A PARSER THAT CONVERTS A BOUNDARY REPRESENTATION INTO A FEATURES REPRESENTATION

Thomas R. Kramer
Guest Researcher, National Institute of Standards and Technology &
Research Associate, Catholic University

October 3, 1988

Funding for the research reported in this paper was provided to Catholic University under Grant No. 70NANB7H0716 from the National Institute of Standards and Technology.

Certain commercial equipment and software are identified in this paper in order to adequately specify the experimental facility. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the equipment or software identified are necessarily the best available for the purpose.

ABSTRACT

The VWS2 B-rep Parser is a computer program written in LISP that takes a file giving the boundary representation of a part as input and produces a file giving a feature-based representation of the part as output. The format of the input file is a PDES/STEP boundary representation, and the format of the output file is that required by the VWS2 system of the NIST AMRF. The parser deals with a limited range of two-and-a-half dimensional parts. The general approach to parsing is to expect that the part is parsable and look for arrangements of faces which are the signatures of features. The initial implementation of the approach recognizes five feature types. The approach is extendible to a wider range of feature and subfeature types, and to parts which have features made from several sides. Parts having features which intersect in a complex manner are likely to test the limits of this approach, or be beyond the limits. With the addition of this parser, the AMRF Vertical Workstation is capable of making a part from a PDES/STEP file without human intervention.

INTRODUCTION

One of the objectives of research at the National Institute of Standards and Technology (NIST) Automated Manufacturing Research Facility (AMRF) is “to study questions of standardization, measurement, and quality control in the automated factory environment” [NANZ]. NIST was formerly the National Bureau of Standards.

PDES (Product Data Exchange Specification) is a national project developing standards for exchange of complete product model data. The PDES standard is being developed in conjunction with the International Organization for Standards, whose product data standard is the Standard for the Exchange of Product Data (STEP). Recently, the AMRF has begun to serve as a testbed for PDES. The opportunity to do the research reported in this paper arose from CALS support for the NIST PDES testbed.

The AMRF Vertical Workstation has been able to produce a range of two-and-a-half dimensional parts automatically from a feature-based design for some time [K&J1]. Roughly speaking, the range of parts is prismatic parts with fillets and chamfers. The data preparation software for the Vertical Workstation is called the VWS2 system. It is written in the computer language LISP. In the VWS2 design protocol [K&J2], a part is represented by beginning with a block-shaped solid and adding features to it. There are nine feature types in the protocol. For clarity, they will be given in *italic* type in this paper. The nine feature types are: *side_contour*, *contour_pocket*, *contour_groove*, *pocket*, *hole*, *groove*, *straight_groove*, *text*, and *chamfer_out*. Each feature may be thought of as a removed volume. In addition, there are four subfeatures that may be added on some features: *chamfer_out*, *chamfer_in*, *countersink*, and *thread*. For *groove* features there is choice of bottom type between flat and round. *Text* may be round-bottomed or vee-bottomed. *Holes* may be flat-bottomed or conical-bottomed.

What the Vertical Workstation has heretofore been lacking, is a tie-in to design information prepared in other formats. The parser provides that tie-in. It will be called the “VWS2 B-rep Parser” in this paper. Version 1 of the VWS2 B-rep Parser is described here. It is planned that other, more capable, versions will be written.

Many of the design data formats used by existing commercial CAD systems, as well as the AMRF format [HOPP], and one of the PDES formats [ISO1] [ISO2] are boundary representations. As its name implies, a boundary representation describes the outer surface, the boundary, of the part [REQU]. The problem of converting one boundary representation to another is tractable, though not trivial. Other researchers at the AMRF have built systems that convert a Computervision boundary representation into an AMRF boundary representation [FOWL], or an AMRF boundary representation into a PDES/STEP boundary representation [LE&R].

Another important type of part representation is the constructive solid geometry (CSG) representation. In CSG, parts are described in terms of geometric primitive solids which may be combined by union, subtraction, and intersection. The VWS2 design protocol is a CSG representation in which the only operation is subtraction, and the primitives may be complex in shape.
The problem of converting a boundary representation into a CSG representation, or into some other type of features representation, is generally regarded as difficult. Some progress has been reported

by researchers [HEND], but the problem is still largely unsolved. A good survey of techniques used by Grayer in 1976, Kyprianou in 1980, Woo in 1982, and Henderson in 1984, is given in [WILS]. The VWS2 B-rep Parser provides a solution for a limited range of part types.

The intent of the research reported in this paper was to test the usability of the PDES/STEP boundary representation in a computer software system and to integrate a standard part representation with the VWS2 system. The scope of the research was kept limited to a modest subset of what may be represented by the PDES/STEP boundary representation and the VWS2 design protocol. It is expected that the scope will be enlarged in subsequent work.

Another AMRF researcher, Mr. Mark Unger, is developing an interactive graphics-based parser. This is a tool which a human can use to manually augment a boundary representation with a feature-based representation.

RANGE OF PART TYPES

The range of part types handled by the VWS2 B-rep Parser (roughly those parts which can be made with two-and-a-half-axis machining with end mills) is best described constructively. There are two stages to the construction, which is shown in Figure 1.

