Manufacturing Feature Instances: Which Ones to Recognize?*

Satyandra K. Gupta
Mechanical Engineering Department
Institute for Systems Research
skgupta@src.umd.edu

William C. Regli[†]
Computer Science Department
Institute for Systems Research
regli@cs.umd.edu

Dana S. Nau
Computer Science Department
Institute for Advanced Computer Studies
Institute for Systems Research
nau@cs.umd.edu

University of Maryland
College Park, MD 20742 USA

Abstract

Manufacturing features and feature-based representations have become an integral part of research on manufacturing systems, largely due to their ability to model correspondences between design information and manufacturing operations. However, several research challenges still must be addressed in order to place feature technologies into a solid scientific and mathematical framework. One challenge is the issue of alternatives in feature-based planning.

Even after one has decided upon an abstract set of features to use for representing manufacturing operations, the set of feature instances used to represent a complex part is by no means unique. For a complex part, many (sometimes infinitely many) different manufacturing operations can potentially be used to manufacture various portions of the part—and thus many different feature instances can be used to represent these portions of the part. Some of these feature instances will appear in useful manufacturing plans, and others will not. If the latter feature instances can be discarded at the outset, this will reduce the number of alternative manufacturing plans to be examined in order to find a useful one. Thus, what is required is a systematic means of specifying which feature instances are of interest.

This paper addresses the issue of alternatives by introducing the notion of *primary* feature instances, which we contend are sufficient to generate all manufacturing plans of interest. To substantiate our argument, we describe how various instances in the primary feature set can be used to produce the desired plans. Furthermore, we discuss how this formulation overcomes computational difficulties faced by previous work, and present some complexity results for this approach in the domain of machined parts.

Keywords: Feature Recognition, Manufacturing Planning, and Feature-Based Representations.

*This work was supported in part by NSF Grants IRI9306580, DDM-9201779, EEC 94-02384 and a forgivable loan from General Electric Corporation awarded to the first author. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

[†]Author's current address: National Institute of Standards and Technology, Manufacturing Systems Integration Division, Building 220, Room A-127, Gaithersburg, MD 20899

1 Introduction

Feature-based manufacturing technologies hold great promise in bridging the information divide between design and manufacturing activities. Manufacturing features and feature-based representations have become an integral part of research on manufacturing systems, largely due to their ability to model correspondences between design information and manufacturing operations.

Over the last decade, significant advances have been made in development of technologies that involve manufacturing features. For example, feature recognition techniques have been developed and successfully employed for a variety of applications including automated process planning, design analysis, and part-code generation for group technology. However, such advances have created new research challenges, one of which is discussed in this paper.

In this paper we will only consider domains in which parts are produced by sequences of discrete manufacturing operations (machining, sheet metal bending, forging, and so forth). Within these domains, different researchers use different definitions of manufacturing features, but these different definitions usually have a number things in common (cf. [17, 23]). In particular, in these domains a *manufacturing feature* is normally considered to be a parameterized geometric object that corresponds to a particular kind of manufacturing operation. Thus, specific manufacturing operations for a particular manufactured part correspond to *feature instances*, which are specified by giving values for the parameters.

Usually, several alternative sets of manufacturing operations can potentially be used to manufacture the same part. Since each operation will normally correspond to a different feature instance, the set of feature instances used to represent a part is by no means unique. For complex parts, it usually is not feasible simply to enumerate all of the feature instances, because the number of them can be very large, or even infinite.

In most cases, very few of the potential feature instances for a part will make practical manufacturing sense. Thus, most approaches to feature recognition will generate only a few of the possible feature instances. However, the criteria for choosing *which* instances to generate are typically *ad hoc* heuristics that are based on local and incomplete information. This makes it difficult to specify the behavior of the feature recognition system and to generate alternative plans in a comprehensive yet well-controlled manner.

This paper addresses the question of which feature instances should be generated. In particular, we argue that for most reasonable definitions of manufacturing features, there is a set of *primary* feature instances that are sufficient for generating all promising manufacturing plans. We describe how primary feature instances can be used to overcome computational difficulties faced by previous work, and present complexity results for the domain of machined parts.

The remainder of this paper is organized as follows. In Section 2, we describe manufacturing features and show that for certain parts, there might be infinitely many feature instances. In Section 3, we define feature-based representations and show that in worst case, the number of feature-based representations might be exponential in the size of a given set of feature instances. In Section 4, we describe how feature recognition can be used to generate feature-based representations. In Section 5, we describe how the notion of useful and primary instances can be used to constrain the possible number feature instances. In Section 6, we describe how feature-based representations can be generated from the set of primary feature instances. Finally, in Section 7, we present our conclusions and describe the benefits that can be achieved by using our formulation.

2 Manufacturing Features

A number of attempts have been made to define and classify manufacturing features [1, 7, 11, 24, 2]. Although there are differences among these approaches, many of them share important similarities. For example, a machining feature usually corresponds to the volume of material that can be removed by a machining operation. In general, manufacturing features usually have associated with them geometry and tolerance information that can be matched with the design attributes of the part and be used to parameterize the manufacturing operations.

