of the potential for AI in industrial automation, but it at least presents a representative sampling of the work being done. The emphasis is on the utility of the ideas rather than the ideas themselves. Readers are presented an integrated view of the factory of the future, its technological demands, and real-world solutions made possible through the use of artificial intelligence techniques.

The Need for AI in Industrial Automation

AI as Advanced Computer Science

AI is sometimes jokingly defined as "that which we cannot do," thus forcing a continual redefinition of the field as it progresses. While sarcastic on the surface, this definition actually has merit, for while in theory AI researchers aim for human intelligence, in practice they must deal with the current limitations of computers and software, and thus AI can be conservatively viewed as advanced computer science.

The problem with today's programs is their inflexibility and inability to adapt. Design assumptions, original requirements, and the designer's problem analysis are forever lost in a forest of code with sparse and cryptic documentary annotations which become inaccessible after compilation. In essence, these programs are black boxes which hide their contents with dogged determination. While the design of such black boxes has become the goal of software engineering, this often leads to programs whose conclusions must be accepted on faith and whose operating assumptions are no longer visible.

AI techniques show promise of providing "glass-box" programs[5] which can explain their reasoning and assumptions to skeptical users so that it is more apparent when assumptions are invalid and the program requires maintenance. Such maintenance can also be facilitated if programs can explain what they know (their knowledge) and thus help programmers bring this knowledge up to date. AI programs can more easily explain their knowledge, assumptions, and reasoning because the knowledge is stored explicitly and the reasoning routines record justifications for conclusions. However, to assume that current rule-based systems meet these ideals of articulation would be wrong, although techniques being used in university projects concerned with explanation come much closer.[3]

Current computer programs are also too inflexible, in that they require carefully crafted input so they can perform sound mathematical transformations on this data. For a piece of software to survive in a dynamic environment, it must be able to deal with a wider variety of inputs and may even have to reason

Texas Instruments Announces the Winners of the AAAI-87 Call for Papers

IN July at AAAI-87, Texas Instruments announced the winners of its Call for Papers on the application of advanced artificial intelligence techniques to industrial automation problems. These papers are featured in this issue of the *TI Technical Journal*.

The Call for Papers, issued in April 1987, requested submissions from students and faculty of leading universities who are using AI to address critical problems in industrial automation. Entries were judged not only on the quality of the research, but also upon the participation of an industrial partner in the research project.

Six winners were selected from over 70 entries from 50 universities around the world. The winning paper, titled *Expert Systems without an Expert: Fault Diagnosis Based on Causal Reasoning*, was submitted by J. Douglass Whitehead and John Roach of Virginia Polytechnic Institute and State University (VPI). To further the authors' research, VPI will receive an Explorer™ II symbolic processing computer and a $10,000 fellowship.

VPI's research deals with the use of hypothetical reasoning and its diagnostic application to the lower hoist of a naval turret gun. The paper demonstrates techniques that can have broad application in the development of advanced diagnostic capabilities for advanced equipment. The paper was noted for the overall quality of exposition and the fact that it encompassed both cutting-edge AI research and direct practical implementation; the initial system was delivered to FMC, VPI's industrial partner, in fall 1986.

Five entries received honorable-mention awards. Each author's university will receive a $10,000 fellowship to support continued research on the project. The five recipients are as follows:

- Norman Hung-Chia Chang, Kuang-Kuo Lin, Chi-Yung Fu and David A. Hodges from the University of California at Berkeley for "BIPS (Berkeley Intelligent Processing System): Application for LPCVD Polysilicon," which was supported by a number of industrial partners including TI;
- Jerry E. Jones, Dawn R. White, Xu Xiaoshu, Paul A. Oberly and Tamara L. Funk from the Colorado School of Mines for "Development of an Off-Line Weld Planning System," with the support of the American Welding Society and the Welding Research Council;
- Michael Shaw of the University of Illinois for "Knowledge-Based Scheduling in Flexible Manufacturing Systems," in partnership with Cincinnati Milicron;
- Dana S. Nau of the University of Maryland for "Automated Process Planning Using Hierarchical Abstraction," in collaboration with the National Bureau of Standards and with support from TI's Defense Systems & Electronics Group;
- Rui J.P. deFigueiredo of Rice University for "A Framework for Automation of 3D Machine Vision," in support of NASA at the Johnson Space Center.

Automated Process Planning Using Hierarchical Abstraction*

Editor's note: Perhaps the most important aspect of this paper is the use of hierarchical abstraction. Most rule-based systems employ a non-stratified collection of rules. For complex problems like process planning, however, computational complexity can be avoided only through the use of varied levels of abstraction; that is, difficult problems must be solved at a high level and then the solution must be successively refined until it is completely solved.

