1993

On the Semantics of Generative Geometry Representations

Christoph M. Hoffmann
Purdue University, cmh@cs.purdue.edu

Report Number:
93-010

ON THE SEMANTICS OF GENERATIVE GEOMETRY REPRESENTATIONS

Christoph M. Hoffmann

CSD-TR-93-010
January 1993

On the Semantics of Generative Geometry Representations*

Christoph M. Hoffmann

Department of Computer Science
Purdue University
West Lafayette, IN 47907

January 1993

Abstract

We argue that geometry constructs and solid operations currently under development in feature-based CAD systems are without an appropriate semantic foundation, and that a rigorous semantics poses significant research problems. In particular, we argue that a formalization of *feature attachment* and other generative constructs cannot be based solely on conventional boundary representation or on current geometry standards in view of implications of constraint-based definitions.

1 Introduction

The recent emergence of feature-based CAD systems such as Pro/Engineer from Parametric Technology and related efforts by other CAD vendors including Unigraphics, SDRC, and Matra Datavision constitute first steps towards departing from the CSG paradigm [8] that up to now has provided a conceptual framework for solid design. That is, instead of building complex shapes from simpler ones using regularized Boolean set operations, the user constructs in these emerging systems new shapes from primitives that are not solids. In particular, the arguments to operations such as *cut* or *protrusion* are not shapes that are independent of the geometry the operations modify, and are not conceptually parametrically defined solids. Furthermore, new solid operations such as *shelling* and *ribbing* can be expressed in terms of CSG, but are not so implemented for performance considerations.

*Supported in part by ONR Contract N00014-90-J-1599, by NSF Grant CDA 92-23502, and by NSF Grant ECD 88-03017.

A fundamental complication in the emerging paradigm arises from the possiblity that a design may be expressed using constraints of dimension, positional interrelation, alignment, and so on. Thus, one could say that in these new systems the user does not define the geometry itself, but rather defines rules of *how to construct* the geometry by instantiation.

This development in solid design has implications that go beyond designing slicker user interfaces to package old core geometric modelers. Moreover, the precise definition of such modeling operations requires a semantics that needs more than an agreement on standard interpretation and poses foundational research problems. We attempt to articulate these issues and isolate several paradigmatic research problems.

2 Illustrations

Current feature-based modeling operations are essentially defined by a particular implementation. We illustrate the nature of this implementation in the case of Pro/Engineer, apparently the first system to use the emerging style of "feature-based" design. In particular, we illustrate some of the topological implications of instantiating a constraint-based design over a range of different parameter values.

2.1 Examples of Cuts

A rather surprising aspect of Pro/Engineer is the fact that while the user defines how to generate the geometry, the generation rules are not formalized. Rather, the generation is a particular sequence of operations on the boundary representation. We demonstrate this in the case of *cuts*.

The cut operation is specified by defining an open profile for the cut, attaching it to existing geometric elements, and specifying a set of attributes for the cut such as extruded, revolved, swept; through next, through all, blind; and so on. The operation appears to be a Boolean subtraction in disguise. As illustrated in the following example from Pro/Engineer, Version 8.0, and shown in Figures 1–4, this is not necessarily the case. In the example, an extruded cut is defined by a line with attributes that the cut begin on one side of the sketching plane and end when having cut through the next feature, here the next face in opposite orientation. The cut is instantiated for several different values of the dimension variable sd_0 that positions the height of the cut line with respect to the base face. From the experiment, we conclude that the following method is used for implementing cuts:

1. Extrude the drawn line obtaining a surface of depth determined by the next face whose normal points opposite.

2. Intersect the extruded face with the two adjacent faces and the opposite face.

3. Substitue the trimmed "side" faces, add the new face, and discard from the Brep all structures locally external to the new face.

Observe that the implementation is not equivalent to a Boolean subtraction, but is instead analogous to modifying a Brep when rounding a sequence of edges. Thus, while a cut could be mapped to a Boolean, doing so would result in a slower operation because of the aditional work of having to define additional faces of the solid defining the cut and intersecting them with the other faces of the first solid. Thus, both the user and the system would be penalized. On the other hand, the cut operation is not defined in many cases. Some generic situations are:

1. *Unattached Cut:* Sketching a cross section of a profiled cut involves defining the geometric shape and positioning it relative to the geometry already defined. By changing some dimensions, the cut may be partially or fully outside the existing geometry, and is therefor redundant, as in the case of Figure 4.

2. *Incomplete Profile Definition:* Consider the cut sketched in Figure 5. Because the side faces flare out, the profile must be extended as the cut proceeds across the object. Pro/Engineer generates an unattached cut in this case. In the example it is easy to imagine laterally extending the line defining the cut. However, if the profile is a spline, no automatic extension can be defined.

