SRC TR 87-106

Knowledge Representation and Reasoning Techniques for Process Planning: Extending SIPS to do Tool Selection

by

D.S. Nau and M. Luce

Knowledge Representation and Reasoning Techniques for Process Planning: Extending SIPS to do Tool Selection

Dana S. Nau[1]
Associate Professor
Computer Science Dept., and
Institute for Advanced Computer Studies
University of Maryland
College Park, Maryland 20742

Mark Luce
Manufacturing Engineer
Texas Instruments
Dallas, Texas 75266

SIPS (Semi-Intelligent Process Selector) was originally developed to use AI techniques for generative process selection. SIPS considers a metal part to be a collection of machinable features—and for each feature, it generates a sequence of machining process to use in creating that feature. It does this by reasoning about the intrinsic capabilities of each manufacturing operation.

SIPS is being extended to do handle other process planning tasks. This paper gives an overview of SIPS, and describes recent extensions which enable SIPS not only to do process selection, but also to select cutting tools and calculate process parameters such as feed rates and cutting speeds.

1 Introduction

One problem facing modern industry is the lack of a skilled labor force to produce machined parts as has done in the past. In the near future, this problem may become acute for a number of manufacturing tasks. One such task is process planning. Since process planning requires intelligent reasoning and considerable experiential knowledge, almost all existing computer aided process planning systems require a significant amount of supervision by experienced human beings.

In most process planning systems their are two types of planning activities, global planning and detailed planning. Global planning is performed by a process engineer and includes a plan for a part throughout a manufacturing facility. Detailed planning is performed by a n/c programmer and includes a plan for a part on a specific machine in the facility.

Given a part to be produced, the process engineer determines what machines are capable of producing the desired part, and determines what types of machining operations to use in creating the part. The instructions produced by the process engineer generally refer to the class of machining process to be used (e.g., "mill this face") rather than the exact machining process to use (e.g., "rough-end-mill this face"). Occasionally, the process engineer may suggest more of the process details, such as feed rates and cutting speeds, but these details are generally left up to the n/c programmer.

Given the process engineer's instructions, the n/c programmer performs detailed planning of a part at the machine level. This includes selecting the exact machining processes to be used, the tools, tool holders, and adapters to be used, and the feed rates and cutting speeds. Once he has produced this information, the n/c programmer uses it to produce the n/c program for the machine tool.

AI techniques can be used to automate (at least partially) several of the reasoning activities involved with process planning. One example of this is SIPS (Semi-Intelligent Process Selector), which uses AI techniques for generative selection of machining operations for the creation of metal parts. SIPS is being integrated into the process planning system in the Automated Manufacturing Research Facility (AMRF) project at the National Bureau of Standards (NBS) [1]. In addition, Texas Instruments is considering the possibility of using SIPS as an aid to n/c programmers, on an experimental basis.

As reported in [2,3], SIPS was originally developed to do process selection. SIPS considers a metal part to be a collection of machinable

[1]This work has been supported in part by the following sources: an NSF Presidential Young Investigator Award to Dana Nau, NSF Grant NSFD CDR-85-00108 to the University of Maryland Systems Research Center, IBM Research, General Motors Research Laboratories, Martin Marietta Laboratories, and the National Bureau of Standards.

features—and for each feature, it generates a sequence of machining process to use in creating that feature. It does this by reasoning about the intrinsic capabilities of each manufacturing operation. The process selection done by SIPS includes both the high-level process selection done by the process engineer (e.g., "mill this face") and the lower-level process selection done by the n/c programmer (e.g., "rough-end-mill this face").

The manufacturing engineer is faced with a number of activities that must be performed after each machining process is selected. Some of these activities include determining what cutting tools to use, and calculating process parameters such as feeds and speeds. As described in the current paper, we are extending SIPS to do these tasks. For future work, we plan to extend SIPS even further, so that it will be capable of doing machine selection, global optimization, and sophisticated reasoning about three-dimensional objects.

This paper gives an overview of SIPS, and describes the recent extensions of SIPS which allow it to select cutting tools and calculate process parameters. Section 2 describes process selection. Section 3 describes tool selection and the determination of feed rates and cutting speeds. Section 4 contains concluding remarks.