The abstraction techniques described in this paper are basically taxonomic; each level of abstraction is seen as a specialization of some parent. Similar techniques are used in object-oriented programming languages, where each object inherits the properties of more abstract object classes. In fact, the current implementation of this work is on an Explorer™ LISP machine using the Flavors object-oriented extension to Common LISP. AI systems employing such a hierarchical organization of objects are often referred to as "frame based."

Planning has a rich history of research in the AI community. It has proven to be a very difficult problem due to the combinatorial explosion of possibilities. Dr. Nau briefly discusses related research in AI planning, and discusses the particularly difficult problem of handling the interaction between subgoals.

In process planning, this means that planning the machining steps to create individual features is rather straightforward, but that coordinating these machining steps into a globally optimal plan is more difficult. After discussing this problem, Dr. Nau briefly mentions the work of one of his Ph.D. students in this area.

It is significant that the system described in this paper is being applied to real problems in the National Bureau of Standards. In combinatorially explosive problems like planning, the real world provides an unforgiving test bed that can identify system weaknesses far better than more commonly used "toy problems."

ONE problem facing modern industry is the lack of a skilled labor force to produce machined parts as has been 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.

Process planning usually consists of 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. The instructions produced by the process engineer generally refer to the class of machining process to be used, rather than the exact machining process to use. Detailed planning is performed by an NC programmer and includes a plan for a part on a specific machine in the facility. Occasionally, the process engineer may suggest some of the process details, such as feed rates and cutting speeds, but these details are generally left up to the NC programmer.

AI techniques can aid in automating several of the reasoning activities required for process planning. For example, Semi-Intelligent Process Selector (SIPS) is a system which decides what machining operations to use in the creation of metal parts. SIPS considers a metal part to be a collection of machinable features — and for each feature, it generates a sequence of machining processes to use in creating that feature. It does this by reasoning about the intrinsic capabilities of each manufacturing operation. SIPS does both the high-level process selection normally done by a process engineer (e.g., "mill this face") and the lower-level process selection normally done by an NC programmer (e.g., "rough-end-mill this face").

SIPS was initially developed by Dr. Nau and his students at the University of Maryland.[8,9] For subsequent work on SIPS, Nau has collaborated with employees of the National Bureau of Standards,[1] General Motors Research Labs,[12] and Texas Instruments.[10]

The development of SIPS was done with two long-term goals in mind: the use of AI techniques to develop a practical, generative process-planning system, and the investigation of fundamental AI issues in representing and reasoning about three-dimensional objects.

*This work has been supported in part by the following sources: An NSF Presedential Young Investigator Award, NSF Grant NSFD CDR-85-00108 to the University of Maryland Systems Research Center, an IBM Faculty Development Award, General Motors Research Laboratories, and Texas Instruments.

Dana S. Nau

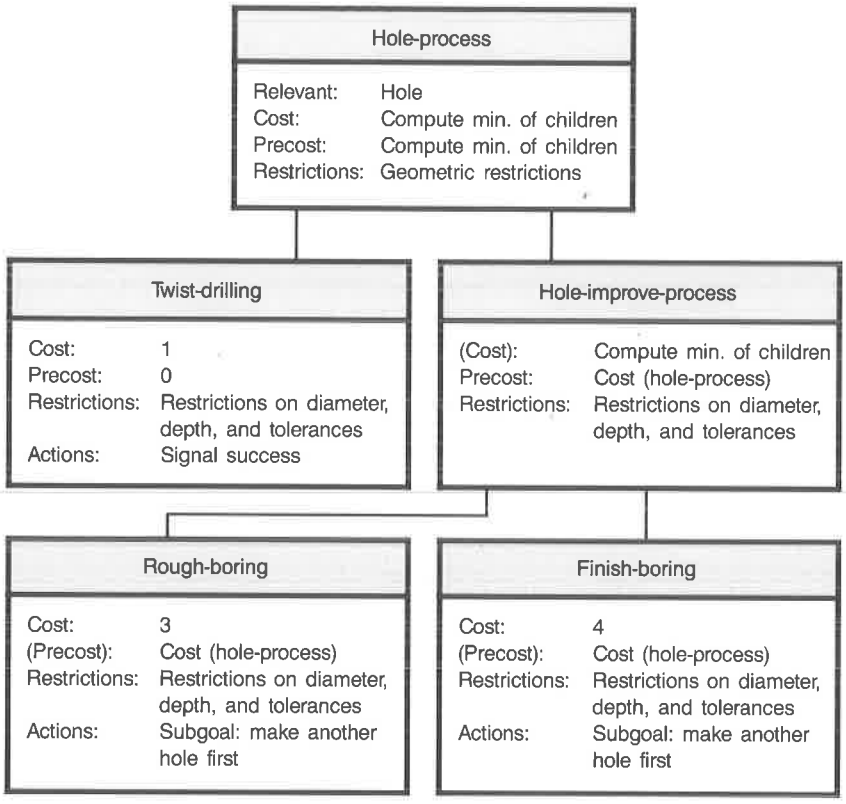

Figure 1. A Simple Set of Process Frames. If a slot name is written in parentheses, its value was inherited from the frame's parent.

