Appears in Working Notes of the NASA Workshop on Planning and Scheduling for Space
Oxnard, CA, October 1997

CIRCA and the Cassini Saturn Orbit Insertion: Solving a Prepositioning Problem

David J. Musliner and Robert P. Goldman
Automated Reasoning Group
Honeywell Technology Center
3660 Technology Drive
Minneapolis, MN 55418
{musliner,goldman}@htc.honeywell.com

Introduction

The Cooperative Intelligent Real-time Control Architecture (CIRCA) was designed for mission-critical applications requiring autonomous planning and control. As space technology becomes mature and budgets shrink, NASA is looking towards precisely this type of autonomy to support future missions including the New Millennium Deep Space One probe and the Mars Rover projects. We are actively investigating the application of CIRCA to a variety of domains including spacecraft planning and control and autonomous aircraft control. In this paper, we discuss one particularly challenging type of planning problem that arises in mission-critical applications, drawing on an example from the Cassini Saturn mission.

Prepositioning problems arise when certain actions must be taken to preposition assets or otherwise prepare for contingencies, before those contingencies could possibly occur. In the Cassini example, a backup inertial reference unit (IRU) must be preheated long before an engine burn is planned, so that if the primary IRU fails during the burn, the backup will be immediately available. The IRU preheating operation is a "prepositioning action." To build plans that include this type of prepositioning, a planner must "look ahead," recognize the contingency, and identify appropriate prepositioning actions. CIRCA's new Dynamic Abstraction Planner (DAP) efficiently builds plans that include prepositioning.

This paper is not meant to be an introduction to CIRCA; instead, our goal is to describe how CIRCA can address the Cassini prepositioning problem, and discuss the various strengths and weaknesses of the approach. Accordingly, we refer readers to other publications (Musliner, Durfee, & Shin 1993; 1995; Goldman *et al.* 1997) for CIRCA overviews and details on the planning algorithms. This paper begins by describing key elements of prepositioning problems in general, and then gives the details of the simplified Cassini problem. We then present the CIRCA solution to this problem. We review how other systems attempt to solve this type of problem, comparing their features with CIRCA, and conclude with a discussion of the more general directions for future work on CIRCA.

Prepositioning Problems

Prepositioning problems arise when actions must be taken to prepare for contingencies, *before* those contingencies could possibly occur. For example, military prepositioning involves the movement of assets to various forward deployment areas when no conflicts are in progress, in anticipation of future conflicts in the area. In the Cassini example, a backup inertial reference unit (IRU) must be preheated long before an engine burn is planned, so that if the primary IRU fails during the burn, the backup will be immediately available. These examples share several common features that we use to characterize prepositioning problems:

Exogenous Events — Without events beyond the system's control there would be no need for prepositioning. If the domain is fully controllable, prepositioning becomes simply establishing preconditions for planned actions. Prepositioning is different because it involves anticipating contingencies outside of the system's control.

Potential Failure — It must be possible for the system to fail, or reach an undesirable state/outcome, if the prepositioning action is not performed correctly. The plant must blow up, the spacecraft miss its orbit insertion, etc., else there is little motivation to preposition.

Temporal Information — By definition, prepositioning actions must be performed in a timely fashion, so that they are completed before the contingency can occur.

These problem characteristics place stringent requirements on planners that must address prepositioning problems: a suitable planner must be able to represent and reason about exogenous events, nondeterminism, failure, and time. Because prepositioning actions must be taken before the disturbance occurs, reactive approaches are not sufficient; projection is needed. On the other hand, because *external* contingencies are involved, "classical" AI projective planners are not sufficient; they assume the planning agent is the only active part of the domain.

CIRCA strikes a particular balance along the spectrum of representational power vs. computational complexity. This balance point was explicitly chosen to represent the information necessary for mission-critical planning problems without unnecessary complexity.

CIRCA and Prepositioning Problems

In particular, CIRCA provides the following features that can be applied to prepositioning problems:

Event and Temporal Transitions — CIRCA can model two types of exogenous changes, event transitions and temporal transitions. Both are modeled as completely outside of the agent's control. Events are instantaneous changes, while temporal transitions represent processes with temporal extent.

