SUPPORTING CREATIVE DESIGN USING ADJACENCY STRUCTURES

STEVEN MEYER and STEVEN J. FENVES
Department of Civil Engineering
Carnegie Mellon University
Pittsburgh PA 15213 USA
and
FANG ZHAO
Department of Civil and Environmental Engineering
Florida International University
Miami FL 33199 USA

Abstract. Whether engineering designers aim to be creative or simply to meet the design requirements their design processes often produce creative solutions. We view creativity in engineering design as a title given after the design product is evaluated in the context of its requirements and historical setting. The potential for creative design is increased as more of a solution space can be explored and as new problem formulations are utilized. In this light, we discuss a representation that allows both a reformulation of our design problem and an expanded search space. The graph-based representation, *adjacency structures*, is presented along with recent work on its use within particular design methods. The representation integrates a geometric model with functional information as a means of expressing a more complete formulation of the design problem. A systematic approach is discussed as a search method increasing the potential for creative design by exploring larger regions of a solution space. In this paper we focus on the use of adjacency structures within the domain of architectural and structural design.

1. Introduction

Creative design involves both the generation of novel products and the recognition of their usefulness. It is a two step process in which interesting products are generated, and then these products are evaluated and recognized as useful for meeting a desired goal. This definition may be rearranged to say that design involves generation and evaluation, and then the design process or object is said to be creative if the object or process is novel. Put either way, design requires the integrated generation and evaluation of relevant domain products. Modeling design as a search process, a realistic search focuses on the utility of an object or process rather than on its novelty. In the course of this search focusing on the utility of the object or process many creative designs may be produced when the search space is sufficiently encompassing and evaluations can recognize novel structure, behavior and functionality. Adequately flexible representations are crucial to the modeling of creative design in order to provide a broadly expressive specification of the design space. Adequately flexible and expressive representations are also essential in order for design processes to use the information stored in partial solutions for guiding operations within the design process. The systematic approach has been proposed as a search method for creative engineering design under the conjecture that creative solutions occur when a new formulation of the design task generates new solutions or when a new solution is found within a given formulation in an unexplored region of the design space [Coyne and Subrahmanian, 1989]. In order to close the representation gap between geometric modelers and knowledge-based systems, and to provide a representation that can express the multiple levels of abstraction necessary for supporting creative design, a representation called *adjacency structures* has been proposed [Meyer and Fenves, 1992]. In this paper, we discuss adjacency structures as a supporting representation for creative design in the domain of discrete static systems by describing the nature of the domain that motivates this representation, the formulation of the design problem taking advantage of this representation, and a systematic approach to exploring the design space.

A serious hindrance to research in the design of highly spatial artifacts has been the lack of a representation that can adequately express the geometry and topology of the systems at levels of abstraction suitable for symbolic reasoning. Currently in our domain of structural systems for buildings, a design is represented at a static abstraction level. A structural system is represented either in the graph-based representations used by a solid modeler, in the matrix formats used by finite element or frame analysis methods, or as a set of attribute–value pairs in rigid data structures. The solid modeling representations are too verbose and detailed for many creative design processes. The matrix representations do not lend themselves to the type of guided search necessary in a creative design process. Finally, the collection of object–attribute–value structures composing frames, prototypes and most rule-based system's data structures may describe some of the spatial and topological attributes of a structural system, but are not complete and open-ended enough to express more than the expected relationships based on past experi-

ence. A flexible intermediary representation that can relate the low-level representation of geometric modelers and analysis methods with the high-level, but incomplete, abstractions of frames, prototypes, etc. is missing. This inability to express the geometry and topology of structural systems, directly and at various levels of abstraction, seriously restricts the ability of computers to support creative structural design.

We motivate a representation for building design by a small example of the characteristics of our domain that must be expressed in a design process models. The adjacency structures representation captures several important types of information used during the design process of design highly spatial artifacts including geometry and topology, function, and behavior. Adjacency structures are particularly promising as a representation for the specification of a design space because of their ability to operate on the geometry and topology of a structural system representation both individually and in combination.