SIPS represents an important step toward these goals, and a number of extensions and enhancements to SIPS are either underway or planned.

How SIPS Works

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 using rules (for details, see References 9 and 11). To overcome these problems, SIPS uses an approach to knowledge representation called hierarchical knowledge clustering, a hierarchical abstraction technique 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 simple knowledge base shown in Figure 1. This knowledge base is much simpler than the one SIPS actually uses, but it gives an idea how SIPS represents process information. (Information about SIPS' real knowledge base is shown in Figures 3 and 4.)

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' search when SIPS is told to make a plan for a hole.

The cost slot is intended to be a lower bound on the cost of performing a process. For hole-process and hole-improve-process, this lower bound is computed by taking the minimum of the cost slots of the child frames. The cost slots for twist-drilling, rough-boring, and finish-boring frames contain the relative costs of these machining processes.

Similarly, precost is intended to be a lower bound on the cost of any other processes which need to precede a process. For hole-process, this bound is found by computing the minimum of the precost slots of the children. Since twist-drilling is never preceded by other processes, its precost is 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 contains the value of hole-process's cost slot, and this value is inherited by rough-boring and finish-boring.

The restrictions slot tells what restrictions must be satisfied for a process to be feasible for achieving 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.).

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 to be done beforehand — so twist-drilling's actions slot states that twist drilling succeeds immediately. However, rough boring and finish boring produce a better hole from another hole which must already exist —

so the actions statements for rough-boring and finish-boring set up the subgoal of first creating the hole which must already exist.

Figure 2 shows 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 or complete) plan for creating a hole h1. Whether or not a plan is feasible will depend on the nature of h1, with the exception that the restrictions in the knowledge base guarantee that the plans marked "infeasible" in Figure 2 will never be feasible. When a plan is found to be infeasible, its children will never be generated.

