Adjacency Structures as Mappings Between Function and Structure in Discrete Static Systems

Steven Meyer
Steven J. Fenves

Department of Civil Engineering
Carnegie Mellon University
Pittsburgh, PA 15213 USA

We present a graph-based method for mapping between functional requirements and physical structure in discrete static systems. Through forward or backward chaining, this method may be used in a generative mode to suggest instances of system structure satisfying the desired functionality, or in a parsing mode to uncover the behavior and function of a given system. The graph may be composed from a geometric model, but the method is independent of any specific geometric modelling representation. We focus on the domain of structural systems in buildings to describe this method.

Keywords: computer-aided structural design, geometric modelling.

1 Introduction

In many domains a finished design consists of a description of the physical artifact's structure.[1] The artifact's structure is a description of its topology, geometry, material, and manufacturing details. However, this description does not normally contain a full specification of the artifact's desired functionality. In contrast, at the start of the design process the design mandate consists almost entirely of specifications of the artifact's desired functions. Between the initiation of the design process—when no physical structure has been given to the projected artifact—and the beginning of the manufacturing process—when a complete physical description must be specified—the design process consists of an iteration of mappings between function and structure. That is, a structure for a system or subsystem of the design is suggested which is thought to meet the given functional

[1] In this paper we adopt the terminology of (Gero, 1991) using the term *function* to describe the intentions and purposes of the artifact being designed; the term *behavior* to describe how the functions are achieved in the design; and the term *structure* to describe the physical components specified by the design. An example of these distinctions in structural engineering is that the function of a building is to resist lateral and gravity loads within a prescribed limit of deflection, vibration etc.; the structure of the building may be a set of beams, columns and slabs of a particular material arranged in a specific configuration; the behavior of the building may be that it resists the applied loads through the flexural strength of the components.

requirements through its expected behavior. Then, this structure is evaluated to determine whether it meets the behavioral and functional requirements and what its side effects might be. This loop continues until the current structure can be mapped to the required functions, and vice versa. Thus, in an evaluation–synthesis–analysis design loop, the mapping from function to structure is a high level view of the synthesis stage, the mapping from structure to behavior is a view of the analysis stage, and the mapping from behavior to function is a high level view of the evaluation stage. We describe a method for performing the two–way mappings between structure and function within discrete static systems based on the adjacency structure representation, a representation that extends the concept of adjacency graphs by adding geometric, material, and behavioral information to each node.

This paper is organized into six sections. Section 2 reviews some of the related research and defines discrete static systems. Section 3 introduces adjacency structures beginning with an example to describe the type of information we would like to express using this representation, then describes the elements and composition of adjacency structures. The next two sections present the two directions of the function—structure mapping. Section 4 presents the parsing of a given adjacency structure to uncover the function and behavior of a given structure. Section 5 describes how a set of functional requirements and a minimal geometric description can be used to synthesize adjacency structures in a behavior–oriented derivation of structure. The paper ends with a summary and brief discussion of adjacency structures within a larger synthesis system.

2 Background and Motivation

In the domains of interest, form variables are given values in order to satisfy functional requirements, or function variables. The form variables may also generate additional function variables or help to complete the specification of existing function variables. These function variables are generally at a higher level of abstraction than the form variables. A useful design method and representation must mediate between the multiple levels of abstraction used during the design process, tracking the detailed representation of geometry as well as the abstract representation of function and behavior. The adjacency structure method may be viewed as a geometric modelling representation focusing on design components at higher levels of granularity than typical geometric modellers. A boundary representation, for example, represents a design component such as a rectangular beam as a graph containing vertex nodes, edge-half nodes, face nodes, etc. As the number of design components increases, the representation becomes extremely cumbersome to manipulate. Adjacency structures represent an individual design component such as a beam as a single node, and form a graph of adjacent nodes to represent the system of connected components. Additionally, the geometric model is a purely syntactic representation having few facilities for expressing non-geometric aspects or domain dependent semantics of the objects being represented. The inclusion of material and behavioral information in the nodes of the graph allows the expression and propagation of functional and behavioral aspects of the system.

The representation of design elements in a specific domain using graph structures has been the subject of considerable investigation, notably for the specification of computer languages (Knuth 1968), in architectural research (Mitchell 1976, Flemming 1986a), and in dynamic systems design (Ulrich 1987, Finger 1989). Each of these areas has provided domain or theoretical background for the ideas presented here.

2.1 Graph Grammars in Computer Science

The formality of graph rewriting systems, and their utility in design through the expression of complex structural interrelationships at multiple levels of abstraction, is attractive from the viewpoint of both the development of system logic and its implementation. The ability of graphs to represent complex structural relationships between many types of objects modelled by the nodes of the graph allows a hierarchical and multidimensional system to be formally evaluated. Additionally, the similarity between the representation of the program's specification language and the internal representation of the program's data structures simplifies the implementation of the logic invested in the language.

The application of graph grammar variants has begun to receive attention in the engineering design field (Finger 1989, Pinilla 1989). The necessity for representing both the geometric and non-geometric aspects of partial designs, and the need for basing the transformations in the design process on both aspects of the representation, focuses our attention on the attribute grammar formalism begun by Knuth (Knuth 1968). An attribute grammar begins from a context-free grammar and associates with it an *attribute system.* The attribute system consists of an attribute vocabulary and a set of *semantic rules* which define how to compute the attributes of each symbol in the production. Each non-terminal symbol (except the starting symbol) is associated with a set of inherited attributes and a set of synthesized attributes. The inherited attributes of a symbol are computed from the attributes of its parent symbol, whereas the synthesized attributes are computed from the children symbols' attributes.

