# A KNOWLEDGE-BASED APPROACH TO GENERATIVE PROCESS PLANNING

Dana S. Nau[1]
Computer Science Dept.
University of Maryland
College Park, MD 20742

Tien-Chien Chang
School of Industrial Engr.
Purdue University
West Lafayette, IN 47907

## ABSTRACT

This paper describes a generative process planning system which produces plans for the creation of metal parts using metal removal operations. The system makes use of knowledge-based reasoning techniques from Artificial Intelligence to make process plans completely from scratch, using only the specification of the part to be produced and knowledge about the intrinsic capabilities of each manufacturing operation.

The system, which is written in Prolog, incorporates a knowledge representation language written in Prolog, a frame-based hierarchical representation of the problem solving knowledge, and a kind of best-first Branch and Bound strategy which for finding process plans of least possible cost.

The paper gives an overview of the system, discusses its good and bad points, and describes implications for further research.

## 1. INTRODUCTION

This paper describes a generative process planning system called SIPP (Semi-Intelligent Process Planner), which uses Artificial Intelligence (AI) techniques to produce process plans for the creation of metal parts using metal removal operations. In contrast to existing conventional approaches to generative process planning, SIPP uses knowledge-based reasoning techniques to make process plans completely from scratch, using only the specification of the part to be produced and knowledge about the capabilities of each manufacturing operation.

SIPP consists of (1) a frame-based knowledge representation system, (2) a knowledge base (written using the frame system) which contains information about the

[1]This work was supported in part by Naval Surface Weapons Center Contract N60921-82-C-0083 and National Bureau of Standards Contract NB83SBCA2124 to the University of Maryland, and by a Presidential Young Investigator Award to Dana S. Nau, including matching funds from IBM, General Motors, and Martin Marietta.

characteristics of various kinds of machinable surfaces and the capabilities of various machining processes, and (3) a control structure which manipulates the knowledge base in order to construct process plans. SIPP has the following unusual features:

(1) In most frame-based approaches to knowledge representation and reasoning, the problem-solving knowledge that manipulates the frames is stored separately in the form of production rules. In SIPP, the problem-solving knowledge is represented not as rules, but hierarchically in the form of frames.

(2) Rather than the depth-first forward-chaining or backward-chaining schemes used in many knowledge-based computer systems, SIPP uses a best-first search strategy based on Branch-and-Bound. This approach produces process plans which have the least possible cost (according to whatever cost criterion the user desires).

(3) The entire system is written in Prolog.

## 2. BACKGROUND

*Process planning* (also called manufacturing planning or material processing) is the task of determining what machining processes and parameters are to be used in manufacturing a part. Process planning is distinct from *production planning*, which involves taking the process plans for all parts to be produced and scheduling the factory's resources to perform the activities specified by the process plans.

Devising a process plan automatically based on a part's specifications is very difficult. However, the substantial benefits to be gained from automated process planning have led to substantial research in computer-aided process planning.

In several existing computer-aided process planning systems, a Group Technology (GT) code [4] is used to classify a part as being in a family of similar parts. When a process plan for a part is desired, a human user enters the GT code for the part into the system, and the system retrieves a process plan which was previously used for some part in this family. The process plan is then modified by the user to produce a plan for the part. Examples of such systems are CAPP [9] and MIPLAN [16].

Such systems are quite useful in industry, as they allow process plans to be made very quickly. However, if process planning is to be automated more fully, it will be necessary for the computer system to have more complete information about the part than just a GT code. In particular, detailed information must be available about each surface to be machined.

A few generative process planning systems have been developed experimentally which use as input a represention of each of the machinable surfaces of the part.[2] Examples include CPPP [10] [7], APPAS [18], CADCAM [2] [3], and TIPPS [5]. Due to the difficulty of truly generative process planning, each of these systems has various limitations on the kinds of parts for which it can produce process plans.

More sophisticated approaches to generative process planning will require sophisticated representational and reasoning techniques [12] [14]. The use of AI rule-based reasoning is being tried experimentally in Gari [6] and TOM [11]. Another system [17] is said to use some elementary expert system techniques, but the paper describing it does not give any further details. Each of these systems has various limitations on the kinds of process plans it can produce.

---

[2] By a *machinable surface* we mean any geometric surface or combination of geometric surfaces which can be produced by a single machine tool operation. For example, a hole is a single machined surface, although it consists of both a cylinder and a cone. For more information on machining operations, see [1], Chapter 1.

## 3. KNOWLEDGE REPRESENTATION

In order to represent knowledge, SIPP includes a frame manipulation language which was written in Prolog. The frame manipulation language allows for two types of frames: *items*, which represent various objects, and *types*, which represent sets of objects. SIPP's knowledge base of machining knowledge consists of types, and the data that it manipulates to solve each specific process planning problem consists of items. Each type or item is defined by a frame.