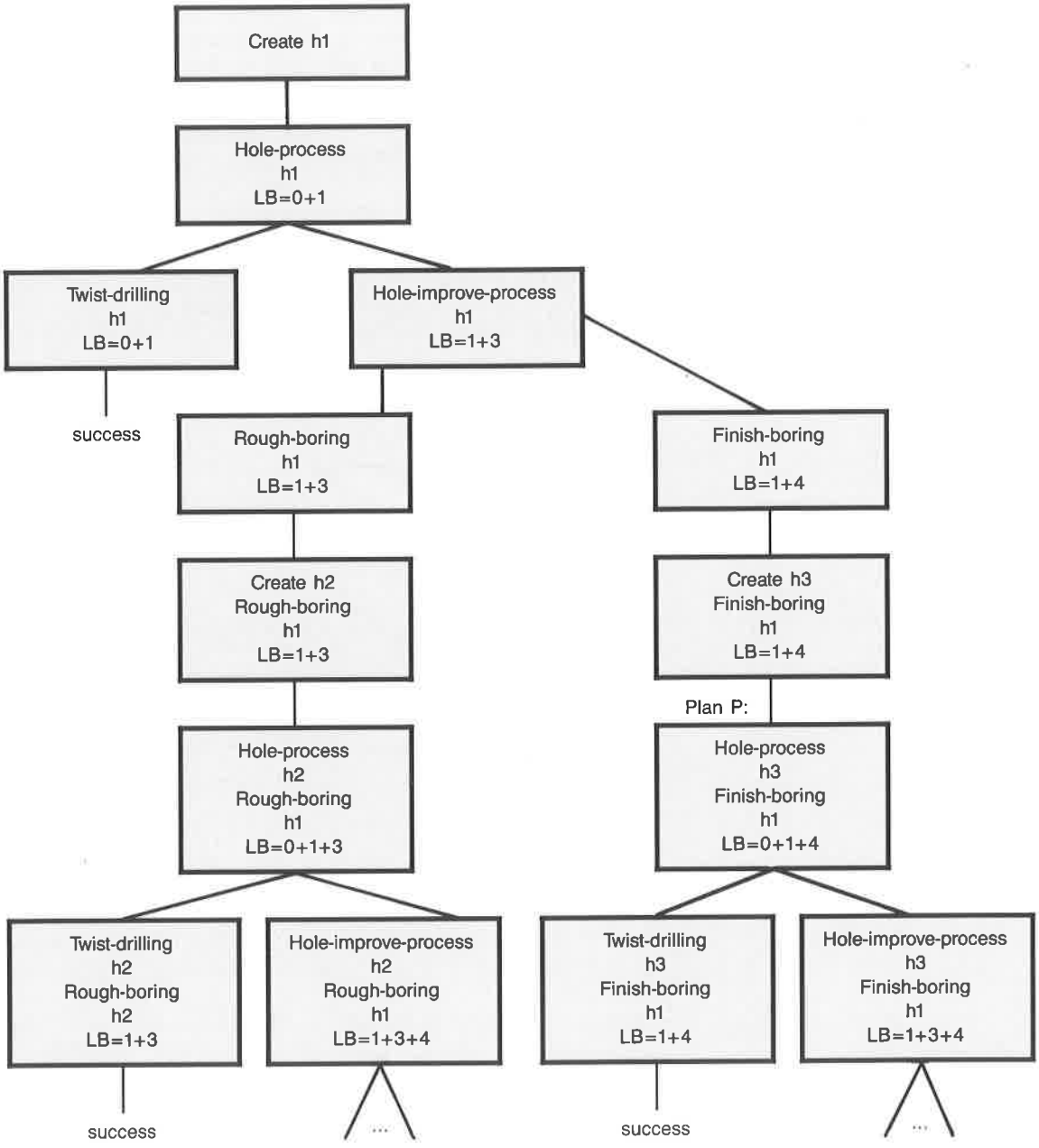

Figure 2. Part of a Search Space for Creating a Hole h1. Plan P is labeled for reference in the text.

SIPS searches the state space using a best-first Branch and Bound procedure.* 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,

LB(P) = precost (hole-process)
+ cost (hole-process)
+ cost (finish-boring)
=0+1+4
=5.

The first solution found by SIPS is guaranteed to be the least costly one.

Relation to Other Work

Process Planning

Computer systems for process planning can be classified into two types. Most commercially available systems are variant systems, in which the user retrieves from a data base a process plan for the desired part. Examples of such systems are CAPP[5] and MIPLAN.[19] For a more detailed discussion, see Reference 2.

A more ambitious approach is generative process planning, in which the computer system attempts to devise a complete process plan for the exact part desired. Since this task is very difficult, most generative process planning systems are still experimental and have limited capabilities.

Interest in generative process planning has increased markedly during the last two years; descriptions of various systems being developed can be found in References 15, 16, and 17. Although several of these systems use AI techniques, the techniques consist primarily of

SIPS' search procedure may also be thought of as an adaptation of A,[13] with LB as the heuristic function.