2 Process Selection

In most knowledge-based problem-solving systems, problem-solving knowledge consists of rules of the form "IF *conditions* THEN *action*". Even in frame systems, where the data (and possibly the knowledge base) are represented using frames, the knowledge base still usually consists of rules.

For process selection, there are several problems with representing the process knowledge in the form of rules. These problems are described in detail in [3,4]. To address these problems, SIPS uses a new approach to knowledge representation called *hierarchical knowledge clustering*, in which the knowledge about machining processes is organized in a taxonomic hierarchy. Each process in the hierarchy is represented by a frame.

For example, consider the frames shown in the simple hierarchy shown in Figure 1. The information in these frames is simpler than the information actually present in SIPS's knowledge base, but it gives an idea how SIPS represents process information.

The **relevant** slot in the **hole-process** frame specifies that a hole process is relevant for making a hole. This information is used to start SIPS's search when SIPS is told to plan the creation of a hole.

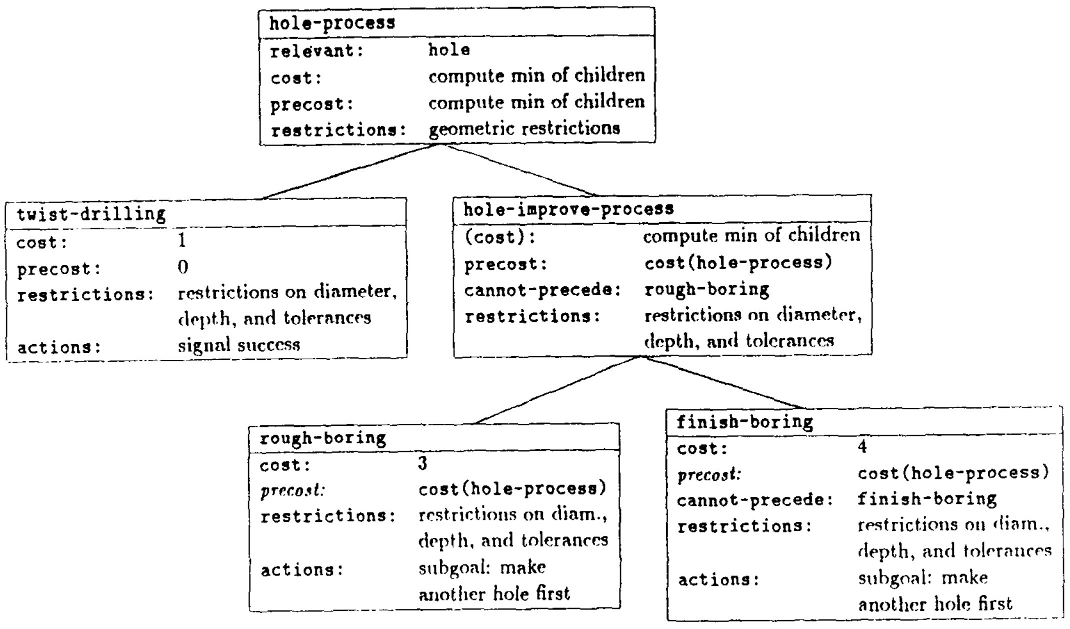

hole-process
relevant: hole
cost: compute min of children
precost: compute min of children
restrictions: geometric restrictions
twist-drilling
cost: 1
precost: 0
restrictions: restrictions on diameter, depth, and tolerances
actions: signal success
hole-improve-process
(cost): compute min of children
precost: cost(hole-process)
cannot-precede: rough-boring
restrictions: restrictions on diameter, depth, and tolerances
rough-boring
cost: 3
precost: cost(hole-process)
restrictions: restrictions on diam., depth, and tolerances
actions: subgoal: make another hole first
finish-boring
cost: 4
precost: cost(hole-process)
cannot-precede: finish-boring
restrictions: restrictions on diam., depth, and tolerances
actions: subgoal: make another hole first

Figure 1: A simple set of process frames. The use of italics for a slot name indicates that the slot is inherited from the frame's parent.