Two interrelated issues limit the applicability of attribute grammars to engineering design. First, in an attribute grammar there are essentially two coupled grammars; a symbol grammar and an attribute grammar, with a strict partitioning of the total vocabulary. For a compiler, it is relatively easy to partition the vocabulary into programming language symbols and machine code "attributes." For an engineering design grammar, it is not as simple to partition the relevant vocabulary into a syntactic set and a semantic set. Secondly, the influence between symbols and attributes is unidirectional. The attributes present a second view of the derivation tree; attributes form a semantic interpretation of the syntactic state, but take no part in guiding the productions towards a goal state. This approach parallels the unending debate within linguistics of the relative primacy of syntax versus semantics. In engineering design the interaction of syntax and semantics necessitates bidirectional influence between the two descriptive systems.

The partitioning of the vocabulary is a difficult issue when syntax and semantics are fuzzy definitions in a domain. It may seem natural to associate the form variables with the grammar symbols and the function variables with the attributes. However, if the semantic rules are a mapping to the attributes of a symbol only, the function variables can have no impact on the form variables. If the opposite choice is made, associating the function variables to the symbols and the form variables to the attributes, the form variables have no feedback to the function variables. Either partitioning tactic reduces the interplay between form and function variables, severely restricting the expressiveness of the transformations and therefore restricting the utility of attribute grammars for building design.

2.2 Orthogonal Structures in Configuration Studies

In configuration studies, Flemming's orthogonal structures represent the topological relations of a set of rectangles placed within a bounding rectangle. This representation has formed the basis of a two–stage method for exploring the possible rectangular dissections of a bounding rectangle (Flemming 1986b). Two classes of constraints are successively employed: topological constraints which specify the spatial relationship between rectangles, and geometric constraints which restrict dimensional properties (e.g., maximum or minimum area of a rectangle). The topological description specifies a class of solutions containing all the geometric instantiations. This approach is exemplary in abstracting the topological and geometric aspects of a rectangular dissection in order to develop a formal representation of potential solutions as graphs whose nodes represent the rectangles and whose directed arcs represent one of two spatial relationships, above or to the right. A well-formed solution is assured by proving that the representation is closed and complete under the application of a small set of generative rules. Thus, the representation is used to prove theorems about the solutions, placing the approach on a firm theoretical basis.

Two aspects of orthogonal structures discount their use for the structural systems of buildings. First, the rules for generating orthogonal structures are embedded in a design method in which the precise number of elements to be used must be known beforehand and the elements are added one at a time while meeting a set of *a priori* adjacency constraints. In a structural system the number of elements employed may change from one potential solution to the next; the exact number of elements is rather unimportant. Likewise, the adjacency requirements on the elements of the design are a function of the partial solution rather than a part of the problem statement. Secondly, we would like to be able to employ elements whose edges are not necessarily parallel to an orthogonal grid. That is, we would like to be able to use diagonal elements such as those in trusses or braced frames. Therefore, we are inspired by orthogonal structures and their formation of the basis of a design method, but we are searching for a representation more appropriate to our domain.

2.3 Bondgraphs for Dynamic Systems

Bondgraphs are a formal representation for describing the transformation of energy in lumped-parameter systems classified according to their type of energy transfer: electrical, fluid, mechanical translation, and mechanical rotation (Paynter 1961). A bondgraph is composed of ports and bonds—the nodes and edges of the graph. Each port represents a functional object and each bond represents a path for the flow of power, described by a flow variable and an effort variable. Thus, power flow in the mechanical translation domain is the product of force and velocity—the effort and flow variables, respectively.

There are many attractive features of the bondgraph concept. First, bondgraphs represent the important components of a system along with their behavioral attributes in a computable form. By explicitly representing the effort–flow relationships among elements of the graph, the qualitative representation of bondgraphs can be used to derive a quantitative description of the system's behavior. Second, bondgraphs may represent subsystems or complete systems, and therefore two complete bondgraphs may be composed into a larger bondgraph using an n-port and its bonds. Therefore, the concept of prototypes(Gero 1987) is readily extended to bondgraphs through the use of standardized subgraphs in the composition of system level bondgraphs. Finally bondgraphs are a formal language with a well-defined grammar allowing methods for ensuring well-formed graphs.

Although structural systems are often represented as mass–spring systems during analysis, this is a cumbersome representation for designing systems. Also, bondgraphs are not applicable for systems with components whose behavior is context-dependent. For example, the reduction ratio of a gear is independent of the rotation direction or applied torque, but the deflected shape of a column is dependent on its support conditions. Furthermore, it may be argued that by the time a building can be modelled as a lumped-parameter system, the configuration problem, and therefore a major part of the structural design problem as a whole, has been completed. While the analogy between the specification of the power variables of effort and flow and the work variables of force and displacement for the description of a "static" system may lead to a quantitative description within our qualitative adjacency structures, we contend that the geometry of a building is a crucial element of this domain and must have a major role in the representation and design process, an aspect which bondgraphs ignore. Nevertheless, another inspiration in the development of adjacency structures is the schematic description of systems using bondgraphs.