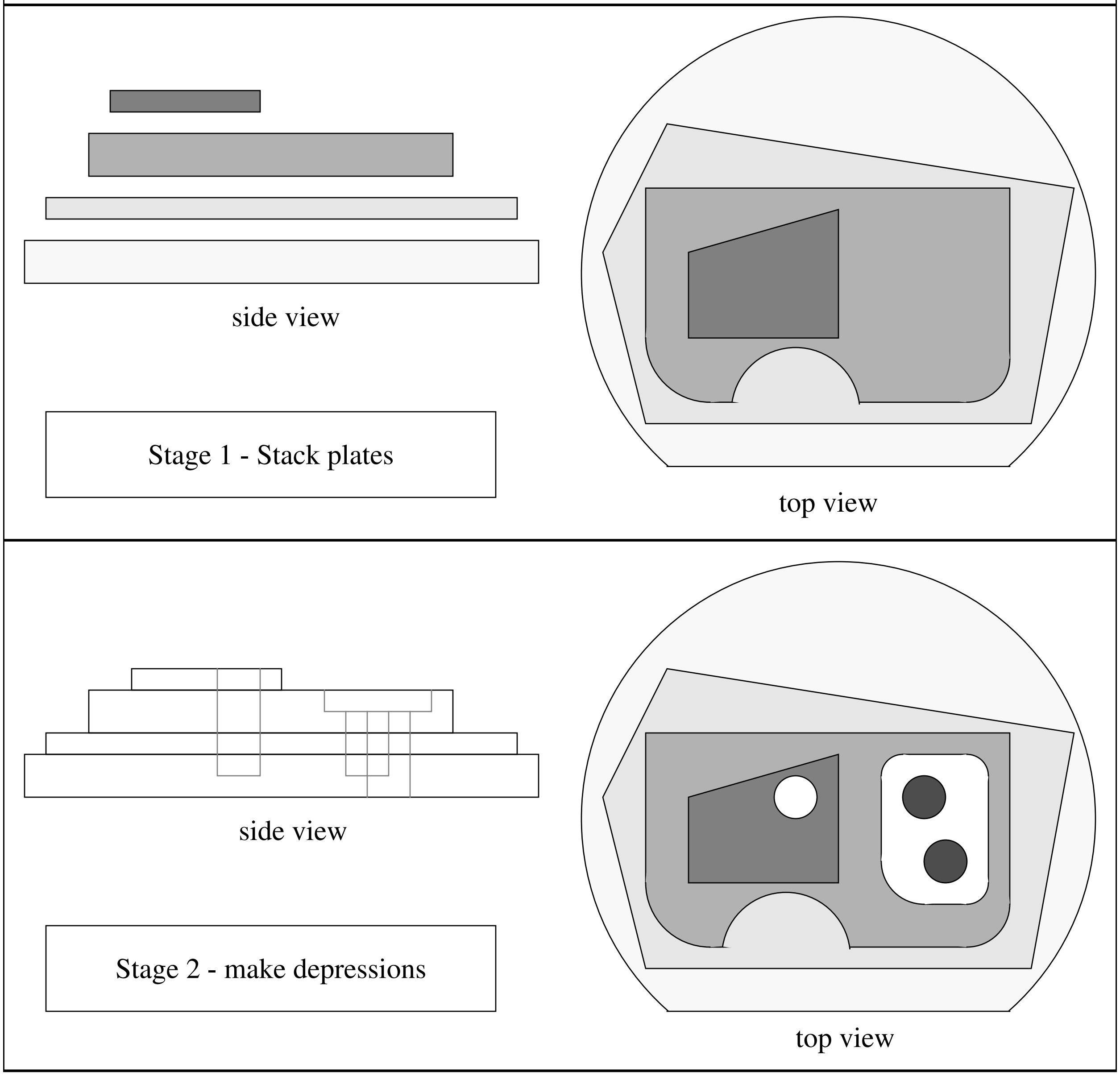
Imagine a stack of metal plates of various thicknesses. There must be at least one plate, but there is no upper bound to the number of plates. Each plate is flat on the top and bottom and vertical on the sides. The outline of each plate is a series of straight line segments and arcs of circles. The outline of each plate fits within the outline of the plate below it, and does not intersect it. Weld the stack together to make a solid.

On every upward-facing surface of the solid make as many depressions as you like. The outline of each depression is a series of straight line segments and arcs of circles. Each depression has a flat bottom and vertical walls but may be of any depth, except that once the bottom of the part is reached there is no more material to make a depression in. The outlines of the depressions may not intersect. At the bottom of each depression, make as many more depressions of the same sort as you please, with the restriction that the outlines of the depressions and the outer edge of the face in which the depressions are made may not intersect. Continue to make depressions in the bottom of depressions as long as you like.

Any solid which may be constructed in this manner can be parsed by the VWS2 B-rep Parser. No solid which cannot be constructed this way can be parsed. The absence of intersections of outlines is critical.

Although the construction method is not likely to bring any familiar object to mind, many parts produced in machine shops for practical use fall in this class, and there is wide variation in the appearance of parts within the class. The class represents only a small fraction of machined parts, however.