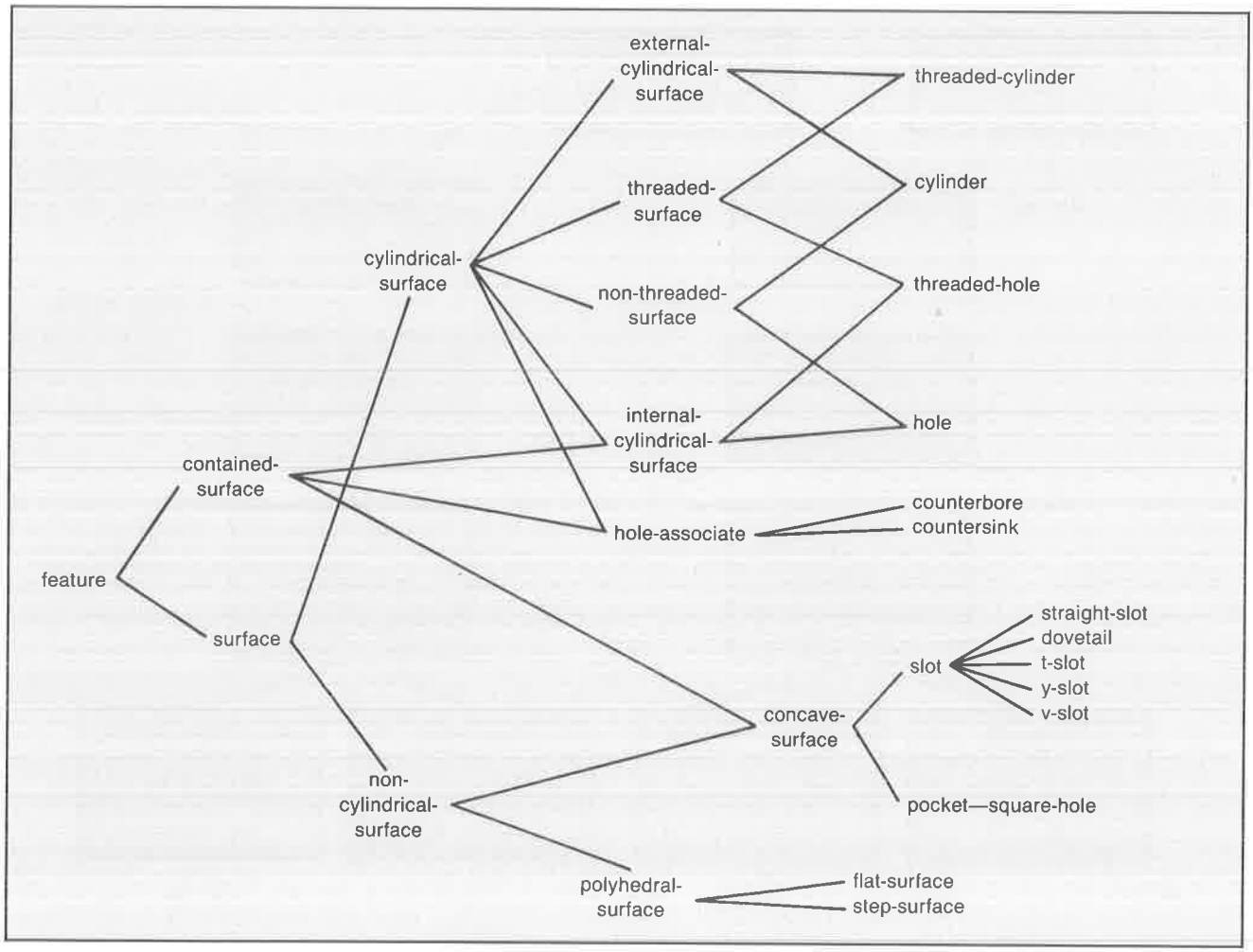

Figure 3. The Names of the Frames in SIPS' Feature Hierarchy

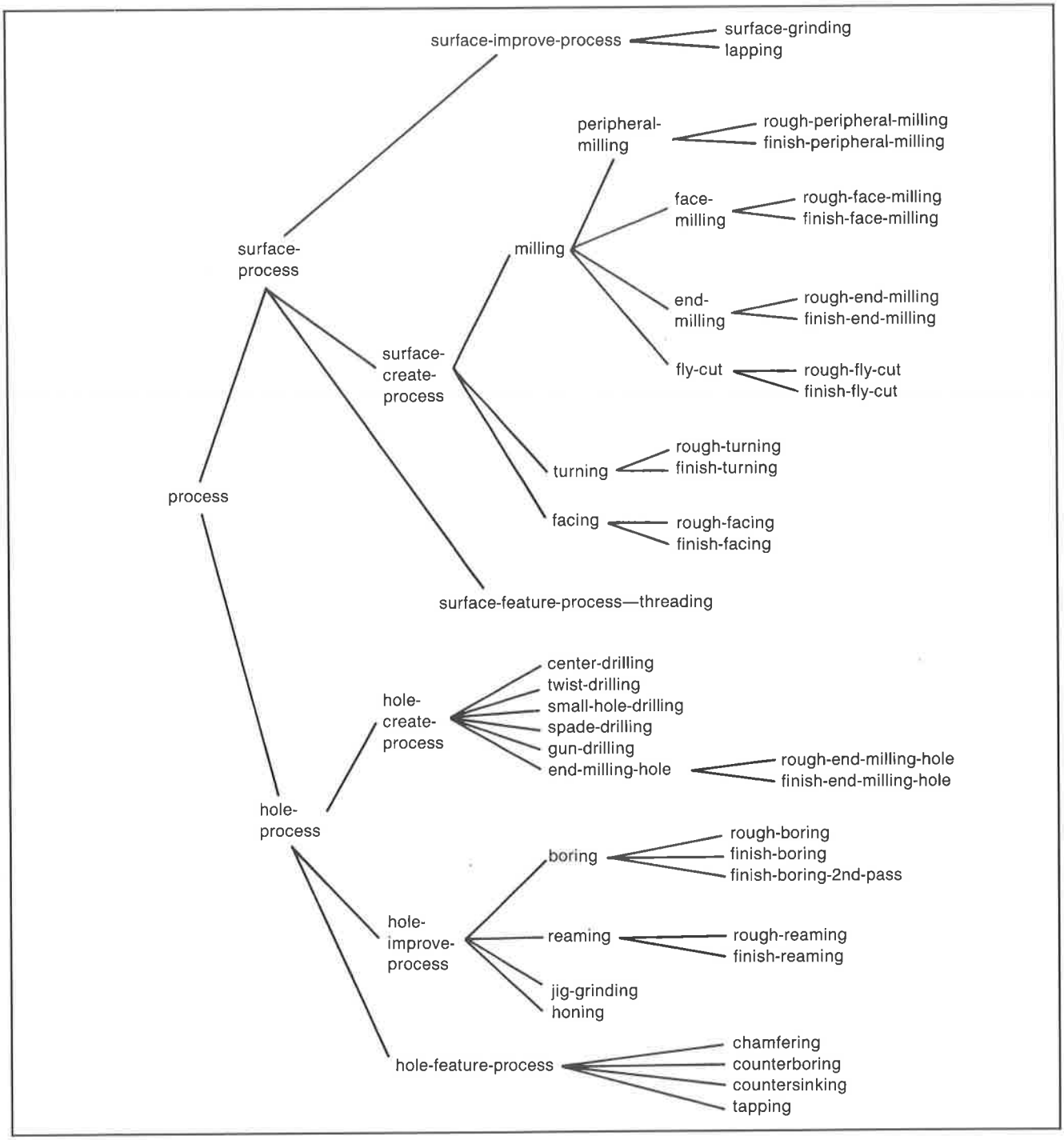

Figure 4. The Names of the Frames in SIPS' Process Hierarchy

conventional rule-based inference (one exception is SIPP,[7] a predecessor to SIPS).

Planning with Abstraction

Hierarchical knowledge clustering can be viewed as a way to do planning based on abstraction. For example, the hole-process frame in Figure 1 represents an abstract machining process which has two possible instantiations: twist-drilling and hole-improve-process.

Several types of abstraction have been explored in the literature on planning. For example, in NOAH,[14] an action A is an abstraction of actions A_1 and A_2 if A_1 and A_2 are alternate instantiations of A.

ABSTRIPS[13] uses another kind of abstraction: it constructs an abstract plan by ignoring some of the preconditions of each action, and then repeatedly modifies the plan to meet more and more of the preconditions which were ignored. Thus, in ABSTRIPS, an action A_1 is an instantiation of A if the preconditions of A are a proper subset of the preconditions of A_1. Conceptually, this is also true in SIPS — but there are some important differences. First, SIPS completely instantiates each action in a plan before considering what actions should precede this action, whereas ABSTRIPS generates a complete (but possibly incorrect) plan and then tries to fix it up. Second, an abstract action in SIPS has several possible alternate instantiations. However, in ABSTRIPS, only one instantiation is possible — there is no way to consider alternate instantiations of an action and choose the one of least estimated cost.

Tenenberg[18] proposes another type of abstraction which is quite close to that used in SIPS. This approach is similar to SIPS in the sense that each abstract action may have more than one possible instantiation. It is potentially more general than what is currently used in SIPS, in the sense that both the preconditions and the consequences of actions are abstracted — but so far, Tenenberg's approach has not yet been implemented.

Several systems for diagnostic problem-solving make use of certain kinds of taxonomic hierarchies. Both MDX[6] and Centaur[4] use taxonomies of various diagnostic problems, in which knowledge about each class of problems is located at the node in the hierarchy which represents that class. These approaches yield some of the same benefits as SIPS in terms of representational clarity and efficiency of problem-solving. However, since these systems do diagnosis rather than planning, they represent and manipulate their knowledge rather differently from SIPS.