2.4 Discrete Static Systems

In discrete static systems there is no action at a distance. There are no magnetic or electrical fields. There is no transmission of electric energy between components of the system. In a discrete static system, such as the structural system of a building, all communication of forces occur through the physical, *virtual* "pushing and pulling" of adjacent members of the system. Systems are composed of adjacent members arranged in a particular orientation. This orientation may be with respect to other members of the system or with respect to the environmental conditions (loads and displacements) the system is meant to counteract. A common informal language for describing the behavior of a building's structural system is in terms of load paths. These paths consist of networks of adjacent members and subsystems which communicate the loads from their sources to their supports (sinks). The introduction of any physical discontinuity in the network may change a load path. The introduction of a new connection within the network can quantitatively change the distribution of forces within the system or qualitatively change its behavior. The confirmation of expected load paths, and the discovery of new load paths and their quantitative assessment, is one view of behavioral evaluation. The circulation and communication of building occupants between the architectural volumes of a building is another example of a functional requirement of a system predicated on the adjacency of elements (rooms, corridors, floors, etc.) of the building.

Of course, building structures are not completely static—they bend, compress, and shear in response to lateral, gravity, and temperature loads. Yet, the members remain in a relatively fixed position and orientation in relation to other members in the system. Also, it is a complex process to quantitatively describe the motion and forces within a given building as it responds to these loadings. Many mathematical methods such as matrix and finite element methods have been used to quantitatively describe the transmission of forces within structural systems. However, during the design process we would also like to have a qualitative and computationally inexpensive method of describing the interrelation of physical members. Furthermore, we would like to be able to describe the potential interrelationships of physical members before the complete geometry (e.g., member sizes) of the system has been determined. Nevertheless, another major inspiration for adjacency structures is the assembly step within the matrix and finite element analysis methods.

3 Adjacency Structures

This section introduces adjacency structures and their expressiveness using an intuitive example of a truss. Then, we begin a detailed presentation of the concept by describing the atomic and primitive elements that compose adjacency structures. Next, the types of composition operations for developing higher-level adjacency structures are presented, after which a few of the possible compound components from the domain are described. Finally, the types of constraints incorporated into non-atomic components are described and a method for discretizing a geometric model to produce a canonical translation to and from adjacency structures is presented. The translation of the complete geometric model into an uninterpreted adjacency structure will be referred to as the *overall adjacency structure*. Our extension of adjacency graphs to adjacency structures adds basic geometric and material information to each node in a graph with undirected arcs.

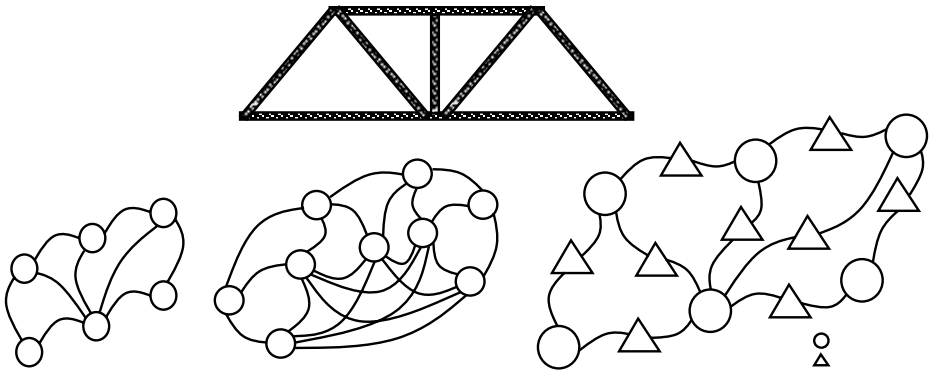

Figure 1. Nine-bar truss and three associated adjacency graphs.

We begin the presentation of adjacency structures by considering a truss. Figure 1 shows a simple nine-bar truss and three graphs. Graph *a* represents the truss joints as the graph's nodes and the truss bars as its arcs. This representation is adequate for describing the spatial relationships of the bars— each node is a member of a minimal cycle with a length of three. Also, if each node contains its coordinates it could be determined whether the truss lies in a single plane. However, since this representation contains no information about the elements connecting the nodes, it is inadequate for answering questions about the functional adequacy of the truss such as its resistance to applied loads. Graph *b* represents the bars of the truss as nodes, with the arcs representing adjacencies among the bars. It can be seen that the proper cycle information has been lost. For example bar *BC* is a part of a cycle of length three containing bars *AB* and *BD*, three bars adjacent at node *B*, but which do not form a triangle. This representation fails on its inability to represent the topological character 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 of graph *c* 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 representation of joint nodes allows for a simple determination of the planarity of the truss and 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.

3.1 Elements of Adjacency Structures

The physical objects represented by the adjacency structures described in this section are those objects representable in any geometric modelling system. Section 3.5 briefly describes the translation requirements of various geometric modelling schemes. In this section, the two lowest levels of a representation hierarchy are presented. The classification begins with the types of nodes in adjacency structures and the objects they may represent. Next, two primitive elements of the structural engineering domain are described. Section 3.2 presents three graph composition operations before a number of system components from the representation hierarchy are presented.

The leaf nodes of the representation hierarchy are the *atomic elements*, the indivisible nodes of the graph structure. These atomic elements are translations of the objects in the geometric model, and are classified according to their gross dimensional proportions. We allow four classes of nodes or *atomic elements* based on the gross dimensionality of the element; we employ zero-, one-, two- and three-dimensional nodes.

- **A zero-dimensional node** has position but no size or form. This node may represent an interface between two adjacent elements such as a joint in a truss or frame.
- **A one-dimensional node** has a length much greater than both its cross-sectional dimensions and may be used to represent a bar of a truss, a column or a beam. Its geometric information may be represented by its two end points.
- **A two-dimensional node** has a breadth and depth much greater than its height, and may represent a wall or floor plate. The geometric information of a two-dimensional polygonal node may be represented by an ordered list of its vertices.
- **A three-dimensional node** has each dimension of roughly the same scale, may represent an architectural volume or a foundation footing, and may have its geometric information represented as a nested list of the vertices of its bounding faces.