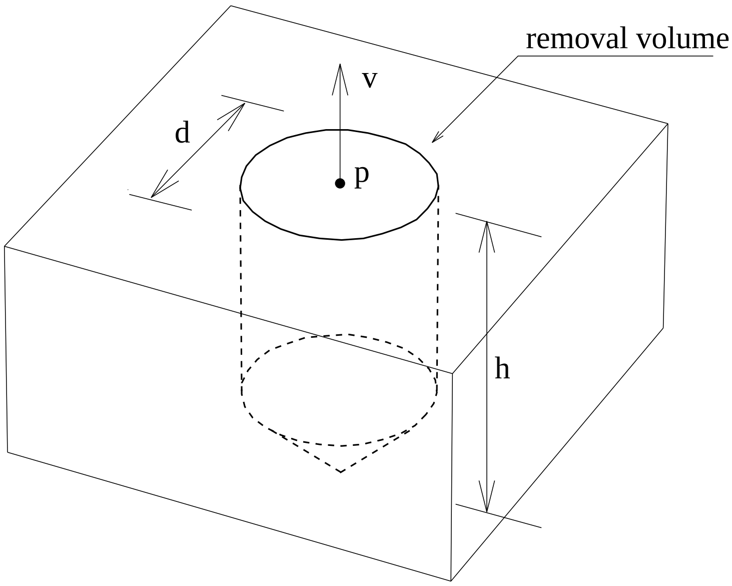
For manufacturing domains that involve discrete manufacturing operations (such as machining, sheet metal bending, forging, etc.), a feature can be thought of as a parameterized object. The parameters of a feature either directly relate to or can be used to derive the parameters of the underlying manufacturing operation. For example, Figure 1 shows examples of features for the machining domain. The feature shown in Figure 1(a) suggests that, if design has a cylindrical surface which needs to be created, drilling may be considered as a possible machining operation. In general, various parameters of a feature can be assigned values from either a discrete or a continuous data set.

In a planning problem, one is typically interested in the feature instances that lead to correct plans. A plan is considered *correct* if it is realizable with available manufacturing resources and produces the part from the stock. A feature instance f is *valid* if there exists at least one correct plan that includes f; otherwise f is *invalid*. In a domain of machined parts there are many conditions under which a feature is invalid. For example, any volumetric feature that intersects with the final part geometry is considered invalid. Including any such feature in a plan would result in over-machining of the part. The set of all valid feature instances is called the *valid feature set*. We use $\mathcal{F}$ to denote valid feature set. Intuitively, one can think of the features in the feature set as the "feature space" of a given part.

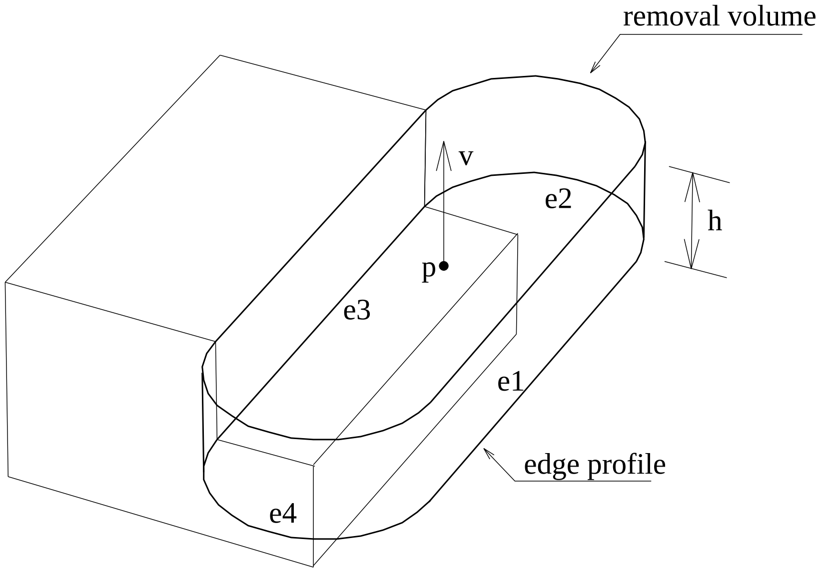
Observation. *There exist parts for which the valid feature set is infinite, i.e., there are infinitely many valid feature instances.*

As an example, consider the part shown in Figure 2(a). This part has a slot that needs to be machined from a hollow cylindrical stock (as shown in Figure 2(b) using standard end-milling operations. As shown in Figures 2(c) and (d), two end-milling operations are needed to create this slot. Therefore, we need to represent this slot as two end-milling features f and f'. Any value of w between $w1$ and $w2$ can be selected as the width of end-milling feature f. This leads to infinitely many possible instances of f. Similarly, any value of w' between $w1'$ and $w2'$ can be selected as the width of end-milling feature f'. This leads to infinitely many possible instances of f'. Which of these feature instances are most appropriate depends on the available manufacturing resources and the optimization criteria. If this part had some other features, those features would have also affected the most desirable feature instances.

In general, if a feature parameter can be assigned values from a continuous scale (such as from a range of real numbers) and none of the values result in an invalid feature (i.e., making every plan that includes this feature incorrect), there will be an infinite set of feature instances for the part.

