The 5th International Symposium on Robotics in Construction June 6-8, 1988 Tokyo, Japan

Geometric Reasoning in Computer Integrated Building Construction

Robert F. Woodbury,[i] Steven J. Fenves,[ii]
Nelson C. Baker,[iii] Richard W. Quadrel,[iv] Clauss B. Strauch,[v]

Carnegie Mellon University
Pittsburgh, Pa. 15213
tel. (412) 268-8853
email rw@cad.cs.cmu.edu

ABSTRACT

Geometric reasoning, the integration of geometric representation and inference in advanced computer systems, is presented as an issue at the forefront of research in construction automation. The unique demands that construction automation poses on such reasoning are discussed. An architecture that provides a structure for geometric reasoning is presented and results from a prototype implementation are shown. A project to develop geometric reasoning in the construction domain of panelized building systems is introduced. Within this project, two exemplary applications, structural/architectural design and construction sequence planning, each supported by the same geometric reasoning facility, are being demonstrated.

1. INTRODUCTION

A most challenging issue for current construction automation research is representation of and reasoning on geometric information. Geometry is ubiquitous in construction and reasoning upon it is an essential constituent of the major intellectual issues in the construction domain. With increasing automation of the construction industry, from design to fabrication, comes a need for capable geometric facilities to underly new computer based systems for representing, designing, communicating, planning, and acting. This need becomes all the more acute when the integration of many programs into a large-scale construction system is contemplated. The rich exchange of information in any integrated system appears to demand that the various components can communicate with common representations, most of which will be largely phrased in geometric terms.

Future construction systems will need to distinguish themselves by several characteristics:

- *Multiple Levels of Abstraction.* Effective problem-solving within the construction domain, be it physical design, construction planning, or robot motion planning, is usually hierarchical. Problems are initially solved at a high level of abstraction and then refined to consider increasing levels of detail. This top-down approach provides: a reduction of complexity, an ability to reason with incomplete information, and an ability to divide the problem among specialists. It is very useful if the representations employed by the various parts of the overall process reflect this hierarchical organization and provide special abstract views for each of the processes.
- *Evolutionary Development over Time.* The models upon which problem-solving processes act are generally developed *over time*, usually by a group of agents, human or machine. For most of its existence the model is incomplete and undergoing evolutionary (incremental) change. It is useful to be able to perform tasks prior to the complete development of a model for two reasons: first,

[i]Assistant Professor, Department of Architecture

[ii]Sun Company University Professor, Department of Civil Engineering

[iii]Graduate Research Assistant, Department of Civil Engineering

[iv]Graduate Research Assistant, Department of Architecture

[v]Researcher, Department of Architecture

greater flexibility is achieved by a system that can still function in the absence of complete information; and secondly, problem-solving based upon incomplete information can provide reasoning needed by other parts of the construction process before it would otherwise be available.

- *Use of Advanced Computational Concepts.* An important aspect of new construction capabilities is the emerging use of artificial intelligence methods such as declarative, object-oriented and rule based programming; search; planning; and constraint manipulation. These techniques interface awkwardly with existing systems for geometric modeling which are based on rigid modeling schemes and cumbersome procedural interfaces.

These features impose compelling and difficult requirements on construction reasoning and on the geometry at the core of that reasoning. In this paper we report our approach and current work towards a geometric reasoning facility which meets these challenges. Our strategy has been to consider two very different activities, structural/architectural design and construction sequence planning, chosen to reveal diverse geometric requirements. For each of these activities, we are developing automatic programs to achieve a significant part of the activity's goals. Each of these programs use identical geometric representations, manipulations and queries, that is an identical geometric reasoning facility. The initial design of the geometric reasoner is based on an organization developed by one of us (Woodbury) as part of his PhD thesis[1].

2. AN ARCHITECTURE FOR GEOMETRIC REASONING

One of the purposes of a geometric reasoner is to allow other programs access to geometric information in a manner transparent to actual methods of representation and computation. Our architecture (presented at length in[1]) employs four concepts which support such a transparent view onto geometry:

- Class descriptions
- Features
- Abstractions
- Constraints

For purposes of exposition we introduce a simple, abstract world consisting of two cuboids, and a single relation, *ON*, defined in terms of other, simpler geometric relations. This world is analogous to the *Blocks World* of various theoretical task planners[2, 3], with the crucial exception that it provides a geometrically meaningful definition of *ON*, in contrast to the situation in the blocks world where *ON* represents nothing more than a LIFO stack. We show, using all four of our concepts, how the *ON* relation is expressed and computed.