Each node is represented by a common data structure regardless of its dimensionality. The requirements of the data structure include the ability to represent nonphysical attributes which express aspects of the node's behavior and the ability to model the environmental conditions for which we are constructing load paths, i.e., loads and displacements. A single data structure represents both physical or *member objects* as well as *virtual objects*. Member objects are the physical objects of the design, e.g., truss bars, 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 fields: dimensionality, geometry, magnitude, composition and stiffness. An additional field is used for a node identifier and another field contains a list of pointers to other data structures. Other 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 building design component.

	Objects			
Fields	Member	Interface	Load	Displacement
Identifier	String	String	String	String
Dimensionality	$\in\{1,2,3\}$	$\in\{0,1,2\}$	$\in\{0,1,2,3\}$	$\in\{0,1,2,3\}$
Geometry	Vertex Coords.	Vertex Coors.	Vertex Coords.	Vertex Coords.
Magnitude	X-Sect. Dims.	X-Sect. Dims.	Load Vector	Displ. Vector
Composition	$\in\{$R.C., Steel$\}$	$\emptyset$	"Load"	"Displacement"
Stiffness	{E, I}	$\{K_\theta, K_\phi, K_\psi\}$	$\emptyset$	$\emptyset$
Arcs	List of Pointers	List of Pointers	List of Pointers	List of Pointers

Table 1. Data structure for adjacency structure nodes.

Aggregations of atomic elements are organized into a hierarchy based on topological and behavioral distinctions. A *primitive element* consists of a specific number of nodes arranged in a specific topology. The exact geometry of each node is not specified by the component's definition; it is only constrained to a specified relation to other nodes in the primitive. A *system component* is composed of an indefinite, but finite, repetition of primitive elements also constrained to specified geometric relationships. A system component is defined by its constituent primitives, retaining the constraints of its primitives but using additional constraints to define its composition into a system. A system component may be composed as a repetition of a single primitive element, or from multiple types of primitives. We call these two system types *uniform systems* and *composite systems*, respectively.

The atomic elements of adjacency structures represent the syntactic elements of all discrete static systems. Any classification of node aggregations gives a domain-dependent relevance to certain subgraphs. Thus, the definition of non-atomic elements and system components identifies relevant syntactic structures and attaches a particular semantic importance to them. Therefore, we call the collection of non-atomic components specified as being semantically relevant to the domain the *semantic templates* of that domain. The formal specification of these templates forms the basis for discovering the semantics of the purely syntactic overall adjacency structure. In this section we present an informal specification of two primitive elements from the domain of building structures.

- **Truss Panel.** A truss panel is represented as a graph whose nodes form a cycle of length six composed of three pairs of alternating zero- and one-dimensional nodes. The elements define a plane parallel to the orientation of the loads they resist, and any applied loads occur at the zero-dimensional nodes.

- **Bent.** A bent is represented as a graph whose nodes form a non-cyclic series of five nodes, composed of three one-dimensional nodes each of which is separated by a zero-dimensional node. The nodes are arranged in a single plane parallel to the orientation of the loads the bent resists, but loads may be applied to either type of node. Additional constraints on the relative angles and absolute orientations of the one-dimensional nodes must be included in the graph template.

3.2 Template Composition Operations

The expressive richness of the adjacency structure representation hierarchy is defined by the semantic templates for the domain. The composition operations described in this section facilitate the composition of systems from elements and the decomposition of systems into elements. Three types of operations useful for constructing higher level graphs are the *merge*, *abut*, and *embed* operations. Merging combines two separate graphs by unifying nodes which have the same location and dimensionality in both graphs. Merging is useful for systems, such as *plane trusses*, which are defined in terms of subgraphs sharing atomic elements. Abutting connects geometrically distinct, but adjacent, nodes in two subgraphs by adding a 'bridge' of two arcs separated by an interface node at each adjacency of the subgraphs. The dimensionality and geometry of the inserted node is equivalent to the intersection of the nodes being 'bridged.' Abutting is useful for combining systems which do not share components, e.g., two orthogonal shearwalls. Embedding inserts one subgraph within another by replacing arcs of the 'mother' subgraph with new arcs into the 'daughter' subgraph. 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 beams to the shearwall.

These composition operations are used in describing the representation hierarchy, and in the parsing and generation processes. When parsing orthogonal frames into two sets of plane frames, for example, the single column at each frame intersection must be copied into two graphs. Therefore, the representation hierarchy requires an operational definition of the transformation of one level of templates into the templates of another level.

3.3 System Components

Each of the primitives described in Section 3.1 are planar in graph–theoretic terms as well as representing objects which are relatively two–dimensional. This section describes groupings of components into larger graphs which continue to be planar and then extends these systems into graphs representing three-dimensional physical objects. Uniform compound systems are composed by repeating a single component subject to a set of geometric constraints. For example, a *plane truss* must have all its panels constrained to a single plane. In contrast, a *space truss* may be composed of the same type and number of compound components, but by using different geometric constraints the semantic template of a different system is specified. Compound systems are specified by juxtaposing multiple multiple uniform or composite systems within a set of geometric constraint. For example, a *braced frame* may be specified as a horizontally adjacent set of *plane frames* and vertical *plane trusses*, all in the same plane. In this way, a hierarchy of higher-level graph templates is defined using a small number of simple graphs which may be repeated an indefinite number of times during their instantiation.