2. Representation

This section introduces adjacency structures as a knowledge representation beginning with an intuitive example. For a more detailed description refer to [Meyer and Fenves, 1992]. As typically used, an adjacency graph is a purely topological description of a system. Each element of the system is represented as a node and each directed or undirected arc represents an adjacency relation between the two elements it connects. The extension of adjacency graphs to adjacency structures adds basic geometric and material information to each node in a graph with undirected arcs. Thus, the underlying principle of adjacency structures is the formation of a graph whose organization is based on physical adjacency and whose nodes contain the union of an abstracted geometric model with essential functional information.

To begin the example, consider the nine-bar truss and adjacency graphs in Figure 1. Graph (a) represents the joints of the truss as vertices, accurately describing the topology of the truss. However, any material or cross-section information is lost when the arcs are represented merely as pointers. Graph (b) represents the bars of the truss as nodes of the graph, but does not accurately describe the salient topology of a truss. Graph (c) contains two types of nodes; (triangular) nodes represent the truss bars and (circular) nodes represent the truss joints. The arcs simply represent the adjacency relations between the bars and joints of the truss. In this representation each node is a member of a minimal cycle of length six composed of three pairs of alternating 'bar' and 'joint' nodes. Furthermore, the inclusion of geometric information within the node representation allows for a simple determination of

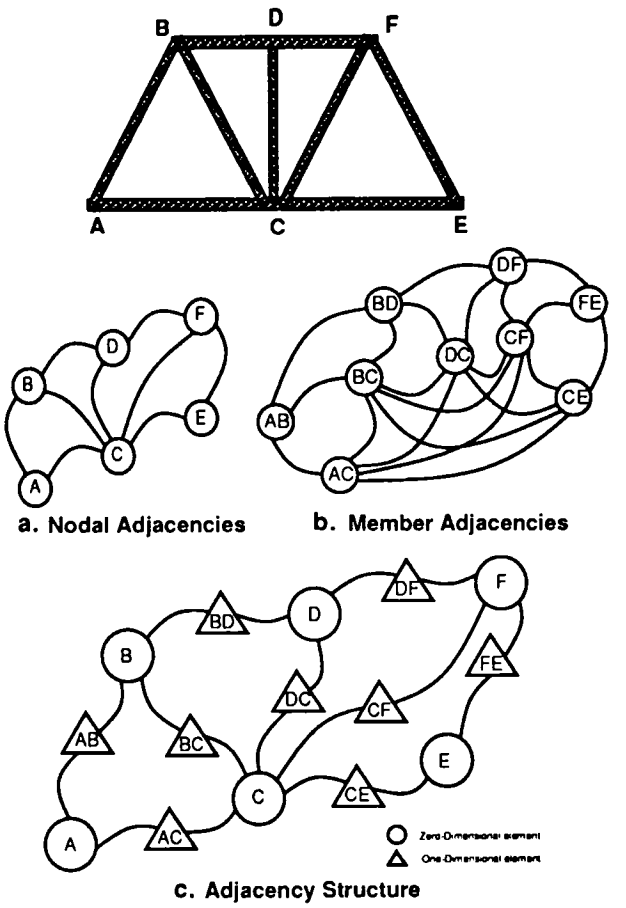

Figure 1: Nine-bar truss and associated adjacency graphs.

the planarity of the truss. Thus, the topological and geometric definition of a plane truss is directly expressed in the syntax of the representation. Furthermore, the geometric and material attributes associated with the bar nodes allows the determination of the functional adequacy of the truss. The remainder of this section presents a more specific description of the composition of adjacency structures.

Individual nodes in an adjacency structure represent physical elements such as columns, functional elements such as applied loads, and interfaces such as joints. The systemic nature of the domain is expressed through a representation hierarchy of atomic elements, primitive elements, and system components. The atomic elements of adjacency structures represent the syntactic basis of systems. Any classification of node aggregations gives a domain-dependent relevance to certain subgraph structures. Therefore, the definition of non-atomic elements and system components identifies relevant syntactic structures and attaches a particular semantic importance to them. For this reason, the collection of atomic and primitive elements specified as being semantically relevant to

the domain are called the *semantic templates* of that domain. The formal specification of these templates can form the basis of an algorithm for discovering the semantics of the purely syntactic adjacency structure.