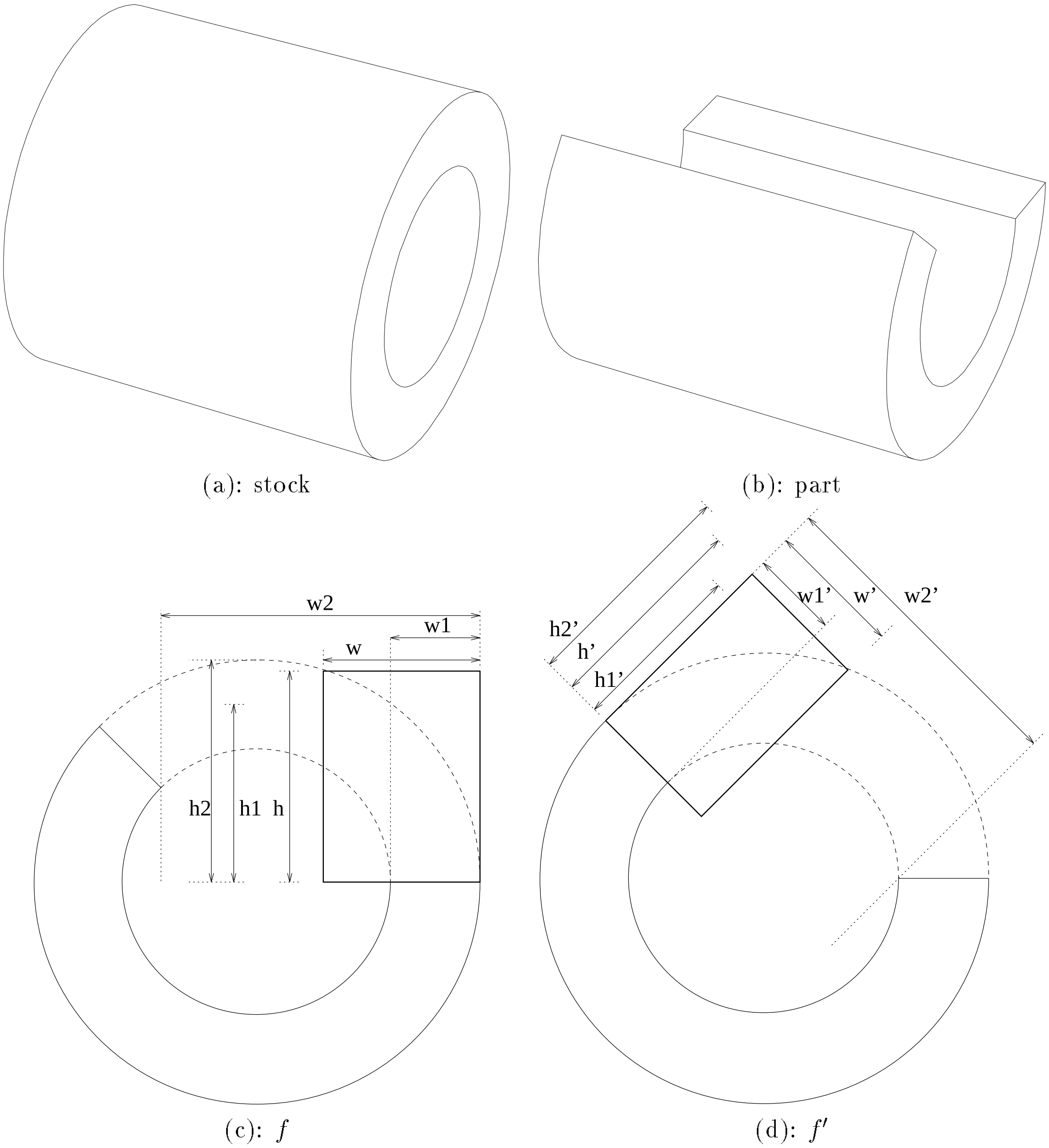


(a): a drilling feature



(b): a milling feature

Figure 1: Examples of machining features.



(a): stock (b): part

(c): f (d): f'

Figure 2: A part geometry leading to infinitely many feature instances.

3 Feature-Based Representations

Feature-based planning usually involves constructing one or more *feature-based representations* (FBR) of the part. Each FBR is a collection of feature instances which can then be mapped into plans. More formally, a set of valid feature instances G is a *feature-based representation* for a given part P and stock (or, blank) S, if it has the following properties:

1. *Sufficiency.* The features in G are sufficient to describe P, i.e., if we apply manufacturing operations corresponding to the elements in G on S, we get P. This ensures that an FBR will have enough features to result in a plan that can manufacture the part to desired specifications.

2. *Necessity.* No feature f in G is redundant, i.e., if we eliminate any feature from G, then the remaining features are not sufficient to produce P from S. This condition means that each feature of a feature-based representation will contribute to some necessary portion of the plan.

Observation. *In the worst case, for a finite subset $\mathcal{F}_r$ of the valid feature set $\mathcal{F}$, the number of alternative feature-based representations that can be produced from $\mathcal{F}_r$ is exponential in the size of $\mathcal{F}_r$.*

Let $\mathcal{F}_r$ be a finite subset of the valid feature set (i.e., $\mathcal{F}_r \subset \mathcal{F}$) and let $\mathcal{I}$ be the size of $\mathcal{F}_r$ ($\mathcal{I} = |\mathcal{F}_r$). Let $\mathcal{A}$ be the number of alternative feature-based representations that can be defined using the feature instances in $\mathcal{F}_r$.

Consider the case where a part can be expressed as m spatially disjoint regions to be manufactured and that there are n_i choices of possible feature instances for i^{th} region. Therefore,

$$\mathcal{I} = n_1 + n_2 + \cdots + n_m.$$

The number of alternative feature-based representations for this part if

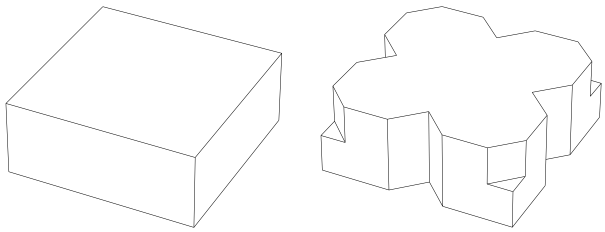
$$\mathcal{A} = n_1 \times n_2 \times \cdots \times n_m.$$

The worst case for $\mathcal{A}$ will be when $n_1 = n_2 = \cdots = n_m = n$. Substituting this value, we get $\mathcal{I} = n \times m$, and $\mathcal{A} = n^m$. Now by substituting $m = \mathcal{I}/n$, we get

$$\mathcal{A} = (n^{1/n})^{\mathcal{I}}.$$

The worst case occurs when $n = 3$, where substituting we get $\mathcal{A} = (\sqrt[3]{3})^{\mathcal{I}}$. From this expression, we can see that in the worst case, the number of feature-based representations for the part is exponential in number of feature instances (i.e., $\mathcal{A} \in O(k^{\mathcal{I}})$).

Consider the part and the 16 feature instances shown in Figure 3. There are 8 disjoint regions each having two possible choices of feature instances. Therefore, these 16 feature instances result in 256 different feature-based representations for the part.



(a): stock (b): part

(c): set of valid feature instances

Figure 3: Feature instances leading to exponential FBRs.