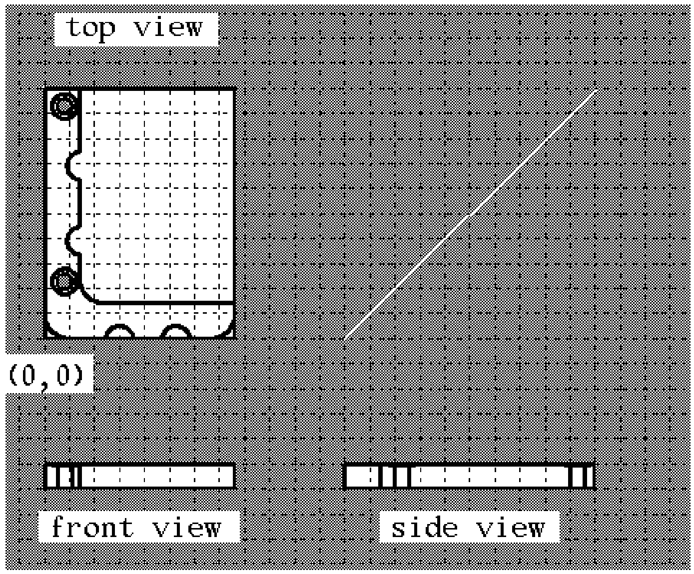
Figure 1. Constructive Description of Part Range



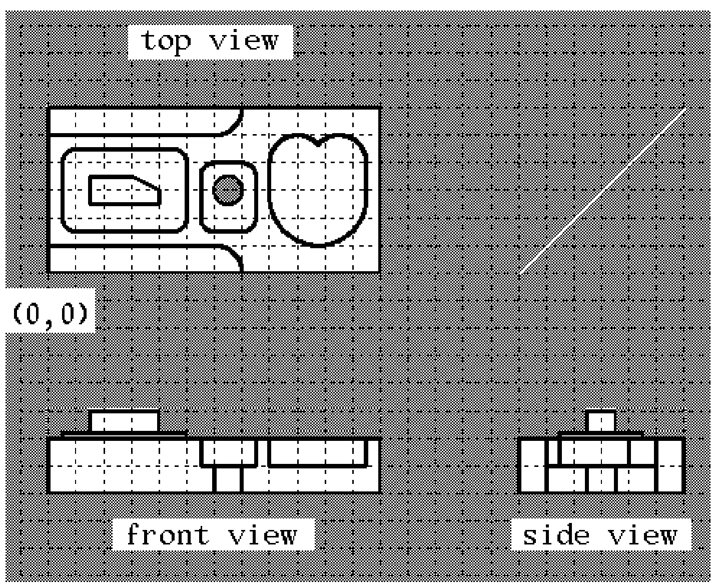
Two parts which fall in the class are shown in Figure 2. As noted on the figure, the features representation is more concise than the boundary representation by a factor of 10 or more, in terms of ASCII characters, when both files are nicely formatted for easy reading by humans, but without huge amounts of white space. This is typical.

The VWS2 B-rep Parser produces a VWS2 design file with five types of features: *side_contour*, *contour_pocket*, *straight_groove*, *pocket*, and *hole*. The only bottom type of the parsed features is flat. No subfeatures are parsed.

Figure 2. Two Parsable Parts



The "pipe clamp" part. The STEP file for this part has 200 entities and 27564 characters. The VWS2 format design has 5 features and 2015 characters. Parsing required about 80 seconds.



A test part, "test5". The STEP file, which could be made more concise, has 817 entities and 60969 characters. The VWS2 format design has 6 features and 1851 characters. Parsing required about 105 seconds.

OVERALL APPROACH

Introduction

The input file is a boundary representation in PDES/STEP format. Six stages are used:

1. Convert the STEP format file into a very similar LISP-readable boundary representation file. Then read the new file into the LISP environment to set up a simple data structure in active memory.

2. Reformat the simple (but hard-to-use) LISP data structure into easy-to-use hierarchical property list format. This property list is another boundary representation, which we will call the VWS2 B-rep.

3. Parse the VWS2 B-rep into features which are either *side_contours* or *contour_pockets*. Parsing consists of identifying groups of faces of the part which may be identified with single features. The parsing process should account for all faces of the part (with minor exceptions described in the next section).

4. Take the list of features extracted in stage 3 and the design_id given by the user and make a property list which is a complete design.

5. Examine the features which are *contour_pockets* to see if they qualify as *straight_grooves*, *holes* or *pockets*, and change the representation of any that qualify.

6. Print the design into a file.

Stage One

The first stage is trivial. It is performed using a UNIX utility called “lex” [SUN] which replaces single characters or strings of characters with other strings. Lex can do much more, but only its character switching ability is used in this system. The intent of this stage is primarily to make the STEP file LISP-readable, so that LISP can deal with it. Characters such as commas, equal signs, @ signs, etc., are replaced by blanks, right and left parentheses, etc., so that the revised file creates a LISP property list when it is loaded into the LISP environment.

Stage Two

The second stage is more substantive. The objective of this stage is to produce a hierarchical property list data structure, the VWS2 B-rep, that is easy to manipulate in LISP. The hierarchical property list data type has repeatedly proved itself to be flexible, comprehensible, readable, and reasonably efficient.

