No Planner is an Island: Lessons from the Nextcut Process Planner

Subbarao Kambhampati*
Department of Computer Science and Engineering
Arizona State University, Tempe AZ 85287-5406
rao@asu.edu; (602) 965-0113

My experience with "practical planning problems" is mainly through the process planning domain. I will describe some of the aspects of this domain that set it apart from the simulated toy-domains, and address their implications for the "typical" AI planning algorithms. The work itself is described in the references [8, 6, 7, 4, 5].

The paper is organized into three parts. In the first part, I describe the process planning domain in general terms and explain its interesting characteristics from AI planning view point. In the second part, I provide a brief overview of the NEXT-CUT process planning system. In the third part, I attempt to answer the questions raised in the symposium CFP in terms of our process planning work. It is possible to read Parts I and III to get a high level picture, and then get details from Part II.

Part I

Characteristics of Process Planning Domain

Given the geometric and/or feature based description of a part and its dimensional and tolerance specifications, *manufacturing process planning* is the problem of finding the sequence of machining operations, the setups and the fixtures to be used to manufacture the part (see Figure 1). Process planning is known to be a very time-consuming and knowledge-intensive problem in automating manufacturing. In the past, the majority of process planning tasks were done by humans -- either starting from scratch, or by manually modifying existing process plans. The lack of automation in process planning has not been a big bottle-neck until now because of the focus on "mass-manufacturing."

Of late however, there has been an increased interest in rapid prototyping and in flexible, customized small-lot manufacturing. This has made the automation of process planning task very critical. Unfortunately however, despite the recent advances in CAD/CAM and information technologies, automated process planning has still not achieved any widespread use in industry. This is partly due to the fact that most existing approaches use *ad-hoc* heuristic techniques for plan generation [1]. These approaches lack any sound theoretical basis, and are consequently brittle. In most cases, the resultant "planners" are simple bookkeeping tools that delegate the majority of the planning tasks to the human planner. One way to improve the state of affairs would thus be to base the development of process planners on a systematic and scientific theory of planning.

Planning, as a domain-independent problem, is studied in Artificial Intelligence, where it is cast as the problem of composing a course of actions capable of transforming the world from a given initial state to a desired goal state. There has been a significant amount of work on this problem within the past twenty years, with more recent work (e.g. [9, 10]) clarifying the formal foundations and relative tradeoffs of the various planning models. Although the classical planning techniques themselves have hither to been applied only to tightly constrained synthetic domains, there has been increased interest in applying them to realistic problems.

The lack of systematic basis for automating process planning, coupled with the interest in realistic applications of AI planning models suggests an obvious prescription to remedy both: *apply classical planning techniques to automate process planning*. In our previous work [8, 4, 5], we explored this course, and found that it is more complicated than a simple "application" of existing AI planning techniques. In particular, we found that the AI planning techniques need to be extended in the following *fundamental* ways before they can be applied to manufacturing planning:

1. Much of the work in AI planning has been aimed at generating plans from scratch. However, in process planning, the domain has been standardized so as to make "variant planning," --- the technique of (manually) modifying existing process plans to solve new planning problems -- the dominant method [1]. For successful operation in this domain, we need planning models that allow reuse of previously generated plans.
2. In classical planning framework, the planner is often modeled as an isolated and independent module which is solely responsible for plan generation, and which has all the knowledge relevant to plan generation at its sole disposal. In contrast:

*Much of this work was done when I was a research associate at the Center for Design Research at Stanford University. I think Mark Cutkosky, Marty Tenenbaum, Soo-Hong Lee, Andrew Philpot and other NEXT-CUT group alumni for their active help and participation in the project. Writing of this symposium paper is supported in part by NSF research initiation award (RIA) IRI-9210997, NSF young investigator award (NYI) IRI-9457634 and ARPA/Rome Laboratory planning initiative grant F30602-93-C-0039.

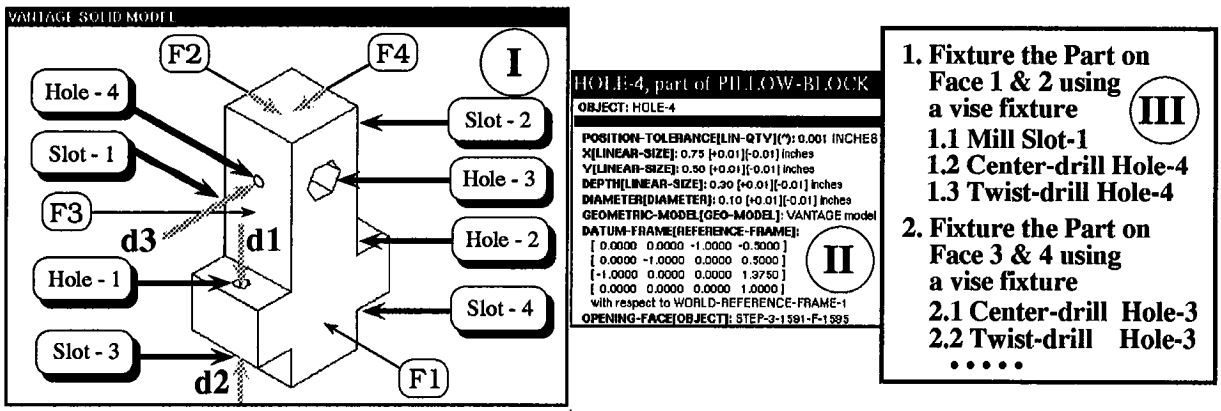


Figure 1: Geometric and feature-based specification of a part and a fragment of the plan for machining it

- The process planning problem is typically too complex for complete automation.
- Even if there are techniques that allow complete automation, human process planners have considerable experience and creativity that should not be replaced but enhanced to make the planners more productive.[1]
- The assumption of an omniscient planner is inadequate for process planning which requires significant amounts of deep domain-specific reasoning involving geometry, kinematics and cutting and clamping forces -- which is either awkward or inefficient to encode into a classical planner.

These differences necessitate a model of planning that is "*hybrid*" in the sense that the planning task is shared between the automated planner, and a host of other human and automated reasoners.