4 Feature Recognition for Generating Alternative FBRs

In this section, we describe various ways in which feature recognition has been used to generate alternative FBRs from a single CAD model. This is followed by a discussion of the main computational problems in handling alternative feature instances. Finally, we describe how these computational problems can be overcome.

4.1 Approaches

Many different approaches have been developed over the past decade to recognize feature instances and feature-based representations. Many of the existing approaches to recognizing feature instances address the problem as one in 3-dimensional geometric pattern recognition to be approached with techniques from AI (such as frame-based reasoning, graph- and plex-grammars, expert systems, neural nets etc.) [8, 16, 5, 12, 24], pattern matching [15, 20], graph searching [4, 9, 22, 3, 13], or geometric algorithms [10, 6, 19]. Feature instances recognized by these systems are grouped into FBRs using the two approaches described in the next two sections.

4.1.1 Generating FBRs Directly

In this approach, FBRs are generated "on the fly," as the feature instances are recognized. These approaches typically produce a single FBR for the given part. In this approach, whenever alternatives are encountered, a decision is made "on the fly" using a greedy heuristic to select the most promising feature or to discard others. Such greedy heuristics consider only the current feature in relation to the part (and sometimes the stock) and those features found up to that point in the recognition process. In this way, features are discarded based on only partial information and a potentially useful FBR could be eliminated from consideration.

This approach has several drawbacks. First, until we have information about all of the other features that might be in the feature-based representation, applying a greedy heuristic to build the representation on a "best-fit" basis may not lead to optimal results without extensive backtracking. Second, designing a system that includes a domain specific evaluation criteria as part of feature recognition is very difficult. Thus, this approach is not appropriate for complex parts that have a large number of alternative FBRs.

4.1.2 Generating FBRs from a Feature Set

In this approach, the following two steps are used to generate FBRs:

1. *Recognize a set of alternative features.* First from the given part, recognize a set of alternative features. Note that, at this level, all the features that appear promising are retained in this set of features.

2. *Generating and evaluating alternative FBRs.* Once we have recognized a set of alternative features, we can generate FBRs from this set. Intuitively, one can think of the set of alternative features found through feature recognition as vectors forming a basis for the "feature space" of the given part. Knowing a good set of spanning features allows us to better define upper and lower bounds for the evaluation functions to efficiently navigate through the space of FBRs.

For parts with many different FBRs, this approach appears to be the more promising one. However, as pointed out earlier, the set of valid feature instances could be infinite, therefore the set of all valid feature instances cannot be used as an objective for the feature recognition component in Step 1 of this approach. In most cases, very few of the potential feature instances for a part will make practical manufacturing sense. So, in order to make this approach work effectively, a system will need to choose which instances to recognize. Whether or not a system produces correct results will depend on the set of features recognized in Step 1.

4.2 Computational Problems in Generating Alternative Feature Instances

It has been pointed out previously by Marefat [13, 14] that existing feature recognition methodologies have had only limited success in identifying and describing alternative feature instances. There are several reasons for this. For example, since features can intersect with each other, the introduction of a new feature into a design can divide other features into spatially disjoint components; components which may be computationally expensive to identify and recombine. This poses difficulty for traditional approaches: rule-based methods must capture all geometric situations that arise from the choice of feature hints and the ambiguities inherent in manipulating multiple interpretations in many separate rules. Graph-based algorithms must syntactically or structurally capture these complexities.

Current approaches to addressing the issue of alternative feature instances often lack a systematic means of selecting the appropriate set of feature instances for planning. The criteria for choosing *which* instances to generate are typically *ad hoc* heuristics that are based on local and incomplete information. For decomposition approaches, the features are primitive cells or combinations thereof. Which specific cells are used depends on implementation and the geometry of the given part. For knowledge-based approaches, the behavior of the system is embedded in the rules for completing features from the traces left in the CAD model. The feature classes addressed by these approaches are byproducts of rules and their interactions in a reasoning system. Thus, the particular set of features that get recognized is a byproduct of the implementation of the system. This makes it difficult to specify the behavior of the feature recognition component and to generate alternative FBRs in a comprehensive yet well-controlled manner.

One criterion put forth for assessing how well a feature recognition system addresses the above problems is to ask whether the system is *complete*. Intuitively, completeness refers to the ability of a system to produce all features appearing in a specific, well-specified class of feature instances. If a system produces all features in a given class $\mathcal{C}$, then we say that the system is *complete* over $\mathcal{C}$. In the existing literature, there have been several efforts toward guaranteeing completeness. Sakurai [21] presents a system that decomposes the volume to be machined into disjoint cells and then recombines them to form compound feature instances. This method is complete over the class of features that can be built from compositions of these primitive cells. Similarly, Marefat [13] states that his hypothesis testing approach is complete over his class of hypothesis generators for features. Above mentioned systems were capable of producing a well-specified subset of the valid feature set.

Note that, in existing systems, completeness has not been addressed in terms of any factors that directly relate to manufacturing planning. In these cases, completeness is with respect to criteria that are artifacts of the computational paradigm they used to recognize the features. What needs to be addressed is how to best define completeness in terms of its relationship to planning.

4.3 Completeness Versus Efficiency

In most problems we are looking for FBRs that optimize some abstract cost measures. Thus, simply generating a single FBR is not enough—we need to make sure that the system is capable of generating the desired solution. Thus, completeness in generating alternative features (as discussed in the previous section) is important in order to achieve completeness in generating FBRs.

In cases where there are a very large number of alternatives, we also need to ensure that FBRs are generated in a controlled manner. If a system tries to select the best alternative by simply enumerating all possible alternatives, complex problems will be computationally intractable. As was noted, very few of the possible alternatives make sense in practical situations. Thus, pruning techniques are needed to avoid generation of unpromising alternatives. The ratio of the alternatives examined to the total number of alternatives can be used as an indicator the efficiency of a system.