Aggregations of atomic elements are organized into a hierarchy based on topological and behavioral distinctions. The hierarchy begins with lower level primitive elements which are, in turn, used to construct higher level system components. A *primitive element* consists of a specific number of nodes arranged in a specific topology. The exact geometry associated with each node in a primitive element is not specified by the element's definition; the geometry of each node is merely constrained to a specified relationship with other nodes in the primitive. A *system component* is composed of an indefinite, but finite, repetition of primitive elements constrained to specified geometric relationships. A system component class is defined in terms of its constituent primitives, and therefore system classes retain the constraints of its primitives and subsystems, but uses additional constraints to define their composition into a system. A system class may be composed of the repetition of a single primitive or a pattern of different types of primitives. These two types of systems are called *uniform systems* and *composite systems*, respectively. Operationally, the hierarchy may be composed through a graph grammar, and we present three graph composition operations before presenting a number of system components from the representation hierarchy.

2.1 Atomic Elements

The leaf nodes of the representation hierarchy are the *atomic elements*, the nodes of the graph structure. There are four classes of atomic elements based on their gross dimensionality. *Zero-dimensional nodes* have position but no size or form, and may represent an interface between two adjacent elements of a geometric model such as a joint in a truss or frame. *One-dimensional nodes* have a length much greater than both cross-sectional dimensions, may be used to represent a bar of a truss, a column or a beam, and have their geometric information represented by their two end points. *Two-dimensional nodes* have a breadth and depth much greater than their height, may represent a wall or floor plate, and have their geometric information represented by an ordered list of vertices. *Three-dimensional nodes* have each dimensions of roughly the same scale, may represent an architectural volume or a foundation footing, and have their geometric information represented as a nested list of the vertices of its bounding faces.

Each node is represented using a common data structure regardless of its dimensionality. The requirements of the data structure include representing non-physical attributes which express aspects of the node's behavior and the ability to model the environmental conditions such as loads and displacements for which we are constructing load paths. A single data structure is used to represent both physical or *member objects* as well as *virtual objects*. Member objects encompass those physical objects being modeled in the geometric modeler, e.g., truss bars and floor slabs, whereas virtual objects include the loads and displacements imposed on a system as well as the interface between adjacent member objects. The data structure contains five attribute fields: dimensionality, geometry, magnitude, composition and stiffness. There are two additional fields, one for a node identifier and another for a list of pointers to adjacent nodes. Additional fields may be used for other domains, but these fields provide a compact yet expressive representation of the geometric and non-geometric aspects of a single design component in the domain of building design.

2.2 Primitive Elements

As mentioned above the composition of nodes into relevant graphs describes a domain-dependent representation hierarchy. Two important subgraphs of the example domain are the truss panel and the bent. Along with atomic elements, these primitives are used to compose system graphs addressing the functional requirements of the building's structure. A *truss panel* is represented as a graph forming a cycle of three pairs of alternating zero- and one-dimensional nodes. The elements define a single plane parallel to the orientation of the loads it resists, and any loads must be applied only at the zero-dimensional nodes. A *bent* is represented as a non-cyclic graph of five nodes comprised of three one-dimensional nodes separated by a zero-dimensional node. As in the truss panel, the elements are arranged in a single plane parallel to the loads the bent resists. However, in a bent the loads may be applied to either the zero- or one-dimensional nodes. Additional constraints on the relative angles and absolute orientations of the one-dimensional nodes must be included in the graph template.

2.3 Composition Operations

The expressive richness of the adjacency structure representation hierarchy may be defined by the collection of semantic templates for the domain of interest. A representation hierarchy should express the elements and systems relevant to the domain because these elements and systems are the building blocks of the generation and evaluation processes. The composition operations facilitate composing systems from elements during design generation and their decomposition

while parsing design solutions. Three types of operations that are useful for combining subgraphs are the *merge*, *abut*, and *embed* operations. The merge operation combines two separate graphs by unifying nodes which have the same dimensionality and the same location in both graphs. This is useful for systems, such as *plane trusses*, which are defined in terms of primitive elements (truss panels) that share atomic elements (a one-dimensional node and its two adjacent zero-dimensional nodes, see Figure 1.) The abut operation connects two geometrically adjacent subgraphs without unifying nodes in the two subgraphs. This operation adds a 'bridge' composed of two arcs separated by an interface node. The dimensionality and geometry of the inserted 'bridge' node is given by the intersection of the nodes being connected. This operation is useful for combining two systems that do not share components, e.g., two orthogonal shearwalls. The embed operation inserts one graph within another by replacing arcs (and possibly nodes) of the host graph with new arcs into the immigrant graph. Embedding is useful for rearranging the components of a system being combined, e.g., when combining a frame and a shearwall by removing columns at the intersection and reattaching the beams to the shearwall.