3. Classical planning assumes that planning is a one-shot process of inputting the problem specification and outputting the plan. In contrast, in many situations, process planning is a *continual and iterative* process. For example, in a concurrent design situation, the designer may generate a design, evaluate its feasibility by generating and inspecting the process plan, and based on the results of the inspection, modify the design and re-initiate the planning and evaluation cycle. Handling such iterative specification changes necessitates a model of planning where the planner is *incremental* and *interactive*.

As the foregoing shows, before the AI planning frameworks can be used as a basis for automating process planning, they need several foundational extensions.

[1]In fact, many process planning researchers found that the industries are more willing to accept systems that "*help*" human process planners than those that attempt to "*replace*" them.

1 Overview of Approach taken

Given that process planning involves extensive reasoning about geometry, kinematics, and cutting and clamping forces, the most efficient way of generating plans may involve intelligently interfacing AI planning techniques with specialized reasoners such as geometric and fixture based reasoning. We thus took a hybrid approach for plan generation, as shown in Figure 2. A classical HTN planner was used for doing machining planning (coming up with the sequence of machining operations), while a solid modeler and geometric reasoner were used to handle geometric and force related reasoning.

Such hybrid architectures of course raise a host of open issues about the modes of coordination and communication between the planner and the specialized reasoners (see [8, 6] for a detailed discussion, and some preliminary approaches). They also have implications for the planning algorithms. Planning in such architectures is a continual rather than a one-shot process. The constraints imposed by the specialists on the plan force the planner (and the specialists) to contend with a continually evolving problem specification. The evolutionary nature of planning has implications for the internal operation of the planner and the specialists. For example, hierarchical abstraction, and the ability to represent plans with partial commitment (partial ordering etc.) are important for allowing the specialists maximum latitude in specializing the plan according to their considerations. More importantly, since inconsistent commitments between the planner and the specialists cannot be completely avoided, incremental operation, in terms of the ability to reuse previous results while accommodating new constraints [2], is essential for efficiency. (Note that in contrast to the classical planning model, where such replanning ability is justified purely in terms of the internal efficiency of the planner, here it is also motivated by the desire to promote efficient interaction between the planner and the specialists.)

Finally, planners in domains such as process planning need also interact flexibly with the human users. In most realistic

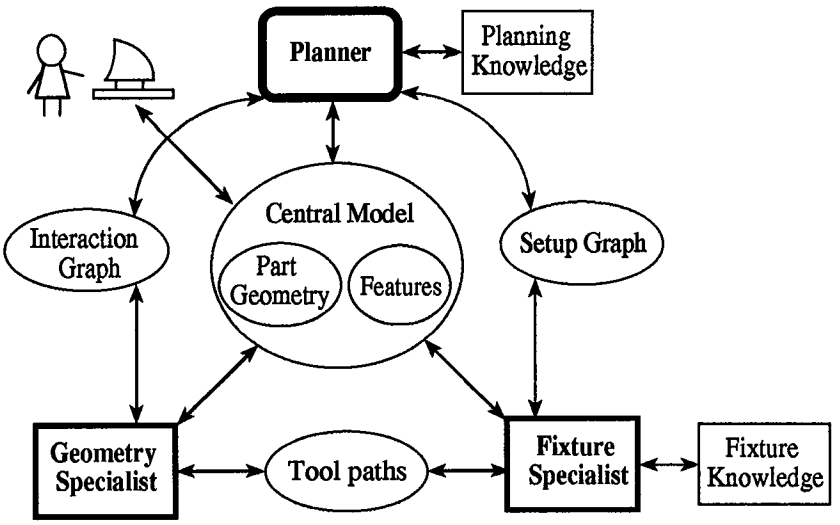


Figure 2: Schematic Diagram of the Planning Architecture in Next-Cut

planning domains (other than the "*robot roaming the hallways*" domain), automated planners of the current day will have to work as decision-support systems to humans. This has several implications on the underlying planning methodology. First, the planner should allow the user to specify easily available domain structure and control information (rather than *insist* on reinventing the information by itself). In our (admittedly limited) experience, we found that the task reduction planning frameworks provide more support for this than the purely subgoal-establishment oriented planners (see [10] for a discussion as to why this might be so). Second, the planner should also provide structured modes in which the user can influence (*steer*) the planner's search process. Most current planners have very little support for this sort of planner steering, and facilitating it presents several open problems (including *where* to allow user control, and *how* to capture and store the rationale for the user decisions so as to exploit them in latter planning episodes).

Part II

Planning Architecture in Next-Cut

In this part, I present a more detailed description of the hybrid process plannig architecture used in NEXT-CUT, discuss the planning process, and illustrate it with examples. The material in this part is excerpted from [8]. Interested readers are encouraged to refer to the latter paper for a more elaborate presentation.