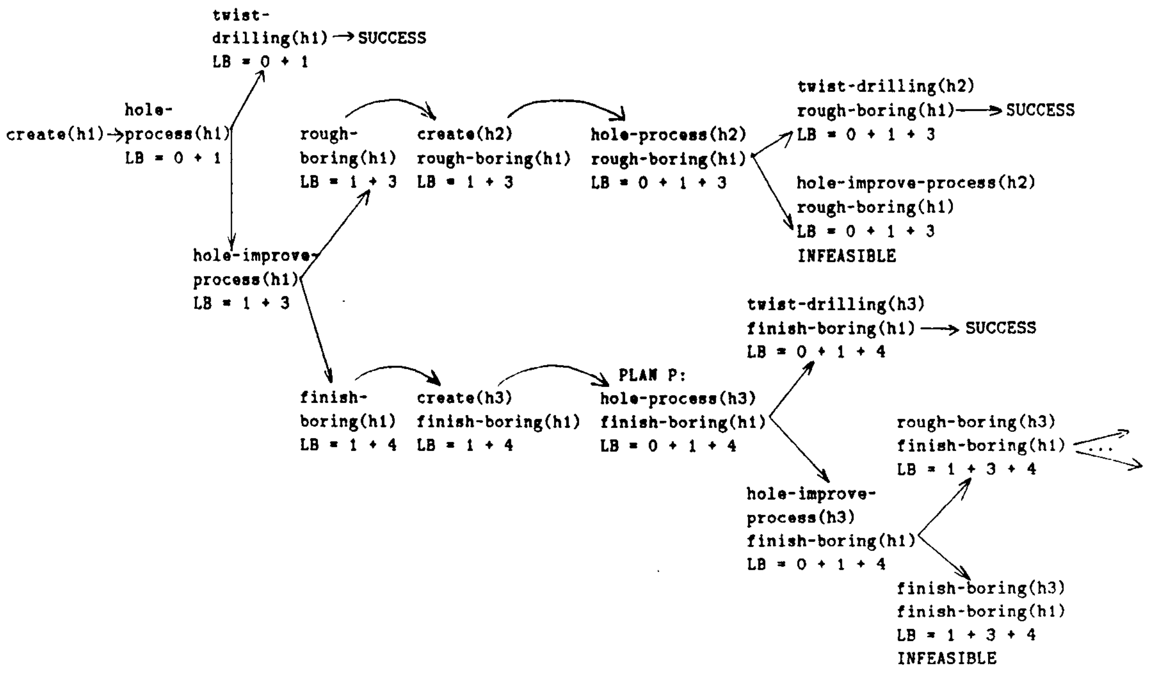
The cost slot is intended to be a lower bound on the cost of performing a process. In the case of hole-process, this lower bound is computed by an attached procedure which takes the minimum of the cost slots of the child frames. hole-improve-process inherits this procedure from hole-process, so its cost will also be computed as the minimum of the costs of its children. Since the twist-drilling, rough-boring, and finish-boring frames represent single kinds of machining processes rather than classes of machining processes, the relative costs of these processes are put into their cost slots.

Similarly, precost is intended to be a lower bound on the cost of any other processes which might be required before doing the hole process. For hole-process, this bound is computed by an attached procedure which computes the minimum of the precost slots of the children. Since twist-drilling does not need to have any other processes occur before it, its precost slot contains the value 0. But a hole improvement process takes an existing hole g and transforms it into the desired hole—and since g must be created by some kind of hole process, the cost of creating g will be at least the minimum cost for a hole process. Thus, the precost slot for hole-improve-process is the value of hole-process's cost slot. Both rough-boring and finish-boring inherit this value from hole-improve-process.

A process's restrictions slot tells what restrictions must be satisfied in order for that process to be a feasible way to achieve the desired goal. For hole-process, the restrictions are mainly geometric ones—for example, restrictions on the angle between the hole and the surface in which it is to be created. For the other processes in Figure 1, the restrictions are mainly restrictions on the hole dimensions and on the best machining tolerances achievable by the process (parallelism, roundness, true position, etc.).

The cannot-precede slots for hole-improve-process and finish-boring state that in no sensible process plan will these processes be followed by certain other machining processes. This slot is not really necessary for correct operation of SIPS, but it makes SIPS more efficient by decreasing the size of the search space.