- **Plane Truss.** A plane truss is represented as a graph composed by merging *truss panel* primitives. Each truss panel has at least one one-dimensional node and its two adjacent zero-dimensional nodes as members of one other panel. The nodes are in a plane parallel to the orientation of the forces the truss resists, and the forces are applied only to zero-dimensional nodes.

- **Plane Frame.** A plane frame is represented as a graph composed by horizontally merging, and vertically abutting, *bent* primitives. Each node is in a single plane in-line with the forces the frame resists.

- **Braced Frame.** A braced frame is composed by merging one or more *plane frames* and one or more vertical *plane trusses*, all in the same plane. Each merger is along a vertical line of shared alternating zero– and one–dimensional nodes.

Higher level templates can be defined to specify three–dimensional systems. A few three-dimensional systems are described below beginning with gravity load resisting systems and then the lateral load resisting systems with which they must eventually be unified. A bay of a flooring system is represented as a graph containing a horizontal two-dimensional node with loads oriented normal to its plane. Various flooring types have different adjacency characteristics, and three examples are described below.

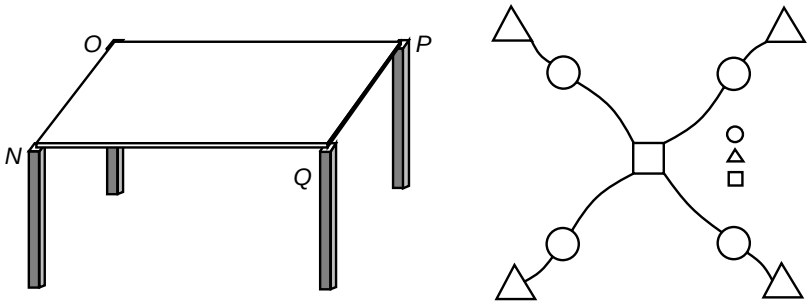

Figure 2. Pictorial and graph representations of flat plate flooring system.

- **Flat Plate.** A bay of a flat plate flooring system is represented as a graph composed of a horizontal two-dimensional node and *n* vertical one-dimensional nodes. A zero-dimensional interface node is adjacent to the two-dimensional node and each of the one-dimensional nodes as shown in Figure 2.

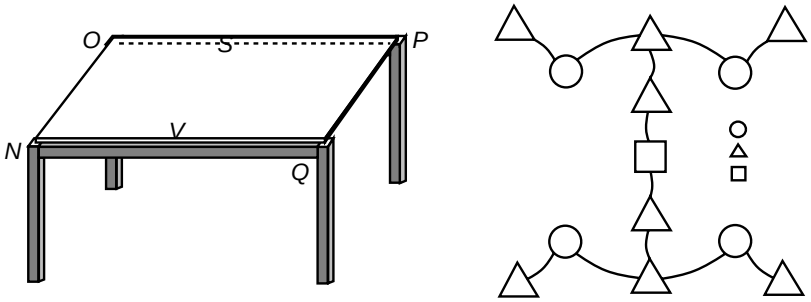

Figure 3. Pictorial and graph representations of one-way flat slab flooring system.

- **One-way Slab.** A one-way flat slab system is represented as a graph composed of a horizontal two-dimensional element abutted to *bent* components on alternating edges of the two-dimensional element. As shown in Figure 3, the two-dimensional element and each of the *bent* components is mediated by an adjacent horizontal one-dimensional interface element introduced during the abut operation.

- **Two-way Slab.** A two-way flat slab system is represented as a graph composed of a horizontal two-dimensional node abutted to horizontally merged *bent* components on each edge of the two-dimensional node. As shown in Figure 4, the two-dimensional node and each of the *bent* components is mediated by a one-dimensional interface node introduced during the abut operation. The number of merged *bent* components equals the number of edges of the two-dimensional node. In addition to these topological and orientation requirements, a bound on the aspect ratio of the two-dimensional node of a two-way flat slab system must be included in the graph template.

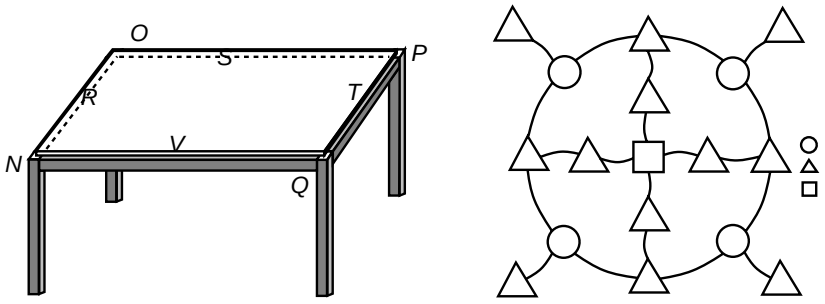

Figure 4. Pictorial and graph representations of two-way flat slab flooring system.

A more complex three-dimensional composition is a hat truss, combining a number of plane trusses into a system which must also be defined in terms of its adjacency with other subsystems in the building and in terms of its location within the overall system. A hat truss is a three-dimensional arrangement of plane trusses placed at the top of a building. The function of the hat truss is to reduce the building's lateral deflection. It accomplishes this function through the behavior of tying together the building's core and perimeter frames, and thereby altering the shape of the building's deflection curve. The specialized location, topology, and desired behavior of a hat truss leads to complex constraints which must be incorporated into the semantic template of a hat truss, and which can sufficiently describe a hat truss without overburdening the process of matching on the overall adjacency structure. A belt truss fulfills a similar function through a similar behavior, but is located in a geometrically different relation to the overall structural system. The identification and distinction of these two systems is a severe measure of the type of representation and process which we are investigating. A frequent remark from experienced structural engineers, when discussing historical developments in structural design, has focused on developing the ability to think (compute) about structures and visualize their behavior in three dimensions. The specification of three-dimensional adjacency structures provides the ability to reason in three dimensions.