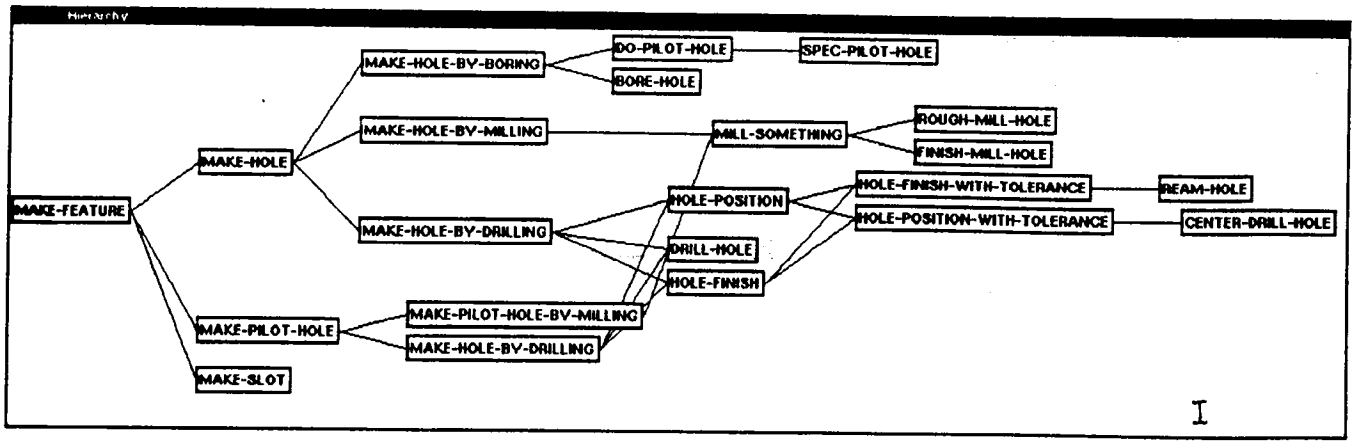
2 Overview of the Architecture

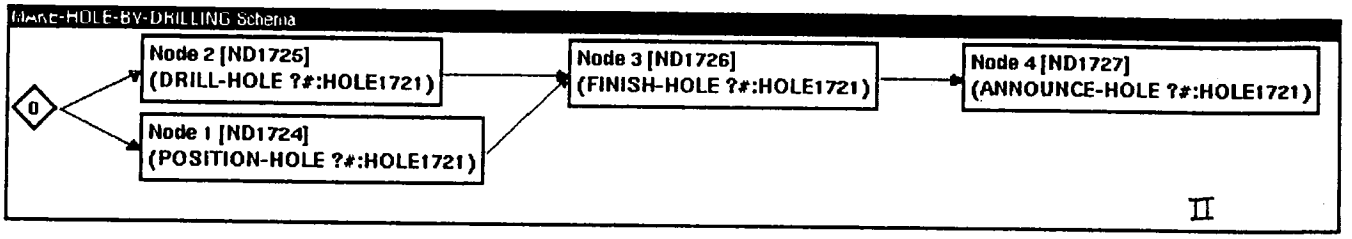
Figure 2 shows the schematic of the planning architecture in the NEXT-CUT environment. A general purpose planner is used for selecting appropriate machining processes and tools and composing them into a machining plan. A geometry specialist is used to detect and resolve geometric interactions that arise during machining, and a fixturing specialist is used to decide the orientations and clamping forces for holding the part during machining.

There are two forms of communication between the planner and the specialists in the NEXT-CUT environment. The first, and more straightforward, is through the shared central model. The planner and specialists can also communicate directly through specialized interfaces (e.g. interaction graph, setup graph). These interfaces facilitate efficient reasoning about interactions between the modules. In this paper, we will be concentrating on the interfaces between the planner and the two specialists.

2.1 Planner

The basic planning paradigm that we use is that of nonlinear hierarchical planning [12, 13, 10]. In this paradigm, the planning tasks involve the satisfaction of a conjunction of goals and the planning process consists of successively refining the planning tasks with the help of a set of a prespecified task-reduction schemata. The reduction schemata consist of plan fragments for achieving various goals. As an example, Window I in Figure 3 shows the various schemata for machining holes, while window II shows an individual task reduction schema, MAKE-HOLE-BY-DRILLING in detail. As can be seen from Window II, the MAKE-HOLE-BY-DRILLING schema is a collection of partially ordered planning steps for machining a hole by drilling it. In particular, it specifies that the hole has to be positioned, a drilling operation has to be carried out, and finally the drilled hole should be improved as necessary (e.g. improving diametral tolerance, surface finish etc.). Notice that this fragment does not provide details about how the position and diametral tolerances should be achieved; those details are left for subsequent reductions. It is in this sense that the schema specifies an abstract plan fragment. As shown in Window III, the internal representation of the schema also includes information about which conditions have to be satisfied, and which effects are asserted at each step. The conditions dictate to a large extent whether a particular schema is suitable for accomplishing a particular task. For example, if either the tolerance requirements of a hole are very high, or the hole happens to be of a nonstandard size, MAKE-HOLE-BY-DRILLING will not be a candidate






```
{SCH1731}
   MAKE-HOLE-BY-DRILLING::(MAKE-HOLE ?HOLE1713)
      Expansion:
           0 {<0::ND1715>[:DUMMY]}
           1 {<1::ND1716>[:ACTION(POSITION-HOLE ?HOLE1713)]}
           2 {<2::ND1717>[:ACTION(DRILL-HOLE ?HOLE1713)]}
           3 {<3::ND1718>[:ACTION(FINISH-HOLE ?HOLE1713)]}
           4 {<4::ND1719>[:PRIMITIVE(:ANNOUNCE-HOLE :NAME ?HOLE1713)]}
      Conditions:
           <<SC1720>> :USE-WHEN (HOLE-SPEC ?HOLE1713) :at 1
           <<SC1721>> :USE-WHEN (SPEC (DIAMETER ?HOLE1713) ?DIAMETER) :at 1
           <<SC1722>> :USE-WHEN (TOOL ?TOOL) :at 1
           <<SC1723>> :USE-WHEN (EQUAL (TOOL-TYPE ?TOOL) :TWIST-DRILL) :at 1
           <<SC1724>> :USE-WHEN (EQUAL (DIAMETER ?TOOL) ?DIAMETER) :at 1
           <<SC1725>> :USE-WHEN (SPEC (BOTTOM-CONDITION ?HOLE1713) ?BOTTOM-CONDITION1714) :at 1
           <<SC1726>> :USE-WHEN (BOTTOM-DRILLABLE ?BOTTOM-CONDITION1714) :at 1
           <<SC1727>> :COMPUTE (SLB-CL::FIND-TOOL-HOLDER-INTERFERENCES (QUOTE ?HOLE1713) (QUOTE ?TOOL))
      Effects:
           <<SE1728>> :ASSERT (<= (POSITION-TOLERANCE ?HOLE1713) 0.004) :at 2
           <<SE1729>> :ASSERT (<= (DIAMETRAL-TOLERANCE ?HOLE1713) 0.005) :at 2
           <<SE1730>> :ASSERT (MAKE-HOLE ?HOLE1713) :at 4
      Vars: (?HOLE1713 ?BOTTOM-CONDITION1714)
                                                                    III
```

Figure 3: Specification of machining operations as task reduction schemata. A complete domain description is available from the author electronically.

for machining that hole.