The VWS2 B-rep has three main subdivisions: faces, edges and vertices. The parser can handle only one shell, so no shells section is required. A check is made for multiple shells, and the parser prints an error message and halts if there is more than one. A solid part is always contained in one shell. Geometry and topology are intermingled in the VWS2 B-rep. Leaf nodes may be atoms or simple lists.

Vertices

Each vertex is given a name of the form vertex*n*. Each vertex has three properties: the x, y, and z coordinates of a point in space. The coordinate system is the same system used in the STEP file.

Edges

Each edge is given a name of the form edge*n*. Edges may be either straight line segments or arcs of circles (including full circles). A straight line segment is defined by giving the names of the two vertices at either end. The order of the two vertices is not significant. A circular arc is given by giving the coordinates of the center of the circle and the components of a unit vector perpendicular to the plane of the circle. If the arc is a full circle, a radius is also given. If not, the names of the vertices at the ends of the arc are given. All arcs are assumed to go in a counterclockwise direction (right hand rule) from the first point to the second when viewed from the tip of the unit vector (assuming it is placed with its base at the center of the circle). The radius of an open arc is not given, to avoid duplicate information. The radius may be computed as the distance from the center of the circle to either end. To simplify computations at later stages, the size of the included angle of each open arc is calculated (in radians) and stored as the value of the property "turn". Each edge must belong to two faces. A list of the two faces is stored as the value of the property "faces". The order of the list is not significant.

Faces

Each face is given a name of the form face*n*. Faces have two properties: surface and edge_loops. The surface always has the property "type", whose value is either "plane" or "cylinder".

For a plane, other properties besides type include xco, yco, zco and const which correspond to (a, b, c, and d) in the equation of a plane ax+by+cz=d. xco is an abbreviation of "x-coefficient".

A cylinder is specified by giving the coordinates of a point on the axis of the cylinder, plus the three elements of a unit vector to specify the orientation of the axis, plus the radius. The names of the properties are cx, cy, and cz (for the point), vx, vy, and vz (for the unit vector), and radius. If the unit vector is parallel to the x, y, or z-axis, it must point in the positive direction of that axis.

Edge_loops are numbered sequentially starting with 1. If a face is entirely within a loop, that loop is loop 1. A planar face is always within loop 1, of course. Each edge_loop has two properties: "loop", the value of which is an ordered list of edges that make up the loop, and "corners", the value of which is a corners structure in VWS2 format representing the same information. The loop list may go in either direction around the loop. The direction is not significant (which differs from most boundary representations).

If a horizontal face which is not the top face of the part has more than two edge_loops, the second will be the bottom of an upwards extension, and the third onward will be the top of depressions. A vertical cylindrical face which is closed around its circumference will have the first loop as the upper bounding loop and the second loop as the lower bounding loop. In this case, both loops are full circles.

Because the edge loops in a STEP file are not assumed to be in any particular order, a fair amount of calculation is necessary to put them in order. The STEP file does have a field for identifying the outermost loop, but its use is optional. The VWS2 B-rep Parser assumes this information will not be available and always orders the loops itself.

Stage Three

Stage three is the parsing process itself, in which *side_contour* and *contour_pocket* features are identified. It is described in the next section of this paper.

Stage Four

The height of the starting block for the VWS2 format design is found by adding together the depths of all the *side_contours*. Its length is the maximum x value attained on the outline of feature 1, which is the lowest (hence largest) *side_contour*. Its width is the maximum y value on the same outline. The system expects (and checks) that the location of the part in the coordinate system used by the STEP file is such that the part does not extend into the negative-x side of the yz-plane or the negative-y side of the xz-plane. It does not matter, however, if the part extends into the negative-z side of the xy-plane.

A header for the design is created from the design_id and the block_size. The design itself is assembled as a hierarchical property list with two top-level properties: header and features.

Stage Five

The *contour_pockets* are examined, and those which may be expressed as *straight_grooves*, *holes*, or *pockets* are identified. Since these feature types are simpler than *contour_pockets*, the *contour_pockets* are replaced, if possible.

A *contour_pocket* which is a *hole* is easily recognized: it is necessary and sufficient that the centers of the arcs in each corner of the *contour_pocket* be at the same point.

A *contour_pocket* which is a *straight_groove* is recognized by checking that it has four corners, that the radii in the four corners are the same, and that the centers of the arcs in two pairs of adjacent corners are at the same point.

A *contour_pocket* which is a *pocket* is recognized by checking that it has four corners which lie at the corners of a rectangle whose sides are parallel to the x and y axes, and that the radii in the four corners are the same.

All the checks of equality, parallelism, etc. mentioned above allow a small margin of error.

Stage Six

The design is printed in an easy-to-read format to a file in the “design” subdirectory of the directory in which LISP is running.

APPROACH TO PARSING USED IN STAGE 3

Introduction

In stage three of the overall approach, the VWS2 B-rep is parsed into features. The parsing is based on faces. Groups of faces are identified which form the boundaries of features. We will call such a group of faces the "signature" of a feature. Only two kinds of features are identified at this stage: *side_contours* and *contour_pockets*. The side contours are identified first, sequentially from the bottom up. This is equivalent to identifying the stack of plates described earlier.

Then the *contour_pockets* are identified by looking into all the edge loops on the upper faces of all the *side_contours*. Each such edge loop may be the top edge of a tree of nested *contour_pockets* extending downward into the part, or it may be the top edge of single *contour_pocket*.