These three composition operations are used to specify the transformation of graphs that compose the representation hierarchy. The parsing and generation processes require these transformations because the graphs are not simply split when parsing or intuitively connected when generating. Therefore, the specification of the representation hierarchy requires an operational definition of the transformation of one level of templates into the templates of another level. These operations are used in describing a few system components presented below.

2.4 Constraints Within Semantic Templates

The semantic templates specify a graph structure relevant to the domain. The topology of the graph clearly specifies how the represented objects are connected but the nodes have only their dimensionality fixed by the template; the template does not assign them a geometry, magnitude etc. The constraints within templates are used to define aspects of the domain systems that cannot be expressed by the dimensionality of individual nodes or by the topology of the semantic templates. However, these constraints are not intended to represent design knowledge in the sense of when a template should be used or what are preferred geometric dimensions. For example, the geometric constraint on the aspect ratio of a one-way flat slab does not say that a one-way flat slab is preferable for a particular situation, but simply allows the association of a particular geometry with a particular behavior; it is a *definitional* constraint. This view of the role of constraints embodies our ideal of minimal constraints that foster creative design, limiting the generation of meaningless solutions without restricting the process to the results of previous design experience.

Constraints are represented as part of the transformations which compose (or decompose) the templates during generation (or parsing.) The five type of constraints are geometric constraints, constraints on the location of the template, constraints on the association of node types within the template, functional constraints which are expressed in terms of loads, and behavioral constraints which relate applied loads to the resulting deflections or rotations.

2.5 System Components

Uniform systems are composed by repeating a single element type within a set of prescribed geometric constraints. For example, a truss, in order to be a *plane truss*, must have each of its panels constrained to a single plane. In contrast, a *space truss* may be composed of the same type and number of components, but by using different geometric constraints the semantic template of a different system is specified. More specifically, a plane truss is represented as a graph repeating the truss panel primitive, i.e., each of whose nodes is a member of a minimal cycle of length six composed of alternating zero- and one-dimensional nodes. Each panel is merged with at least one other panel, i.e. each cycle has at least one one-dimensional node and its two adjacent zero-dimensional nodes as members of one other cycle, the nodes are arranged in a single plane parallel to the forces the truss resists, and the forces are applied only to the zero-dimensional elements. A *plane frame* is represented as a graph repeating the bent component, achieved by horizontally merging bent primitives and vertically abutting bent primitives. Additionally, the zero-dimensional elements are arranged in a single plane parallel to the orientation of the forces the frame resists.

Composite systems are specified through the juxtaposition of different elements or uniform systems within a set of prescribed geometric constraints. For example, a *braced frame* may be specified as the horizontal merging of *plane frames* and a vertical *plane trusses*, all in the same plane. A *shearwall* is represented as a vertical series of two-dimensional nodes mediated by a horizontal one-dimensional interface node. A shearwall–plane frame combination is represented as an embedding of a shearwall within a plane frame. All one-dimensional nodes within the intersection of the shearwall and the plane frame are removed from the plane frame adjacency structure before arcs are inserted to combine the two systems.

In this way a hierarchy of templates is specified using a small number of simple subgraphs which may be repeated an indefinite number of times. During the generation process a system is instantiation by repeating its components a specific number of times to fit the design case, forming a specific adjacency structure. This partial description of a representation hierarchy may appear to preclude the generation of creative solutions to a design problem. However, with this representation the design process can be based on matching functional components of the system definitions (loading and geometric constraints) and functional requirements of the design case. Therefore the generation of potential solutions takes place in an expanded design space promising the expression of a larger solution space.

3. Design Process

With these representation issues in mind, we consider the systematic approach and its potential in creative design, serving as background for our description of a generative grammar that uses an adjacency structure representation to support creative design. We then move to a discussion of the use of this representation for complementary generative mechanisms, focusing on analogy and mutation. The systematic approach as a useful computational model for creative design is discussed in the context of the structural design of buildings. We conjecture that the keys to using a systematic approach to perform creative design are (a) a rich representation, (b) a minimum of constraints, and (c) an effective evaluation method. The potential of systematic approaches in creative design depends on the integration of these three elements.

