1993

Semantic Problems of Generative, Constraint Based Design

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

Report Number:
93-062

SEMANTIC PROBLEMS OF GENERATIVE, CONSTRAINT-BASED DESIGN

Christoph Hoffmann

CSD-TR-93-062
September 1993

Semantic Problems of Generative, Constraint-Based Design*

Christoph Hoffmann†
Department of Computer Science, Purdue University
West Lafayette, IN 47907-1398

Report CSD-TR-93-054, August 1993‡

1 Introduction

A new generation of CAD systems has become available in which geometric and dimensional constraints can be defined and solved. The methods such systems offer to instantiate generically defined models from user-supplied dimension values constitute capabilities that far outstrip our current understanding. Basic research will be needed to gather the facts with which to discuss the potential and implications of this modeling technology, and such discussions must be focused by major user groups articulating their needs in the context of broad application areas.

This process will have to run its course if there is to be a dependable emerging data exchange standard that exploits the new technologies. Even if this goal is postponed, a clear understanding of the semantics should precede any major system implementation — unless one considers it acceptable that a CAD system behaves in ways unexpected by the user and at variance with the user's design intent.

In this note, I illustrate the nature of some of the problems that constraint-based,

*Presented at the Conference *Practice of Computer Aided Geometric Design — What CAD systems are really capable of*, Tauberbischofsheim, Germany, September 1993.

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

‡This report and others are available via anonymous ftp to arthur.cs.purdue.edu, in directory pub/cmh and subsidiaries

generative design entails with the help of very simple examples that were computed with Pro/Engineer Version 11.0. The choice of this system for illustrating semantic problems is incidental: I used Pro/Engineer merely because it was conveniently available. Moreover, the examples given clearly indicate problems that are inherent to the nature of constraint-based, generative design, and so must be addressed not only by this but by all CAD systems of comparable or greater capabilities.

In addition, I give a brief overview over ongoing research that addresses these problems in a systematic way.

2 Three Problem Areas

2.1 Cuts and Blends

Feature attachment is a term that has been used to describe solid modification operations in which, conceptually, new geometry is fitted to existing geometry in a manner that appears to be based on Boolean operations [6]. For example, as illustrated in [2], most users are unfamiliar with the way in which a profiled cut is constructed in Pro/Engineer. It is often assumed that a profiled, one-sided cut is eventually implemented by a Boolean operation between two solids. But the behavior of cuts under a range of different dimension values indicates that this is not the case. In particular, a one-sided cut as defined in Figure 1 on the left may well modify geometry on both sides of the sketching plane, as seen in the right variant of the design. For an explanation of the semantics of the operation see [2].

The example does not expose a hard problem, but it drives home the point that the familiar concepts CAD users have developed based on their experience with Boolean operations and the design of specific shapes does not necessarily constitute a reliable guide to the details of generative design. Clearly, there ought to be a discussion, among users and vendors, whether the manner in which cuts are made in Pro/Engineer is desirable. Should a one-sided cut ever extend to both sides of the sketching plane? How should one define one-sided cuts if a profile has been sketched on a curved surface that does not clearly partition space into two separate regions on either side? Should a cut operation ever result in an "unattached cut" — or should redundant cuts be ignored? These are only some of the simpler questions that must be addressed.

Another common misconception is exactly how a fillet or round should be constructed. There are no commonly accepted standards how a constant-radius blend of an edge should end at complex vertices, especially if blending requires extending surfaces incident to one or both ends of the edge. Figure 2 shows a regeneration variant from Pro/Engineer Version 9.0 that most would agree is in error.[§] Providing