Principal Data Structures

The parsing process is facilitated by the use of five lists:

1. the unaltered VWS2 B-rep,
2. a list of horizontal faces called "horizontal_faces",
3. a list of vertical faces called "vertical_faces",
4. a list of the top faces of *side_contours*, called "sc_tops", and
5. a list of features, called "features".

Horizontal faces are identified as those planar faces which have vertical unit vectors. Horizontal_faces is ordered according to height above the xy-plane, with the first face being the lowest.

Vertical_faces is all faces on surfaces which are either planes whose z-coefficient is zero or cylinders whose axis is vertical. It is not ordered.

If any faces remain after horizontal_faces and vertical_faces have been extracted, the user is asked whether the parser should continue or quit. Any leftover faces will be ignored and not parsed. Allowing leftover faces permits the parsing of some parts which have holes in their sides, although no features representing those holes will be created in the parsed design.

Horizontal_faces and vertical_faces are constructed at the beginning of the parsing process. As the parsing process proceeds and faces are identified with features, the faces are removed from horizontal_faces and vertical_faces, while features are added to the features list. Sc_tops starts out as an empty list and is built up as *side_contours* are parsed. Then faces are removed from sc_tops as the *contour_pockets* are parsed, so that it is empty again when parsing is over.

Parsing Side_Contours

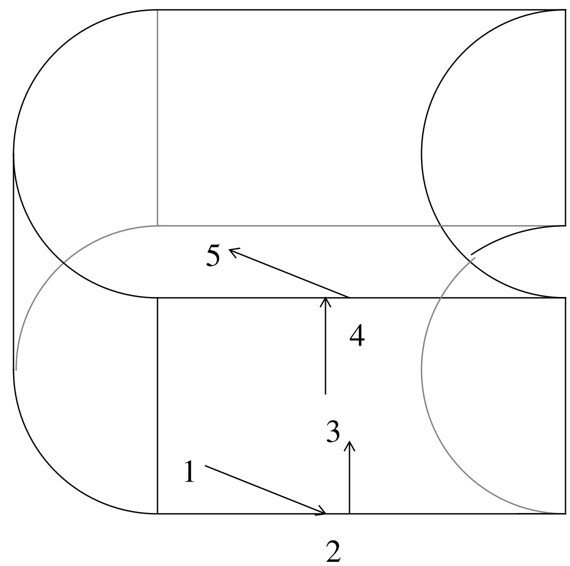
The essence of parsing the *side_contours* is to identify the plates which make up the stack described earlier. Each plate corresponds to a *side_contour* feature.

The first horizontal face is the bottom of the part and the bottom of the lowest plate. Call the outer edge loop at the bottom of this plate the “bottom loop”.

As shown in Figure 3, the upper surface of the plate is identified by picking any edge from the bottom loop, finding the other face attached to that edge, finding the upper edge of that face, and then finding the other face that edge is attached to. Then it is verified that all the vertical faces attached to the bottom loop are vertical and share an edge with the first edge loop of the upper face of the plate.

Figure 3. Finding Upper Surface of Side_Contour

1. Start at lower face
2. Pick an edge of bottom loop
3. Find vertical face on edge
4. Find upper edge of vertical face
5. Find upper face



The faces around the sides of the plate are deleted from vertical_faces, the top face of the plate is put on the sc_tops list, and the bottom face is deleted from horizontal_faces.

A side contour feature is put on the list of features. The depth of the side contour is the distance between the top and bottom faces, and the “corners” data structure for the *side_contour* is extracted from the outer edge loop of the upper surface of the plate. If the corners structure is counterclockwise, an equivalent clockwise structure is constructed and used instead. Other actions are taken to keep track of feature numbers and reference features.

The system then finds the outer edge loop of the next plate up in the stack. The outer loop is the second edge loop of the upper surface of the lower plate. The actions of the preceding three paragraphs are repeated to identify the next plate up and add a feature to the list.
This process is continued until it turns out that the top of the most recently found plate is the uppermost horizontal face of the part, at which point a few concluding actions are taken. The parsing of *side_contours* is completed.

Parsing Contour_Pockets

The parser applies a function named “parse_depressions” to each face in sc_tops. Parse_depressions simply applies “parse_dep” to each edge loop on the face other than the outer loop and the bottom loop of the next plate up (if there is one).

The parsing of a *contour_pocket* by parse_dep proceeds much like the parsing of a *side_contour*. The outer edge_loop of the bottom of the *contour_pocket* is found by tracing down a vertical face of the *contour_pocket* from the top edge loop, which is known, to the bottom. Then all the faces attached to the top loop (except for the current sc_top) are checked for verticality and for being attached to the bottom loop. Those faces between the two loops are removed from vertical_faces. Then parse_depressions is applied to any edge loops found on the interior of the bottom face of the *contour_pocket*, and the bottom face is removed from horizontal_faces.

A *contour_pocket* feature is put on the list of features by parse_dep. The depth of the *contour_pocket* is the distance between the top and bottom faces, and the “corners” data structure for the *contour_pocket* is extracted from the top edge loop and switched from clockwise to counterclockwise, if necessary. Other actions are taken to keep track of feature numbers and reference features.