Models of objects are a necessary precondition to expression of spatial relationships. These models are defined through a hierarchy of *class descriptions*,[vi] each of which provides some information on the modeling formalism, data structures and methods that are necessary to create a complete model. In our world, this class hierarchy has four strictly geometric levels, corresponding to: method definition for solids; topology definition for cuboids; definition of topological cuboids as solids in three dimensional space; and, instantiation of those solids to an actual location in space. From classes, we can create the cuboids needed to model the objects in our world.

Simple cuboid models are not in themselves sufficient to express spatial relationships between objects. Precise specification of these relationships must involve parts of these models. We call these parts *features* and provide a mechanism for their representation and selection. A feature is a labelling mechanism. It permits the assignment of a name or identification to a portion of a model for the purpose of consequent query or reasoning on that part. Consider the cuboids from our abstract world. When the two cuboids are related by *ON*, two of their faces will be adjacent. Features provide a capability to identify the appropriate faces and a place to store data and methods that concern them. A feature may be part of the description of a class, in which case its specific label or identifying procedure is inherited automatically whenever an instance is created. In our world, once the features of *TOP-FACE*, *BOTTOM-FACE*, and *ANY-SIDE-FACE* are defined, the arguments for actual spatial relationships exist.

These features are themselves complex objects. They contain too much information and in a form inappropriate for development of a mathematical description of *ON*. To solve this, we create *abstractions* of the

[vi]Readers unfamiliar with object oriented programming are encouraged to refer to Keirouz[4].

features. These *abstractions* concisely express, in a simple data structure, all of the atomic values required to completely describe *ON*. Abstractions are partial representations of features. They are created upon demand, and their potential existence is inherited from the class definitions of the features which they partially represent. In our world of cuboids, an abstraction of a face to a *face-equation* and to a *center-line* is sufficient to completely express *ON*.

With abstractions, we finally have sufficiently simple models to provide a precise definition of *ON*. Informally, our definition of *ON* implies: alignment between center-lines of top and bottom faces; parallelism between side faces; and, plane equality (with direction reversal) between top and bottom faces, as shown in Figure 2-1. We use a *constraint description language* and a solution procedure (propagation) on statements of the language to first express and then solve these mathematical assertions. The constraint language is declarative, in that it only contains expressions of the states that we wish to hold in the models; actual determination of those states is left to a computation device independent of the description. In our world, this means that we can make statements about the relationships between the cuboids and simply invoke the constraint solution procedure to determine actual locations.

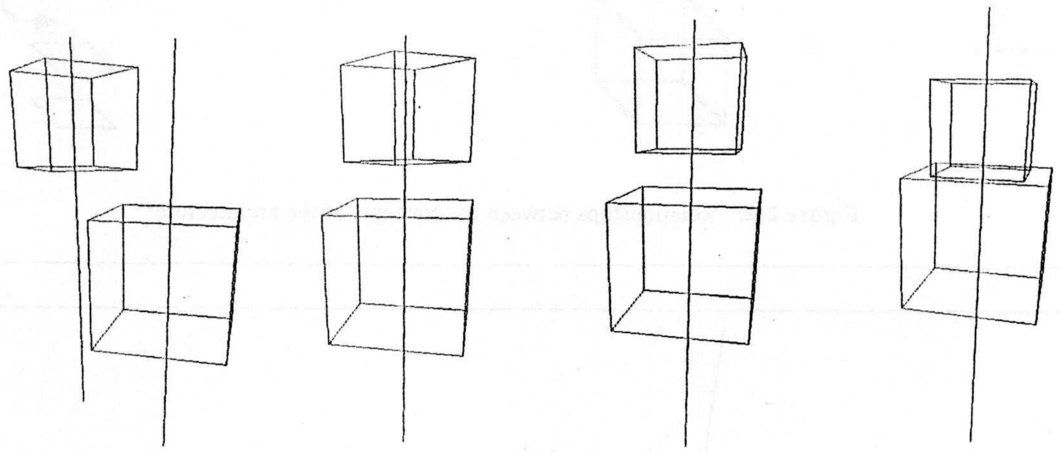


Figure 2-1: Successive stages in the specfication of *ON*

The architecture is summarized in Figure 2-2. It uses classes to describe types of objects including properties and procedures which can act on those objects. Instantiation from those classes produces models of geometric objects. Features denote portions of a model of interest for a specific purpose. Abstractions are used to isolate from those features specific representations for use by reasoning mechanisms. One such reasoning mechanism is that of constraint satisfaction, which uses a declarative constraint description language and an independent, general solution procedure.