Explicit Failure — CIRCA models a distinguished failure state which it must plan to avoid at all costs. CIRCA builds plans that guarantee that failure cannot be reached. This is the key feature that makes CIRCA suited to mission-critical applications.

Simplified Temporal Model — The temporal model in CIRCA is reduced to the minimum amount of information necessary to build plans that ensure safety. Transitions are only assigned worst-case timing values. Actions that are under the planner's control are assigned maximum delays; the system can rely on these actions taking effect by the maximum delay. On the other hand, events and temporals have minimum delays; the system can rely on these transitions *not* occurring before the delay elapses. Together, these upper and lower bounds impose the minimum and maximum limits on the dwell time in a state. For example, a planned action can be assigned a maximum delay value D to ensure that the system can dwell in a state for no more than D time units before the action must occur. The CIRCA execution engine (the Real-Time Subsystem) will enforce this execution timing requirement.

State-space Projection — CIRCA's state-space planner builds plans by interleaving a forward simulation of the environment with the selection of actions for each simulated state. Lookahead search heuristics allow the planner to make intelligent action choices, and smart backjumping improves the search for a useful plan when lookahead is not sufficiently prescient.

The Cassini Prepositioning Problem

The Cassini prepositioning problem was originally outlined by Erann Gat in his paper "News From the Trenches: An Overview of Unmanned Spacecraft for AI" (Gat 1996). The problem concerns the Saturn orbit insertion maneuver planned for the Cassini spacecraft. The spacecraft has a narrow time window during which it must fire its thrusters to decrease its speed and enter the desired Saturn orbit. To successfully navigate during a thruster burn, the Cassini spacecraft must have an inertial reference unit (IRU) warmed up and functioning. The spacecraft has a primary and a secondary IRU. The problem is to foresee the possibility of a primary IRU failure during the orbit insertion burn and warm up both IRUs early enough that they will be available for navigation during the burn. If the primary fails but the backup has been warmed up, the backup can be switched in and the burn can continue uninterrupted.

Gat originally claimed that "there is currently no AI planner that can figure out, given this information, that it is a good idea to turn on the backup IRU before orbit insertion begins so that the burn doesn't have to be terminated if the primary IRU fails." We claim, on the contrary, that CIRCA can easily build this plan, given quite sparse information about the problem. In fact, it is precisely the sort of planning that CIRCA is good at. In the following section, we give full details on one version of the domain encoding that CIRCA can solve, discussing several necessary quirks of the representation as well as its strengths.

CIRCA Solving the Cassini Problem

Figure 1 shows the domain encoding we used to solve the Cassini prepositioning problem. Of course, this is a highly simplified representation of the real problem, focused only on illustrating how CIRCA decides to preheat the backup IRU (IRU2). In that sense, this domain encoding is like a blocks-world problem; we are not claiming it captures the real complexity of the Cassini domain.

The representation is fairly straightforward, specifying the initial state and the possible events, temporal transitions, and control actions. For CIRCA, the goal of avoiding failure is always implicit. The `*goals*` expression describes "task-level" goals that the planner should try to achieve, but which are not mission-critical (there are none in this domain). We have made the `(engine on)` goal a safety-critical "control-level" goal using the `fail_if_dont_burn` transition. This transition specifies that, if the engine stays off for

```
(setf *goals* nil)
(make-instance 'action :name "warm_IRU1"
      :preconds '((IRU1 off))
      :postconds '((IRU1 on))
      :delay 60)
(make-instance 'action :name "warm_IRU2"
      :preconds '((IRU2 off))
      :postconds '((IRU2 on))
      :delay 60)
(make-instance 'action :name "engine_on"
      :preconds '((engine off))
      :postconds '((engine on))
      :delay 1)
(make-instance 'action :name "select_IRU1"
      :preconds '((IRU1 on))
      :postconds '((active_IRU IRU1))
      :delay 1)
(make-instance 'action :name "select_IRU2"
      :preconds '((IRU2 on) (IRU1 broken))
      :postconds '((active_IRU IRU2))
      :delay 1)
(make-instance 'temporal :name "fail_if_burn_with_broken_IRU1"
      :preconds '((engine on)(active_IRU IRU1)(IRU1 broken))
      :postconds '((failure T))
      :delay 5)
(make-instance 'temporal :name "fail_if_burn_without_guidance"
      :preconds '((engine on)(active_IRU none))
      :postconds '((failure T))
      :delay 1)
(make-instance 'temporal :name "fail_if_dont_burn"
      :preconds '((engine off))
      :postconds '((failure T))
      :delay 1000)
(make-instance 'event :name "IRU1_fails"
      :preconds '((IRU1 on))
      :postconds '((IRU1 broken)))
(setf *initial-states* (list
      (make-instance 'state
              :features '((failure nil) (engine off) (IRU1 off) (IRU2 off)
                          (active_IRU none)))))
```

Figure 1: Simple Cassini prepositioning domain description.