We want to consider some subset of the valid feature set, hence what is needed is a means of defining the restricted set of feature instances $\mathcal{F}_r$ that will be of interest when generating manufacturing plans. This definition needs to take into account the likely existence of alternative feature-based representations for the part. We would to be able to calculate, in advance of feature recognition and planning, what specific class of features needs to be recognized and what class of alternative interpretations will need to be considered in order to obtain a good plan. Such a specification enhances our ability to do feature recognition by telling us exactly what to look for. Given such a specification, a system can be implemented with any of the previously mentioned approaches.

If $\mathcal{F}_r$ includes all features of interest for planning and, at the same time, excludes those features that are not useful for planning, then the knowledge that a system is complete over $\mathcal{F}_r$ has very useful implications. In particular, one would know precisely which manufacturing plans are within consideration and which are outside the scope of the system. As $\mathcal{F}_r$'s properties can be defined with respect to planning, one would also know that most of the computational effort is being used to generate and evaluate realistic manufacturing plans.

Section 5 describes how the notions of useful and primary instances have been used to constrain the valid feature set. Section 6 describes how FBRs are generated from the restricted set of valid features.

5 Constraining the Valid Feature Set

In this section, we first classify the feature instances that help in constraining the valid feature set. After that, we show how the valid feature set for machined parts can be constrained using these feature instances.

5.1 Useful and Unuseful Instances

The most natural way of classifying the features is to partition them into those that we consider *useful* for manufacturing planning, and those that we consider *unuseful* (i.e., unlikely to occur in any reasonable plan). Below, we consider several possible ways to do this.

Plan level unusefulness. The simplest way of defining unusefulness by stating that a feature f is considered *unuseful*, if f does not appear in the optimal plan. However, in most realistic planning problems, the cost of a feature in a plan is affected by other features in the plan. Thus, this set of unuseful features cannot be determined a priori without actually generating and evaluating all possible plans and, hence, all possible FBRs. Therefore, this notion of unusefulness cannot be used to constrain the valid feature set in practice, and is only of theoretical interest.

Furthermore, in manufacturing planning problems, models of cost are not very accurate. Estimated costs of most operations have associated variations. Therefore, formally defining the concept of the optimal plan is not possible. Instead, if we rank plans by their estimated costs, on the top there will be a set of desirable plans. Any of these plans might turn out to be equally good and the planning system needs to be able to produce one of these plans. Certain operations may get classified as "very expensive or undesirable" without having any quantitative information. Operations requiring special purpose manufacturing equipment would be examples of such category. Violation of common manufacturing practice can be considered another example of this category. Such violation may pose risks of equipment failure or reduce the probability of successfully completing the operation. Any plan containing these undesirable operations will be considered undesirable. Therefore, at the bottom of the list of plans there will be a set of undesirable plans. Quite a few plans may lie between these two extremes. In order to improve computational efficiency of planning, we want to prune all undesirable plans.

From practical point of view, we need another definition of usefulness at the level of the individual feature instances.

Feature level unusefulness. In this type of unusefulness, a valid feature instance f will be considered *unuseful* if:

Condition 1: f is redundant for every possible plan. A feature f will be considered irredundant in a plan $\mathcal{P}$ if:

(a) Even if the operation corresponding to f is eliminated from $\mathcal{P}$, $\mathcal{P}$ can still produce the part from the stock.

(b) $\mathcal{P}$ includes a feature g that corresponds to undoing a portion of f;

Condition 2: All plans including f will be undesirable.

Condition 3: There exists a feature g, such that replacing f by g in every plan containing f improves the plan.

In many planning domains, testing Condition 3 a priori will be very difficult. The exact cost contributed by a feature f to a plan $\mathcal{P}$ can only be determined if all other operations in the plan are known. For example, whether f will require a new setup or not would depend whether there is any other feature in $\mathcal{P}$ that requires the same setup as f. Thus, in general, an a priori test for determining all unuseful features is not possible.

Therefore, we define another notion of usefulness based on a priori testability. This notion of unusefulness finds a subset of features found by the feature level unusefulness.

A priori testable unusefulness. In this type of unusefulness, Condition 3 of feature level unusefulness is replaced by the following condition:

Condition 3': Let g be a feature that subsumes the portion of workpiece created by f. Let C_g^u be the upper-bound of cost contributed to P by g. Let C_f^l be the lower-bound of cost contributed to P by f. f is unuseful if $C_f^l > C_g^u$.

In practice this notion of unusefulness can be used as a pruning guideline for discarding unpromising feature instances from the valid feature set. Effectiveness of the pruning would depend on how sophisticated a test can be implemented to achieve Condition 3'.

Examples. For machined parts, any feature instance having no intersection with the delta volume (i.e., volume to be machined) is an example of an unuseful instance. Another example of an unuseful instance is an end-milling feature instance that is completely subsumed by a face-milling feature instance creating the same portion of the part at a significantly lower cost.

For sheet-metal bending, feature instances resulting in overbends will be considered unuseful. Every plan including these type of instances will require another feature instance that will correspond to undoing some portion of the bending performed by the unuseful instance.

5.2 Primary Instances

The set of all useful features $\mathcal{F}_u$ as determined by an a priori testable unusefulness criterion may still be quite large (even infinite). Thus, we need additional restrictions on the set of features being recognized.

What we would like to do is to recognize a set of representative instances from the set of all useful features. Such representative instances will be called *primary instances* and can defined by imposing restrictions on the set of useful features. In selecting these representative instances, one needs to make sure that all other instances of interest can be generated by manipulating these primary instances.

Primary instances are defined as follows. Suppose we can define an equivalence relation E on the set of all useful features $\mathcal{F}_u$. This equivalence relation E partitions $\mathcal{F}_u$ into several different equivalence classes. From each class we select a representative instance. Whenever required, a representative feature can be manipulated to produce other feature instances in the same equivalence class. The representative instance for each class is called the *primary instance*. A primary instance should also be able to provide good upper and lower bounds on the cost of including other instances in the same equivalence class to a plan. If we can identify primary instances for a planning domain, then just recognizing the set of of all primary instances is adequate for performing the manufacturing planning.