- A representation that captures the structure, function and behavior of the domain and provides sufficient information for design operations.
- A powerful generative method that can exploit the representation and search the design space as completely as possible. The appropriate search method is a tradeoff between a purely syntactic generator and one that employs rigorous evaluation and early pruning. The former risks getting bogged down in a combinatorial explosion of potential solutions and the latter reduces the likelihood of finding creative solutions if they do not show early promise. Therefore a minimum of constraints should be employed, that is only constraints that are formally grounded in the necessary, physical definition of the domain.
- Effective evaluation methods that can quickly determine the feasibility and merits of generated solutions. The evaluation methods must be able to recognize behavior of a potential solution in the current design context.

We focus on the expressiveness of the adjacency structure representation, the minimum constraint concept, and the cooperation between a grammar that will systematically search the design space and evaluation methods for describing the potential of a partial solution. In regards to this last point, the strength of the systematic approaches, e.g. generate-and-test or those approaches that also incorporate some partial evaluation, lies in the promise of a thorough search of the design space. In reality, it is difficult to verify that the generative mechanism would cover the complete design space. Also, to assume that a design space may be completely covered by a systematic method is to say that this design space is closed and all the possibilities can be represented. However, assuming a closed design space does not necessarily eliminate the possibility of finding creative solutions because these opportunities come from revealing solutions that are not obvious, that might be missed by other search methods, or that might be inadvertently pruned before they can show their promise. An additional difficulty associated with this approach is the need to evaluate a large number of potential solutions. One advantage of the representation we describe is its ease of translation into the matrix methods of analysis. A qualitative analysis method is also described that parses the syntactic structures to verify intended semantics and to discover emergent semantics.

3.1 A Rule-Based Approach

To illustrate the use of adjacency structures in a rule-based design process let us assume that we have a geometric model representing a preliminary architectural design of a building[1]. This geometric model, the applied loads, and the constraints on the deflection and vibration form the functional requirements of the preliminary structural design problem. The search for structural systems that satisfy these requirements and constraints defines the design process. The result of the design process is a structural description of systems (the physical elements and their connectivity) that meet these requirements and constraints. Using adjacency structures as a representation for the design state, the search process can be modeled using a graph grammar and the resulting design is a graph representing a structural description via the syntax

[1] In the context of current building design, the architect is relied upon to develop the building's overall form and, therefore, we begin our problem from the architectural geometric model.

of the nodes and arcs, plus a behavioral description via the procedurally defined behavioral constraints.

In more detail, the design process begins with an architectural description including a definition of the building's envelope plus any interior partitions such as the service core, floors and permanent internal walls. The geometric model of the building is translated into the overall adjacency structure according to the minimal extent principle [Meyer and Fenves, 1992]. Through design standards or experience, lateral and gravity loads are specified for the building form as a function of height and occupancy type. These lateral and gravity loads are added to the overall adjacency structure as *load* nodes. Also through design standards or experience, allowable deflections are specified as a function of the applied loads. Together, the architectural envelope and internal elements, and the applied loads and allowable deflections form the specific functional requirements for the structural design. The structural design phase consists of instantiating sets of subgraphs which together satisfy the specific functional requirements for an individual building, along with the general requirements for all buildings such as constraints on member forces.

Design generation begins by seeking matches for the semantic templates of the representation hierarchy giving priority to three-dimensional systems. The control mechanism for the generation process seeks multiple instantiations which satisfy the functional requirements; the control of the generation process attempts to branch the single set of design requirements into multiple potential solutions. The complete instantiation of semantic templates to form an adjacency structure which satisfies the functional requirements is divided into two phases: topology and parameterization. In the first phase the building envelope is populated with specific types and numbers of semantic templates. In the second phase the (primarily geometric) unassigned data fields of the templates nodes are assigned values.

In the first phase, an attempt is made to topologically instantiate the highest level semantic templates possible through matching on the object and load nodes in the overall adjacency structure while satisfying the templates' constraints. The load nodes are propagated subject to the constraints on the allowable dimensionality, location and orientation for load nodes on the semantic templates being instantiated. Originally the lateral and gravity loads of the functional requirements are distributed loads inserted into the overall adjacency structure as two-dimensional nodes. The lateral loads must be propagated as one-dimensional nodes when instantiating such templates as frames or framed tubes, as zero-dimensional nodes when instantiating truss templates or remaining as two-dimensional nodes when instantiating shearwall templates. Another fundamental constraint on each high-level system during topological instantiation is that the system must be composed of an integer repetition of its constituent templates. Thus, topological instantiation involves a determination of the dimension of the target region for the potential instantiation of a template and a comparison of the template's definitional application limits to arrive at possible dimensions for the region's division. The order of these two steps is dependent of the design process in which it is used.