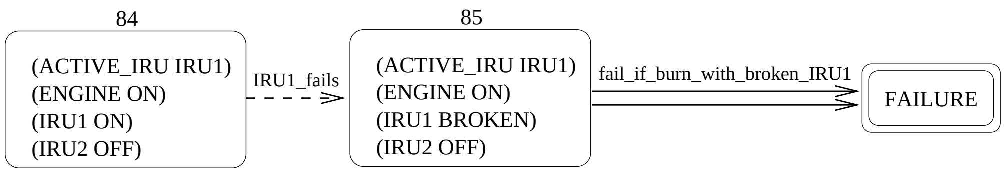

Figure 2: Partial state space during planning, where failure is reachable.

1000 time units, the system may fail catastrophically. Since CIRCA builds plans that must guarantee to avoid `(failure T)`, the final plan will definitely turn the engine on.

To illustrate how CIRCA's planner solves this problem, we'll trace a small portion of the search activity. Consider the partial plan shown in Figure 2. In this sequence, the planner considers the `IRU1_fails` event transition leading out of state 84 to state 85. From this state, a temporal transition (`fail_if_burn_with_broken_IRU1`) leads quickly to failure, which the planner must prevent. The planner recognizes that it cannot take the `warm_IRU2` and `select_IRU2` actions in time to beat (or "preempt") the potential failure. In other words, the planner has projected a failure and its temporal model has shown that no corrective actions are available once state 85 is reached. The only possible solution is to take an earlier, prepositioning action aimed solely at preventing state 85 from ever occurring. The planner labels state 85 as a (transitive) failure state for future reference, and backtracks. The smart backjumping function is able to recognize the first decision that made state 85 reachable, backtrack to that point, and alter that and future decisions to preheat IRU2, avoiding state 85.

Note that this plan fragment is actually a hypothetical example — the real DAP planner running on the domain in Figure 1 generates a plan in somewhat more complex ways because it interleaves the process of splitting (refining) the abstract state descriptions with the process of choosing actions for states. In fact, DAP recognizes the potential failure and plans the prepositioning action at a much earlier point in the search. Space limitations do not permit us to show the actual DAP planning sequence, but the final plan is shown in Figure 3. There are several intriguing aspects of this plan and the domain encoding that deserve brief discussion.

Representation Hacks and Plan Quirks

IRU2 Cannot Fail Observant readers will have noted that IRU2 cannot fail according to Figure 1. We have omitted this transition because if IRU2 can fail, there is no guaranteed safe plan because both IRUs can fail, causing the burn to possibly fail. Since CIRCA will only produce safe plans, adding an `IRU2_fails` event leads the planner to (correctly) conclude that the domain is overconstrained and unsolvable, and that some performance tradeoffs must be made. One typical performance tradeoff, for example, would be to ignore certain lower-probability transitions such as the failure of a backup system. The current DAP state-space planner is not yet integrated with a higher-level tradeoff engine.

Preferring IRU1 Another unusual aspect of the domain encoding is the inclusion of `(IRU1 broken)` in the preconditions of the `select_IRU2` action. This has the effect of making the planner "prefer" to select IRU1, since it cannot use IRU2 unless IRU1 is broken. If this condition is omitted, the planner is smart enough to exploit the fact that IRU2 never fails, by always selecting IRU2. This has the beneficial effect of making IRU1 irrelevant, but it also sidesteps the prepositioning problem. Thus the extra precondition in the `select_IRU2` action is an idiomatic way of forcing the planner to respect a primary/backup ordering over the IRUs.