This architecture has been implemented as a prototype and demonstrated in the domain of robot task planning. With a very small amount of code (about 100 lines), we were able to build a software demonstration of a robot laying a brick wall. In this demonstration, shown in Figure 2-3, a robot, represented as an *abstraction* consisting of its *A* and *T* matrices, is directed to construct a brick wall around an outline, also expressed as an abstraction. Brick locations are determined with a simple set of rules that employ geometric reasoning functions in their conditions. Constraints are used both to move the robot and locate the bricks in their chosen places.

3. KAHN: A PROJECT FOR GEOMETRIC REASONING IN CONSTRUCTION

The geometric reasoning architecture outlined above provides a general approach to the organization of geometric information into a comprehensive system, but is mute on the specific requirements of building construction. As a means of revealing these requirements and as a focus and goal for our efforts we have chosen two complementary activities from very different stages of the building process: structural/architectural design

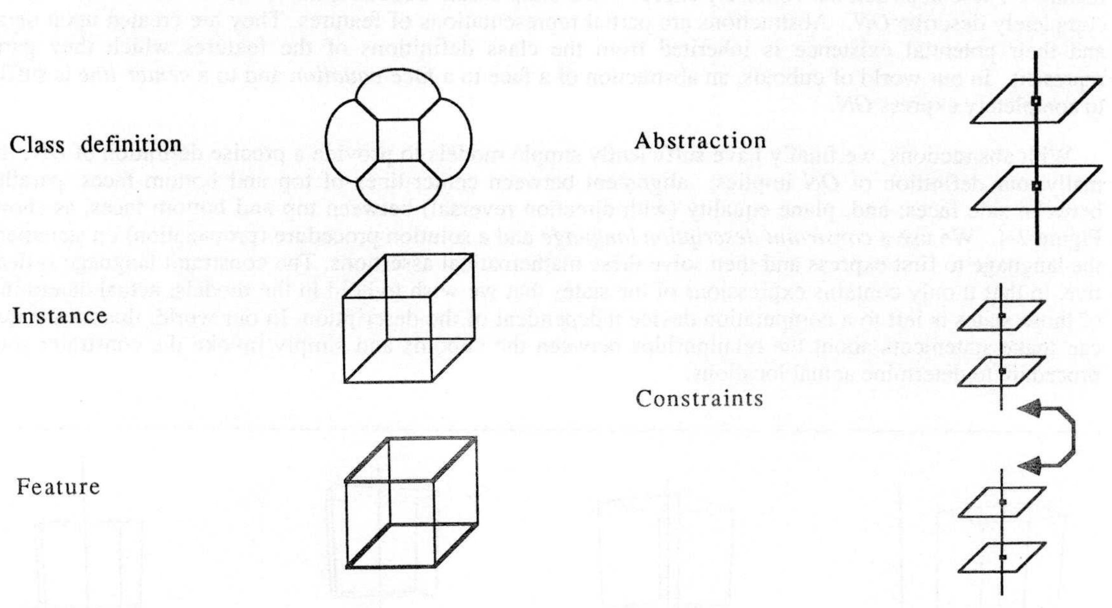


Figure 2-2: Relationships between the concepts of the architecture[vii]

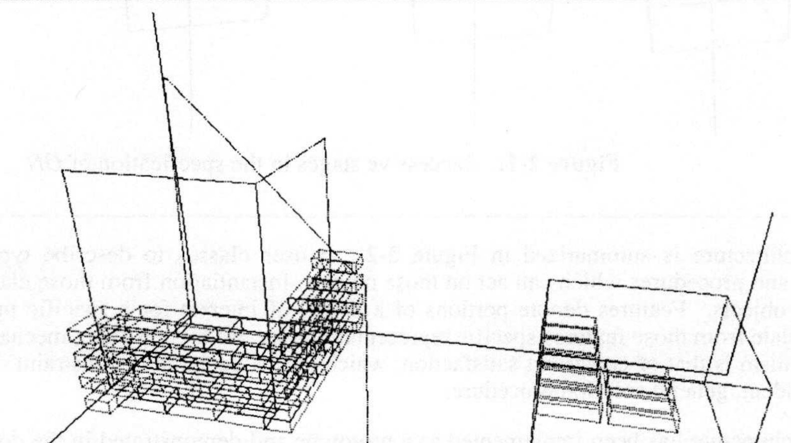


Figure 2-3: Demonstration of simple robot task planning using the prototype geometric reasoner

and construction sequence planning. In each of these domains we are developing programs that are predicated