3. *Unclear End-Rule:* In the cut defined in Figure 6, the cut should terminate at the next face that locally bounds material. On the left, the profiled hole ends at the curved face. On the right, the profile has been moved slightly so that the hole should terminate at the very bottom of the round cross slot. Here, Pro/Engineer is unable to generate the cut properly, not recognizing that the curved face should be the termination face.

It is obvious that these errors can be avoided with a proper conceptualization of what it means to do solid operations based on surface manipulation. In the case of incomplete profiles, one could engage the user in a dialogue that defines a proper extension when no simple strategy would make sense. In the example of Figure 5, it is obvious that the cut ought to be extended by extending the line segment.

Next, we discuss how such semantic rules become more complicated in the presence of geometric constraints, a common device for defining geometric designs.

2.2 Sketching and Constraint Examples

An attractive technique for defining shapes in 3-space is to build them from 2-dimensional drawings. Examples include extruding or revolving cross sections, or sweeping contours. In each case, two-dimensional sketches are helpful because they are immediately comprehended.

One can sketch with precision using constraints such as dimensioning distances and angles, as well as imposing geometric relationships such as tangency, concentricity, and so on. There is a rich literature on this subject, mostly concerned with the mechanics of solving constraints; e.g., [1, 2, 9, 3, 14].

Aside from this very important subject of how best to optimize performance of the constraint solver by restricting, but not overly limiting, the expressive power of allowed constraints, it appears that the question of interpreting constraints in a wide range of situations has not been deeply addressed.

The examples given before of instantiating cuts hint at the problems that arise in constraint-based geometries. A major problem is how to interpret correctly the user's intent in the presence of multiple solutions, and how to devise ways that allow the user to choose alternative solutions if necessary. We illustrate the issue with some examples obtained with Owen's constraint solver [7], a solver based on formulating a system of nonlinear equations ensuing from the sketched geometries and specified constraints.

Since geometric constraints routinely involve nonlinear equations, there are generally more solutions than one. As pointed out in [7], there are simple configurations where n constraints lead to $O(2^n)$ distinct solutions. Some of the solutions are symmetries, but some are, from a geometric perspective, structural alternatives that may or may not be useful to the application. As a symmetry example, consider the dimensioning of the triangle shown in Figure 7. There are four congruent solutions that can be obtained from each other by reflections.

A structurally different solution is seen in Figure 8. Here, the two solutions arise from the choice of where to center the connecting arc with respect to the lines. Clearly, the left choice is the plausible one in design applications. Let us adopt the left interpretation, and consider varying the angle. For the critical value of 90° we obtain the shape shown in Figure 9. The reason is probably that in the topological description of the configuration no provision has been made for zero-length arcs. We call this a *degeneracy error*. For the angle of 135° we obtain the shape shown in Figure 10. Here the problem is the fact that in the description of the constraint problem, the shape semantics has been identified based on a certain root selection of the constraint equations. This is a *root-identification error*.

To summarize, in addition to the problems of specifying and dimensioning features such that redundancy of the feature (e.g., unattached cut) or incomplete shape specification of the feature can be tolerated in a predictable, mathematically well-founded way, the presence of geometric constraints adds the problems of degeneracy and of root identification.

3 Research Problems

Having demonstrated some of the idiosyncracies found in current implementations of feature-based geometric design and constraint solving, we now discuss several basic research problems that arise.

3.1 Formal Geometry Languages

The trend towards generative geometry should be founded on a formal geometric design description that specifies a *family* of designs rather than a single design instance. In analogy to procedures in programming languages, specific instances of geometric design would be a function of parameter values such as specific dimensions, or geometric relationships that are prescribed. This approach has been called *parametric design* in the literature; e.g., [9].

In the past, parametric designs have been described by a CSG tree in which the individual primitives and their spatial interrelationship are defined by a set of constraint equations whose free variables constitute the design parameters. The advantage of this approach is that it is semantically well-founded. In consequence, with such an approach there would never be the problem of instantiating ill-defined geometries. The principal draw-back is that it appears to be an approach that is unresponsive to the design process, viz. the commercial success of Pro/Engineer, and that full Boolean operations are associated with a performance penalty.