[§]This error seems to be corrected in Version 11.0

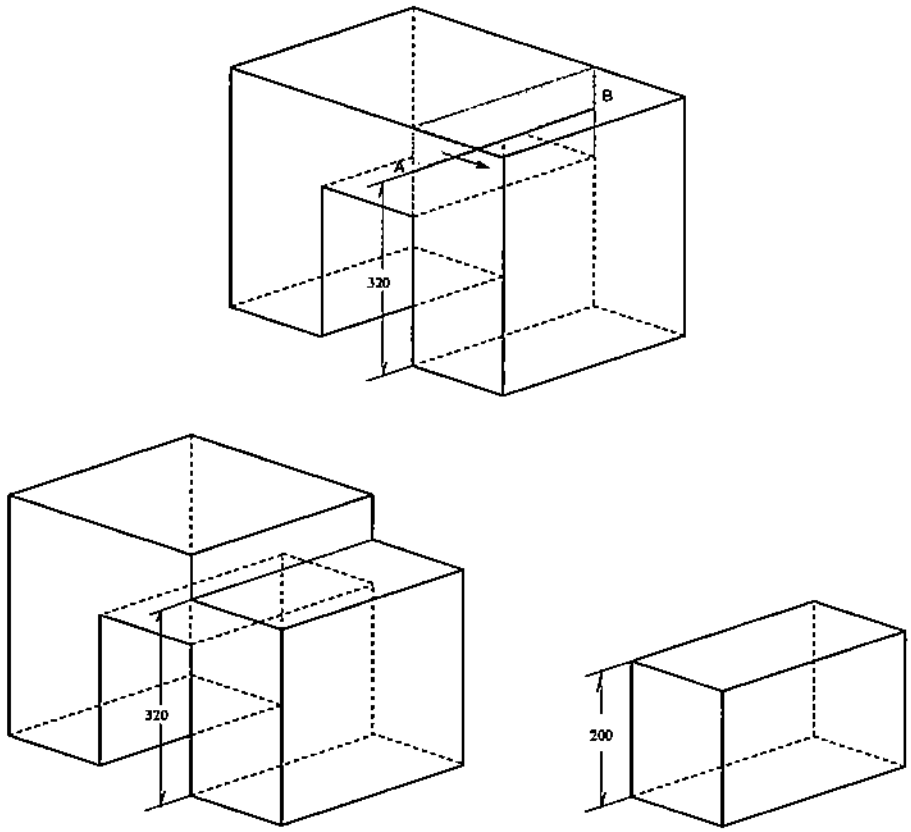

Figure 1: A one-sided cut in Pro/Engineer, as sketched above, can have a two-sided effect, as shown in the bottom right variant.

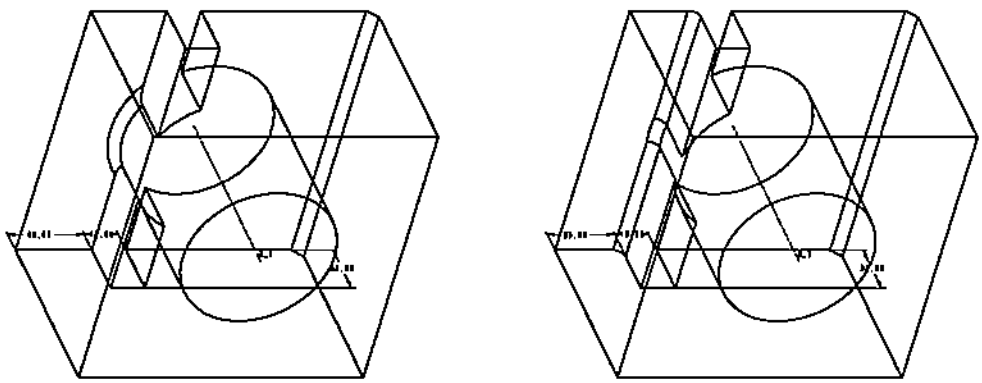

Figure 2: Blend end rule error in variant.