[vii]The graphic opposite the *Class definition* label refers to the Schlegel diagram used in portraying boundary representation topologies.

upon fluid access to appropriate geometric information. We are simultaneously developing theory for, and an implementation of, a geometric reasoning facility to serve both applications. As a further restriction, we are using the construction type known as precast panelized building systems as the domain for our work. We call this project KAHN, after Louis Kahn, a 20th century architect well-known for the geometric rigor of his designs.

3.1. Precast, Panelized Building Systems

The scope of this study addresses the structure and space of prefabricated panelized building technologies, specifically precast concrete construction. We selected this class of structures for three reasons:

1. It represents a meaningful building type in current construction practice.
2. It displays an appropriate level of geometric significance.
3. It is a candidate for early applications of robot assisted automated construction.

Within the chosen building type we limit attention to the physical elements that compose the structure and space dividing partitions of the building. In many common applications of panelized buildings these elements have predominant influence on the creation of architectural spaces. Other building components, while important do not have as great an impact on the definition of space.

Prefabricated construction has proven to be economically viable and has been used extensively as a building methodology throughout the last fifty years. Modular precast components are less costly and can be erected faster than standard steel frame or cast-in-place concrete structures. Despite their modularity, today's precast structures demonstrate both uniqueness in design and remarkable complexity in form, configuration and geometry.

The components of precast concrete construction consist of a relatively small set of columns, beams, and planks of standard dimensions. These are typically hoisted into place and fastened together to create both single and multi-story configurations. In bearing-wall construction, precast wall panels carry vertical loads, eliminating the need for structural frame at the building perimeter. Beam-column frame construction is characterized by a grid of precast framing members enveloped in a curtain wall of prefabricated panels. Columns often extend continuously over several stories, and embody connections for horizontal members. Prestressed beams, supported by corbels or haunch casts, complete the framing and provide support for tee-beams or planks that compose the floors and roof.

The regularity of prefabricated construction and the modularity of its building components imply a well-defined construction sequence. The initial and final positions of individual elements can be planned in a formal manner, albeit not a simplistic one. Prefabricated construction requires the use of heavy lifting equipment, and construction planning requires knowledge about the placement, range, and load-carrying capabilities of these devices. In addition, the designs of precast buildings are becoming increasingly complex: curvilinear forms, long-span construction and mixed use of materials are commonplace in today's structures. Prefabricated buildings are likely candidates, however, for robot assisted automated construction since the component geometries are well-defined and a standard method of assembly exists.

Although the prefabricated building type seems unique, the concept of geometric reasoning is extensible to many other types of beam-column framing methodologies. Steel framing, for example, follows a similar pattern of on-site assembly of standardized components. Problems in the manipulation of geometric elements and in assembly planning are germane to many building types, and it is expected that this study will find applications in a variety of building systems.

3.2. The Design System

The effective design of buildings requires information and knowledge from many engineering and architectural disciplines. Our design system addresses two of these disciplines, structural engineering and architecture. These two disciplines are very intertwined; the design of one directly influences what can be designed in the other. This coupling occurs predominantly in the domain of geometry.

Within each of these domains spatial and functional attributes are strongly interdependent. For example, in achieving its function of providing structural support, a structural system is largely determined by its spatial location. When other disciplines are considered, for example architectural spatial layout, the placement of the structural system poses real and challenging constraints. We present several examples to support this statement.

The placement of a steel rigid frame along the width of a building provides lateral and gravity stability to the facility. The placement of this system directly influences the kind and type of other structural systems located in a normal direction. For instance, the placement of the normal system should usually be of the same material for constructability. Thus the existence of a structural system in one direction imposes constraints on a structural system in a normal direction. These constraints normally do not apply unless both structural systems are present in the spatial configuration. That is, both systems are acceptable independently, but their combination in a particular spatial arrangement is not acceptable.