By recursively applying parse_depressions to the bottom face of each *contour_pocket*, a downward-extending tree of nested *contour_pockets* is fully parsed. The end of a branch of the tree is recognized as a *contour_pocket*, the bottom of which has only one edge loop.

The parsing of *contour_pockets* is complete when each potential tree of *contour_pockets* extending downward from every interior edge loop of each sc_top has been parsed.

Exception Handling

The VWS2 B-rep Parser can parse parts which have the type of shape described above. It tries to extract such a part from the boundary representation it is given. It does not assume that the part is of that type, however. Rather, once it has preliminary evidence of a feature, it checks that all faces of the feature are as they should be before concluding that the feature exists.

At the end of the parsing process, horizontal_faces, vertical_faces and sc_tops must be empty, or an error message is printed and the system halts.

In general, if the part turns out to be unparsable at any stage, an error message is printed and a LISP program break occurs, so that the current state of the parsing process may be examined. Moreover the LISP function in which the break occurs generally returns a cue to the function which called it, and that function also prints a message and breaks for further examination of the parsing process.

EXTENSIBILITY

Version 2

The VWS2 B-rep Parser, version 1, was built in about a month. Only a limited range of feature types was intended to be recognized by version 1, and a rather small range of surfaces (planes and cylinders) and curves (circular arcs and straight line segments) is recognizable. The technique of looking for arrangements of faces which are the signatures of features may be extended quite a bit. In version 2 it should be feasible to add conical, spherical, and toroidal surfaces, and to add arcs which are elliptical, parabolic, and hyperbolic, at least.

Additional arrangements of faces could be added to what the parser will recognize, so that it will handle: chamfers and countersinks on the upper edges of features, fillets between horizontal and vertical faces, round-bottomed and vee-bottomed features.

Feature simplification (stage 5) could be extended to include *contour_grooves*. The level of difficulty of extracting *contour_grooves* varies from easy to hard, depending upon the particular *contour_pocket*. In principle, any *contour_pocket* may be expressed as a *contour_groove*.

Future Version

As just described, Version 2 requires no change to the VWS2 design protocol. By making changes to the protocol and corresponding improvements to the parser, the range of parts could be expanded further. The most desirable changes are:

1. allow islands in *contour_pockets* and multiple raised areas on *side_contours*.
2. allow features to be made from two, six, or many sides of the part.
3. add new feature types.
4. add more subfeatures and options, such as chamfering a contour feature, filleting the bottom of a *pocket*, making the chamfer angle a variable, or rotating a *pocket*.

The range of parts that would be handled by the VWS2 B-rep Parser with these changes would include a significant proportion of the parts a typical machining job shop makes. For some classes of parts, such as electronics chassis, the proportion should exceed fifty percent.

OBSERVATIONS

Timing

Running uncompiled on a Sun 3/160 microcomputer with a numerical coprocessor and 8 megabytes of internal memory, the VWS2 B-rep Parser parsed a STEP file with 200 entities in 80 seconds, and a 23-page STEP file with 817 entities in 105 seconds.
Since the approach used by the parser is a constructive one, and does not involve searching in large spaces, the amount of time taken to parse a part should grow roughly linearly with the number of entities in the input file.

Other Types of Part Model Parsing

The advantage of parsing a boundary representation into a VWS2 format design is that a system already exists which can make parts automatically from such a design. The VWS2 design protocol, however, is not standard. PDES has been working for some time on a standard feature-based design protocol, but the physical file format for this protocol has not yet been defined. When the file format has been defined, it should be feasible to use the approach of the VWS2 B-rep Parser to construct a system which will parse a PDES boundary representation into a PDES features representation. The simplest method of constructing such a system will probably be to use the VWS2 B-rep Parser to convert the PDES boundary representation into a VWS2 feature-based representation, and then convert VWS2 features into PDES features. In the long run, a separate parser to convert between the two PDES representations seems more sensible.

It should be feasible to construct a parser that goes in the opposite direction of the VWS2 B-rep Parser and converts a VWS2 design into a PDES/STEP boundary representation. Indeed, all commercial CSG modeling systems which can make pictures of the faces of parts must have this type of capability: the ability to find the boundaries of a part when its solid form is known.

Integration

The VWS2 B-rep Parser ties the VWS2 system to existing boundary representations, so that parts may be machined automatically from designs in any of three formats. Figure 4 shows the extent to which integration has been achieved. As shown in Figure 4, the following sequence of activity is feasible:

1. The Computervision Company has lent several of its CADDS stations to NIST for use in research. A part is designed on a CADDS station using solid modeling. The design must be done so that no spurious faces or vertices are in the file representing the part.

2. A boundary representation of the part in AMRF standard format is extracted from the CADDS representation of the part by the facility build by James Fowler [FOWL]. This facility has been integrated with the CADDS software and runs on the CADDS station. The conversion is done by issuing a single command from CADDS.

3. The AMRF boundary representation file is transferred to a SUN3 and converted into a STEP boundary representation file by the facility built by Tina Lee and Sanford Ressler [LE&R]. The conversion is done by issuing a single "stepparse" command from UNIX.

4. The STEP boundary representation file is converted into a VWS2 features design file by the VWS2 B-rep Parser. The VWS2 Part Design Editor [K&J2] may be used to create a VWS2 features file directly.