Current State and Proposed Work

SIPS represents a step towards two long-term goals: the use of AI techniques to develop a practical generative process-planning system, and the investigation of fundamental AI issues in representing and reasoning about three-dimensional objects. This section describes the current state of SIPS, and proposes further work to be done.

Integration with the AMRF Project

SIPS currently runs on the Texas Instruments Explorer™ and Symbolics lisp machines in Zetalisp,™ and on Sun workstations and Silicon Graphics Iris workstations in Franz LISP.* As shown in Figures 3 and 4, SIPS' knowledge base currently contains nearly 80 frames describing machinable features and machining processes. A knowledge base for tooling is also under development.[10]

SIPS is being integrated into the Automated Manufacturing Research Facility (AMRF) project at the National Bureau of Standards.[1] Once this work is complete, the AMRF planning system will be able to send descriptions to SIPS about machinable features of various parts to be manufactured. SIPS will tell the planning system what machining operations to use for these features. This information will be passed to an NC code generator, which will produce the NC machine code for creating the part.

Mark Luce, an industrial engineer employed by Texas Instruments, has worked with SIPS during a one-year appointment at the National Bureau of Standards. Mark has found SIPS' knowledge representation scheme to be a natural way to represent not only machinable features and machining processes, but also cutting tools — and he has made significant enhancements to SIPS' knowledge base.[10]

Feature-based Modeling

In most existing solid modelers, the user constructs a solid model of an object by doing Boolean operations on various geometric primitives. But if the object is to be manufactured using machining operations, it is often necessary to express the object as a set of features (holes, pockets, etc.) which correspond more directly to manufacturing operations. This has inspired interest in feature-based modeling, in which the user builds a solid model of an object by specifying its machinable features directly. For example, one might start with a model of a piece of metal stock, and modify it by adding holes, slots, pockets, and other machinable features.