3.4 Constraints within Semantic Templates

The topology of a semantic templates clearly specifies how the nodes are configured, but the nodes representing physical objects have only their dimensionality fixed by the template; the template does not assign the geometry, magnitude, etc. of the constituent nodes. Previous sections have mentioned the constraints associated with components and systems. This section lists the types of constraints which may be associated with the semantic templates. Constraints are represented as part of the transformations which decompose (or compose) the templates during parsing (or generation). The types of constraints are geometric constraints, functional constraints expressed in terms of loads, and behavioral constraints which relate applied loads to the resulting displacements.

- Geometric constraints:
  - Planarity: The nodes of a graph or subgraph must reside within a single (oriented) plane.
  - Symmetry: The topology or geometry of a subgraph must be equivalent to that of another subgraph in the template under a reflective transformation.
  - Single node aspect ratio: The geometric proportions of a two– or three–dimensional node must be within a specified range.
  - Complete graph aspect ratio: The geometric proportions of the convex hull of a semantic template must be within a certain range.
  - Location: A subgraph must be in a particular geometric relation to the overall graph.
  - Associativity: The nodes of one subgraph must be joined to the nodes of a specific dimensionality, location or orientation in another subgraph.
- Functional constraints:
  - Dimensionality: The applied loads or displacements must be of a specific dimensionality, e.g., zero–dimensional loads applied to a truss template.
  - Location: The applied loads or displacements must be located at specific positions of the graph to which they are attached, e.g., to zero–dimensional nodes of a truss template.
  - Orientation: The applied loads or displacements must be oriented in a specific direction relative to the graph, e.g., distributed loads oriented perpendicular to nonzero-dimensional nodes.
- Behavioral constraints:
  - Stiffness: The lateral deflection at a specific location of the template is directly proportional to the applied load and inversely proportional to the flexural stiffness of constituent subgraphs.
  - Strength: The internal forces of constituent nodes in the template are related to the applied loads, the material properties, and cross-section of the node.

3.5 Translation and Discretization of the Geometric Model

Adjacency structures are a graphical representation of individual, physical objects and their adjacencies. Geometric modelling systems represent these objects in various ways. An algorithm for translating an arbitrary geometric modelling representation into a canonical adjacency structure is given in (Meyer 1992). It is the responsibility of a translation function to recognize, for a particular geometric modelling system, the characteristics which determine which data structures represent a physical object in that modelling system. For example, when using a boundary representation solid model each node of an adjacency structure will represent one solid, or the discretization of one solid, in the geometric model. In a non-manifold modelling system, in contrast, dangling faces and edges may also represent physical objects from the domain. Once the individual object's representations have been distinguished, these objects may need to be discretized into smaller objects before becoming part of the overall adjacency graph. One motivation for discretizing the model is to arrive at a scheme for producing a canonical representation of any geometric model. For example, when representing a 40-story building, is a geometrically continuous column to be a 40-story tall object? We could use the maximal line method from shape grammars to produce a canonical representation. However, looking back at the truss example of Figure 1, with the maximal line method the top and bottom chords of a truss would each be modelled as one node rather than as separate nodes for each panel. Instead, we work under a minimal extent scheme, discretizing every modelled object at any location of intersection or adjacency with another modelled object.

4 Semantic Interpretation of Syntactic Structure

The formation of the overall adjacency structure from a geometric model is a translation of one model's syntax into the syntax of another modelling system. The expense of this translation is only worthwhile if the new modelling system is more useful for some particular purpose. In this section, we present how the adjacency structure representation of a geometric model may be used to discover emergent semantics within the syntactic representation, and to check the consistency of intended semantics with the syntax of the adjacency structure representation. This discovery process is the parsing mode mentioned at the beginning of this paper.

The parsing of an overall adjacency structure into semantic templates is an inductive procedure, translating a specific syntactic structure into a set of semantic templates general to the domain. To illustrate this procedure, let us assume that we have translated a geometric model representing the structural system of a building into an adjacency structure as described in Section 3.5. Beginning from the highest level of the adjacency structure taxonomy, an attempt is made to unify semantic templates with the adjacency structure. In structural engineering the class of three–dimensional systems incorporates both a lateral load resisting system and, with the addition of floor slabs, a gravity load resisting system. This class includes the varieties of tube structures and orthogonal framing systems. Therefore, the first templates which may be unified with the overall adjacency structure are these three–dimensional system templates. No more than one semantic template should match on a given overall adjacency structure during parsing. For example, a framed tube structure should not be described as an orthogonal rigid frame structure even though the framed tube may consist of four rigid frames oriented orthogonally.

After the highest level adjacency structure has been unified with the overall adjacency structure, matches are sought for any remaining subgraphs. These ancillary adjacency structures must be compatible with the previously unified adjacency structures. Adjacency structure compatibility entails being able to combine multiple subgraphs through the available composition operation described in Section 3.2, while satisfying the applicable constraints on each adjacency structure.[2] The need for ancillary adjacency structures will be common for building with non-rectangular plans and massings, for example.