The structural design phase encourages creative design when non-behavioral constraints are relaxed. However, creative design cannot be purely unconstrained design due to the purposeful nature of building design. Creative design, in this context, must still satisfy the architectural purpose of the building as defined by the client. For example, if an office building is being designed with a 45 foot core-to-perimeter dimension which has no permanent interior walls besides the core, is it acceptable to place a column between the core and the perimeter? If it is not architecturally acceptable to do so, then this constraint must be considered during the instantiation of any internal frame or flooring system. This suggests the need for an interactive ability during constraint satisfaction within the topological instantiation of semantic templates; their own limits of application are underconstrained.

When the number of element templates composing a system is determined the topological instantiation itself can be accomplished by generating the proper number of subgraphs which realize a system template, assigning values to the appropriate data fields in each node of the template, and embedding it in the overall adjacency structure. After the first template system is instantiated and added to the overall adjacency model, subsequent instantiations are also constrained to accommodate the existing adjacency structures in the model through the composition operations presented earlier. For example, if the lateral load system is satisfied first by an orthogonal rigid frame, the gravity load system is constrained to using the existing beams and columns for instantiating the gravity system templates.

The parametric instantiation is primarily concerned with defining member cross-sections. In order to accomplish this parameterization some level of analysis is needed. One advantage of the representation we have described is its ease of translation into the matrix methods of analysis and we will describe an analysis-based parameterization. Each member node, if it had its cross-section defined, would contain enough information to form the member stiffness matrix. The overall adjacency structure would then contain enough information

to compose the element stiffness matrices into the global stiffness matrix. Then, the product of the inverse global stiffness matrix (the flexibility matrix) and the force vector results in the deflection vector. However, the topological instantiation does not provide the information needed to complete the element stiffness matrices; it only provides enough information to compose the global matrix from defined element stiffness matrices. Thus, parametric design may proceed as follows:

1. The topological instantiation defines the length of 1D elements and the breadth and depth of 2D elements during the subdivision of the building envelope.

2. The **relative** stiffness of member nodes is defined by the user, thereby defining element stiffness matrices for each member node as a function of its moment of inertia I and the modulus of elasticity E.

3. The relative stiffness matrix of each element and the defined topology is used to construct the relative global stiffness matrix.

4. The deflections of the building structure are found as a function of the relative stiffness and applied loads.

5. A stiffness is assigned to limit the deflections. This allows the back calculation of member stiffnesses and the defining of their cross-sections.

The instantiation process continues until the overall adjacency structure becomes a complete connected graph, that is, when the applied propagated loads have been connected to system templates for resisting these loads, and when the system templates have been completely instantiated down to their atomic elements. In this way we perform the generative mapping from functional requirements, stated in terms of the architectural form and the applied loads, through the behavior of load propagation and flexural stiffness to derive the structure of a design solution represented as a network of adjacent members.

A design process must also include provisions for changing the model as new information is introduced or existing information is altered. We have discussed the introduction of adjacency structures into the design model, but must also consider the requirements of editing existing adjacency structures. As in their introduction, the editing of adjacency structures may be divided into topological alterations and parametric alterations. A graph grammar may be used for topological edits such as embedding a shearwall or braced frame template into an existing plane frame system within the overall adjacency structure. On the other hand, the object oriented programming technique of *methods* which operate on a restricted set of abstract data types appears more appropriate for the parametric editing of a specific system component. Instantiation and editing, thus, form the basic operations of the design process with adjacency structures.

3.2 Semantic Interpretation of Syntactic Structure

The expense of translations between a geometric model and adjacency structures is only worthwhile if the new model is more expressive for some particular purpose. With the help of a parsing algorithm, the adjacency structure representation of a geometric model may be used to discover both intended and emergent semantics within the syntactic representation. Behavioral semantics are crucial to creative design as a way of evaluating the syntactic structures being manipulated, and a purely syntactic matching process using adjacency structures can discover emergent systems because it inspects the graph structure to find what **is** contained in the model, not what is **said** to be in the model. For example, if the structural system is generated as a set of plane frames in the x–z plane, and then each frame is connected by beams (at the joints of each frame) in the y–z plane the *plane frame* template will match on frames in both the x–z plane and in the y–z plane, thereby discovering that there is an orthogonal plane frame system, i.e., plane frames exist in both directions.