It is easy to see that, while there are a large number of useful instances, a relatively small number of their characteristics (such as operation type, orientation etc.) are shared by these instances. Therefore, in most manufacturing domains, an equivalence relation can be devised based on these characteristics to partition the set of useful features and select primary instances. The set of primary feature instances for the part is called the *primary feature set* $\mathcal{F}_p$. The following section describes how to define primary instances for machining features.

5.3 Primary Instances for Machining Features

Once we select a specific domain and a scheme for defining manufacturing features, we can formulate specific conditions for identifying valid, unuseful, and primary instances. In this section, we demonstrate how these conditions can be formulated for machining features that correspond to operations on a 3-axis vertical machining center. For simplicity, we will restrict ourselves to drilling and milling operations. Note that the following presents just one set of conditions for machining features. There may be other equivalent conditions which also adequately get at the notions of valid, unuseful, and primary instances.

Machining features. Consider a class of volumetric machining features, each of which has type, location, orientation, tool, and a set of attributes describing removal volume as its parameters. The *removal volume* of the feature is the volume that can be removed by the feature from the workpiece. For example, Figures 1(a) and (b) show removal volumes of a drilling and milling features. For a feature f, the removal volume is denoted by $\mathrm{rem}(f)$. Note that the actual volume removed by a feature from a workpiece is not necessarily its removal volume; instead, it is its effective removal volume. The *effective removal volume* $\mathrm{eff}(f, W)$ of a feature f is defined with respect to a workpiece W. It is given as $\mathrm{eff}(f, W) = \mathrm{rem}(f) \cap^* W$.

Conditions for valid instances. A feature instance f is *valid* for a given part P, if the removal volume of f does not intersect with P.

Conditions for unuseful instances. A feature instance f is *unuseful* for a given part and stock, if:

1. f does not create any portion of the part boundary.
2. The orientation of f is not in the set of fixturable orientations O_f (how to compute O_f is described below) for the part and stock.

The set of preferred orientations O_f is computed as follows:

1. For every planar face u in the part and stock, add a vector perpendicular to u to O_f.
2. For every cylindrical/conical face u in the part and stock, add a vector parallel to axis of u.
3. For every planar face u in the part and stock, do the following:

   if no vector in O_f is parallel to u, then add a vector parallel to u to O_f.
4. For every cylindrical/conical face u in the part and stock, do the following:

   if no vector in O_f is perpendicular to the axis of u, then add a vector perpendicular to the axis of u O_f.

Conditions for primary instances. For machining features, maximality of the removal volume can be used to formulate conditions for primary instances. Such instances correspond to the maximal realistic machinable volume made by a single machining operation in a single machining setup. Such instances can be easily truncated later to produce other feature instances that correspond to the machining volumes removed in the actual machining plans.

Given a valid feature instance $f \in \mathcal{F}_u$, we define the *primary container* of f to be the feature instance $p_c(f) \in \mathcal{F}_u$, such that:

1. $p_c(f)$ has the same orientation, tool and machining operation as f.

2. The removal volume of $p_c(f)$ contains the removal volume of f.

3. For every valid feature $g \in \mathcal{F}_u$ (of the same orientation, tool and machining operation as f) whose removal volume contains $p_c(f)$'s, g has the same effective removal volume as $p_c(f)$.

4. For every valid feature $g \in \mathcal{F}_u$ (of the same orientation, tool and machining operation as f) whose removal volume is contained in $p_c(f)$'s, g has a smaller effective removal volume than $p_c(f)$.

Now we define the equivalence relation R on $\mathcal{F}_u$. Two instances in $\mathcal{F}_u$ are considered R-equivalent if they have the same primary container. It is quite straightforward to show that R forms an equivalence relation on $\mathcal{F}_u$. For the sake of brevity, we are omitting the details here.

A feature instance that is the primary container of itself is the representative of its equivalence class and is defined to be a *primary instance.*

Complexity results for machining features. We would like to calculate an upper bound on the number of primary features that might exist for a given part. Specifically we would like to show that the number of primary feature instances is polynomial in the "size" of the part. In this analysis, size refers to the number of geometric and topological entities in the model of the part; i.e. n is $O(E)$ where E is the number of edges of the part.[1]

To show the number of primary feature instances is polynomial in the size of the part involves three observations. First, within the set of useful features there are $|O_p|$ possible orientations. As defined above, there are at most 2 orientation vectors added to O_p for each face of the part. Hence, $|O_p| \in O(n)$ and, for each entity in the part boundary, there are $O(n)$ possible orientations for the features to produce that entity.

As noted in Section 2, there may be an infinite number of such valid feature instances. For each different tool and machining operation, let T be the set of feature instances producing that entity with the same tool and operation. We show that T contains one primary instance of a valid feature. If f is a feature in T, there is a primary feature $p_c(f)$. We know that, for all features g in the set $calF_u$ with the same orientation, tool, and operation (and hence also those in the set T), then $\text{rem}g \subseteq \text{rem}p_c(f)$. If $\text{rem}p_c(g) \subseteq \text{rem}f$, then $\text{eff}g = \text{eff}p_c(f)$, otherwise $\text{eff}g \subset \text{eff}p_c(f)$. Hence, $p_c(f)$ is a primary feature for all features in T, and the number of primary feature instances is $O(n^2)$ (i.e., one primary feature instance of each feature type in each orientation is capable of creating each portion of the part boundary).