Current practice seeks to supplant the CSG-based approach with a boundary-based approach in which the design steps are understood and implemented by local surface operations. Industry practice is to concentrate on the boundary representation and to define implicitly the semantics by the ensuing implementation. Of course, this is not satisfactory. Instead, we feel that the problem should be refocused on a higher level of abstraction, and that a high-level generative geometry description should be formulated from which the geometry instances are computed by a *geometry compiler*. We have called such representations *Ereps*, i.e., *editable representations*, and have argued in [6] that the benefits of such an approach include interoperability of geometric modeling systems, the ability to federate different systems into a problem solving environment, and an elegant solution to the legacy problem as explained later. It would also allow a firm conceptualization of the semantics of design operations without having to resort to an understanding of the particulars of any core modeling system.

A major problem is to comprehensively understand how a constraint-based feature definition that has been expressed in terms of cross sections and surfaces, will interact with the semantics of the solid operation it is supposed to specify.

The examples above have illustrated some of the difficulties achieving this, and there are others. We doubt that the problem can be satisfactorily solved by extending geometric data standards. To understand why, consider the problem of defining an angle between two line segments, as shown in Figure 11. In order to identify the acute angle α, rather than the exterior angle, we need to include in the geometric description some information about the orientation of the segments and of the angle. Suppose we have fixed the angle orientation as counter-clockwise, assuming both segments are oriented away from the angle. That is, α is obtained by turning the oriented segment $\overline{B,C}$ counter-clockwise into the oriented segment $\overline{B,A}$. Now assume that the position of A is determined from a distance constraint of A from the segment (B,C), and that we negate this distance so that A now must be positioned on the other side of (B,C). Keeping the angle α fixed, we should obtain the configuration of Figure 12. But then the angle orientation would have to be reversed in the geometry description. In other words, the topological description is influenced by the dimension values.

Ultimately, there are two ways of recording orientation:

1. By coordinatization of the geometric elements and evaluating the sign of certain expressions.

2. By specifying the order of points incident to lines and other geometric elements.

The first technique is the one adopted by all geometry representations in use. It is not suitable in this context, however, because the elaboration of constraints can change the coordinatization such that the ordering based on the old coordinatization no longer applies. The second method is untested in applications. It relates to the investigation of arrangements in classical mathematics [5], and is likely to profit from current research on oriented matroids; e.g., [13, 12].

3.2 Topological Degeneracy and Root Identification

In the example of Figure 9 we showed that in certain situations a topological element, here the circular arc, vanishes. In the example of Figure 4, a topological element should be ignored outright as irrelevant. It seems that the literature has not addressed this problem of *topological degeneracy*. The problem is connected to the problem of identifying which of several possible solutions of a system of constraint equations is the intended one (cf. Figure 7) in that, in a continuous deformation of a parameter value from the old to the new value, the geometry described passes through one or more topological degeneracy. It is important to investigate and understand how to isolate and handle topological degeneracies in an effective, and possibly interactive manner. In view of the possibly exponential number of potential solutions this analysis needs to research how to isolate which variables would participate in the root change, and whether they can be arranged in a dependency structure in which the change of one variable entails an implied change of some others.[1]

Such work would also make a contribution towards the important issue of identifying under- and overconstrained geometries and suggesting to the user certain changes that would remedy the situation.

3.3 Characterization Theorems

We have stated that solving geometric constraints is a problem that, from the applications point of view, requires restricting the expressive power of the constraint system. The restrictions found in the literature are often ad-hoc, with no theoretical characterization of scope, and hence do not admit a strict semantics. Some characterizable restrictions have been proposed, such as limiting to ruler-and-compass constructible configurations. This has the advantage that the constraints can be solved using only linear algebra and computing square roots, [4]. In practice, such a system may not realize the full expressive power of the restricted class because of efficiency considerations. Note also that some very simple dimensioning schemes cannot be solved with ruler and compass alone, such as the one shown in Figure 13.

Given an efficient approach to constraint solving, research that characterizes the expressive power of the solver would be invaluable to certify the completeness of any implementation and would be a basis for giving a precise semantics.

3.4 Legacy Considerations

A geometric modeling system is a considerable investment of time and effort, and it is no surprise that the preservation of existing core modelers is a high priority of commercial vendors as well as many research groups. In [6] we argued that devising a semantically well-defined Erep is an elegant way to preserve the core modeler while adapting to targeted applications and potential changes in user-requirements. Despite the performance penalty mentioned before, compiling generative geometries to core modelers that have a CSG architecture is therefore a serious consideration. However, doing so naively would entail some very serious research problems.

Consider, for example, the shape drawn in Figure 14, and assume that it is the cross section of a solid of extrusion. A clever geometry compiler might decide to consider this shape the union of a block and a disk. Doing so has the advantage that a valid cross section is obtained under all valuations of d_1, and, in particular, with $d_1 = 10$, which obtaines a square. However, the more natural interpretation is the one shown in Figure 15, in which a corner of the square

[1]It is interesting to note that ruler-based constructions, some of the simplest geometric constructions possible, must necessarily have an associated system of algebraic equations that has 2^n real solutions [4].

is rounded. This more natural interpretation is, however, inconsistent with the CSG semantics.