The purely syntactic matching process 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 to its neighbor 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 existing syntactic compositions. That is, the matching process may also be utilized to find the presence of extraneous elements or the omission of necessary elements. 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 the overall adjacency structure 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 floor slab and column placements may be specified by the architectural design. However, this design is not meant to rule out the use of beams in the structural design or to preclude the use of a one-way or two-way flat slab gravity system. 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.

These three results of parsing can summarize the syntactic condition of the design: syntactically complete, intentionally or unintentionally redundant, or syntactically incomplete. First, the overall adjacency structure may be completely parsed into atomic elements, describing the overall adjacency structure in terms of the templates at successive levels of decomposition. This signifies a syntactically complete design. Second, the overall adjacency structure may be decomposed, but leave 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.

[2] The recurrent combining of adjacency structures not contained as single systems in the representation hierarchy during the parsing process may afford one method of learning new and useful system components.

5 Adjacency Structures in a Design Process

The various design processes in which an architect and a structural engineer interact may be viewed along the two axes of phase and domain, shown in Figure 5. The predominant architect–engineer design process begins with an architectural parsing of the client's design brief, then proceeds with an architectural generative phase. Next, structural design begins with a parsing of the architectural design to find pertinent geometric and functional information, then proceeds through a structural generative phase. In this design process, the architectural phase sets the geometry of the building envelope and constrains the location of any internal elements which the structural engineer may introduce. The role of the structural engineer during preliminary design becomes one of proposing a small number of "good" structural systems which can support, and be accommodated within, the architecturally subdivided envelope. This section presents the use of adjacency structures in the transformation of functional requirements into a structural description.

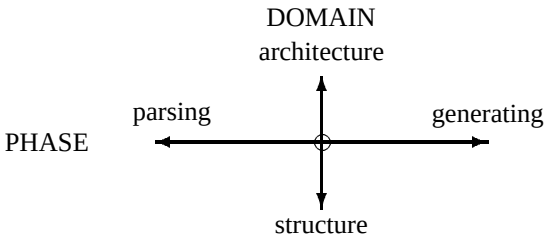

Figure 5. Design as processes along Domain—Phase axes.

The mapping from the functional requirements of a design to instances of design structure which satisfy those requirements can be performed as a successive refinement of elements of a functional hierarchy. A representation for such a process must include attributes which can explicitly represent at least function, if not behavior. The inclusion in the semantic templates of nodes representing loads and displacements can represent both given and propagated functional requirements such as applied loads or imposed displacements. Behavior, in the form of the flexural stiffness and axial forces of the representation hierarchy. Thus, a uniform representation is used to model function and structure, providing a transparent method of propagating functional requirements within a partial design solution.

To illustrate the use of adjacency structures in a design process let us assume that we have a geometric model representing a preliminary architectural design. The architectural description includes a definition of the building's envelope as the external surface of the geometric model, plus any interior geometric entities representing the location of 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 described in Section 3.5. 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, deflection limits are specified. Together, the architectural envelope, internal elements, applied loads and deflection limits

form the specific functional requirements for the structural design. The structural design phase consists of instantiating subgraphs which, together, satisfy these specific functional requirements, along with general requirements such as limits on member forces.

Design generation, like the parsing process, 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 differs from the control of the parsing process in that multiple instantiations are sought which satisfy the functional requirements; the control of the generation process should 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 may be divided into two phases: topological instantiation 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 nodes composing the templates are assigned values. 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. For example, a *plane frame* template must be composed of an integer number of horizontally merged *bent* templates and an integer number of vertically abutted *bent* templates. Thus, topological instantiation involves a determination of the dimensions of the target region[3] for the potential instantiation of a template and a comparison of the template's application limits to arrive at possible dimensions for the region's division. The order of these two steps is dependent on the design process in which it is used. In the design process described at the beginning of this section, it is appropriate to first determine the acceptable range of the subdivision dimensions. 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 the constraint satisfaction necessary in the topological instantiation of semantic templates; their own limits of application are underconstrained. When the number of component templates composing the specific system is determined, the topological instantiation can be accomplished by generating the proper number of subgraphs which realize a compound template, assigning values to the appropriate data fields in each node of the template, and embedding it in the overall adjacency structure. After the

[3] A region is a general geometric space in R^1, R^2 or R^3 i.e. the division of a region may be the division of a beam, floor plan or architectural volume.

first template system is instantiated and added to the overall adjacency model, subsequent instantiations are also constrained to accommodate the existing templates 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.

The parametric instantiation is primarily concerned with the defining of member cross-sections. In order to accomplish this parameterization some level of analysis is needed. One advantage of the node and system representation we have described is its ease of translation into the matrix methods of analysis. Each member node, if it had its cross-section geometry 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 a relative global stiffness matrix from relative element stiffness matrices. Thus, one design process is as follows:

1. The topological instantiation defines the length of one-dimensional elements and the breadth and depth of two-dimensional elements when the building envelope is subdivided into an integer number of templates.

2. The relative stiffness of all member nodes is defined. This relative stiffness is used to define element stiffness matrices for each member node in the overall adjacency structure as a function of its moment of inertia I and the modulus of elasticity E.

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

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

5. The stiffness is assigned to limit the deflections to an allowable amount. The assigned stiffness allows the back calculation of the member stiffnesses and, therefore, 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.