The placement of structural systems also affects architectural design decisions (or vice versa) such as the movement of people within a building. For example, a shear wall in one direction would usually eliminate a shear wall in the normal direction as circulation would be restricted. The functions of some enclosed spaces, such as auditoriums, preclude the placement of structural systems within the space (in this particular example columns at any spacing would not be allowed). The structural components must perform their load carrying function without imposing on the architectural functions of the enclosed space.

Architectural design also mediates structural possibilities. Very often, structure is used as a sculptural element in architectural compositions in addition to providing support. The Fort Wayne Fine Arts Center and the Salk Institute, both by Louis Kahn, stand as prime examples of this type of tight integration between structure and form. This integration not only poses constraints on construction, but on the fundamental choices of structural action in a design.

The design system must reason in the spatial terms of structure and architecture. Such geometric reasoning has been lacking in previous systems for design[5]. These systems have depended on the human designer to place the structural systems, or permitted the placement to be selected from apriori defined locations (for example, HIRISE[6]). Fenves argues the need for improved spatial representations that allow the design process to reason with its representation[7]. The next generation of design systems should provide spatial reasoning features that allow generation of locations for structure and space. Reasoning about the location and function of structure and architectural space must be provided.

In the design system for this project, we propose a solution to these requirements. We are developing a language based on context-sensitive grammars to provide a formalism for generating spatial configurations[8]. With this language, possible structural systems and architectural layouts can be designed. The approach has been shown to work in two dimensions for a simpler design problem involving retaining wall design[9]. The grammar[viii] which defines the language for the building specifies how the structural framing system and architectural features can be generated. The generation is in terms of spatial connectedness, where a given object can generate new objects in a particular spatial location. The grammar is independent of a particular geometric representation; the specification of rules purely in relative spatial terms (such as *ABOVE*, *TO-THE-RIGHT-OF* and, *ADJACENT*) permits an isolation from an underlying geometric reasoner and a sharper focus on the other issues in design.

3.3. The Construction Sequence Planner

As an example application in construction we have chosen construction sequencing, a select part of the overall planning process, for its strong dependence on geometric information. Our goal is to develop an organization for planning that takes advantage of characteristics in the domain of building construction. These properties, of which geometry is crucial, are often intermingled. For example, determining support relations in a structure involves a consideration of both geometric configuration and material properties.

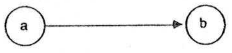
The order of construction is affected by many diverse factors. In addition to support relations, these include stability, constructability, location of utilities (such as HVAC), considerations relating to specific

[viii]The grammar used here is similar to the shape grammars developed by Stiny and Gips[10]. The shape grammars have been previously used to produce the floor plans of Frank Lloyd Wright's Prairie houses[11] and other architectural layouts. The building grammar differs from the shape grammars in that it also includes semantic information in addition to the purely syntactic spatial information.

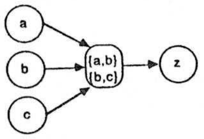
characteristics of the building site, and others. A common facet of these concerns is the geometric knowledge required to reason about them. It is our belief that the expression of knowledge in a planning system should be independent of specific geometric situations. The handling and representation of geometry should be transparent to the other parts of a planner.

We conceptualize the process of creating a construction sequence plan as a pipeline. Within this pipeline, a plan is represented as a partial order on a set of construction operations. Each operation corresponds to the placement of a building component. At the start of the pipeline, a plan is simply an unordered set of such operations. At each stage in the pipeline, a knowledge source is applied. These knowledge sources use domain-specific knowledge, within the context of a common geometric model, to add constraints to the partial plan. After all knowledge sources have been applied, a total order consistent with the final partial order is chosen.

Since plans are partial orders(see Sacerdoti[3] for a discussion of the advantages of conceptualizing plans in this manner), one may represent them as directed graphs. Each node in such a graph either represents a construction task or contains information as to the relative ordering of construction tasks. For example, the interpretation of

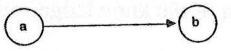


is that operation **a** must take place before operation **b**, and the interpretation of

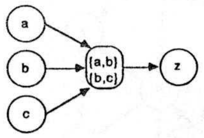


is that either operations **a** and **b** *or* **b** and **c** must take place before operation **z**. Knowledge sources in the planner act upon the graph of a plan through two operations. These operations are(where the letters **a**, **b**, **c**... denote construction tasks):

- From **a** and **b** construct:

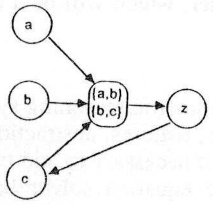


- Given a set, S = {**a**,**b**,**c**...} and a task **z** not in S, construct:

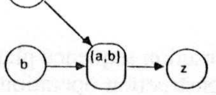


(This is only one of many possible constructions over this set of tasks.)