A hierarchical plan can be formally characterized as a 3-tuple,

$$\mathcal{P} : \langle \ \langle T, O, \mathcal{V} \rangle \ , T^* \ , D \ \rangle ,$$

where T is a set of plan steps (tasks) with O defining a partial ordering over them; and T^* is the union of tasks in T and their ancestors with D defining a set of parent, child relations among the tasks of T^*. Planning consists of refining abstract planning tasks (such as `(Make-hole Hole-2)`) into concrete subtasks with the help of these task reduction schemata, until every task in the plan is "primitive" (i.e., the planner knows how to perform that task)[2]. Figure 4 shows how the `(Make-hole Hole-2)` task is refined, with the help of MAKE-HOLE-BY-DRILLING schema (in Figure 3), into three sub-tasks `(Position-hole Hole-2)`, `(Drill-Hole Hole-2)` and `(Finish-Hole Hole-2)`, which in turn are reduced to more concrete subtasks.

During this refinement process, any interactions between the newly introduced steps and the existing steps are resolved by posting ordering and binding constraints on the plan. As a classical hierarchical planner, the planner only detects the interactions that become evident in terms of clobbered preconditions. The partially ordered plan for machining the `cross-product` is shown in Figure 6 (see Section 3 for further discussion). The planning strategy is "least-commitment" in that the planner starts with the assumption that the various design goals can be achieved in *any* order and imposes ordering relations only to remove interactions or to satisfy constraints. Avoiding over-commitment in this way facilitates subsequent processing of the generated plan for satisfying optimality criteria (e.g., merging machining steps to reduce setups and tool changes) [4].

As pointed out in Part I, the planner needs the ability to modify its plans incrementally both to promote efficient interactions with the specialists and to deal with user-imposed changes in the design of the part. Our planner supports incremental plan modification by maintaining the causal dependencies among the individual steps of a plan, and the decisions underlying the development of that plan, in a representation called a "validation structure." It utilizes the PRIAR modification framework [4, 2] for carrying out the modification (see Section 3.1 for details).

2.2 Specialists

The specialists in our framework either augment the specification of the problem as seen by the planner and detect interactions that the planner itself cannot detect, or utilize the generated plan to make their own further commitments. In our system, the geometry specialist (see below) is of the former type, while the fixturing specialist is of the latter. The analyses by the specialists impose implicit constraints on the plan developed by the planner (and vice versa). The interfaces -- the interaction graph, and the setup graph -- help the modules in keeping track of these constraints.

1. Geometry Specialist: The geometry specialist in the NEXT-CUT environment uses solid models of the part and features to detect a variety of geometric interactions that may affect the machining or fixturing of parts. Examples of such interactions include interferences between the tool paths for machining a feature, and the volumes of other features (or the part itself). In the case of the `cross-product` shown in Figure 1, the tool access path for machining `hole-4` (shown by the shaded arrow `d3` in the figure) interferes with the feature volume of `slot-1`. Window I in Figure 5 shows the geometry specialist's description of the interference detected in this case. Such interactions are ubiquitous in machining and are therefore computed with every design or plan change. Since the exact details of the tool paths are not yet known[3], and also since exact volume intersections can be time consuming, our geometry specialist uses conservative rectangular bounding box approximations of the material that a tool could remove from the part, and of the total volume swept out by a tool [7].

Once such interferences are detected, appropriate actions must be taken to resolve them (if possible). The geometry specialist does this by analyzing the interferences. In particular, suppose an interference $\mathcal{I}_{f_d}$ is detected for the tool approach direction d of feature f. The geometry specialist checks to see if the volume of the detected interference $\mathcal{I}_{f_d}$ is wholly subsumed by the volumes of some subset $\mathcal{F} = \{f_i\}$ of other features of the part. If this is the case, then the interference $\mathcal{I}_{f_d}$ can be avoided by machining the features in $\mathcal{F}$ first (if no such set $\mathcal{F}$ is found, then the feature f cannot be made in tool approach direction d). This essentially imposes a set of constraints O_{f_d} on the machining order of the individual features:

$$O_{f_d} = \{(f_i \prec f) \mid f_i \in \mathcal{F}\}.^4$$

In the case of interference between `hole-4` and `slot-1`, the analysis by the geometry specialist shows that the interference between the part, and the tool path for making `hole-4` in the direction `d3` is completely subsumed by the feature volume of `slot-1`. Thus, this interaction can be avoided by machining `slot-1` *before* machining `hole-2` if `hole-2` is to be made in the direction `d1`.

In this fashion, the geometry specialist detects the interactions for each feature and each possible tool approach direction for making that feature, and computes the appropriate ordering relations for avoiding those interactions. Once this is done, the geometry specialist heuristically selects a single tool approach direction for each feature (based on such criteria as the number of geometric interactions to be resolved in that direction) and conveys the corresponding feature orderings to the planner by constructing (or updating) the interaction graph (see Section 3)[5]. Window II in Figure 5 shows the

[2]Sometimes a task does not require further reduction because all of its effects already hold in the current situation (in planning terminology [11], such tasks are called *phantom goals*). For example, in the plan for machining `cross-product`, shown in 6, the finishing step was not required for HOLE-2, since the specified diametral tolerance for HOLE-2 is guaranteed by the drilling step itself. Thus the (DIAMETRAL-TOLERANCE HOLE-2) step is ***phantomized*** (shown with dashed lines in the figure) and does not constitute a step to be executed in the final plan.

[3]The detailed geometry of tool path depends on the exact tool that is selected for machining the feature, which will only be known after the machining planning is over

[4]The symbol "$\prec$" is used to denote precedence relation between two entities. Thus the expression $a \prec b$ means a should precede b.

[5]Thus, the orderings imposed by the geometry specialist are conditional on the tool approach directions chosen, in the sense that if at a later point, the fixturing specialist decides to make a feature in a different orientation, then the ordering in the interaction graph

interaction graph corresponding to the `cross-product` (note the ordering relation between `slot-1` and `hole-2`).