We may adopt a two-stage translation step in which the cross section is reinterpreted based on the topology of the boundary and is then mapped to a new CSG expression. This entails clearly the very difficult subproblem of Brep to CSG conversion, which is only partially solved at this time; [10, 11]. Another approach would be to generate primitives from instantiated geometries with a boundary-based semantics. The research issues associated with doing so have been discussed before, and concern primarily finding a rigorous and reasonable semantics of generative geometry representations, i.e., understanding the implications of orienting geometric configurations without reliance upon a-priori coordinatization.

4 Summary

We have illustrated that the trend towards generative geometric design with features poses a number of research problems whose proper solution involves foundational work. The full treatment of these issues requires developing coordinate-free geometry representations, and to identify, when using these representations, all topological degeneracies that affect which of several possible solutions of the given constraints is the proper one. Furthermore, one needs to address the problems of diagnosing and correcting under- and overconstrained configurations, especially in view of the fact that the native representation of the constraint solver does not necessarily correspond directly to the way in which the user has formulated these constraints.

References

[1] B. Bruderlin. Symbolic computer geometry for computer-aided geometric design. In *NSF Conf. Advances in Design and Manuf. Sys.*, Tempe, AZ, 1990.

[2] B. Buchberger, G. Collins, and B. Kutzler. Algebraic methods for geometric reasoning. *Annual Reviews in Computer Science*, 3:85–120, 1988.

[3] T. L. De Fazio and *et al.* A prototype of feature-based design for assembly. Technical Report CSDL-P-2917, The Charles Stark Draper Laboratory, Inc., Cambridge, MA, 1990.

[4] D. Hilbert. *Grundlagen der Geometrie.* B. G. Teubner, Stuttgart, 1956.

[5] D. Hilbert and S. Cohn-Vossen. *Geometry and the Imagination.* Chelsea Publications, 1983.

[6] C. M. Hoffmann and R. Juan. Erep, a editable, high-level representation for geometric design and analysis. Technical Report CER-92-24, Comp. Sci., Purdue Univ., 1992.

[7] J. Owen. Algebraic solution for geometry from dimensional constraints. In *ACM Symp. Found. of Solid Modeling*, pages 397–407, Austin, Tex, 1991.

[8] A. Requicha. Mathematical models of rigid solids. Technical Report Memo 28, University of Rochester, Production Automation Project, 1977.

[9] J.R. Rossignac, P. Borrel, and L.R. Nackman. Interactive design with sequences of parameterized transformations. Technical Report RC 13740, IBM Research Division, T.J. Watson Research Center, Yorktown Heights, New York, 1988.

[10] V. Shapiro and D. Vossler. Construction and optimization of CSG representations. *Computer-Aided Design*, 23:4–20, 1991.

[11] V. Shapiro and D. Vossler. Efficient representations of two-dimensional solids. *Transactions of ASME, Journal of Mechanical Design*, 113:292–305, 1991.

[12] B. Sturmfels. Computational algebraic geometry of projective configurations. *J. of Symbolic Computation*, 11:595–618, 1991.

[13] N. White. *Combinatorial Geometries.* Cambridge University Press, 1987.

[14] Wu Wen-Tsün. Basic principles of mechanical theorem proving in geometries. *J. of Systems Sciences and Mathematical Sciences*, 4:207–235, 1986.