Nick Ide, a master's degree student, is building an interface between SIPS and PADL (a solid modeling system developed at the University of Rochester).[21] Once this is done, it will enable the user to specify a part as a set of machinable features, display the part using PADL, and select machining processes for it using SIPS.

*Certain commercial equipment, instruments, or materials are identified in this article in order to adequately specify the experimental procedure. Such identification does not imply recommendation or endorsement by the National Bureau of Standards, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

SIPS and MBF

One problem with feature-based modeling is that it requires the designer to determine the machinable features of an object — a task which is normally done by a process engineer. This is different enough from the traditional way of designing a part that it may not be immediately acceptable to industry. Another approach to extracting a part's machinable feature is provided by a system called MBF (Machining By Features).

MBF, which was built at General Motors Research Labs by Sarvajit Sinha and Jim Mildrew, is intended to be used by more than one user. The first user (presumably a designer) designs a part using ordinary solid modeling operations. MBF provides tools which allow a second user (presumably a process engineer) to identify the machinable features present in this part, for use in manufacturing.

During a summer appointment at General Motors Research Labs, Nau worked with Mildrew and Sinha to build an interface between SIPS and MBF. This interface allows MBF to send feature descriptions to SIPS, so that SIPS can decide what machining processes to use for these features.[12] Nau hopes further work can be done to develop and expand this interface.

Global Optimization

SIPS considers a part to be a collection of machinable features, and it selects machining operations for each feature separately. It considers local interactions among features (for example, the angle between a hole and the flat surface containing the hole), but does not consider non-local interactions (such as the relative location and orientation of two holes). Thus, the plan that SIPS produces for a single machinable feature is locally optimal (it is the least costly way to create that particular feature), but it is not necessarily globally optimal (it may not lead to the last costly way to create the entire part).

For example, consider an object containing two holes h1 and h2, both having the same diameter and the same machining tolerances. Suppose h1 can be created by either twist drilling or spade drilling. Then the least costly way to make h1 is twist drilling. If the depth of h2 is sufficiently large, h2 may require spade drilling rather than twist drilling. In this case, the cheapest way to make the entire object is to use spade drilling for both h1 and h2 in order to avoid a tool change — even though spade drilling would not be the cheapest way to make h1 if h1 were the only hole being made.

The problem described previously is basically a subgoal interaction problem. Chiang Yang, a Ph.D. student, is investigating the problem of finding the least-cost plan in the presence of such interactions. He has shown that the problem is NP-hard. However, he and Nau believe that efficient approximation algorithms can be developed (i.e., algorithms which will produce results that are close to optimal). This would enable SIPS to produce better process plans than it does currently.

Reasoning about Three-dimensional Objects

Currently, the way SIPS reasons about three-dimensional objects is rather primitive, because SIPS does not have adequate three-dimensional representations of these features. In order to handle problems such as the one described previously, SIPS must be thoroughly integrated with a solid modeler — not just by using the solid modeler as a "front end" to SIPS as described, but also by having SIPS communicate extensively with the modeler while it decides how to manufacture a part.

George Vanecek, a Ph.D. student, is developing a new way to represent three-dimensional objects which offers several computational benefits over currently known solid modeling techniques.[20] Vanecek has implemented this approach for two-dimensional polygons, and for his Ph.D. work he will build a three-dimensional solid modeler implementing this approach. This solid modeler will be implemented in LISP on a Texas Instruments Explorer, and will be fully integrated with SIPS.

Extending SIPS for Other Process Planning Tasks

SIPS can be extended to handle several process planning tasks other than just process selection. Current and proposed work in this area is described below.

Mark Luce, an industrial engineer employed by Texas Instruments, worked with SIPS during a one-year appointment at the National Bureau of Standards. During this time, he and Nau began to extend SIPS to do not only process selection, but also tool selection and the determination of process parameters such as feed rates and cutting speeds.[10] Luce and Nau hope to continue this work.

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

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.

The main benefit of this work is that it will allow knowledge about several different process planning tasks to be represented in a single, uniform format which will be natural and easy to understand. This knowledge will be used by a single integrated system, with a uniform user interface.

User Interface

Since the development of SIPS was a research activity, SIPS' user interface is still relatively unsophisticated. If SIPS is to become a practical tool for process planning, a better user interface will be very important.