SIPS does problem solving by searching backwards from the ultimate goal to be achieved. Therefore, the actions slot for a machining process must specify what SIPS needs to do *before* it can perform the machining process. For twist-drilling, nothing need be done beforehand—so twist-drilling's actions slot states that twist drilling succeeds immediately.



twist-drilling(h1) → SUCCESS
LB = 0 + 1
create(h1) → hole-process(h1)
LB = 0 + 1
hole-improve-process(h1)
LB = 1 + 3
rough-boring(h1)
LB = 1 + 3
create(h2) rough-boring(h1)
LB = 1 + 3
hole-process(h2) rough-boring(h1)
LB = 0 + 1 + 3
twist-drilling(h2) rough-boring(h1) → SUCCESS
LB = 0 + 1 + 3
hole-improve-process(h2) rough-boring(h1)
LB = 0 + 1 + 3
INFEASIBLE
finish-boring(h1)
LB = 1 + 4
create(h3) finish-boring(h1)
LB = 1 + 4
PLAN P:
hole-process(h3) finish-boring(h1)
LB = 0 + 1 + 4
twist-drilling(h3) finish-boring(h1) → SUCCESS
LB = 0 + 1 + 4
hole-improve-process(h3) finish-boring(h1)
LB = 0 + 1 + 4
rough-boring(h3) finish-boring(h1)
LB = 1 + 3 + 4
finish-boring(h3) finish-boring(h1)
LB = 1 + 3 + 4
INFEASIBLE

Figure 2: Part of a search space for creating a hole h1. Plan P is labeled for reference in the text.

However, rough boring and finish boring produce a better hole from a hole which must already exist—and SIPS needs to figure out how to make the hole that must already exist. The **actions** statements for **rough-boring** and **finish-boring** set up the creation of this hole as a subgoal for SIPS.

Figure 2 shows part of the state space which can be generated from the set of frames shown in Figure 1. Each state in the state space is a (partial) plan for creating a hole h1. Whether or not this plan is feasible will depend on the nature of h1—except that the plans marked "infeasible" in Figure 2 can never be feasible, because of the **cannot-precede** slots in the knowledge base. When a plan is infeasible, its children will never be generated.

SIPS searches the state space using an adaptation of Branch and Bound. The lower bound function LB which guides this search is computed from the **cost** and **precost** slots of the machining processes. For example, for the plan labeled P in Figure 2,

$$\begin{aligned}\mathrm{LB}(P) &= \texttt{precost(hole-process)} + \texttt{cost(hole-process)} \\ &\quad + \texttt{cost(finish-boring)} \\ &= 0 + 1 + 4 \\ &= 5.\end{aligned}$$

So that SIPS will avoid generating expensive plans when cheaper ones can be used, SIPS's search strategy is best-first. Thus, SIPS's search procedure may also be thought of as an adaptation of A* [5], with LB as the heuristic function. The first solution found by SIPS is guaranteed to be the least costly one.

3 Tool Selection

The selection of cutting tools is part of the detail planning of a part, and is performed after process selection has been done. Selecting a tool involves determining the correct material, size, and shape of the tool. This requires a great deal of information about both the machinable feature to be created, and the machining process to be used to create that feature.

Selecting the proper cutting tool material for a given machining operation is one of the more important factors involved in obtaining high production at low cost. The need for selecting the proper tool material is apparent when one recognizes that performance of an expensive n/c machine tool depends to a large measure on the capability of a very inexpensive cutting tool.

The industry provides a wide range of tool sizes and materials, for use by n/c programmers in manufacturing facilities. The process engineer must de-