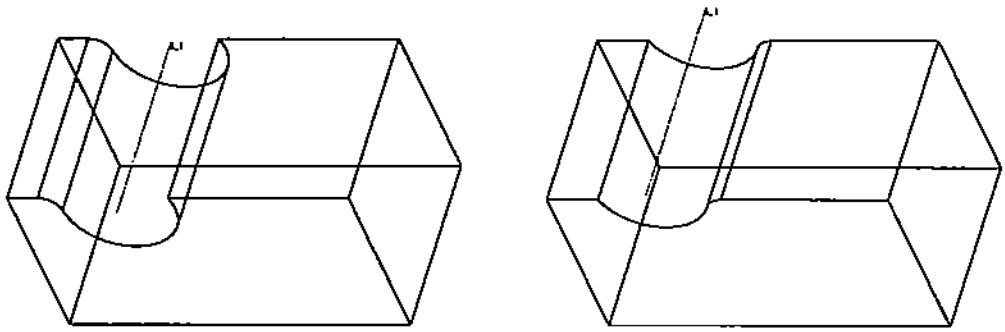

Figure 3: Persistent ID error in variant.

extensive detail about the blending surface is in conflict with convenience and design speed, yet without clear rules and conventions that are meticulously implemented by the CAD system a user can neither develop a firm grasp of what to expect from the CAD system, nor can he or she reliably exchange generative model specifications and between CAD systems.

2.2 Persistent Identifiers

For obvious reasons, design interfaces are centered around visual design gestures. While visual design gestures interact with images that are representations of *instances* of a generic model, their intent is to modify the generic model itself. However, geometric elements identified visually need not correspond explicitly to design gestures ever made, and to capture design intent on the basis of the explicit gestures is a profound challenge that has neither been solved to-date, nor accounted for well in recovery mechanisms. Consider Figure 3. The shape shown there has been designed in three steps:

1. A block has been created by drawing a rectangular profile and extruding it by a specified depth.

2. A round slot has been cut by extruding a circular profile across the block.

3. An edge round of constant radius has been defined by visually identifying one edge and specifying a radius for the round.

After so defining the model on the left, the dimension value locating the center of the slot profile is changed. On regeneration, the edge round "jumps" to a different edge.

The basic problem is to identify the correct edge. The edge that was identified on the left corresponds implicitly to the intersection of the slot surface with the top of the block. Both surfaces, in turn, are the trajectory of explicitly drawn geometric elements, and so can be generically represented. However, the edge on the right, to which the blend jumps after relocating the slot center, is also the intersection of the

two surfaces. No generic way is evident that could be generated automatically and distinguish algorithmically between the two edges. Since regeneration of the slot must precede the existence of the two edges in the new shape instance, any annotation of the previous model instance in which the edge was selected has been lost.

From a technical point of view, the generic describability of the edge in a unique way appears to be related to the ideas involved in converting from Brep to CSG [6]. It may also be a matter that can be solved better by three-dimensional variational constraint solving, a technology that is currently only rudimentary. With no good solutions at the moment, one should have effective mechanisms for the user to redirect variant regeneration. Surprisingly, commercial CAD systems have not yet developed them.

2.3 Constraints Vs. Design Intent

It is widely known that a well-constrained geometric problem may have exponentially many solutions; e.g., [5, 4]. The reason is that constraints such as the distance between two points correspond to quadratic algebraic equations, so that the mathematical semantics of the constraint problem is described by a system of simultaneous nonlinear equations. If the geometric problem is well-constrained, the equation system is zero-dimensional, and the number of solutions corresponds to its Bezout number.

As example, consider Figure 4. The dimensioning schema shown in the upper left panel is well-constrained when adding that the arc be tangent to the adjacent segments, and that the other two segments be perpendicular. For this problem four distinct solutions are shown, each mathematically satisfying the given constraints. They do not exhaust the possibilities. On the other hand, the design application for which the problem was formulated most likely will require exactly one of the possible solutions. To find the "intended" solution, constraint solvers apply a number of heuristics that try to second-guess the user's intention. The heuristics can be rules that preserve, for example, on which side of a directed segment in the sketch an adjacent arc has its center.

In situations where the user sketch is in some sense "close" to the sketch ultimately needed, the heuristic rules succeed with high probability. However, CAD systems have not developed useful paradigms for selecting the right solution when the heuristic rules fail. Consequently, in such cases the user often has no other recourse than to delete some constraints and attempt to do better with a different constraint schema, by trial and error. Deleting geometric elements, another ad-hoc way for the user to cope with poor solutions, is more risky because attached features could be lost in the process.