2. Fixturing Specialist: The objective of the fixturing specialist is to decide which operations of the plan will be done in which setup, and to arrive at fixture arrangements for locating and restraining the part as it is machined. The windows F1-F4 in Figure 7 show a fixturing plan for manufacturing `cross-product`. An important consideration here is to reduce the number of setups. The operation of the fixturing specialist can be seen as having two phases; with the first phase consisting of proposing setups and the second phase consisting of testing them, employing geometric, kinematic and force calculations. To reduce the number of setups, the fixturing specialist merges the steps of the machining plan based on the expected orientation of the part (given by the tool approach direction selected for that feature by the geometry specialist; see above) during those steps. In the second phase, it checks if the part can actually be fixtured in the proposed setups, and selects fixture elements for restraining the part during machining. This involves selecting a particular sequence (total ordering[6]) of the proposed setups (consistent with the ordering constraints among plan steps that comprise the setup groups), and ensuring that the geometry of the work-piece at the start of each setup allows it to be fixtured satisfactorily. The specific sequence of fixturing groups that are tested by the fixturing specialist then constitutes the fixturing plan. A constraint graph called the "setup graph," which contains information about the chosen setup groupings, and the ordering relations among them, acts as the interface between the fixturing specialist and the planner (see Section 3).

3 The Planning Cycle

When the specification of a part, such as that of `cross-product` as shown in Figure 1, is entered for the first time, the geometry specialist computes the possible geometric interactions between its features (as shown by the example in Window I of Figure 5). Specific ordering constraints to avoid these interactions are then conveyed to the planner via the interaction graph (Window II).

Given the plan representation discussed in Section 2.1, the **interaction graph** can be seen as an augmentation to the top-level specification of the problem. In particular, the interaction graph can be represented by a directed acyclic graph (DAG) $\mathcal{G} : \langle F, O_g \rangle$ whose nodes are the individual features of the part, whose edges define a partial ordering on the machining of different features. From the discussion in Section 2.2, we can see that $O_g = \bigcup_f O_{f_d}$, where d is the chosen tool approach direction for feature f, and O_{f_d} is the set of precedence constraints imposed by the geometry specialist to resolve any tool path interferences in machining f in direction d.

The effect of the analysis by the geometry specialist is that instead of starting with unordered goals, the planner orders them according to the vrestrictions imposed by the interaction

would change. For a more detailed description, see [7].

[6]The need to ground the fixturing checks relative to the particular (intermediate) geometry of the part, and the difficulty of generating and maintaining partial geometries, are the main reasons why the fixturing specialist is forced to select a specific total ordering.

Given a new or changed specification:

1. **Geometry Specialist:** (*Input*: The solid model of the part and the features)
   Compute geometric interferences and update interaction graph
2. **Planner:** (*Input*: Feature specification, interaction graph, setup graph)
   (a) If no machining plan exists, generate one using the feature specification and the interaction graph. If there are any tool-holder collisions, backtrack to the geometry specialist (see Section 3.1).
   (b) If a machining plan exists, modify it to accommodate the new specifications (changes in feature attributes, interaction graph or setup graph), while respecting any implicit constraints imposed by the setup graph and the interaction graph (see Section 3.1)
3. **Fixturing Specialist:** (*Input*: Machining plan, feature geometry, setup graph)
   (a) If a fixturing plan does not exist, construct the setup graph by merging steps of the machining plan. Select a setup sequence and compute the fixturing details for it. If no such total ordering is found, backtrack to the planner (see Section 3.1).
   (b) If a fixturing plan does exist, update the setup graph to reflect changes (if any) in the machining plan. Use it to incrementally revise the existing fixturing plan. Update the setup graph.

Listing 1. High level description of the planning cycle in NEXT-CUT

graph. In particular, the planner starts with an initial task network $\langle T', O' \rangle$, with T' containing the set of tasks of the form t_i : *Achieve*(*feature*$_i$), and orderings of type

$$[t_i : \textit{Achieve}(\textit{feature}_i)] \prec_{O'} [t_j : \textit{Achieve}(\textit{feature}_j)]$$

if and only if $\textit{feature}_i \prec_{O_g} \textit{feature}_j$. The final plan thus incorporates the orderings imposed by the planner, as well as those inherited from the interaction graph.

The machining plan for `cross-product` is shown in Figure 6. (The diamond shaped steps are dummy steps, and steps with dotted boundaries correspond to "phantom" steps, i.e., steps whose intended effects are made true by other steps). Notice in particular that the machining steps for `slot-1` and `hole-4` (in the lowest branch of the plan in Figure 6) are ordered according to the constraints specified by the interaction graph (Window II in Figure 5).

Next, based on this plan, the fixturing specialist chooses setups for fixturing. From the planner's view point, the fixturing specialist is partitioning the plan steps into groups, based on a set of equivalence classes defined in terms of the expected orientations of the part during plan execution. Such a partitioning induces an implicit partial ordering among the setups. As discussed in Section 2.2, this partitioning is followed by checks to ensure that some total order of setups consistent with these this partial ordering can actually be fixtured.

The **setup graph** can thus be seen as a DAG $\mathcal{S} : \langle \mathcal{W}, O_s \rangle$ where each member $\omega \in \mathcal{W}$ is a set of plan steps that can be machined in a particular setup, and O_s is a partial ordering on the setups, induced by the corresponding partial ordering