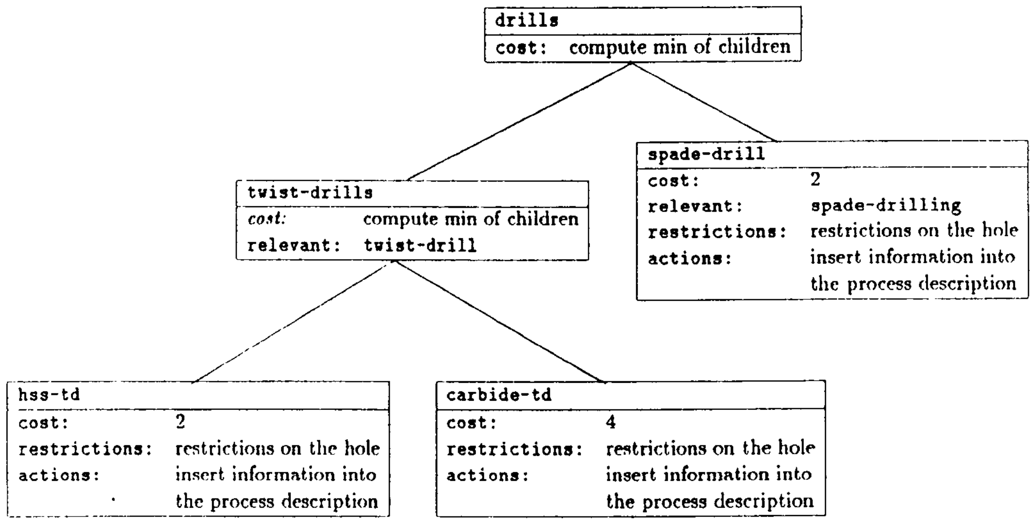

drills
cost: compute min of children
spade-drill
cost: 2
relevant: spade-drilling
restrictions: restrictions on the hole
actions: insert information into the process description
twist-drills
cost: compute min of children
relevant: twist-drill
hss-td
cost: 2
restrictions: restrictions on the hole
actions: insert information into the process description
carbide-td
cost: 4
restrictions: restrictions on the hole
actions: insert information into the process description

Figure 3: A simple set of tool frames. The use of italics for a slot name indicates that the slot is inherited from the frame's parent.

cide which combination of tool size and material will best produce the required finish stated by the design engineer. Too often, the same tool and material is used in a number of different machining processes that would yield better results if only the tool material were changed.

In order to extend SIPS to do tool selection, we have broken tool selection into three specific tasks: determining what type or family of cutting tools can successfully manufacture the part, determining a proper tool size to fit the constraints of the feature, and determining the tool material that can best produce the required finish.

Given a specific machining process, the scope of tool selection in SIPS is limited to identifying a specific family of tools that can be used for the process, and identifying the best tool material to use for the process. This approach does not satisfy all aspects of tool selection—but it does provide an initial approach to the problem, and that we believe it will lead to more effective use and selection of cutting tool materials.

Suppose SIPS is using the simple tooling knowledge base shown in Figure 3. This knowledge base is much simpler than the knowledge base SIPS actually uses for tool selection, but it will give the reader some idea what SIPS's tooling knowledge base is like.

The **relevant** slots in the **twist-drills** and **spade-drill** frames specify what kinds of machining processes these tools are relevant for. This information is used to start SIPS's search when SIPS is told to select a tool for some machining process.

The **cost** slot is intended to be a lower bound on the cost of a particular cutting tool. In the case of **drills**, this lower bound is computed by an attached procedure which computes the minimum of the **cost** slots of **twist-drills** and **spade-drill**. **twist-drills** inherits this procedure from **drills**, so its cost will be computed as the minimum of the costs of **hss-td** and **carbide-td** (high-speed steel and carbide drill bits, respectively). Since the **hss-td**, **carbide-td**, and **spade-drill** frames represent single kinds of cutting tools rather than classes of cutting tools, the relative costs of these tools are put directly into their **cost** slots.

A tool's **restrictions** slot tells what restrictions must be satisfied in order for that tool to be a feasible tool to use for a machining process. For each of the tool frames shown in Figure 3, the restrictions consist of diameter and depth restrictions on the hole (e.g., minimum and maximum diameters, and whether or not a cutting tool is available having the exact diameter required for the hole), as well as restrictions on the best surface finish which

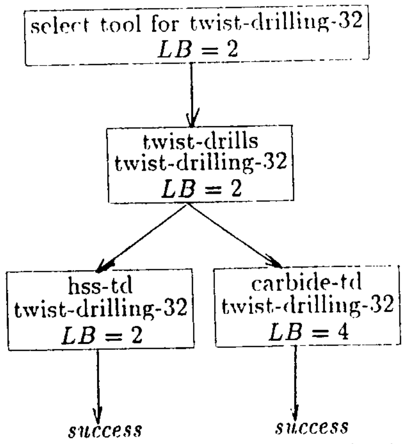


Figure 4: A search space for tool-selection for twist-drilling-32.

can be created using this type of tool material. In the future, we intend to extend these restrictions to include information about the type of material used in the workpiece.