The graph of a plan may also be modified by operations which change the structure, but not the semantics, of the plan. For example, given a graph:



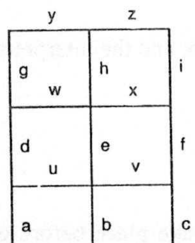
one may construct:



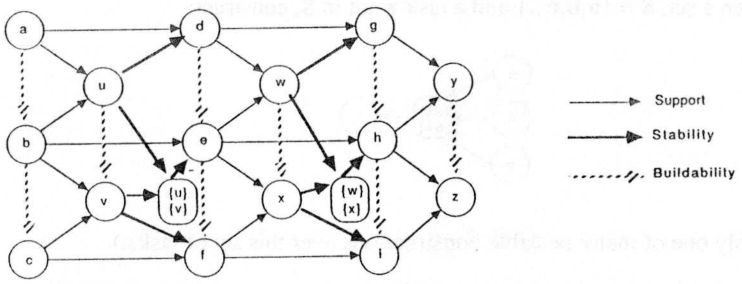
by removing node c and all conditions which involve c. The purpose of such formal operations is to simplify the graph, and thus simplify the application of knowledge sources.

As an example of the application of this planning paradigm, consider the simplified case of a building(shown in cross-section below) with three example constraints on its construction:

- Support -- No component of the structure can be built before the components which support it.
- Stability -- A vertical column must be stabilized at its top before it can support anything.
- Buildability -- Access is from the right, and one cannot be more than one bay away from what is being built.



Each of these constraints corresponds to a knowledge source. Each of the knowledge sources is able to examine the same geometric model to determine what constraints to add to the representation of the plan. A representation of the graph of the plan after application of the knowledge sources is shown below.



From this graph it is possible to construct a linear order, which will be a complete sequence of construction steps.

3.4. The Geometric Reasoner

Underlying both of our example applications is a geometric reasoning system structured after the prototype described in Section 2. Like that system it uses classes, features, abstractions and constraints as a framework for its representation. To these devices we have found it necessary to add two things: a formal graph-theoretic representation of spatial relationships and a numeric equation solver based on the the Newton-Raphson method. Before discussing these new features, we introduce the domain upon which we are defining our reasoning system, namely close packings of polyhedra.

Considered at a certain level of abstraction, the geometry of buildings can be conceived of as a set of polyhedral volumes and a set of planar elements that exist in the interstices between the volumes. Specified by the volumes are the habitable spaces of the building, by the planar elements the structure and space dividing partitions which are the physical realization of the building. With such an abstraction it is possible to do reasoning about the connectivity and "form" of space and the existence and configuration of structural supports.

It is very useful if a model of such volumes and planar elements can be constructed and reasoned upon in terms of spatial relationships, for example *LEFT-OF* and *RIGHT-OF*, without reference to the specific geometry of the elements. This consideration appears to demand an abstract representation, based on these spatial relationships, which has certain formal properties:

- For each syntactically valid representation there must be at least one corresponding realizable configuration of volumes.
- For each configuration of volumes there must exist a syntactically valid representation.

Constructive operations defined on the abstract representation are particularly useful if they can be proved to have their own set of formal properties:

- **Closure.** Every application of a constructive operator to a syntactically valid representation produces a syntactically valid representation.
- **Completeness.** Every realizable configuration can be constructed by a finite sequence of operators.

The invention and proof of such representations and operators is very difficult and usually demands restrictions in the domain of the representation. To the best of our knowledge, no such representation exists for general polyhedra. For the restricted domain of orthogonally oriented 2-dimensional rectangles and various types of relationships (adjacencies, relative geometric location) graph-theoretic representations and their implementations as computer programs do exist[12]. Our strategy is to use an existing formal representation and to attempt its extension to increasingly more general situations. Simultaneous with this development we are developing the design and planning application programs. With careful software engineering this allows us to develop reasoning and language interface capabilities for the applications independent of the specific formal representations currently available.