Items and types contain various slots which may contain various values. For example, the frames below are similar to frames used in SIPP's current knowledge base:

```
type(surface, thing).
slots(surface, [
        [surface_finish, X, number(X)],
        [contains, X, list_of_atoms(X)], /* all surfaces which this one contains */
        [adjacent, X, list_of_atoms(X)], /* all surfaces adjacent to this one */
        [comment, X, true]
]).
defaultvals(surface, [
        adjacent = [ ],
        contains = [ ],
        comment = 'this is an arbitrary surface'
]).

type(flat_surface, surface).
slots(flat_surface, [
        [norm, [X,Y,Z], (number(X),number(Y),number(Z))],
        [flatness, X, number(X)],
        [angularity, X, number(X)],
        [parallelism, X, number(X)],
        [boundaries, X, list_of_atoms(X)]
]).
defaultvals(flat_surface, [
        comment = 'this is a flat surface'
]).

item(f, flat_surface).
slotvals(f, [
        norm=[1,0,0],
        flatness=0.1,
        angularity=0.1,
        parallelism=0.1,
        pos_tolerance=0.1,
        neg_tolerance=0.1,
        surface_finish=100,
        adjacent=[f2,f3,f4,f5]
]).
```

The above frames describe a hierarchical structure in which *f* is a *flat_surface*, a *flat_surface* is a *surface*, and a *surface* is a *thing*.

The "slots" entries for *surface* and *flat_surface* specify what slots these two objects have, as well as what kinds of data values are permissible to be stored in these slots. Slot declarations are inherited from above; for example, since a *flat_surface* is a *surface*, a *flat_surface* has a slot called *adjacent*.

The "defaultvals" entries for *surface* and *flat_surface* specify default values for various of the slots. Any time a default value is specified for a type, it may be overridden at lower levels of the hierarchy. For example, the flat surface f1 has the following slot values:

contains=[ ] (inherited from *surface*);

comment='this is a flat surface', (inherited from *flat_surface*);

adjacent=[f2,f3,f4,f5] (specified explicitly for f1).

In most AI systems that use frames, the frames are manipulated by production rules of the form "IF *condition* THEN *action*". However, this approach seems rather unwieldy for process planning. For example, if two different kinds of drilling operations are represented using production rules, the preconditions of these two rules may consist of a dozen or so tests, most of which are exactly the same for both rules. Such situations can occur quite often in a knowledge base about the capabilities of manufacturing processes--thus leading both to unnaturalness in the representation of the problem-solving knowledge and inefficiency in its application to a manufacturing problem.

SIPP's frame system uses a different way of representing proble-solving knowledge. Instead of using production rules, the problem-solving knowledge is represented hierarchically within the frame system, as illustrated below.

```
type(hole_process,process).
relevant(hole_process,hole).
defaultvals(hole_process, [cost=1]).  /* at least the cost of twist_drill */
restrictions(hole_process, H) :- . . .  various geometric restrictions

type(hole_improve_process, hole_process).
defaultvals(hole_improve_process, [
        precedence=20,
        projected_cost=1,  /* at least the cost of twist_drill */
        cost=3  /* at least the cost of rough_bore */
]).
restrictions(hole_improve_process, H) :- H?special_features eq none.

type(bore, hole_improve_process).
defaultvals(bore, [cost=3, precedence=22]).
restrictions(bore, H) :- . . .  various tolerance restrictions

type(rough_bore, bore).
restrictions(rough_bore, H) :-
        H?pos_tolerance gte 0.002,
        H?neg_tolerance gte 0.002.
actions(rough_bore, P, H) :-
        copy_item(H,G),
        G:diameter  gets H?diameter - 0.005,
        G:pos_tolerance gets 1,
        G:neg_tolerance gets 1,
        G:straightness gets 1,
        G:roundness gets 1,
        G:parallelism gets 1,
        G:true_position gets 1,
        G:surface_finish gets 125,
        subgoal(P,G).
```

In this example, *rough_bore* is a subtype of *bore*, which is a subtype of *hole_improve_process*, which is a subtype of *hole_process*. This means that *rough_bore*

is applicable for creating a particular surface H only if the restrictions for *hole_process*, *hole_improve_process*, *bore*, and *rough_bore* are all satisfied for H.

The above example also illustrates how the frame manipulation language allows the user to retrieve and modify slot values. In the example, "eq" and "gte" are similar to Prolog's "=" and ">=" functions, respectively, except that they interpret constructions such as "H:pos_tolerance" and "H?pos_tolerance" as references to the values of slots in the item H. The difference between "H:pos_tolerance" and "H?pos_tolerance" is that if the *pos_tolerance* slot for H does not have a value, "H:pos_tolerance" causes an error but "H?pos_tolerance" asks the user for the value. "gets" is an assignment operator; for example, "H:pos_tolerance gets 1" puts the value 1 into the *pos_tolerance* slot in the item H.

## 4. CONTROL STRATEGY

SIPP's control strategy is a least-cost-first search based on Branch-and-Bound techniques. SIPP creates a process plan for an object by creating process plans for each of its machinable surfaces. If SIPP is trying to create some surface *s*, it starts by looking at the process types which are flagged as being "relevant" for creating whatever type of surface *s* happens to be. If the restrictions on this process type are satisfied, then its subtypes are eligible for consideration as possible ways to make *s*. No process type is eligible for consideration unless the restrictions for the process type directly above it in the hierarchy have been tested and found to be satisfied. Among the various process types which are eligible for consideration, the one chosen next for consideration is the one which SIPP believes will lead to the least costly process plan.