In the process knowledge base in Figure 1, some of the **actions** slots specified subgoals to be achieved. For example, the **actions** slot for **rough-boring** specified that a hole had to be present before rough boring could be done. Such subgoals are not necessary in the tooling knowledge base: once it is determine that a particular cutting tool is feasible for use in a given machining process, tool selection is finished for that particular machining process. Thus, the actions slots for **hss-td**, **carbide-td**, and **spade-drill** do not contain any subgoals. Instead, they determine what cutting tool should be used, use textbook formulas to calculate the feed rate and cutting speed, and insert this information into the frame describing the machining process.

Consider the hole **h1** of Figure 2. Suppose that the tolerance requirements on this hole are such that twist drilling by itself is not capable of creating **h1**. Then the least costly sequence of operations capable of creating **h1** would be something like what is shown below, where **twist-drilling-32** and **rough-boring-30** are the specific instances of twist-drilling and rough-boring used to create **h1**.

1. use `twist-drilling-32` to create a hole called, say, `hole-10`.
2. use `rough-boring-30` to transform `hole-10` into `h1`.

Suppose we now tell SIPS to determine what cutting tool to use for `twist-drilling-32`. The data required to perform cutting tool selection for this process are the diameter, depth, and tolerance requirements for `hole-10`, and the fact that `twist-drilling-32` is an instance of the `twist-drilling` frame. The relevant data about `hole-10` were determined during process selection (for example, the diameter of `hole-10` is slightly smaller than the diameter of `h1`, since rough boring enlarges a hole).

In Figure 3, since `drills` is the only frame that is relevant for `twist-drilling`, it is where SIPS starts its search. Since the tooling knowledge base does not set up any subgoals, the search space SIPS uses to select a tool for `twist-drill-32` is quite simple. This search space is shown in Figure 4. Just as SIPS did for process selection, it performs a best-first search of the state space shown in Figure 4. Thus, the first tool SIPS finds that is capable of being used for `twist-drill-32` is the least costly tool feasible for `twist-drill-32`.

4 Discussion and Conclusions

4.1 Current State

We have found that the knowledge representation scheme in SIPS provides a natural way to represent, organize, and manipuate knowledge about machinable features, machining processes, and cutting tools. In a general sense, the search technique SIPS uses for problem solving is similar to the reasoning process that a manufacturing engineering goes through every time a part is considered for planning.

SIPS currently runs on Symbolics lisp machines and Texas Instruments Explorers in Zeta Lisp, and on Sun workstations in Franz Lisp. Its knowledge base has been considerably extended since the time of [2], and consists of more than 90 frames describing machinable features, machining processes, and cutting tools. The knowledge base is set up primarily to work on prismoidal parts having machinable features such as holes, pockets, and slots; but it is being extended to deal with lathe-turned parts as well.

Figures 5, 6, and 7 show the names of the frames SIPS currently has for features, processes, and tools, respectively. Of these three knowledge bases, the tooling knowledge base is the least com-

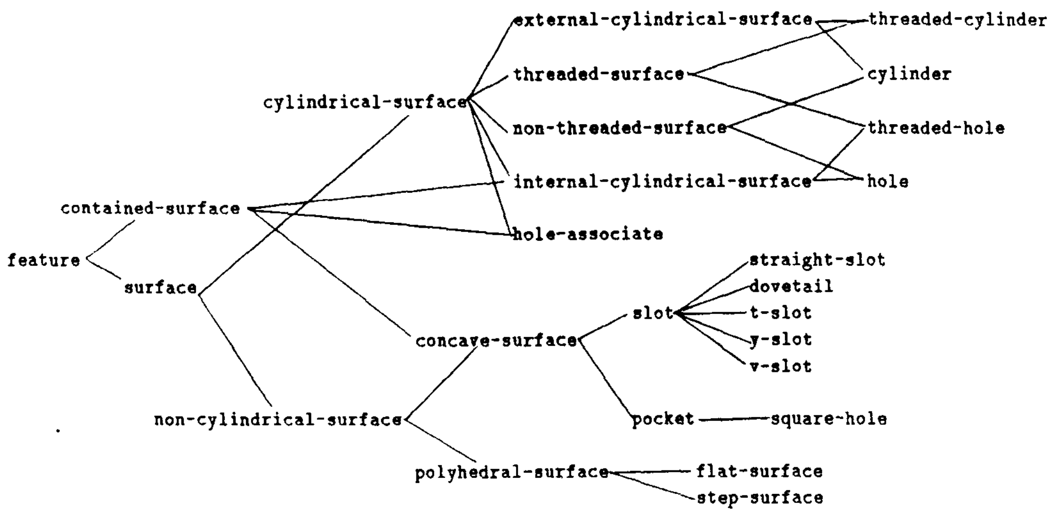


Figure 5: The names of the frames in SIPS's feature hierarchy.