6 Conclusion

This research is motivated by the belief that a hierarchical organization of design components reflects the way human designers think about building structures, and that basing a graph representation on the adjacencies of physical components reflects both the way building structures are built and the way they behave as systems. The representation presented in this paper provides a more convenient, higher-level means of operating on a system of geometric objects than such representations as the split-edge data structure used by boundary representation geometric modelling. At the same time, the graph representation of adjacency structures provides a more explicitly system-oriented representation than objects, frames or prototypes for a domain in which a design is composed of a large number of highly interconnected, primarily geometric objects. For these reasons, we feel that adjacency structures provide an intuitive and convenient representation for design generation and evaluation, a representation which captures the essential geometric and systematic nature of discrete static systems.

The typical result of a design process is a description of the physical shape and material composition of an artifact delivered in response to a set of functional requirements. The drawings and specifications delivered by an architectural/engineering firm as the design of a building are a purely syntactic description of the building. However, the design mandate for the artifact was given in terms of the semantics of the design—the functions the proposed artifact must satisfy. The parallel between the form–function dichotomy and the syntax–semantics dichotomy is apparent in a domain where a design may be specified purely in terms of a description of the artifact's shape and material. The syntax of the design describes the form of the proposed artifact whereas the semantics of the design describes the functional requirements and the behavioral expectations of the proposed artifact.

We have presented a representation which incorporates syntactic aspects of a design in terms of adjacent physical objects with semantic aspects of the domain in terms of systems, loads, and displacements. We have also presented methods for utilizing this representation in the design process as bidirectional mappings between the functional, behavioral, and structural views of a design description. These two mapping directions support design through mapping from function to structure (synthesizing potential solutions), through mapping from structure to behavior (qualitatively analyzing potential solutions), and through mapping from behavior to function (evaluating a potential solution). The adjacency structure representation also provides an important link between design visualization, provided by geometric modelling, and design analysis provided by the matrix methods. Through these mappings, this representation and design process supports a function–to–structure design method for the domain of discrete static systems.

References

Finger, S. and Rinderle, J., 1989. "A Transformational Approach to Mechanical Design using a Bond Graph Grammar," in *Proceedings,* Design Theory and Methodology Conference, ASME.

Flemming, U., 1986. "On the Representation and Generation of Loosely Packed Arrangement of Rectangles," *Environment and Planning B: Planning and Design*, 13, pp. 189–205.

Flemming, U., Coyne, R., Glavin, T. and Rychener, M., 1988. "A Generative Expert System for the Design of Building Layouts – Version 2," In J.S. Gero (ed.), *Artificial Intelligence in Engineering: Design (Proceedings of the Third Internaional Conference, Palo Alto, CA).* New York: Elsevier, p. 513.

Gero, J.S., 1987. "Prototypes: A New Schema for Knowledge-based Design," working paper, Architectural Computing Unit, University of Sydney.

Gero, J.S., Lee, H. and Tham, K., 1991. "Behaviour: A Link Between Function and Structure in Design," in *IntCAD '91*, IFIP Working Group 5.2, Columbus, OH.

Knuth, D. E., 1968. "Semantics of Context-free Languages," *Mathematical Systems Theory,* 2(2), pp. 127–145. Corrections in *MST,* Vol. 5, No. 1, pp. 95-96.

Meyer, S. and Fenves, S.J., 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.

Mitchell, W., Steadman, J. and Liggett, R., 1976. "Synthesis and Optimization of Small Rectangular Floor Plans," *Environment and Planning B*, 3, pp. 37–70.

Paynter, H. M., 1961. *Analysis and Design of Engineering Systems*. Cambridge:MIT Press.

Pinilla, J., Finger, S. and Prinz, F., 1989. "Shape Feature and Recognition using an Augmented Topology Graph Grammar," in *Proceedings,* 1989 NSF Engineering Design Research Conference, Amherst, MA.

Ulrich, K. and Seering, W., 1987. "A Computational Approach to Conceptual Design," in *Proceedings,* International Conference on Engineering Design.

References

[Finger 1989] S. Finger and J. Rinderle. "A transformational approach to mechanical design using a bond graph grammar." In *Proceedings, Design Theory and Methodology Conference*, ASME, Montreal, September 1989.

[Flemming 1986a] U. Flemming. "On the representation and generation of loosely packed arrangement of rectangles." *Environment and Planning B: Planning and Design*, 13:189–205, 1986.

[Flemming 1986b] U. Flemming, M. Rychener, R. Coyne, and T. Glavin. "A generative expert system for the design of building layouts, version 1." Progress report, Center for Art and Technology, Carnegie Mellon University, June 1986.

[Gero 1987] J. Gero. "Prototypes: A new schema for knowledge-based design." Working paper, Architectural Computing Unit, University of Sydney, 1987.

[Knuth 1968] D. E. Knuth. "Semantics of context-free languages." *Mathematical Systems Theory*, 2(2):127–145, 1968. Corrections in MST Vol. 5 No. 1 pp. 95-96.

[Meyer 1992] S. Meyer and S. J. Fenves. "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, 1992.

[Mitchell 1976] W. Mitchell, J. Steadman, and R. Liggett. "Synthesis and optimization of small rectangular floor plans." *Environment and Planning B*, 3:37 – 70, 1976.

[Paynter 1961] H. M. Paynter. *Analysis and Design of Engineering Systems*. MIT Press, 1961.

[Pinilla 1989] J. Pinilla, S. Finger, and F. Prinz. "Shape feature and recognition using an augmented topology graph grammar." In *Proceedings of the 1989 NSF Engineering Design Research Conference*, Amherst, MA, 1989.

[Ulrich 1987] K. Ulrich and W. Seering. "A computational approach to conceptual design." In *Proceedings of the International Conference on Engineering Design*, August 1987.