Combinatorial laws for physically meaningful computational design Extended Abstract

Vasu Ramaswamy **Vadim Shapiro***
Spatial Automation Laboratory †
University of Wisconsin-Madison

Motivation

A typical computer representation of a design includes geometric and physical information organized in a suitable *combinatorial* data structure. Queries and transformations of these design representations are used to formulate most algorithms in computational design, including analysis, optimization, evolution, generation, and synthesis. Formal properties, and in particular problem-specific existence and validity of the computed solutions, should be assured and preserved by all such algorithms. Given a rich variety of representations and algorithms for computational design, there appears to be little reason to believe that most of them may be formulated within a single combinatorial framework. We argue that not only such a framework is possible (Palmer & Shapiro 1993), but it is formal (Branin 1966) because it is based on the tools from algebraic topology (Hocking & Young 1988), practical (Palmer 1995; Egli & Stewart 1999) because it arises naturally from computational considerations, and ubiquitous (Tonti 1975) because it spans most of the geometric and physical laws.

Designs are oriented cell complexes

Most combinatorial representations of design on the computer can be effectively constructed using oriented cellular structures, formally known as *cell complexes*. Combinatorially, cell complexes are compositions of oriented k-cells with k = 0 (vertex), 1(edge), 2(surface), 3(volume), typically embedded in 3-D space without intersections and subject to the additional constraint that the closure of every cell is a union of other cells in the complex. Instances of cell complexes used in computational design are illustrated in Figure 1. Voxel-based representation, such as used in evolutionary design of tables (Hornby & Pollack 2001), are essentially 3-dimensional cubical cell complexes, where 3-cells correspond to individual voxels, and lower-dimensional cells implicitly correspond to adjacency relationships (2-cells for face adjacency, 1-cells for edge adjacency, and 0-cells for corner adjacency). Truss and frame structures that are commonly used in shape-grammar based design (Reddy & Cagan 1995) are 1-dimensional complexes composed of 0-cells (joints) and 1-cells (members). Boundary representations of solid models (Requicha 1977) and subdivision surfaces (Taubin 1995) are 2-dimensional geometric cell complexes with faces, edges, and vertices corresponding to 2-cells, 1-cells, and 0-cells respectively.

Design Properties are cochains

Geometric and physical properties of the design correspond to functions evaluated on the corresponding cell complexes, formally known as *cochains*. A k-cochain associates one coefficient (value of the property function) with each k-dimensional cell in the cell complex. Cochains in our examples include geometric properties (for example, geometric locations of 0-cells of a truss, voxels of a table, and points of a subdivision surface patch), as well as a variety of physical and non-physical attributes, such as forces and displacements on a truss structure or voxel-based models of tables, and weights associated with edges of a subdivision surface in (Taubin 1995). All p-cochains defined over a cell complex form a linear vector space with dimension equal to the number of the p-cells and addition and multiplication operations are defined for p-cochains in a cell-by-cell fashion, very much like for vectors. These and more general operations on cochains have been used in several physical modeling and simulation systems that treat cochains as a basic data type (Palmer 1995; Egli & Stewart 1999) the emergence of which allowed to replace many cell-by-cell operations and algorithms by powerful and elegant algebra and language of cochains. Formal definitions of cell complexes, and cochains can be found in (Hocking & Young 1988) and most texts on algebraic topology.

Design Operations and Laws

The main utility of chains and cochains in computational synthesis lies in that they define a convenient class of objects for algorithmically constructing and simulating complex geometrical and physical models. It follows that computational design concerns largely with generation, transformation and queries of various cochains that are subject to some postulated geometric and physical constraints. Most such con-

*The corresponding author
†Complete address: 1513 University Avenue, Madison, WI 53706, USA. E-mail: ramaswam@sal-cnc.me.wisc.edu, vshapiro@engr.wisc.edu

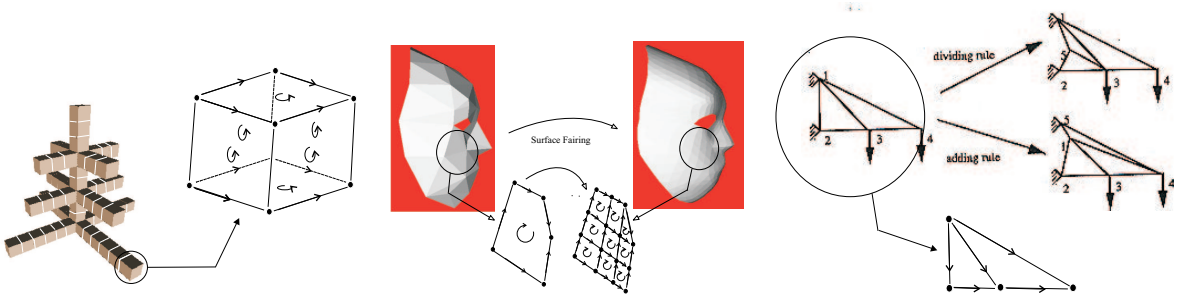

Figure 1: Common examples of combinatorial structures used in computational synthesis; each is an instance of an oriented cell complex.

straints can be characterized as compositions of topological and metric laws (Tonti 1975).