plete. This knowledge base is still under development, and is currently much less complete than the feature and process knowledge bases.

SIPS can either read prepared data from a file, or (if some of this data is omitted) run interactively, asking the user for any needed information. Various user features have been implemented in SIPS. For example:

1. For any frame in SIPS's knowledge base, the user can print out information about that frame in whatever amount of detail is desired.

2. At the National Bureau of Standards, a tool has been implemented which can print out graphical represetations of the hierarchical structure of SIPS's knowledge base, and can print out the search tree being generated by SIPS as it is generated.

3. If SIPS produces a plan for producing some feature, the user can later tell SIPS to go back and find other alternative plans for producing this feature.

4. At General Motors Research Laboratories, SIPS has been interfaced to MBF, an expermental system which does solid modeling and human-supervised feature extraction. Thus, the user can construct a model of an object and identify its machinable features using MBF, and MBF will automatically send descriptions of these features to SIPS so that SIPS can decide what machining processes to use for them.

5. SIPS has been integrated with the interactive process planning system in the AMRF project at the National Bureau of Standards. Using this system, a user can look at a model of a part, retrieve machinable features from the part model, and choose a precedence ordering on them (i.e., which features should be machined before which others). At this point, SIPS will choose processes to use for each feature, transforming the precedence graph of features into a precedence graph of processes. The process information is then passed back to an n/c code generator, which produces the n/c code for creating the part.

4.2 Future Plans

SIPS's knowledge base is set up for the machining of wrought aluminum—a material type that is widely used in industry. Future work on SIPS will include extending it to deal with other material types as well. This will necessitate extending