5. A process plan for the VWS2 machining center is generated from the VWS2 features design file and a tool catalog by the VWS2 Process Planning Module [KRAM].

6. An NC-program for the VWS2 machining center is generated from the VWS2 features design file, the process plan, and a current tooling database by the VWS2 Data Execution Module [KR&W].

7. A workstation level process plan for the Vertical Workstation is used to load a part blank, download the NC-program to the controller for the machining center, cut the part, and unload the part.

Obviously, the process may be started at any of the items 1 through 5. For example, data for the pipe clamp part shown in Figure 2 was entered in AMRF format directly, using a text editor, so the process for the pipe clamp started with item 4. The other part shown in Figure 2 was designed on the Computervision CADDS station.

In practice, a few LISP commands have been added to the Vertical Workstation control system, so that items 4 through 7 will be carried out by issuing a single command to the control system. Thus, parts are being made directly from the STEP file without further human intervention.

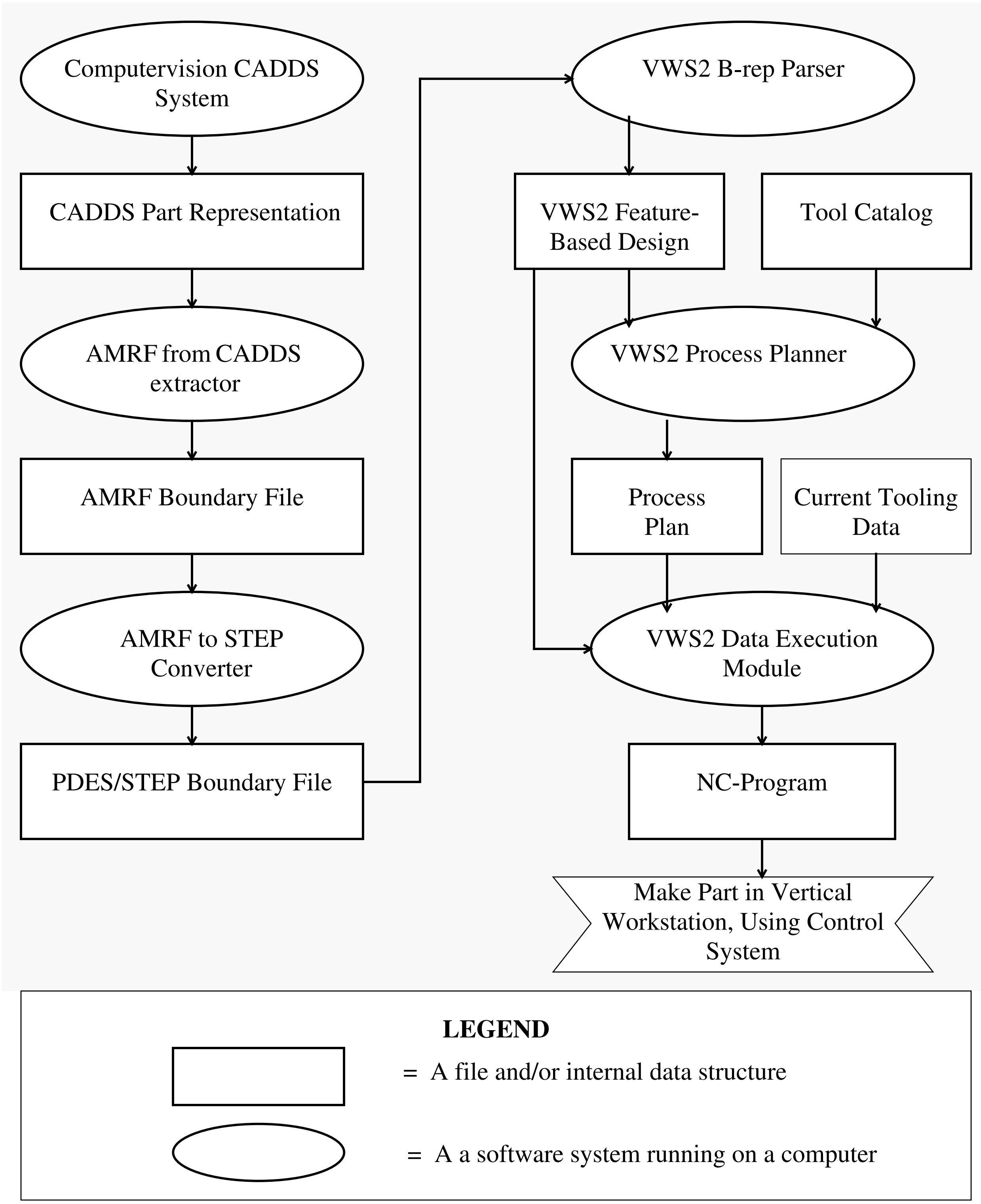
The PDES/STEP Boundary Representation Format

The PDES/STEP boundary representation format was fully adequate for the research reported here, and it appears to be adequate for the continuation of this research. The format has many admirable qualities. It is complete and unambiguous, and individual lines are easily read.