State 127 As is often the case with AI programs, DAP can arrive at non-intuitive but correct plans. In state 127 of Figure 3, IRU1 is broken but selected and the engine is off. Surprisingly, the planner chooses to turn on the engine and *then* select IRU2, rather than vice versa. According to the domain model of Figure 1, the chosen order is acceptable because the `select_IRU2` action can be guaranteed to complete before the temporal transition to failure (`fail_if_burn_with_broken_IRU1` from state 39) can possibly occur.

Related Work

Other researchers have attempted to extend "classical" AI planning techniques to handle problems like the prepositioning problem. One approach to prepositioning problems is offered by "conditional planners." These are conventional AI planning systems that have been extended so that they can generate plans with embedded "if-then-else" structures. The first such planner was Warren's Warplan-C (Warren 1976). More recently, causal link planners have been extended to conditional planners by Peot and Smith (1992), Pryor and Collins (1996), and Goldman and Boddy (1994). The UWL SENSP softbot planner also did some conditional planning (Etzioni *et al.* 1992).

These conditional planners all make the same extension to classical AI planners: they permit nondeterministic actions that have multiple possible outcomes. For example, the Plinth image processing planner (Goldman & Boddy 1994) has an operator for running a neural net region classifier that may succeed or fail. Conditional planners attempt to find plans that will achieve the goal for all possible outcomes of conditional actions. When Plinth constructs a plan using the neural net region classifier, it considers the possibility that the classifier may fail and adds a contingency plan to handle that eventuality.

Conditional planners do not provide a fully satisfactory solution to the prepositioning problem because

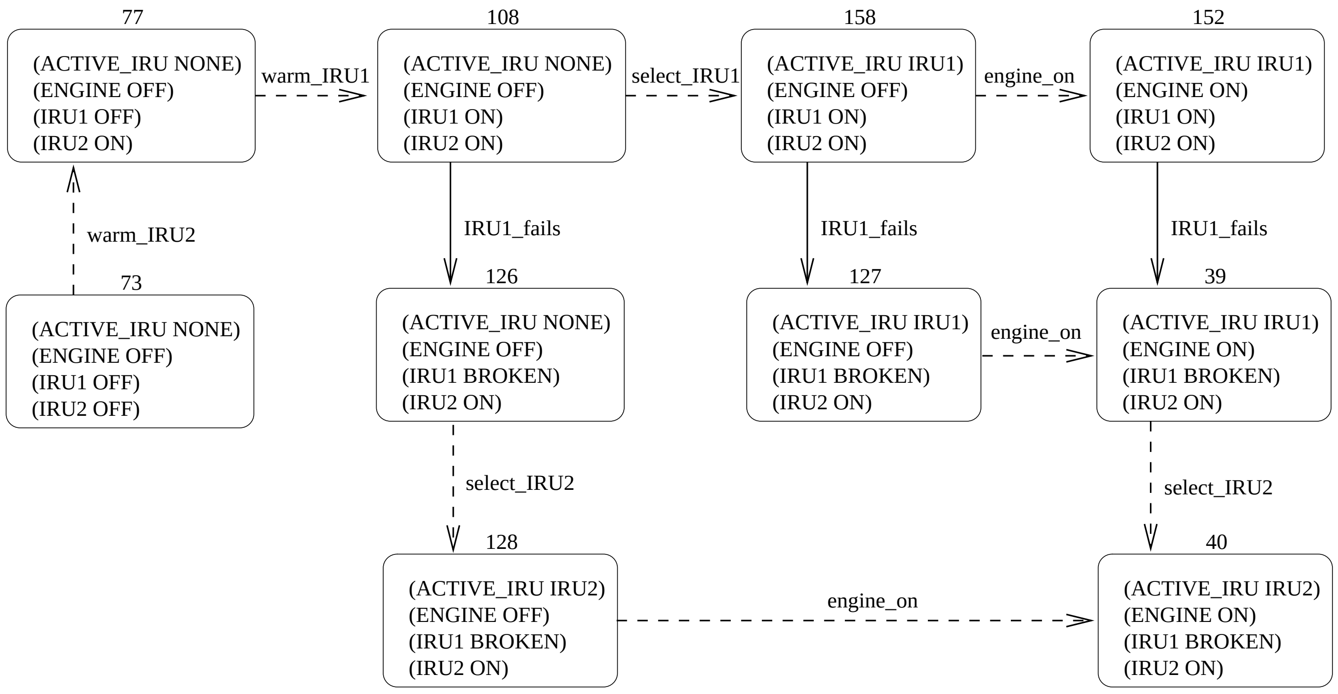

Figure 3: The final Cassini state space.