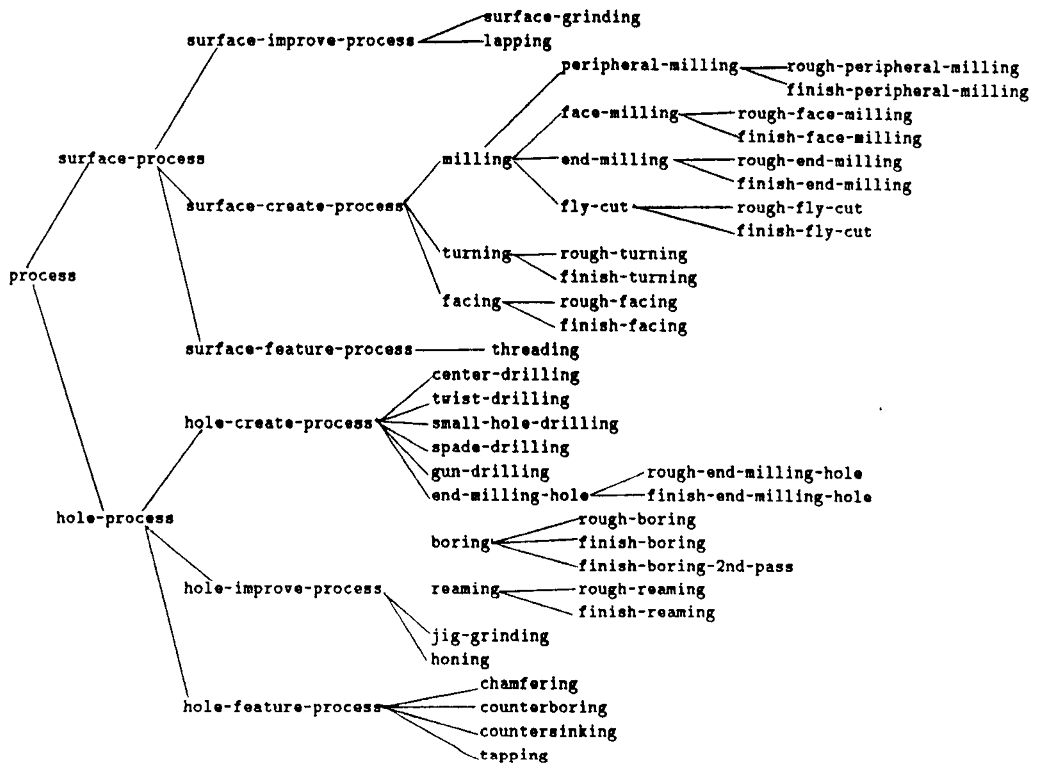


Figure 6: The names of the frames in SIPS's process hierarchy.

the tooling knowledge base to enable calculation of speeds and feeds for these materials.

SIPS currently does not do machine selection. Future plans include extending SIPS to select machines and machining processes in such a way as to minimize costs incurred by machine transfers, setups, and tool changes.

SIPS currently does not reason about the geometry of a part in a very sophisticated way—and thus it will fail to recognize, for example, that a hole cannot be created in a workpiece if some other part of the workpiece interferes with the tool trajectory. To handle this, we plan to build a solid modeler which will be thoroughly integrated with SIPS [6].

References

[1] P. F. Brown and C. R. McLean, "Interactive Process Planning in the AMRF," *Symposium on Knowledge-Based Expert Systems for Manufacturing at ASME Winter Annual Meeting*, Anaheim, CA, Dec. 1986, pp. 245-262.

[2] D. S. Nau and M. Gray, "SIPS: An Application of Hierarchical Knowledge Clustering to Process Planning," *Symposium on Integrated and Intelligent Manufacturing at ASME Winter Annual Meeting*, Anaheim, CA, Dec. 1986, pp. 219-225.

[3] D. S. Nau and M. Gray, "Hierarchical Knowledge Clustering: A New Representation for Problem-Solving Knowledge," *in* J. Hendler, *Expert Systems: The User Interface*, Ablex, 1987, to appear.

[4] D. S. Nau, "Hierarchical Abstraction for Process Planning" *Second Internat. Conf. Applications of Artificial Intelligence in Engineering*, 1987, to appear.

[5] N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga Press, Palo Alto, 1980, pp. 350-354.

[6] G. Vanecek and D. Nau, "Computing Geometric Boolean Operations by Input Directed Decomposition," Tech. Report, 1987.

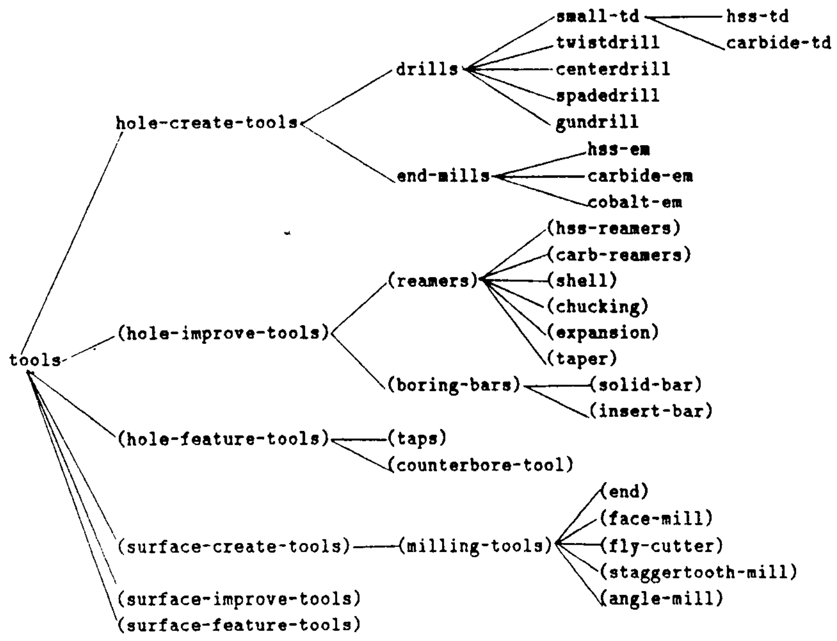


Figure 7: The names of the frames in SIPS's tool hierarchy. The names listed in parentheses have not been put into the hierarchy yet, but will be added soon.