Recognizing the primary feature set for machined parts. In the feature recognition literature, there are many approaches capable of producing the set of primary features for machined parts. Perhaps the best suited of these are the trace-based methodologies [24, 18, 13]. In such an approach, machining features are identified by matching the geometric characteristics of various part faces with various types of features. The boundary of a feature is comprised of different types of surfaces. Each type (planar, conical, etc.) may be a part of the boundary of one or more types

[1] For the worst case, we can say the size is $O(n)$ where $n = E + V + F$ and E, V, and F are the number of edges, vertices, and faces of part respectively. By Euler's equation $2 = V - E + F$, we can simplify this to be $n = 2 + 2E$ or $n = O(E)$.

of features. For example, a cylindrical face could be considered as the side face of a drilling feature and as a corner radius of an end-milling feature. For a given part face, we would like to construct all possible useful feature instances that might be used to create the face. For example, in the case of a cylindrical face, we want to try to instantiate both drilling feature and end-milling features. Any feature instance that intersects with the part is not valid and is discarded.

In our previous work [19, 18], we have developed trace-based algorithms for identifying the set of primary feature instances.

6 Using The Primary Feature Set to Generate FBRs Efficiently

Each primary instance is representative of its equivalence class. Thus, the primary feature set $\mathcal{F}_p$ captures the information about the set of all useful features $\mathcal{F}_u$ useful for planning. It is worth noting here that, in building the desired plan, one might be actually interested in a feature instance not present in $\mathcal{F}_p$. Generation of FBRs from $\mathcal{F}_p$ is an indirect process—primary instances can be used to prune unpromising FBRs using various constraints and feature relationships derived using $\mathcal{F}_p$ (which also extend to various instances in $\mathcal{F}_u$). Therefore, whenever a collection of primary feature instances looks unpromising, all the FBRs that can be generated by replacing various primary instances by other instances in the respective equivalence class of a primary instance are also unpromising, and can be discarded.

On the other hand, whenever a collection of primary features appears to be promising, various primary instances in the collection can be manipulated to create FBRs that consist of the most appropriate instances from $\mathcal{F}_u$, the set of all useful features. In this way, the primary feature set alleviates the need of ever explicitly finding the set of all useful features $\mathcal{F}_u$. This can significantly improve the computational efficiency of feature recognition and FBR generation. In this way, the set of primary features $\mathcal{F}_p$ forms a very effective basis for the "feature space" of a part.

In order to use primary instances in FBR generation, the same basic idea of the approach presented in Section 4.1.2 can be used. However, several augmentation steps are needed to allow efficient use of primary feature instances. The following is a modified version of the approach that can be used to generate FBRs using primary instances:

1. For the given part and stock, recognize the primary feature set $\mathcal{F}_p$.

2. Find various constraints on the primary instances in $\mathcal{F}_p$.
   (These constraints will later be used to discard infeasible collection of feature instances.)

3. Compute the lower-bound of the cost C_l on any plan resulting from features in $\mathcal{F}_p$.
   (Note that C_l also applies to $\mathcal{F}_u$.)

4. Compute the upper-bound of the cost C_u on any plan resulting from features in $\mathcal{F}_p$.
   (Note that C_u also applies to $\mathcal{F}_u$.)

5. Initialize *current_best* $= C_u$.
   (Variable *current_best* is used to store the cost of current-best solution.)

6. Do Steps (a)-(c) repeatedly, until:

   - *current_best* has come close enough to C_l;

- or, no new FBR can be generated.

(a) Find a new FBR F by manipulating a collection of instances in $\mathcal{F}_p$ such that:

  i. lower-bound of the cost on any plan resulting from features in F is better than *current_best*.

  ii. F respects various constraints found in Step 2.

(b) Using various features in F, generate the best possible plan P.

(c) If plan P is better then *current_best*, then update *current_best*.

7. Return the FBR and the plan that resulted in the current value of *current_best*.

Details of various steps of the above described procedure depend on the specific domain. How well a primary set captures the information in the set of all useful features determines the efficacy of this approach. In the the following section, we describe details of some of the steps of this procedure in the domain of machined part.

6.1 Generating FBRs for Machined Parts

Finding constraints to discard unpromising FBRs. Tolerance and symmetry information can be used to generate a set of constraints on features in $\mathcal{F}_p$ that describe which subsets of $\mathcal{F}_p$ are not feasible and which subsets look more promising. For example, any two features having different orientation vectors but associated with the faces having tight tolerances will not result in a feasible FBR. Symmetric portions of the part should be machined with similar features (i.e, having the same type and orientation). Plans with similar features typically have lower cost compared to the plans with dissimilar features. Thus, whenever possible, similar feature combinations should be tried first.

Finding lower and upper bounds at the feature level. Since a primary instance volumetrically subsumes every instance in its equivalence class, it can easily provide an upper-bound of cost of every instance in its class. The irredundant portion of the primary feature instance can be used to provide lower-bounds.

Generating FBRs. In case of machined parts, an FBR is basically an irredundant volumetric cover of the delta volume. Thus, techniques for finding irredundant set covers can be used to generate FBRs in case of machined parts. From implementation point of view, FBR generation step (Step 6(a) of the general approach) can be solved more efficiently using the following two sub-steps:

1. Find volumetric covers of effective removal volumes of various primary features. Note that two feature instances can have the same effective removal volume. For example, the feature shown in the first row and the first column and the feature shown in the second row and the first column of Figure 3(c) have the same effective removal volume.

2. Find FBRs corresponding to a volumetric cover R found in the previous step by adding feature instances that resulted in various effective removal volumes in R.

Finding lower and upper bounds at the FBR level. Given a set of feature instances G, the function $h(G)$ can be used to find the lower bound on the production time of plan resulting from features in G. $h(G)$ is defined as

$$h(G) = L_s(G) \times T_s + (1+\beta) \sum_{g \in G} L_{mt}(g),$$

where

- $L_s(G)$ is a lower bound on the number of setups needed to machine G. For three-axis machining centers, $L_s(G)$ is the cardinality of the set $\{\vec{v}(g) : g \in G\}$, where $\vec{v}(g)$ is the orientation vector for g.
- T_s is the minimum setup time.
- $L_{mt}(g)$ is a lower bound on the time required to machine g. This is the time required to machine the irredundant portion of the effective removal volume of g. Let solid $g_I = \mathrm{eff}(g, S) -^{*} \cup_{f \in G - \{f\}} (\mathrm{eff}(f, S))$, where S is the stock. Now $L_{mt}(g)$ can be computed as

$$L_{mt}(g) = \text{machining time for } g \times (\text{volume of } g_I / \text{volume of } \mathrm{eff}(g, S)).$$

- β is the fraction of machining time that accounts for the auxiliary time.