they do not provide a natural model of the true problem. The true problem is to preposition assets in order to respond to foreseeable disturbances. However, conditional planners keep the classical planning assumption that the planning agent is the only actor. Because of this assumption, exogenous events can only be encoded as an outcome of one (or many) of the planning agent's actions. This encoding complicates the domain knowledge engineering and yields cumbersome encodings in cases where events can occur at multiple times. Furthermore, these planners do not handle the temporal aspects of the prepositioning problem.

Jim Blythe has developed a planner, Weaver, that more directly addresses the problem of reacting to exogenous events (Blythe 1996). This planner generates a plan for a main line of execution, then incrementally adds significant domain events and plans reactions to these events. However, Weaver, like the conditional planners, does not address temporal aspects of the prepositioning problem.

There are probabilistic and decision-theoretic planners that provide facilities similar to those we have discussed above (Drummond & Bresina 1990; Kushmerick, Hanks, & Weld 1993; Draper, Hanks, & Weld 1993; Dearden & Boutilier 1997). We agree that such approaches are, broadly speaking, the Right Thing. However, as a practical matter, the need for a probabilistic model adds a burden over and above the modeling needs of non-stochastic approaches, and does not (in our experience at least) appeal to domain users. We are examining ways to extend CIRCA to handle probabilistic measures of uncertainty, but in a minimalist way. Atkins *et al.* (1996) have also worked on developing a probabilistic CIRCA planner. Their planner is intended to reason about the likelihood of various transitions, so that the planner can use its limited resources to consider the most likely transitions first. This project is complicated by the fact that the CIRCA model is not Markovian: it matters what the preceding state was, and how long one remained there. As far as we know, none of the other probabilistic approaches address the temporal aspects of the problem.

Kabanza *et al.* (1997) have developed a planning method for reactive agents that is similar to the original CIRCA. Their architecture differs in emphasis, however. The NFAs it constructs are "clocked:" they make transitions at times that are the least common denominator of all possible transitions. This scheme will suffer a state space explosion in domains where there is a wide range of possible transition delays, like those to which CIRCA has been applied. Kabanza's group has concentrated on developing a more flexible notation for goals than those used by CIRCA, but they do not make the same distinction between safety and goal achievement.

Conclusions

Our current work on CIRCA extends in four directions: (1) planning with more expressive domain models; (2) determining how best to incorporate CIRCA into an autonomous agent; (3) distinguishing significant (and insignificant) disturbances and (4) developing a more efficient state-space planner (see Goldman *et al.* (1997)).

Currently, CIRCA's planner only chooses either to preempt exogenous processes or to allow them to happen; it cannot *rely* on them happening, because it does not have an upper bound on their delay.This expressive limitation makes it difficult to represent intentional processes like the warming up of an IRU. Currently, we must do one of two things: (1) make "warming up the IRU" an atomic action that takes a very long time or (2) encode the fact that the IRU is warming into the feature space and have a temporal transition from warming to warm. The first alternative is undesirable because it makes it impossible for the CIRCA agent to do anything else while it is waiting for the IRU to warm. The second option is actually worse, because it makes it impossible for CIRCA to rely on the IRU ever becoming warm. To make the second option work correctly, we are extending the CIRCA model to include "reliable temporals" that have upper bounds on their times of occurrence.

We noted above that our Cassini domain encoding incorporates the design decision that it is not worth worrying about two consecutive IRU failures. Ideally, one could use a probabilistic world model to automatically identify significant eventualities, rather than having to identify them manually (cf. Atkins *et al.* (1996)). However, we want to do this without the domain engineering overhead of constructing a full Markov process model; constructing this model is at least as hard as identifying the eventualities for the planner. We are currently examining a technique for model pruning that assumes independent component failures (an assumption commonly used in reliability engineering) and calculating only order-of-magnitude likelihoods on CIRCA state space graphs.