Suppose SIPP considers some process finds that its restrictions are satisfied. If that process has actions associated with it, then the actions are performed. To continue the example above, *rough_bore* is an operation which requires that a hole G already be present before *rough_bore* is performed--but since the purpose of rough boring is to improve the characteristics of a hole, the tolerance requirements for G are not as strict as those for the hole H which will be there after *rough_bore* finishes. The actions for *rough_bore* set up--as a subgoal--the creation of G. The Branch-and-Bound procedure considers the various possible ways to create H along with all the other possible ways to create H--and at each point, the procedure looks next at whichever possibility it believes will lead to the least costly process plan for H. Because of the way in which this is done, the first successful process plan found by SIPP is guaranteed to have the least cost of any process plan for creating H.

## 5. CONCLUSIONS

SIPP is currently up and running as a prototype system whose knowledge base contains about 55 frames. Information about the implementation details is available in [13]. SIPP can either read prepared data from a file, or (if some or all of this data is omitted) run interactively, asking the user for any needed information. Various user features have been implemented--such as the ability to go back and produce other process plans for a machinable surface if the user wants to see alternatives to the first process plan the system produces.

Currently, SIPP's reasoning capabilities about tolerance requirements are quite sophisticated--but (as with other existing process planning sytems) its geometric reasoning capabilities are limited. The addition of sophisticate geometric reasoning capabilities is a major future goal, but one which will require that a solid modeling system be incorporated into SIPP. We are already doing research along this line [15] [8]. Other goals include integrating SIPP with some kind of computer-aided design system, and giving it the ability to decide upon process details such as tool paths, feed rates, and cutting speeds.

## REFERENCES


[1] Boothroyd, G., *Fundamentals of Metal Machining and Machine Tools,* Scripta, Washington, DC, 1975.

[2] Chang, T. C., Interfacing CAD and CAM - A Study of Hole Design, M.S. Thesis, Virginia Polytechnic Institute, 1980.

[3] Chang, T. C. and Wysk, R. A., An Integrated CAD/Automated Process Planning System, *AIIE Transactions* **13**, 3, Sept. 1981.

[4] Chang, T. C. and Wysk, R. A., *An Introduction to Automated Process Planning Systems,* Prentice-Hall, Englewood Cliffs, NJ, 1985.

[5] Chang, T.C. and Wysk, R. A., Integrating CAD and CAM through Automated Process Planning, *International Journal of Production Research* **22**, 5, 1985.

[6] Descotte, Y. and Latombe, J. C., GARI: A Problem Solver that Plans How to Machine Mechanical Parts, Proc. Seventh International Joint Conf. Artif. Intel., Aug. 1981.

[7] Dunn, M. S. Jr., Bertelsen, J. D., Rothauser, C. H., Strickland, W. S., and Millsop, A. C., Implementation of Computerized Production Process Planning, Report R81-945220-14, United Technologies Research Center, East Hartford, CT, June 1981.

[8] Joshi, S. and Chang, T. C., Feature Extraction and Recognition for Automated Process Planning, Work in progress, 1985.

[9] Link, C. H., CAPP--CAM-I Automated Process Planning System, *Proc. 13th Numerical Control Society Annual Meeting and Technical Conference,* Cincinnati, March 1976.

[10] Mann, W. S. Jr., Dunn, M. S., and Pfiederer, S. J., Computerized Production Process Planning, Report R77-942625-14, United Technologies Research Center, Nov. 1977.

[11] Matsushima, K., Okada, N., and Sata, T., The Integration of CAD and CAM by Application of Artificial-Intelligence, *CIRP,* pp. 329-332, 1982.

[12] Nau, D. S., Issues in Spatial Reasoning and Representation for Automated Process Planning, *Proc. Workshop on Spatial Knowledge Representation and Processing,* May 1983. (By invitation; not refereed.)

[13] Nau, D. S., SIPP Reference Manual, Tech. Report 1515, Computer Science Department, University of Maryland, 1985.

[14] Nau, D. S. and Chang, T. C., Prospects for Process Selection Using Artificial Intelligence, *Computers in Industry* **4**, pp. 253-263, 1983. (Also available as Tech. Report TR-1268, Computer Sci. Dept., Univ. of Maryland.)

[15] Nau, D. S., Jones, D., and Masal, K., Converting Boundary Surface Representations Into Constructive Solid Geometry Representations, *IEEE Trans. Systems, Man, and Cybernetics*, 1985, to appear.

[16] TNO,, *Introduction to MIPLAN,* Organization for Industrial Research, Inc., Waltham, MA, 1981.

[17] Wolfe, P. M. and Kung, H.K., Automating Process Planning Using Artificial Intelligence, in *1984 Annual International Industrial Engineering Conference Proceedings,* Chicago, ILL, 1984.

[18] Wysk, R. A., An Automated Process Planning and Selection Program: APPAS, Ph.D. Thesis, Purdue University, 1977.