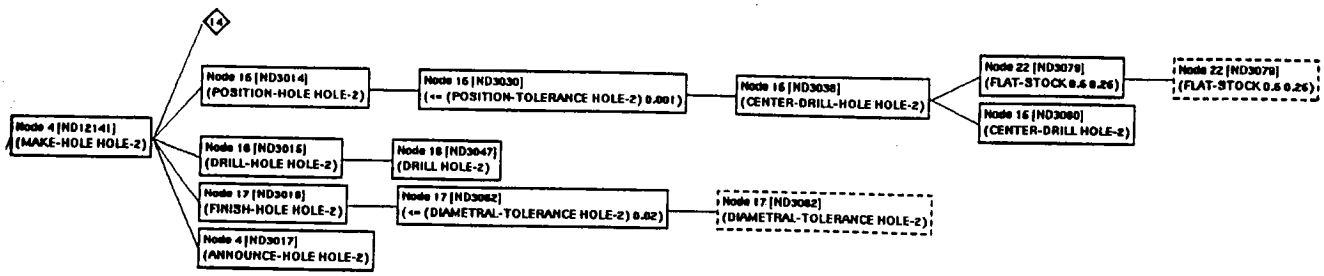


Figure 4: Reducing an abstract task into concrete sub-tasks

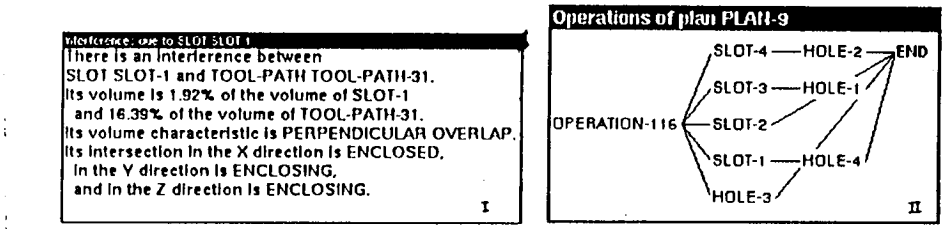


Figure 5: Detecting and Resolving geometric interactions for the `cross-product`

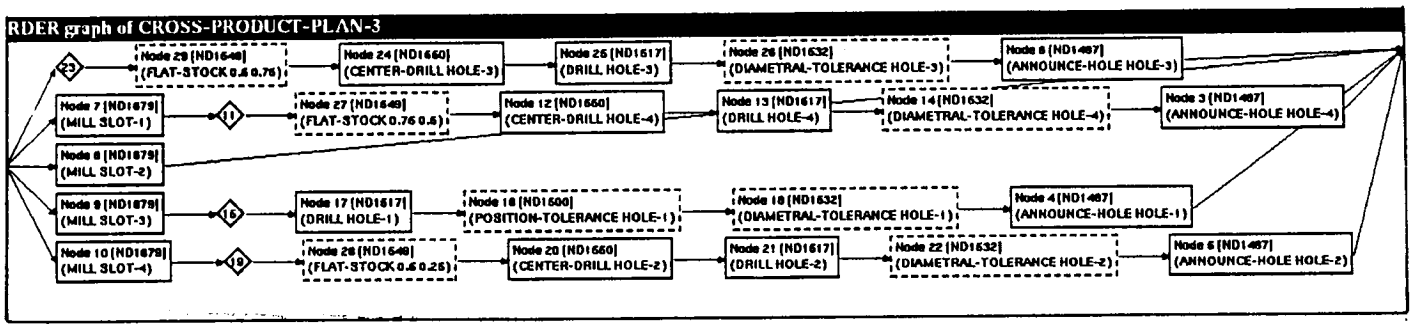


Figure 6: Machining plan for the `cross-product`

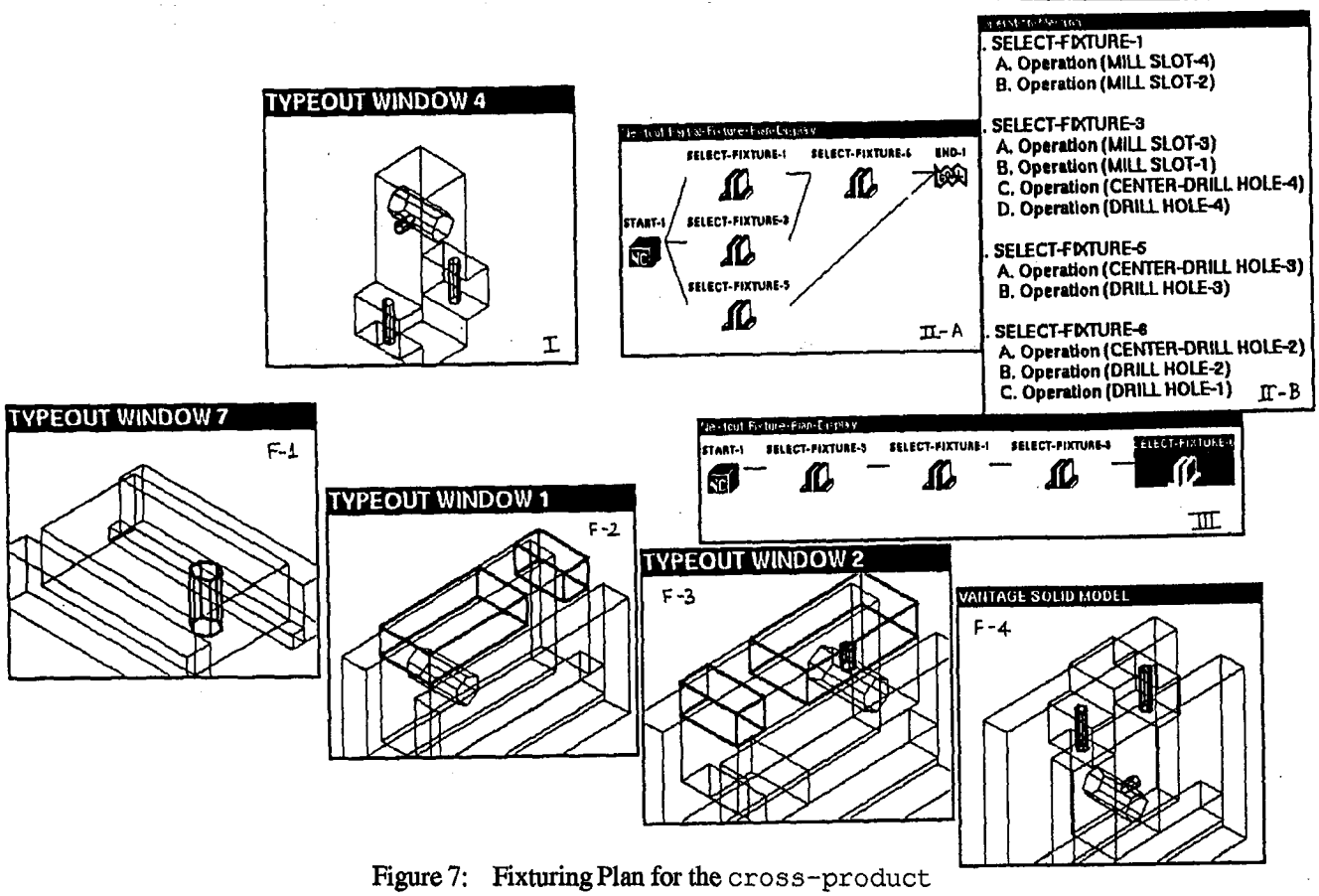


Figure 7: Fixturing Plan for the `cross-product`

on the plan steps.