Another limitation is CIRCA's lack of a system clock. Currently, CIRCA can only reason about duration relative to the time it enters a particular state. To meet deadlines that are related to a global clock (e.g., the actual Cassini orbital burn time), CIRCA's RTS executive must be able to act at an appropriate time relative to a planned future event. We do not want to abandon the unclocked executive, because inclusion of global time into the state space can cause it to explode. We are exploring how to coordinate the CIRCA state-space planner with an overall mission planner/scheduler. The mission planner/scheduler will be able to provide signals indicating important global times to the CIRCA RTS. The RTS will then detect these signals like any other state feature. Our preliminary investigations suggest that we can detect the need for such features through search failures in the AIS.

The higher-level CIRCA mission planner/scheduler or "task-level planner" (Musliner 1993) plays a major role in managing the state-space planner's activities and the sequence of plans executed by the RTS. The task-level planner determines which lower-level planning jobs the DAP planner should work on, what safety-critical and best-effort goals should be pursued, what performance tradeoffs should be made, what low-probability transitions should be ignored, etc. This higher-level planner module is an area of active research, with great potential to complement CIRCA's DAP state-space planner and RTS, yielding a powerful, domain-independent intelligent real-time control system.

Acknowledgments This work was supported by the Defense Advanced Research Projects Agency under contract DAAK60-94-C-0040-P0006.

References

Atkins, E.; Durfee, E. H.; and Shin, K. G. 1996. Plan development using local probabilistic models. In *Proc. Conf. on Uncertainty in Artificial Intelligence*, 49–56.

Blythe, J. 1996. A representation for efficient planning in dynamic domains with external events. in the AAAI workshop on "Theories of Action, Planning and Control: Bridging the gap".

Dearden, R., and Boutilier, C. 1997. Abstraction and approximate decision-theoretic planning. *Artificial Intelligence* 89(1–2):219–283.

Draper, D.; Hanks, S.; and Weld, D. 1993. Probabilistic planning with information gathering and contingent execution. Technical Report 93-12-04, Department of Computer Science and Engineering, University of Washington, Seattle, WA.

Drummond, M., and Bresina, J. 1990. Anytime synthetic projection: Maximizing the probability of goal satisfaction. In *Proc. Nat'l Conf. on AI*, 138–144.

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

Gat, E. 1996. News from the trenches: An overview of unmanned spacecraft for AI. In Nourbakhsh, I., ed., *AAAI Technical Report SSS-96-04: Planning with Incomplete Information for Robot Problems.* American Association for Artificial Intelligence. Available at `http://www-aig.jpl.nasa.gov/home/gat/gp.html`.

Goldman, R. P., and Boddy, M. S. 1994. Conditional linear planning. In *Proc. Int'l Conf. on AI Planning Systems.*

Goldman, R. P.; Musliner, D. J.; Krebsbach, K. D.; and Boddy, M. S. 1997. Dynamic abstraction planning. In *Proc. Nat'l Conf. on AI*, 680–686.

Kabanza, F.; Barbeau, M.; and St-Denis, R. 1997. Planning control rules for reactive agents. Technical Report 197, Computer Science Dept., University of Sherbrooke.

Kushmerick, N.; Hanks, S.; and Weld, D. 1993. An algorithm for probabilistic planning. Technical Report 93-06-03, Department of Computer Science and Engineering, University of Washington, Seattle, WA. to appear in *Artificial Intelligence*.

Musliner, D. J.; Durfee, E. H.; and Shin, K. G. 1993. CIRCA: a cooperative intelligent real-time control architecture. *IEEE Transactions on Systems, Man and Cybernetics* 23(6):1561–1574.

Musliner, D. J.; Durfee, E. H.; and Shin, K. G. 1995. World modeling for the dynamic construction of real-time control plans. *Artificial Intelligence* 74(1):83–127.

Musliner, D. J. 1993. *CIRCA: The Cooperative Intelligent Real-Time Control Architecture.* Ph.D. Dissertation, University of Michigan, Ann Arbor. Available as University of Maryland Computer Science Technical Report CS-TR-3157.

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

Pryor, L., and Collins, G. 1996. Planning for contingencies: A decision-based approach. *Journal of Artificial Intelligence Research* 4:287–339.

Warren, D. H. 1976. Generating conditional plans and programs. In *Proceedings of the AISB Summer Conference*, 344–354.