Additionally, this representation and parsing process may be used to confirm intended syntactic compositions and to evaluate the condition of existing compositions. After the structural system has been parsed, the resulting definition of the system in terms of semantic templates may be compared to the structural system graph translated from the geometric model. If a boolean difference between the overall adjacency structure and the union of semantic templates parsed from it leaves any structural elements remaining, these remaining elements may be said not to participate in the structural system described by the semantic templates. These elements which are not included in the semantic templates are extraneous to the system defined by the templates and possibly may be removed from the design.

Alternately, the initial architectural definition of the building contains a set of components which may form an intentionally incomplete structural system. For example, the architectural design may specify the locations of floor slabs and columns while not ruling out the use of beams in the structural design or precluding other flooring systems. Parsing the overall adjacency structure finds that there are no lateral load resisting system templates that will match on the overall adjacency structure, but that many templates may be unified

with it if the design process is shifted to the generation of a lateral load resisting system.

The three results of parsing can be used to summarize the syntactic condition of the design. The overall adjacency structure may be completely parsed into atomic elements, thereby describing the overall adjacency structure in terms of the templates at successive levels of decomposition. This signifies a syntactically complete design. Alternately, the overall adjacency structure may be decomposed, but the decomposition leaves elements that are not parsed out of the overall adjacency structure. This signifies an intentionally or unintentionally redundant design. Finally, the overall adjacency structure may be incompletely decomposed, halting at the system level unable to match on the existing adjacency structure. This signifies a syntactically incomplete design that requires additional elements for completion.

4. Discussion

Creative design is fostered by an adjacency structure representation in capturing the essential, immutable characteristics of the domain structure while allowing the relaxation of the routine, but non-essential, constraints on the elements and systems which compose that structure. Adjacency structures directly represent function and structure using the syntax of the graph, while indirectly describing the behavior and performance of the design through procedurally defined constraints and evaluation functions. Additionally, adjacency structures are capable of describing these essential attributes of structural systems in a manipulable way, overcoming many of the previous representations' difficulties with representing and operating on geometry and topology. Thus, an adjacency structure representation meets our requirements of a rich representation that can employ a minimum set of constraints and that can be used to integrate many generative mechanisms with a number of effective qualitative and quantitative evaluation methods so that systematic generation can discover interesting points in an expanded design space.

Furthermore, the integration of a representationally complete geometric model within the adjacency structure representation allows the reformulation of computational models of the building design process to address new portions of the problem. The ability to inspect the architectural model for relevant features of the design such as collinear internal walls, vertically continuous internal volumes, etc. allows the design process to evaluate architectural elements in structural terms without an a priori labeling of their function. Thus the process of discovering the architectural context for the structural design is not bound by a geometrically incomplete view of the design state.

It is of particular importance to formulate a set of constraints for the graph grammar so that creativity is not restricted by their applications, while still limiting the generation of meaningless solutions. The constraints used in specifying the representation hierarchy are definitional constraints, reflecting the minimal essential requirements imposed by the mechanics of the domain rather than by human experience and preferences. Our conjecture that creative design using a systematic approach results from the integration of a powerful search method, a minimum constraint set and an effective evaluation method means that creative design is, to a large degree, an issue of formulating the appropriate constraints.

In summary, the adjacency structure representation is promising as a representation for creative design in its ability to represent and manipulate the topological and geometric nature of the systems in the domain, its flexible representation of system hierarchies, and that it relies only on definitional constraints. This representation suits our initial conjecture; that creative design may be performed through a systematic approach by employing a rich representation and a minimum of constraints integrated with an effective evaluation method. The ideas described in this paper are currently being implemented by the authors. The presentation of this paper will contain results based on current developments.

References

[Coyne and Subrahmanian, 1989] Computer supported creative design : a pragmatic approach. Technical Report EDRC-05-46-89, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA.

[Meyer and Fenves, 1992] Adjacency structures as mappings between function and structure in discrete static systems. Technical Report EDRC-12-49-92, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA.