from the table, followed by the action of leaving the room

There are several problematic aspects of this solution that deserve mention First, the solution depends serendipitously on the action of the "put things down in the least crowded place" heuristic The justification for this heuristic is unclear, since other alternatives such as putting the object in the nearest spot, or putting it back from whence it came seem equally sensible IfNOAH were given a variant of the problem in which the pile of red things was initially on the table, it would apparently pick one up, and then put it down in a less-crowded box, thus producing an unsound plan Worse yet, NOAH has method other than this heuristic of rejecting a plan in which the red thing were put back in the pile by the door

Second, NOAH'S response to the perceived bug in its initial plan—not being able to hold on to the red thing while moving the key—uses an apparently sub-optimal repair strategy The problem is an instance of the threatened violation of a protection interval One of NOAH's

[7] But the behaviour *of the set* matcher was fully specified

standard strategies in such cases is to constrain the interfering action to occur before the establishment of the goal, this was termed promotion by Chapman [1987] Notice that promotion is in general preferable to the interruption strategy, since it involves achieving the goal only once, rather than achieving the goat, undoing it, and re-achieving it * It is thus difficult to see why NOAH would elect to use the interruption strategy, in fact, it is difficult to see why it would have such a strategy

One possible answer to this question is that NOAH might use interruption when it recognizes a situation in which, as in the keys and boxes problem, the use of promotion will not lead to a correct plan Once again, however, the knowledge necessary to make such a judgment is entirely lacking NOAH IS unable to recognize the problem caused by heterogeneous piles of objects, as far as it is concerned, were the bug repaired using promotion, the resulting plan would be entirely acceptable

Sacerdoti's approach to the keys and boxes problem thus fails to come to grips with either of the two interesting aspects of the problem A sound plan is constructed only through a series of contingent accidents, and NOAH has no knowledge that would allow it to reject unsound alternative plans In short, it cannot be considered a legitimate solution to the problem

5 Conclusion

Reasoning under uncertainty is an important issue for any planner intended to operate in real-world domains We have constructed a planner, Cassandra, that is capable of constructing viable plans in the face of uncertainty Cassandra has a number of important properties that are not found in other approaches to planning under uncertainty Using Cassandra, we have produced the first known solution to the keys and boxes challenge problem, first proposed by Michie over twenty years ago

5 1 How Cassandra solves the problem

Cassandra's ability to solve the keys and boxes problem, and its variants, stems from its general ability to handle plans affected by uncertainty There are several aspects of Cassandra's approach that lead to its success

- Cassandra's representation of uncertainty allows diverse phenomena to be treated within a. single framework Any set of initial conditions or operator effects can be viewed as representing alternative potential outcomes of a source of uncertainty, each effect can also have conventional preconditions,
- For a given goal, Cassandra can find plans that are unaffected by a given contingency as well as plans that are affected by that contingency This is because it determines the relevance (or lack thereof) of a source of uncertainty to its current plan by waiting until an outcome of that uncertainty is used to establish an effect in the plan,
- Whenever Cassandra encounters an uncertaint that is relevant to its plans, it constructs contingency plans for each relevant outcome of the uncer tainty,
- Any number of uncertainties can be handled within a single plan
- Because of the explicit representation of decisions, Cassandra can construct plans involving making decisions between different contingency plans as well as those involving executing all contingency plans in parallel,
- Cassandra can recognize that plans involving picking an object from a heterogeneous pile in the keys and boxes world are unsound because it recognizes when it is impossible to plan for a relevant contin gency
- Using the one of construct, Cassandra can represent the situation in which different possible out comes of a given uncertainty are contingent on dif ferent antecedent conditions For example, it can represent the uncertainty of drawing a ball from an urn, where the alternative outcomes—particular balls that might be drawn—depend upon different jnitial conditions—whether a given ball is in the urn before the draw,
- Cassandra considers plans that avoid potential out comes of an uncertainty Since *it* can plan to rule out outcomes as well as planning to achieve its goal in the face of outcomes, it can exhaustively consider viable plans for dealing with a particular source of uncertainty

"The interruption strategy might be justified as a response once execution of the plan has begun, ■ e , if the executing system has achieved a goal and subsequently discovers that it must execute an action that will undo it When planning in advance, however, it is clearly preferable to avoid the conflict from the start

Many of these capabilities are unique to Cassandra Other recent planners that address issues connected with uncertainty are SENSp [Etzioni *et al* , 1992], CNLP [Peot and Smith, 1992] and C-BURIDAN [Draper *et al,* 1994] None of these has a fully general representation for uncertainty [Pryor and Collins, 1993] None of them can handle uncertainty when it is impossible to sense the actual outcome that has occurred, as is the case in the keys and boxes domain, and none can allow for the execution of contingency branches in parallel To our knowledge, Cassandra is the only planner to consider ruling out potential outcomes by ensuring they cannot occur, and no other planner can represent the type of uncertainty for which Cassandra uses the on«of construct

5 2 Open issues

The application of Cassandra to the keys and boxes problem highlights several general open issues in planmn under uncertainty Space permits only a brief glance at some of these

First, it is potentially extremely inefficient for Cassandra to consider every object in a given pile as representing a different alternative outcome of the uncertainty involved in picking something up from that pile, since in many instances different objects ID the pile ma

function identically as far as the planner is concerned For example, in the keys and boxes problem it does not matter which red thing is picked up In other words, red things form an equivalence class for this problem, and the planner need only reason about the outcome of picking up a generic red thing The ability to spot equivalence classes among uncertain outcomes is one that would prove highly useful

Second, the one of construct, like the forall construct suggested by Pednault [1989] and implemented by Penberthy and Weld [1992] ranges over every object of a given type defined in the planning universe Explicit consideration of all of these objects may represent a tremendous amount of overhead in a crowded universe The planner should have some filter that allows objects that are technically in the scope of a oneof or forall to be considered explicitly only when there is some reason to do so

Third, Cassandra exhibits the problem of exhaustive search faced by all classical planners The introduction of contingency plans and, especially, the proliferation of choices resulting from the handling of the oneof construct, increase the size of the search space enormously Without effective domain-independent search heuristics, plans for even simple problems take an impractical length of time to construct One way of reducing this problem would be to merge plan branches tliit consist of the same actions unfortunately, this turns out to be extremely com plex (see [Pryor and Collins, 1993])

Finally, Cassandra cannot in principle achieve an explicit, general understanding of the problem that heterogeneous piles pose for the non-sensing robot—every time it considers doing a pickup from such a pile during plan construction, it in effect rediscovere the same problem A better approach would be to incorporate a mechanism whereby the planner could formulate a general description of the problem once, and recognize it quickly whenever it recurred (this is in the spirit of Sussman [1975], see [Collins *et al*, 1991] for ideas about how to formulate this sort of genera] problem characterization)

Future work will tackle these and other central issues in the design of planners that can cope with uncertainly

References

[Chapman, 1987] D Chapman Planning for conjunctive goals *Artificial Intelligence,* 32 333-337, 1987

[Collins *et al,* 1991] G Collins, L Birnbaum, B Krulwich, and M Freed Plan debugging in an intentional system In *Proceedings of the 12th International Joint Conference on Artificial Intelligence,* pages 351-358, Sydney, Australia, 1991 IJCA1

iDraper et *al* , 1994] D Draper, S Hanks, and D Weld Probabilistic planning with information gathering and contingent execution In *Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems,* pages 31-36, Chicago, IL, 1994 AAAI Press

[Etziom *el al* , 1992] E Etzioni, S Hanks, D Weld, D Draper, N Lesh, and M Williamson An approach to planning with incomplete information In *Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning,* Boston, MA, 1992 Morgan Kaufmann

[Fikes and Nilsson, 1971] R E Fikes and N J Nilsson STRIPS A new approach to the application of theorem proving to problem solving *Artificial Intelligence,* 2189-208, 1971

[McAllester and Rosenbhtt, 199l] D McAllester and D Rosenbhtt Systematic nonlinear planning In *Proceedings of the 9th National Conference on Artificial Intelligence* pages 634-639, Anaheim, CA, 1991 AAAI Press

[McDermolt, 1987] D V McDermott A critique of pure reason *Computational Intelligence,* 3 151-160, 1987

[Michie, 1974] D Michie *On Machine Intelligence* Edinburgh University Press, Edinburgh, 1974

[Pednault, 1988] E P D Pednault Synthesizing plans that contain actions with context-dependent effects *Computational Intelligence,* 4 356-372, 1988

[Pednault, 1989] E P D Pednault ADL Exploring the middle ground between STRIPS and the situation calculus In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning* Morgan Kaufmann, 1989

[Penberthy and Weld, 1992] J S Penberthy and D S Weld UCPOP A sound, complete, partial order planner for ADL In *Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning,* Boston, MA, 1992 Morgan Kaufmann

[Peot and Smith, 1992] M A Peot and D E Smith Conditional nonlinear planning In Proceedings *of the 1st International Conference on Artificial Intelligence Planning Systems,* College Park, Maryland, 1992 Morgan Kaufmann

[Pryor and Collins, 1993] L Pryor and G Collins Cassandra Planning with contingencies Technical Report 41, Institute for the Learning Sciences, Northwestern University, 1993

[Pryor, 1994] L Pryor Opportunities and planning in an unpredictable world Technical Report 53, Institute for the Learning Sciences, Northwestern University, 1994

[Pryor, 1995] L Pryor Decisions, decisions Knowledge goals in planning In J Hallam, editor, *Hybrid Problems Hybrid Solutions (Proceedings of AISB-95),* pages 181-192 10S Press, 1995

[Sacerdoti, 1977] E Sacerdoti *A structure for plans and behavior* American Elsevier, New York, 1977

[Sussman, 1975] G J Sussman *A computer model of skill acquisition* American Elsevier, New York, 1975

[Tate, 1975] A Tate *Using goal structure to direct search in a problem solver* PhD thesis, University of Edinburgh, 1975

[Warren, 1976] D Warren Generating conditional plans and programs In *Proceedings of the AISB Summer Conference,* University of Edinburgh, 1976