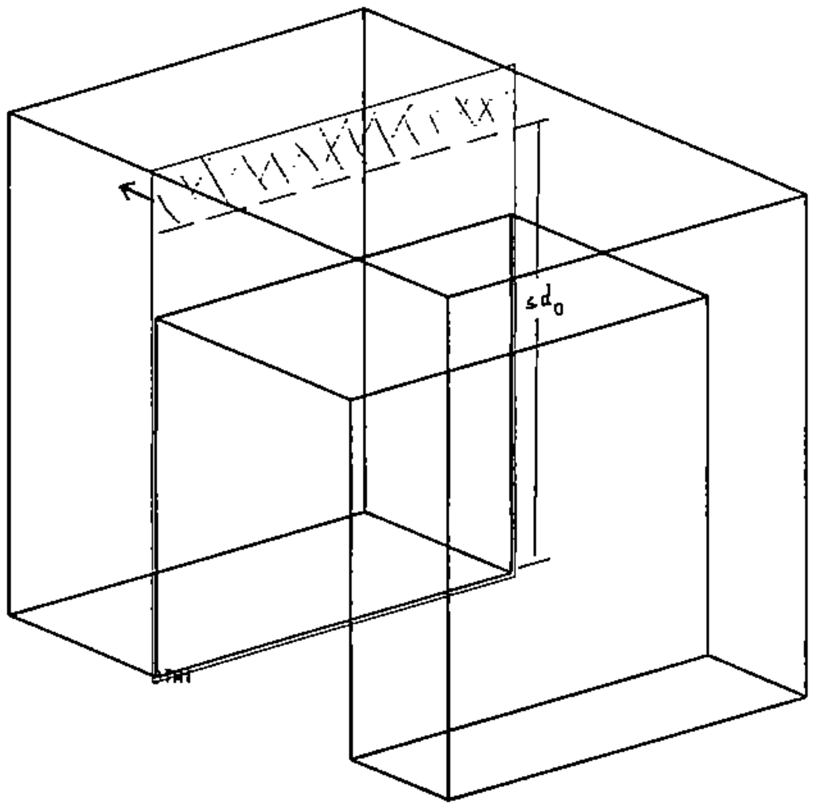

Figure 1: Object and Cut Definition

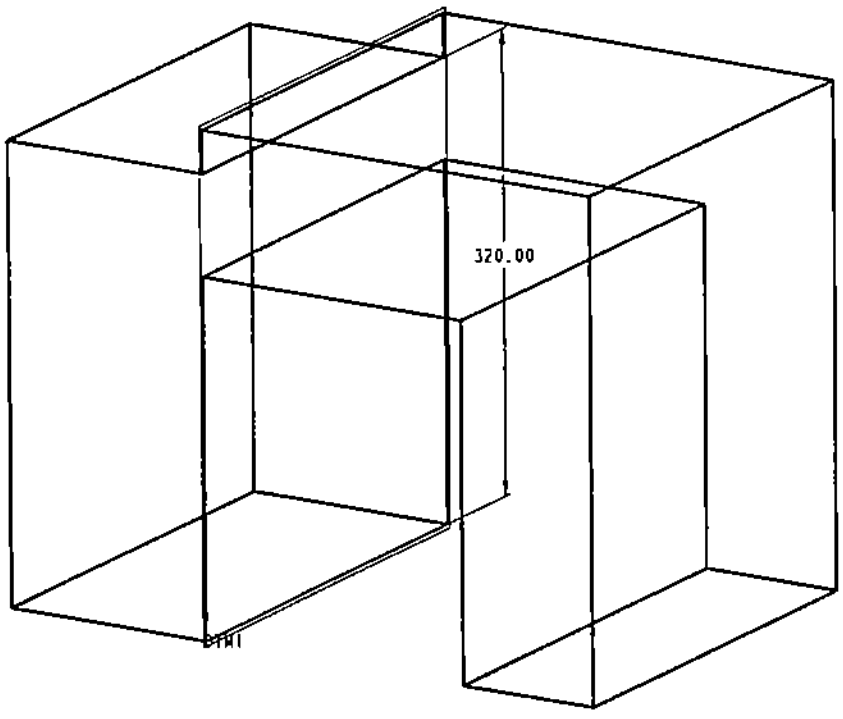

Figure 2: Step Created with $sd_0 = 320$

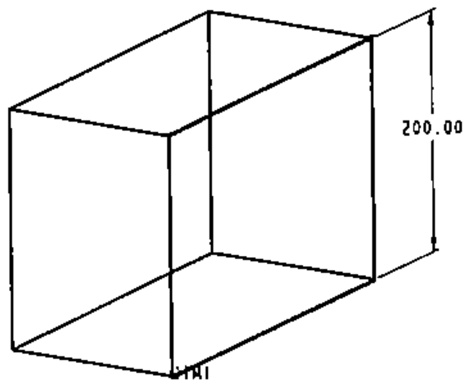

Figure 3: Block Created with $sd_0 = 200$

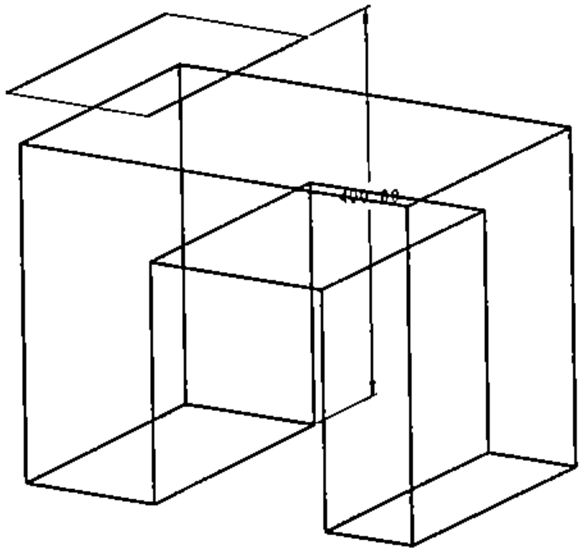

Figure 4: Error Generated with $sd_0 = 400$