The constraints on the setup-graph from the planner's viewpoint are that $\mathcal{W}$ be a set of mutually exclusive and exhaustive subsets of tasks in T, such that the partitioning is consistent with the partial ordering among the tasks. To ensure the latter, the following two constraints must be satisfied:

1. $\forall \omega \in \mathcal{W},\ \forall t_1, t_2 \in \omega\ \nexists t \in T$ s.t. $t \notin \omega \wedge (t_1 \prec t \prec t_2)$
2. $\forall \omega_1, \omega_2 \in \mathcal{W}$ if there exists a task $t_1 \in \omega_1$ and $t_2 \in \omega_2$ such that $t_1 \prec t_2$ in the plan, then it should necessarily be the case that $\omega_1 \prec_{O_s} \omega_2$

O_s thus defines the partial ordering induced among the setups as a result of merging the steps of the plan.

For the `cross-product` example, Window II-A in Figure 7 shows the setup group mergings computed, and Window II-B shows the description of the individual plan steps merged under each setup group. Notice that the graph is partially ordered at this point.

From the point of view of fixturing specialist, each $\omega \in \mathcal{W}$ is a fixturing group. In general, once the fixturing specialist makes a merging of the plan steps according to the above constraints, there is an implicit partial ordering among the fixturing groups (as stated in the condition *ii* above). From the standpoint of fixturing, this merging is consistent as long as the fixturing specialist can find a sequence of the setup groups consistent with this partial ordering, which satisfies the fixturing constraints (see Section 2.2). To this end, the fixturing specialist first selects a total order on $\mathcal{S}$ based on some heuristic considerations [7], and then carries out fixturing analysis in accordance with that sequence[7]. Once a totally ordered sequence of setups is selected, that further constrains the orderings among the steps of the machining plan implicitly. In particular, selecting a total order on a setup graph $\mathcal{S} : \langle \mathcal{W}, O_s \rangle$ is equivalent to adding a set of additional ordering relations O_F among the setups in $\mathcal{W}$ such that $O_s \cup O_F$ induces a total order on $\mathcal{S}$. Every new ordering $\omega_i \prec_{O_F} \omega_j$ among setups translates to additional orderings among plan steps such that $\forall t_i \in \omega_i$ and $\forall t_j \in \omega_j$, $t_i \prec t_j$ (even if t_i and t_j do not have any ordering relations imposed among them by the geometry specialist or the planner). Such implicit constraints have to be respected to ensure conservatism of any future plan revision (see Section 3.1).

For the `cross-product` example, the fixturing specialist selects one total ordering (shown in Window III in Figure 7) consistent with this graph that is satisfactory from the fixturing viewpoint, and computes a fixturing plan (in each fixture setup, the features to be manufactured in that setup are shown highlighted). It then updates the setup graph with additional orderings corresponding to the selected sequence. The windows F-1 to F-4 in Figure 7 show the details of the fixturing plan. At this point, we have a complete process plan for machining `cross-product` (see Section II).

[7]Notice that different setup sequences have differing fixturing properties as they correspond to different intermediate geometries of the part during machining

3.1 Backtracking and Incremental Plan Revision

When inconsistencies arise between the commitments made by the planner and the specialists, the linear control flow discussed in Section 3 disrupted, and backtracking is necessitated. When this happens, there are in general a variety of backtracking alternatives, some intra-module, and some inter-module, each presenting a different set of tradeoffs. The inter-module backtracking is guided by the interfaces between the planner and the specialists. Such inter-module backtracking is often costly. To contain this, and to improve efficiency of the overall planning, it is important for individual modules to have the ability to accommodate changes in their specifications by incrementally modifying their plans. Similar revision is also necessitated in response to designer initiated specification changes. In both cases, the revision needs to be conservative both to ensure internal efficiency of planning, as well as to contain the ripple effects of changes in the plan on the analyses of other modules. Furthermore, to improve the overall efficiency, the planner's ability to reuse its plans will have to be supplemented by the specialists' ability to reuse their previous analyses. In our implementation, both the planner and the fixturing specialist have the ability to reuse previous results. While each module maintains the internal dependencies on its plans, the external (inter-module) dependencies are maintained through the interfaces. The planner uses the PRIAR modification framework, developed in our earlier work [2, 3] to carry out plan revision. See [8] for further discussion.

4 Results

The planning architecture described in the previous sections has been implemented as a prototype on top of the NEXT-CUT planning framework. Several empirical studies have been conducted on this architecture [8]. We found that the architecture avoids duplicating capabilities of the specialists in the planner, thereby eliminating redundancy, and improving efficiency and modularity. Our implementation was able to automatically generate process plans that satisfy the constraints of geometry, machining, and fixturing specialists cooperating in an integrated framework. The architecture provides a first account of how a general purpose planner can be integrated with a set of specialists. We developed interfaces between the planner and the specialists that allow both to explicitly keep track of externally imposed constraints. In the case of the planner, all the external constraints have been modeled as additional orderings and mergings among machining steps. We found that these interfaces allow the planner to function with a minimal understanding of the internal operations of the specialists, or the domain specific knowledge they employ. We also found that the ability to incrementally modify existing plans to accommodate external constraints effectively controls the proliferation of secondary interactions, in the event inconsistent commitments between the planner and the specialists are detected.

Part III
Retropsective Analysis

In the following, I attempt to address some of the questions raised by the symposium CFP in the context of the NEXT-CUT process planning system.

What was the most difficult aspect of this problem?

The difficult aspects of this problem include the necessity of deep geometric and force based analyses, the requirement of optimal partially ordered plans (which can give rise to least number of setups), the requirement for adequate communication between the planner and the specialized reasoners and the humans that it interacts with.