Topological relations can be formulated using strictly topological operations on cochains, such as coboundary δ that maps k-cochains into $(k+1)$-cochains. Examples of topological laws include geometric integrity, balance, equilibrium, compatibility, and conservation. One of the advantages of using δ to formulate physical laws is that it applies to any and all cell complexes. For example, assuming that the cells of the voxel-based table design example are endowed with physics of solid mechanics, the body force F (or 3-cochain) of each 3-cell is balanced by the resistive traction forces S (or 2-cochain) on its incident 2-cells. The force balance can be compactly written as $F = \delta S$.

Metric laws often specify measured or weighted relationships between a k-cochain and a *dual* $(n-k)$-cochain associated with the dual of the cell complex. The common examples of such laws include Ohm's law, Hooke's law, and other constitutive relationships.

A wide variety of broadly applicable laws may be constructed by combining the primitive measure dependent and topological laws. That many of these laws follow similar patterns has been observed by many, notably (Roth 1955; Branin 1966). As shown in (Tonti 1975; 1999), most physical laws may be represented in a factored form as a composition of the primitive topological and measured relationships. Efforts to develop an interactive geometric language for describing, combining, and editing combinatorial laws are currently under way (Chard & Shapiro 2000). The factored form of physical laws may be also used directly to formulate, compare, and improve various numerical discretization and solution schemes (Hyman & Shashkov 1997; Mattiussi 2000). Furthermore, the same approach may be used to represent and enforce a variety of synthetic or user-defined laws, relations, and constraints. An even greater variety of laws may be constructed using operators proposed by (Palmer 1995) and (Egli & Stewart 1999), though their properties have not been studied formally.

Future Directions

Reformulating design representations and algorithms in algebraic topological terms introduces a non-trivial conceptual overhead, but this additional investment is well worth it. The most immediate benefit is a unifying combinatorial framework for computational design and an algorithmic toolbox for constructing a variety of laws and invariants. The combinatorial representation is standard in that it is a straightforward extension of the current geometric modeling practices and systems. The proposed formulation of laws and invariants is broadly applicable across diverse design domains and computer representations. Implementation should be practical because the primitive laws are quite simple. Thus, a promising direction for future research is to reformulate the common techniques of computational synthesis (including L-systems, shape grammars, genetic algorithms, and subdivision) in terms of law-preserving transformations on algebraic topological chains and cochains.

References

Branin, F. H. 1966. The algebraic-topological basis for network analogies and the vector calculus. In *Proceedings of the Symposium on Generalized Networks*, volume 16, 453 – 491. Brooklyn, New York: Polytechnic Institute of Brooklyn.

Chard, J. A., and Shapiro, V. 2000. A multivector data structure for differential forms and equations. *IMACS Transactions, Mathematics and Computers in Simulation*.

Egli, R., and Stewart, N. 1999. A framework for system specification using chains on cell complexes. *Computer-Aided Design* 31(11):669–681.

Hocking, J. G., and Young, G. S. 1988. *Topology*. Dover.

Hornby, G. S., and Pollack, J. B. 2001. The advantages of generative grammatical encodings for physical design. *Congress on Evolutionary Computation* 600–607.

Hyman, J., and Shashkov, M. 1997. Natural discretizations for the divergence, gradient and curl on logically rectangular grids. *International Journal of Computers and Mathematics with Applications* 33(4):81–104.

Mattiussi, C. 2000. The finite volume, finite element, and finite difference methods as numerical methods for physical field problems. *Advances in Imaging and Electron Physics (P. Hawkes, Ed.)* 113:1–146.

Palmer, R., and Shapiro, V. 1993. Chain models of physical behavior for engineering analysis and design. *Research in Engineering Design* 5:161–184.

Palmer, R. S. 1995. Chain models and finite element analysis: An executable chains formualtion of plane stress. *Computer Aided Geometric Design* 12:733–770.

Reddy, G., and Cagan, J. 1995. An improved shape annealing algorithm for truss topology generation. *ASME Journal of Mechanical Design* 117(2(A)):315–321.

Requicha, A. A. G. 1977. Mathematical models of rigid solid objects. Tech. Memo 28, Production Automation Project, University of Rochester, Rochester, NY.

Roth, J. P. 1955. An application of algebraic topology to numerical analysis: On the existence of a solution to the network problem. *Proc. Nat. Acad. Sci.* 41:518–521.

Taubin, G. 1995. A signal processing approach to fair surface design. *Siggraph '95 Conference Proceedings* 351–358.

Tonti, E. 1975. *On the Formal Structure of Physical Theories*. Milan: Istituto Di Matematica Del Politecnico Di Milano.

Tonti, E. 1999. *Finite Formulation of Field Laws*. unpublished.