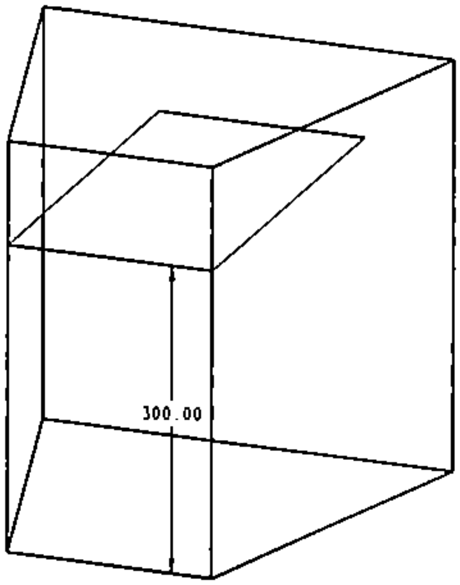

Figure 5: Incomplete Profile

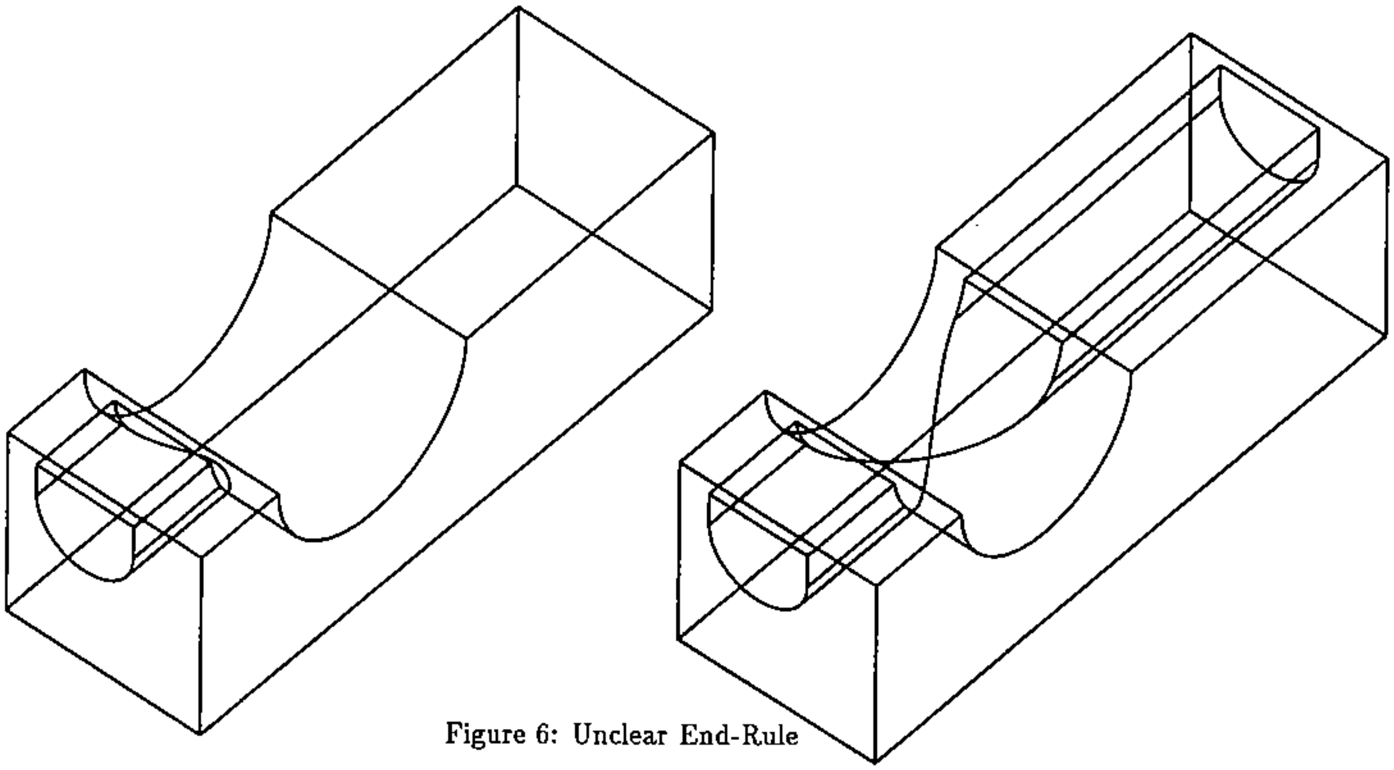

Figure 6: Unclear End-Rule

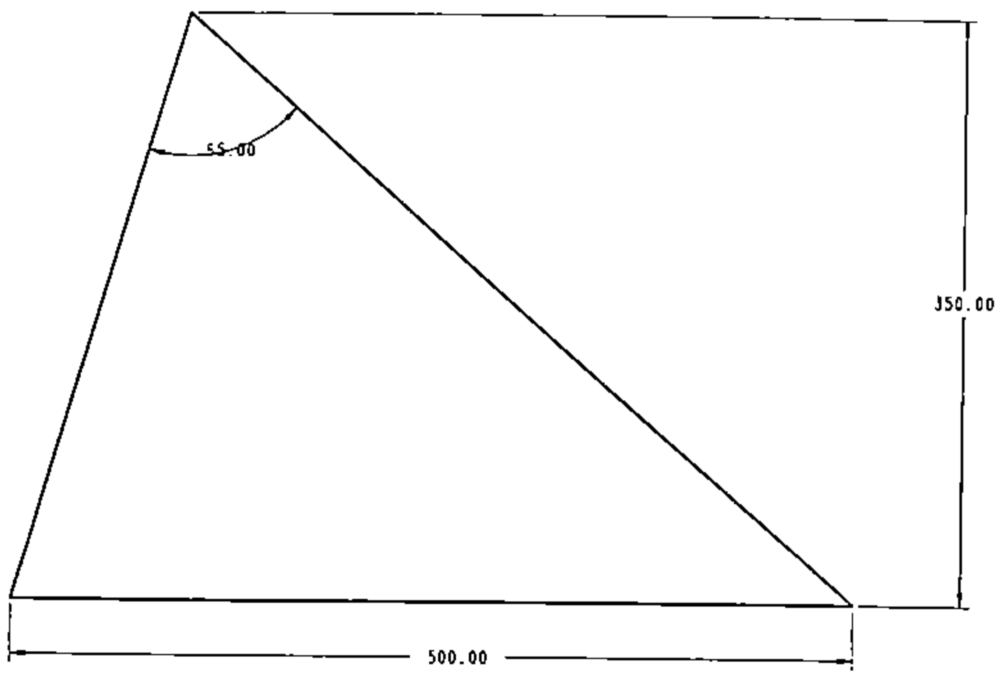

Figure 7: Triangle Dimensioning Schema with Symmetric Solutions

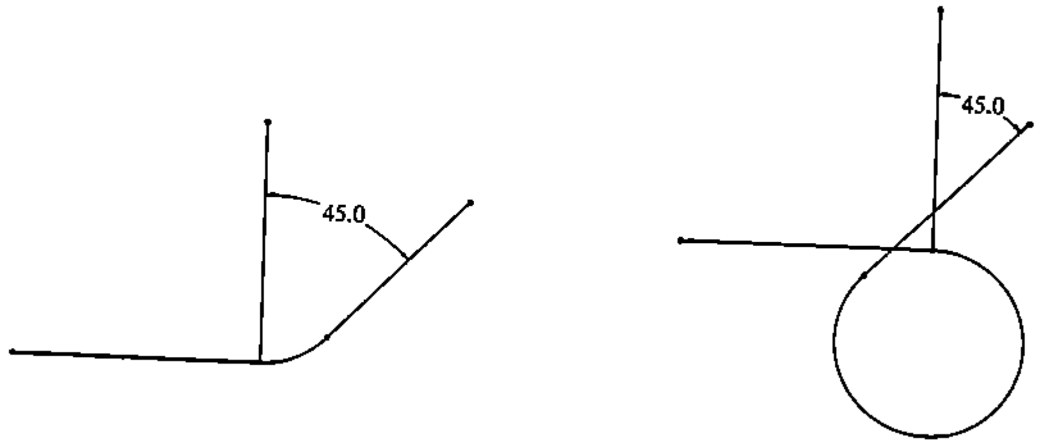

Figure 8: Two Solutions of Angle Constraint