$h(G)$ is very useful for discarding FBRs that involve features from many different approach directions, and therefore require many setup changes.

7 Conclusions

In a variety of application domains, it is useful to employ representational schemes in which parts to be manufactured are represented as collections of manufacturing features. However, even within a single representational scheme, there can be many alternative representations of the same part as different collections of feature instances. For complex parts, the number of feature instances can be so great that it is infeasible to deal with all of them. In order to integrate feature recognition systems with downstream software components, it is important to use only those feature instances that are actually relevant for manufacturing.

In this paper, we have argued that for most reasonable definitions of manufacturing features, there is a set of *primary* feature instances that are sufficient for generating all promising manufacturing plans. Thus, this approach ensures that only a reasonable amount of feature-based representations are examined, while also ensuring that the desired representation will not be overlooked by the system. To demonstrate applicability of this approach, we have provided detailed examples of how this approach can be used in the domain of machined parts.

We anticipate that system designs based on the use of primary feature instances will result in better integration of feature recognition and manufacturing planning. By using primary features, feature recognition methodologies can be focused toward finding only those features most applicable for generating realistic manufacturing plan.

Acknowledgements

This work was supported in part by NSF Grants IRI9306580, DDM-9201779, EEC 94-02384 and a forgivable loan from General Electric Corporation awarded to the second author. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] W. Butterfield, M. Green, D. Scott, and W. Stoker. Part features for process planning. Technical Report R-86-PPP-01, Computer Aided Manufacturing–International, Arlington, TX, USA, November 1986.

[2] Tien-Chien Chang. *Expert Process Planning for Manufacturing*. Addison-Wesley Publishing Co., 1990.

[3] S. H. Chuang and M. R. Henderson. Three-dimensional shape pattern recognition using vertex classification and the vertex-edge graph. *Computer Aided Design*, 22(6):377–387, June 1990.

[4] Leila De Floriani. Feature extraction from boundary models of three-dimensional objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(8), August 1989.

[5] Xin Dong. *Geometric Feature Extraction for Computer-Aided Process Planning*. PhD thesis, Rensselaer Polytechnic Institute, Troy, NY, USA, 1988.

[6] R. Gadh and F. B. Prinz. Recognition of geometric forms using the differential depth filter. *Computer Aided Design*, 24(11):583–598, November 1992.

[7] N. N. Z. Gindy. A hierarchical structure for form features. *International Journal of Production Research*, 27(12):2089–2103, 1989.

[8] Mark R. Henderson. *Extraction of Feature Information from Three-Dimensional CAD Data*. PhD thesis, Purdue University, West Lafayette, IN, USA, 1984.

[9] S. Joshi and T. C. Chang. Graph-based heuristics for recognition of machined features from a 3D solid model. *Computer-Aided Design*, 20(2):58–66, March 1988.

[10] Y. S. Kim. Recognition of form features using convex decomposition. *Computer Aided Design*, 24(9):461–476, September 1992.

[11] T. R. Kramer. A library of material removal shape element volumes (MRSEVs). Technical Report NSTIR 4809, NIST, Gaithersburg, MD, March 1992.

[12] Timo Laakko and Martti Mäntylä. Feature modelling by incremental feature recognition. *Computer Aided Design*, 25(8):479–492, August 1993.

[13] M. Marefat and R. L. Kashyap. Geometric reasoning for recognition of three-dimensional object features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):949–965, October 1990.

[14] Michael Marefat and R. L. Kashyap. Automatic construction of process plans from solid model representations. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(5):1097–1115, September/October 1992.

[15] J. Miguel Pinilla, Susan Finger, and Friedrich B. Prinz. Shape feature description using an augmented topology graph grammar. In *Proceedings NSF Engineering Design Research Conference*, pages 285–300. National Science Foundation, June 1989.

[16] S. Prabhakar and M. R. Henderson. Automatic form-feature recognition using neural-network-based techniques on boundary representations of solid models. *Computer Aided Design*, 24(7):381–393, July 1992.

[17] M. J. Pratt. Application of feature recognition in the product life-cycle. *International Journal of Computer Integrated Manufacturing*, 6(1&2):13–19, 1993.

[18] William C. Regli, Satyandra K. Gupta, and Dana S. Nau. Extracting alternative machining features: An algorithmic approach. Technical Report ISR-TR94-55, The University of Maryland, July 1994.

[19] William C. Regli and Dana S. Nau. Recognition of volumetric features from CAD models: Problem formalization and algorithms. Technical Report TR 93-41, The University of Maryland, Institute for Systems Research, College Park, MD 20742, USA, April 1993.

[20] Scott A. Safier and Susan Finger. Parsing features in solid geometric models. In *European Conference on Artificial Intelligence*, 1990.

[21] Hiroshi Sakurai and Chia-Wei Chin. Definition and recognition of volume features for process planning. In Jami Shah, Martti Mäntylä, and Dana Nau, editors, *Advances in Feature Based Manufacturing*, chapter 4, pages 65–80. Elsevier/North Holland, 1994.

[22] Hiroshi Sakurai and David C. Gossard. Recognizing shape features in solid models. *IEEE Computer Graphics & Applications*, September 1990.

[23] Jami Shah, Martti Mäntylä, and Dana Nau, editors. *Advances in Feature Based Manufacturing.* Elsevier/North Holland, 1994.

[24] J. H. Vandenbrande and A. A. G. Requicha. Spatial reasoning for the automatic recognition of machinable features in solid models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12):1269, December 1993.