From the formal representation above can be developed constraints on the allowable geometry of the volumes denoted in the representation. These constraints usually take the form of non-linear algebraic equations. Solution of these equation sets yields allowable dimensions on specific volumes and planar elements. We express these equations using a declarative modeling language called ASCEND[13, 14], which has as its computational engine a Newton-Raphson solver. Integrating the graph-theoretic representation and equation modeling language is the geometric reasoning architecture outlined in Section 2. Objects in each of these representations are identified, stored and accessed as the features and abstractions of the representation. The equation modeling language ASCEND fills the role of the constraint system in the prototype architecture.

4. SUMMARY AND CONCLUSIONS

We believe that good geometric reasoning facilities will clarify and simplify the creation of computer systems in construction automation. By developing a reasoning system in the context of two geometrically rich applications we hope to not only demonstrate this belief but also to contribute to the body of theory necessary for construction automation to prosper.

5. ACKNOWLEDGEMENTS

This work is supported by the National Science Foundation under grant #MSM-8717307.

References

1. Robert Woodbury, *The Knowledge Based Representation and Manipulation of Geometry,* PhD dissertation, Department of Architecture, Carnegie-Mellon University, December 1987.

2. R.E. Fikes and N.J. Nilsson, ''STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving'', *Artificial Intelligence,* Vol. 2, 1971, pp. 189-208.

3. E.D. Sacerdoti, ''Planning in a Hierarchy of Abstraction Spaces'', *Artificial Intelligence,* Vol. 5, No. 2, 1974, pp. 115-135.

4. Walid T. Keirouz, Daniel R. Rehak, Irving J. Oppenheim, ''Object Oriented Programming for Computer Aided Engineering'', Tech. report EDRC-12-09-87, Engineering Design Research Center, Carnegie-Mellon University, 1987.

5. M. Stefik, et al., ''The Organization of Expert Systems, A Tutorial'', *Artificial Intelligence,* Vol. 18, No. 2, March 1982, pp. 135-173.

6. M.L. Maher and S.J. Fenves, ''HI-RISE: An Expert System For the Preliminary Structural Design of High Rise Buildings'', *Knowledge Engineering in Computer-Aided Design,* Gero, J.S.,ed., North-Holland Publishing Company, International Federation for Information Processing (IFIP), 1985, pp. 125-135.

7. S.J. Fenves, ''Computers in the Future of Structural Engineering'', *Proceedings of the Eighth Conference on Electronic Computation,* Nelson, Jr., James K.,ed., ASCE, February 21-23 1983, pp. 1-8.

8. S.J. Fenves, and N.C. Baker, ''Spatial and Functional Representation Language for Structural Design'', *Expert Systems in Computer-Aided Design,* J.S. Gero,ed., North-Holland, Amsterdam, 1987, pp. 511-530.

9. Steven J. Fenves, and Nelson C. Baker, ''Grammars for Functional and Spatial Reasoning In Design'', *Fifth Conference on Computing in Civil Engineering: Microcomputers to Supercomputers,* sponsored by Technical Council on Computer Practices (of ASCE), Society of Computers in Engineering, Planning and Architecture (CEPA), and the ICES User's Group (UG), March 1988.

10. G. Stiny, ''Introduction to Shape and Shape Grammars'', *Environment and Planning B,* Vol. 7, No. 3, 1980, pp. 343-351.

11. H. Koning and J. Eizenberg, ''The Language of the Prairie: Frank Lloyd Wright's Prarie Houses'', *Environment and Planning B,* Vol. 8, No. 3, 1981, pp. 295-323.

12. Ulrich Flemming, ''On the representation and generation of loosely-packed arrangements of rectangles'', *Planning and Design,* December 1985.

13. Arthur W. Westerberg and Dean R. Benjamin, ''Thoughts on a Future Equation-Oriented Flowsheeting System'', *Computers in Chemical Engineering,* Vol. 9, No. 5, 1985, pp. 517-526.

14. P.C. Piela and A.W.Westerberg, ''A User Interface for ASCEND3, A Process Modeling Aid'', *1986 Spring National Meeting, New Orleans,* AICHE, April 6-10 1986, pp. Paper 42C.