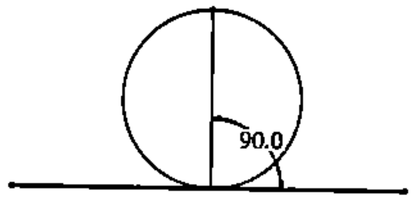

Figure 9: Critical Angle of 90°

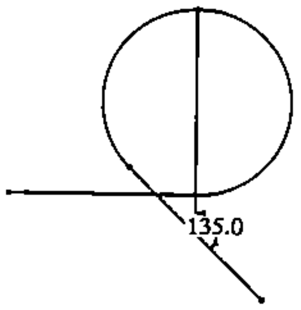

Figure 10: Angle of 135°

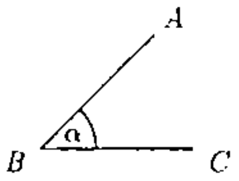

Figure 11: Angle Between Two Segments

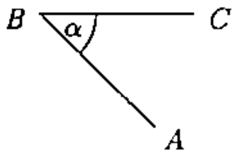

Figure 12: Angle Between Two Segments

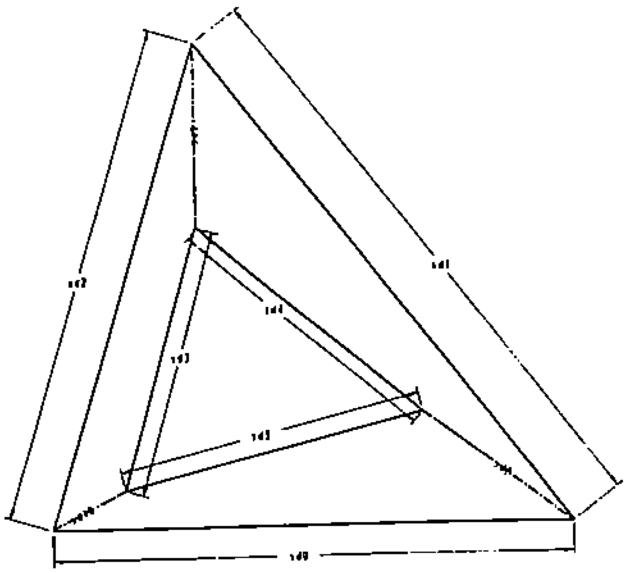

Figure 13: A Dimensioning Scheme not Solvable Using Ruler and Compass

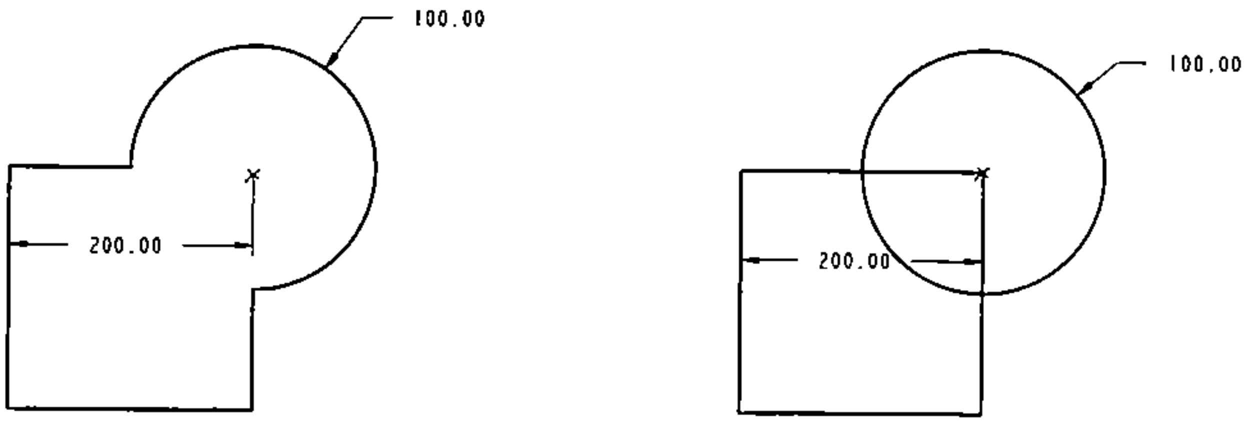

Figure 14: Example Cross Section with CSG Interpretation

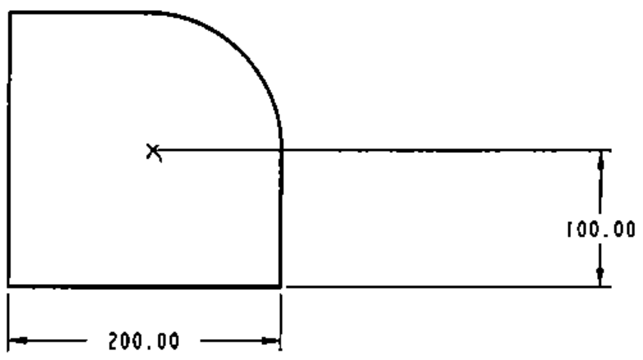

Figure 15: Natural interpretation with $d_1 = 10$, inconsistent with CSG semantics.