As a first step, Steve Ray at the National Bureau of Standards has built a

graphic interface for SIPS which can be used to print out SIPS' search tree as it is being developed. This graphic interface can also print out the names of the frames in SIP's knowledge base, in a hierarchical structure similar to that shown in Figures 3 and 4. We intend to extend this interface into a general tool for displaying, editing, and creating SIP's process plans and knowledge bases graphically.

Re-implementation of SIPS in Common LISP, without Flavors

SIP's frame system was written in LISP, using Flavors. Since the goals and required capabilities for frame systems differ in several significant aspects from the goals and capabilities of an object-oriented programming systems such as Flavors, Flavors was not flexible enough to do everything that was desired. Thus, the use of Flavors turned out to be more of a hindrance than a help. For future work on SIPS, SIPS will be re-implemented without using Flavors.

The re-implementation of SIPS will be done in Common LISP, using a Texas Instruments Explorer. This will allow the knowledge representation scheme to be more powerful and flexible, which will make it easier to develop and modify knowledge bases for SIPS.

Conclusion

AI techniques can aid in automating several of the reasoning activities required for process planning. To explore this potential, we have developed SIPS, a knowledge-based system which does generative selection of machining operations for the creation of metal parts.

The proposed future work on SIPS is intended to address two goals: the development of SIPS into a practical, generative process-planning system using AI techniques, and the development of ways to represent and reason about three-dimensional objects. We believe this work can support future goals of Texas Instruments to implement automated process planning for metal fabrication manufacturing.

References

1. P. Brown and S. Ray, "Research Issues in Process Planning at the National Bureau of Standards," *Proc. 19th CIRP International Seminar on Manufacturing Systems*, June 1987, pp. 111-119.
2. T.C. Chang and R.A. Wysk, *An Introduction to Automated Process Planning Systems*, Prentice-Hall, Englewood Cliffs, N.J., 1985.
3. M.S. Dunn, Jr., et al., "Implementation of Computerized Production Process Planning," Report R81-945220-14, United Technologies Research Center, East Hartford, Conn., June 1981.
4. P. Jackson, *Introduction to Expert Systems* Addison-Wesley, Wokingham, England, 1986, pp. 142-157.
5. C.H. Link, "CAPP—CAM-I Automated Process Planning System," *Proc. 13th Numerical Control Society Annual Meeting and Technical Conference*, Cincinnati, March 1976.
6. S. Mittal, et al., "Overview of MDX—A System for Medical Diagnosis," *Proc. Third Annual Symposium on Computer Applications in Medical Care*, Washington, D.C., October 1979.
7. D.S. Nau and T.C. Chang, "Hierarchical Representation of Problem-Solving Knowledge in a Frame-Based Process Planning System," *Jour. Intelligent Systems* 1:1, 1986, pp. 29-44.
8. 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, Calif., December 1986, pp. 219-225.
9. D.S. Nau and M. Gray, "Hierarchical Knowledge Clustering: A Way to Represent and Use Problem-Solving Knowledge," in J. Hendler, *Expert Systems: The User Interface*, Ablex, 1987, to appear.
10. D.S. Nau and M. Luce, "Knowledge Representation and Reasoning Techniques for Process Planning: Extending SIPS to do Tool Selection," *Proc. 19th CIRP International Seminar on Manufacturing Systems*, June 1987, pp. 91-98.
11. D.S. Nau, "Hierarchical Abstraction for Process Planning," *Second Internat. Conf. Applications of Artificial Intelligence in Engineering*, 1987, to appear.
12. D.S. Nau and S. Sinha, "MBF, X-Solid and SIPS: Solid Modelilng and AI for Process Selection," in preparation, 1987.
13. N.J. Nilsson, *Principles of Artificial Intelligence*, Tioga Press, Palo Alto, Calif., 1980, pp. 350-354.
14. E. Sacerdoti, *A Structure for Plans and Behavior*, American Elsevier, New York, 1977.
15. *Proc. 19th CIRP International Seminar on Manufacturing Systems*, University Park, Penn., June 1987.
16. *Proc. Production Engineering Conference at ASME Winter Annual Meeting*, Miami Beach, December 1985.
17. *Proc. Symposium on Integrated and Intelligent Manufacturing at ASME Winter Annual Meeting*, Anaheim, Calif., December 1986.
18. J. Tenenberg, "Planning with Abstraction," *Proc. AAAI-86*, Philadelphia, 1986, pp. 76-80.
19. TNO, "Introduction to MIPLAN," Organization for Industrial Research, Inc., Waltham, Mass., 1981.
20. G. Vanecek and D. Nau, "Computing Geometric Boolean Operations by Input Directed Decomposition," Tech. Report, 1987.
21. H.B. Voelcker, et al., "The PADL-1.0/2 System for Defining and Displaying Solid Objects," *ACM Computer Graphics* 12:3, August 1978, pp. 257-263.