What did you think would be difficult that was surprisingly easy?

At the outset, I thought that the most difficult thing would be the combinatorics of action-interactions (as is the case in domains like blocks world). In the end, I found that the normal interaction detection and resolution phase takes up relatively small amount of time in process planing domain.

Although there are some interactions between machining operations (e.g., center drilling should come before the corresponding drilling operation), these inteactions can be avoided by packaging primitive machining operations into plan fragments. The task reduction planning framework, used in the nextcut process planning domain, provides good support for this. The real cost is in detecting the geometric interactions, and optimzing the partial plan to get least number of set ups in fixture planing.

If you used a hybrid solution, why?

As mentioned earlier, we did use a hybrid solution, where an AI planner was used to do the machining planning, while a solid modeler and a fixture planner were used to do the geometric reasoning and fixture planning respectively. The reasons for going for hybrid solution are: (*i*) to avoid reinventing geometry and force-based reasoning within a STRIPS action representation (apart from the obvious inefficiencies of such a reinvention, it would also have the drawback of almost surely alienating the user group!) (*ii*) to exploit the already existing methods for dealing with geometric and fixture planning.

Lessons learned from the implementation

The implementation has also taught us several general principles on designing hybrid planning systems; we summarize them briefly below:

- Communication between the planner and the specialists takes several forms, including the shared representation of the design and process plan, specialized representations of mutual constraints (e.g, the setup graph) and standardized messages (e.g., the results of intersection tests from the geometry specialist). In all cases, there is a tradeoff between expressiveness and abstraction. For example, the geometric intersection results, as in Window I of Figure 5, were found after some experimentation to be at the right level of detail for making

ordering decisions in process planning. More generally, it will be impossible to satisfy a variety of modules with messages and representations at a single level of detail. A solution to this problem may be to exploit hierarchical representations.

- Modules in a hybrid planning environment benefit from hierarchical representations and least commitment approach in problem solving which keeps options open and reduces the need for backtracking in the face of specification changes and planning conflicts (by allowing maximum latitude to the specialists in generalizing refining the plan according to their constraints). In our implementation, for example, we maintain partially ordered machining plans, and setup graphs.[8]
- Each module should reuse previous results whenever practical, both for speed and to make the effects of design changes manifest. Reuse of previous results is particularly useful in managing the interactions between the planners and the specialists. Every time a module computes a new result, it is possible that it may invalidate results previously computed by other modules. However, to the extent that each module reuses previous results, the incidence of new side-effects and interactions with other modules is reduced. Thus, if the process planner makes only minor changes to a previous process plan, it is unlikely that major changes will be needed in the corresponding fixture plan.
- The ability to reuse previous plans (and analysis results), as well as to control inter-module backtracking hinges primarily on keeping track of dependencies within the plans and between the plans, the specifications and the external constraints imposed by other modules. Interfaces which keep track of externally imposed constraints can thus play an important role in facilitating reuse. More generally, we found that it is important to keep issues of feasibility (constraints) separate from issues of optimality (costs) since the former are far more likely to remain valid from one plan iteration to the next.

Experience with our implementation makes us believe that hybrid architectures such as the one explored here offer a promising avenue of research for dealing with realistic planning domains.

References

[1] T.-C. Chang and R.A. Wysk. *An Introduction to Automated Process Planning Systems*. International Series in Industrial and Systems Engineering. Prentice-Hall, 1985.

[2] S. Kambhampati and J. Hendler, "A validation structure based theory of plan modification and reuse," *Artificial Intelligence*, Vol. 55, 1992.

[3] S. Kambhampati. Exploiting Causal Structure to Control Retrieval and Refitting during Plan Reuse *Computational Intelligence*, Vol. 10, No. 2, May 1994, pp 213-245.

[4] S. Kambhampati and M. R. Cutkosky, "An approach toward incremental and interactive planning for concurrent product and process design," in *Proceedings of ASME Winter Annual Meeting on Computer Based Approaches to Concurrent Engineering*, November 1990.

[5] S. Kambhampati and J. Tenenbaum, "Planning in concurrent domains," in DARPA *1990 Workshop on Innovative Approaches to Planning, Scheduling and Control*, 1990.

[6] S. Kambhampati, M. Cutkosky, J. M. Tenenbaum, and S. Lee Combining specialized reasoners and general purpose planners: A case study. In *Proc. of 9th AAAI*, 1991.

[7] S. H. Lee, M. R. Cutkosky, and S. Kambhampati, "Incremental and interactive geometric reasoning for fixture and process planning," in *Issues in Design/Manufacture Integration 1991* (A. Sharon, ed.), pages 7--13, ASME Winter Annual Meeting, ASME, December 1991.

[8] S. Kambhampati, M. Cutkosky, J. M. Tenenbaum, and S. Lee. Integrating general purpose planners and specialized reasoners: case study of a hybrid planning architecture. IEEE Trans. on Systems, Man and Cybernetics (Special section on planning, scheduling and control). Vol. 23, No. 6, November 1993.

[9] S. Kambhampati, C. Knoblock and Q. Yang. Planning as Refinement Search: A Unified framework for evaluating design tradeoffs in partial order planning. ASU-CSE-TR 94-002. To appear in *Artificial Intelligence* special issue on Planning and Scheduling. 1995.

[10] S. Kambhampati. A comparative analysis of partial order planning and task reduction planning SIGART Bulletin, Special Section on evaluating plans, planners and planning agents. Vol. 6, No. 1, January 1995.

[11] E. Sacerdoti, *A Structure for Plans and Behavior*. New York: Elsevier North-Holland, 1977.

[12] A. Tate, "Generating project networks," in *Proceedings of 5th International Joint Conference on Artificial Intelligence*, pages 888--893, 1977.

[13] D. Wilkins, "Domain independent planning: Representation and plan generation," *Artificial Intelligence*, vol. 22, pages 269--301, 1984.

[8] Of course, the usual tradeoff holds between the delay of commitment and the amount of computation needed to check the consistency of the plan. Thus, in the example of the fixturing specialist, to avoid extensive geometric simulation, a single total-ordering of the setups is ultimately chosen for detailed fixture planning.