It would seem reasonable to ask the user to sketch with some accuracy, so as to maximize success of the heuristics. But since the strength of constraint-based design is precisely the ease with which to derive variant designs, for instance by changing some dimension values, it is necessary to develop sophisticated paradigms for selecting different solutions of the same constraint system; [1]. For example,

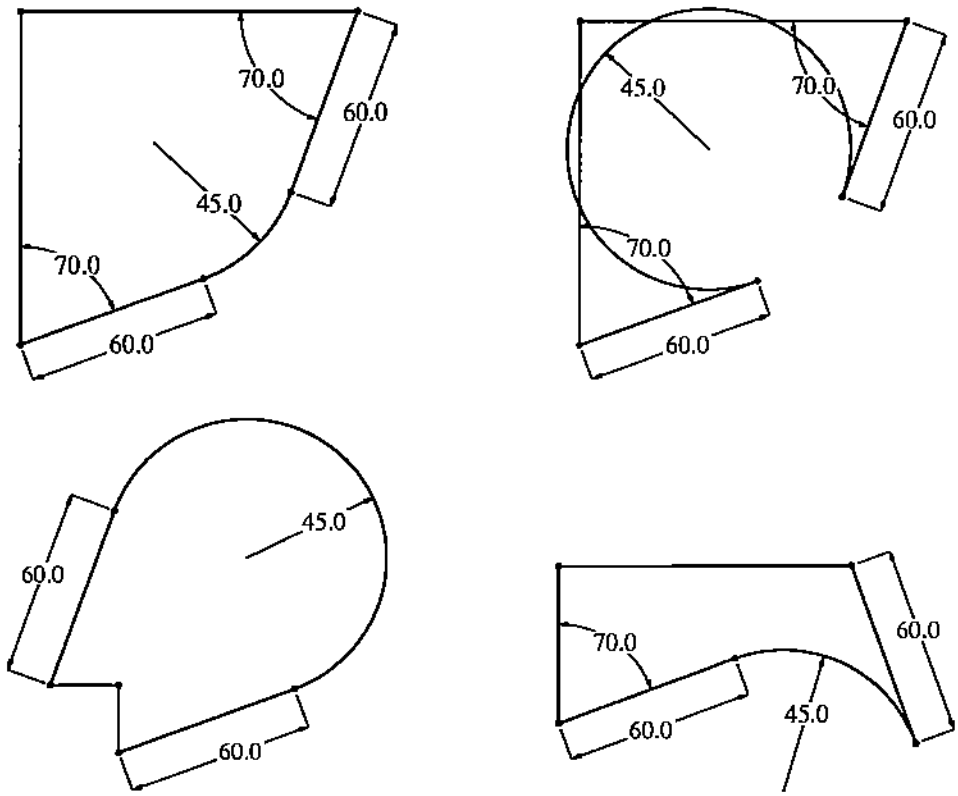

Figure 4: Four structurally different solutions of the same constraint problem.

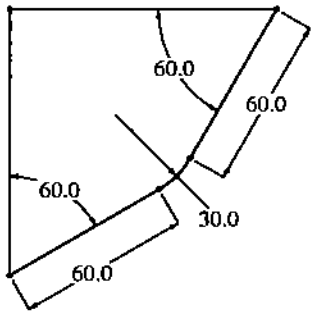

Figure 5: Rounded quadrilateral

consider the constraint problem shown in Figure 5. Quite likely, the intent of the arc is to round one corner of a quadrilateral. But if this intent is instead recorded as the relative position of the center point with respect to the adjacent oriented segments, then diminishing the angles to less than 45 degrees would result in the left solution shown in Figure 6. Instead, the solution to the right would be correct. Moreover, if both angles are exactly 45 degrees, the rounding arc would be redundant and ought to be suppressed. See also Figure 7.