For the purposes of parsing, the format has some drawbacks. First, an understanding of the shape of a part or even a small portion of shape cannot be obtained by a human reading the file. No mental image of shape is obtained by reading the file. Second, to extract data, it is typically necessary to follow a series of up to six pointers from one part of the file to another, across a host of data types, including shells, faces, edges, vertices, surfaces, curves, points, edge loops, and more. The first drawback makes life difficult for the programmer trying to build a parser. The second drawback necessitates more work for both the programmer and the computer. But these are minor problems which are dealt with by extracting the VWS2 B-rep before trying to parse.

It might be desirable to add some restrictions to the PDES/STEP format in order to provide for making the data be minimal. Two potentially useful restrictions were brought to light during the course of building the parser:

Figure 4. Integration



First, do not allow multiple copies of the same entity. Multiple copies simply waste space. For example, all horizontal faces of a part have the same unit vector for orientation, namely (0, 0, 1), but coming out of a commercial CAD system, each horizontal face is likely to have its own copy of this vector.

Second, do not allow unnecessary subdivision of lines and surfaces. Designs coming out of commercial CAD systems may have an unnecessary vertex located in the middle of a continuous line, and the line represented as two parts. This brings the VWS2 parser to a halt. Full cylindrical faces may be divided into two halves by two unnecessary lines between opposite ends of the cylinder. The VWS2 parser can deal with it, but the STEP files contains an extra seven points, four vertices, four curves, four edges, one surface, one face, and three unit vectors, and parsing takes more time. Other unnecessary subdivisions may come out of other systems.

LIMITATIONS

Current Version

Most of the limitations of the VWS2 B-rep Parser have already been noted. In addition, the input STEP file is limited to 1023 numbered lines, since the line numbers are used as properties in the second stage of the system, and the LISP dialect being used, Franz LISP from Franz, Inc., will not use higher numbers as properties. This limitation could easily be removed. It is not necessary to use numbers as properties.

Long Run

In pushing the technique of the VWS2 B-rep Parser to its limits, the most difficult obstacle is expected to be dealing with features which intersect in a complex manner. The basis of the technique is to look for arrangements of faces which are the signatures of features. The set of recognizable signatures must include everything that can occur in the part range the parser is intended to handle. When features are allowed to intersect, the set of signatures will probably grow in proportion to the number of feature types raised to the power of the number of allowable simultaneous intersections of features. This is likely to get out of hand very quickly.

REFERENCES

[FOWL] Fowler, James; personal communication regarding the part model extraction system he built, which converts a Computervision model to an AMRF model; not yet documented; 1988.

[HEND] Henderson, Mark R.; *Extraction of Feature Information from Three Dimensional CAD Data*; Ph.D. Thesis at Purdue University; May 1984; 139 pages.

[HOPP] Hopp, Theodore H.; *AMRF Database Report Format: Part Model*; NBSIR 87-3672; National Bureau of Standards; 1987; 71 pages.

[ISO1] International Organization for Standardization, TC184/SC4/WG1; *The STEP File Structure*; Version 11.0; February 1988.

[ISO2] International Organization for Standardization, TC184/SC4/WG1; *Shape Topical Information Model*; Version Denver; April 1988.

[K&J1] Kramer, Thomas R., and Jun, Jau-Shi; *Software for an Automated Machining Workstation*; Proceedings of the 1986 International Machine Tool Technical Conference; Chicago, Illinois; September 1986; pp. 12-9 through 12-44.

[K&J2] Kramer, Thomas R., and Jun, Jau-Shi; *The Design Protocol, Part Design Editor, and Geometry Library of the Vertical Workstation*; NBSIR 88-3717; National Bureau of Standards; 1988; 101 pages.

[KR&W] Kramer, Thomas R.; and Weaver, Rebecca E.; *The Data Execution Module of the Vertical Workstation of the Automated Manufacturing Research Facility at the National Bureau of Standards*; NBSIR 88-3704; National Bureau of Standards; 1988; 58 pages.

[KRAM] Kramer, Thomas R.; *Process Plan Expression, Generation, and Enhancement for the Vertical Workstation Milling Machine in the Automated Manufacturing Research Facility at the National Bureau of Standards*; NBSIR 87-3678; National Bureau of Standards; 1987; 56 pages.

[LE&R] Lee, Y. Tina, and Ressler, Sanford P.; *Converting the AMRF Part Model Report to a PDES/STEP Subset*; NBSIR 88-3818; National Bureau of Standards; 1988; 39 pages.

[NANZ] Nanzetta, Philip; *Update: NBS Research Facility Addresses Problems in Setups for Small Batch Manufacturing*; Industrial Engineering; June 1984; pp. 68 - 73.

[REQU] Requicha, A. A. G.; *Representations of Solid Objects*; Lecture Notes in Computer Science; Goos, G. and Hartmanis, J.; New York; Springer-Verlag; 1980; pp. 2 - 78.

[SUN] *Programming Utilities for the Sun Workstation*; Sun Microsystems; Part No: 800-1301-03; Revision F of 17 February 1986; 238 pages.

[WILS] Wilson, Peter R.; *Feature Recognition*; SIGGRAPH Course Notes; Advanced Topics in Solid Modeling; Twelfth Annual Conference and Exhibition on Computer Graphics and Interactive Techniques; San Francisco, California; July 1985; 42 pages.