The fact that constraint problems have multiple solutions, and that the constraint solvers differ between different CAD systems, implies in particular that it is not possible to communicate in a neutral format a variational geometric constraint problem that is not fully instantiated and satisfies all constraints. Moreover, it will not be possible to predict which solution another system would determine if the dimensions were changed in any way. Of course, it is also likely that the user will not have a deep understanding of the way the solver works, so that even without exchange between different CAD systems one would have to question whether users can consistently devise dimensioning schemata that work as expected.

All these difficulties underscore the need to develop better paradigms for determining the right solution of a constraint problem. Such paradigms should be intuitive,

Figure 6: Fixed circle orientation leads to erroneous solution left; changed circle orientation preserves rounding function right.

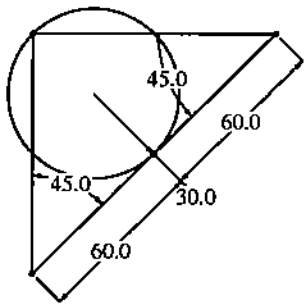

Figure 7: Redundant round not recognized.

mathematically well-founded, and should not unduly hinder implementing different solver strategies. Ideally they would be fully automatic, but they could include user interaction at certain junctures.

As pointed out in [1], there are a number of possibilities that await further exploration. Other ideas will undoubtedly be advanced as the community begins to consider this issue.

3 Geometry Compilation of Editable Representations

In our research we develop a high-level generative design representation that is constraint-based with variational geometric constraint solving [3]. In reference to the convenience with which such design representations can be modified and edited, we have called our representation an *editable representation* (Erep).

To address the fundamental problems illustrated above and in [2], we have chosen to develop this representation without accounting for the individual capabilities and characteristics of any core geometric modeler that might be used to implement the semantics of the representation. Thus, the translation of the high-level geometry representation into specific geometry instances is analogous to the translation of a high-level programming language into machine language. The core modeling engine is then the abstract machine, while the high-level Erep is the program. As explained in [3], the analogy is in some respects inexact, but suggests the strong possibility of federating different modelers, and, in particular, would allow to overcome functional barriers such as the difficulty of integrating design and analysis. At the time of writing, the status of our project is as follows:

1. We have completed a variational 2D geometric constraint solver along with an interface that allows sketching, regeneration, as well as comprehensive recovery mechanisms in case the solver has identified a different solution than the one needed by the application. The solver is extensible and not limited to ruler-and-compass constrictions.

2. We have completed a pilot implementation of an Erep compiler that translates and instantiates a textual geometry representation into design instances in boundary representation. As core geometric modeler we presently use ACIS.

3. We are completing the integration of the pilot compiler with the constraint solver and our graphical design interface.

We have carefully insulated these software components from underlying core modeling algorithmics. If there are dependencies, we believe they ought to be on the design interface rather than the underlying modeler, because we believe that the design interface ought to account to the user's design needs which are likely to be more permanent than the individual operations that current modeling technology can provide.

The coming steps in the project include better 3D constraint capabilities, experimenting with different mechanisms for computing persistent IDs, and new methods for interacting with the user when different design instances are to be explored.

Acknowledgements

For complete literature reviews of related research see [1, 3].

References

[1] W. Bouma, I. Fudos, C. Hoffmann, J. Cai, and R. Paige. A geometric constraint solver. Technical Report TR-93-054, Purdue University, Computer Science, 1993.

[2] C. M. Hoffmann. On the semantics of generative geometry representations. In *19th ASME Design Automation Conference*, 1993.

[3] C. M. Hoffmann and R. Juan. Erep, a editable, high-level representation for geometric design and analysis. In P. Wilson, M. Wozny, and M. Pratt, editors, *Geometric and Product Modeling*, pages 129–164. North Holland, 1993.

[4] D. Kapur. A refutational approach to geometry theorem proving. In D. Kapur and J. Mundy, editors, *Geometric Reasoning*, pages 61–93. M.I.T. Press, 1989.

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

[6] V. Shapiro, H. Voelcker, and D. Vossler. Boundary to CSG conversion: current status and open issues. IFIP WG5.2 Conference on Geometric Modeling for Product Realization, Rensselaerville, 1992.