From Discrete Structures to Mechanical Systems: A Framework for Creating Performance-Based Parametric Synthesis Tools

Kristina Shea
Alex C Starling

Engineering Design Centre
Department of Engineering
Cambridge University
Trumpington Street
Cambridge CB2 1PZ
United Kingdom

Abstract

This position paper outlines the authors' vision for creating effective and innovative computational design synthesis tools. Development of performance-based synthesis methods is discussed in the context of a parametric synthesis framework that consists of four main phases: (1) *investigate*, (2) *generate*, (3) *evaluate* and (4) *mediate*. These four phases are described and analyzed in detail with examples from current research to develop methods for architectural, structural and mechanical design. So far, the prototype tools show great capacity for generating designs that exhibit performance-driven spatial innovation whereas achieving functional innovation is a future research goal.

Introduction

Computers continue to change the world of design; what we can design, what we can fabricate and construct, and as a result even what we imagine. In an increasingly competitive world market, advanced computer tools are a key to faster design of innovative, cost-effective and reliable products. To date, software for engineering and architectural design has progressed from coarse drafting systems to powerful tools that incorporate solid modeling and a wide range of analysis capabilities. Current tools help us to easily create and modify accurate digital models of our ideas as well as determine their feasibility and manufacturability. However, they are limited to modeling our ideas rather than actively contributing to them by generating new design alternatives beyond our own insight.

This research aims to expand the role of the computer in the design process from a tool for modeling and analysis to a synthesis partner. Computational synthesis is a difficult area of study since design synthesis tasks are generally ill-structured (Simon 1973). Creating effective computational synthesis methods requires the development of new ways to represent and intelligently modify both the form and function of designs as well as automatically assess their performance and search vast design spaces to generate promising and novel alternatives.

A general framework for creating performance-based parametric synthesis methods and tools will be described and illustrated through a decade of research in structural synthesis that has now become prototype software called eifForm aimed at both structural and architectural design. Work on developing a method for mechanical synthesis will be discussed to illustrate the domain independence of the approach and compare challenges. Issues relating to developing effective generative representations, function-form relations and encoding of design knowledge are discussed.

Design synthesis background

Since the advent of the computer, research has been underway to develop methods for computational synthesis (Post 1943) and has been ongoing now for three decades. The current state-of-the-art in design synthesis in a variety of domains can be found in (Antonsson and Cagan 2001) and (Chakrabarti 2002). In architectural design, production systems such as shape grammars (Stiny 1980) have been utilized for creating new designs and characterizing designers, such as Siza's housing types (Duarte Forthcoming). However, few synthesis systems in architecture include performance feedback. One example can be found in (Caldas 2002) who uses a genetic algorithm combined with lighting and thermal analysis to generate novel and performance driven building envelopes and room configurations.

In structures, topology synthesis has been investigated since the early 1900s when Michell (1904) developed an analytic means for optimal structural layout of single purpose truss structures. Current methods for structural layout use either continuous or discrete representations. Discrete methods, the focus of work presented here, can be classified as either topology reduction or topology

construction (Antonsson and Cagan 2001). Topology reduction methods transform the topology layout problem to one of sizing a highly connected ground structure (Bendsøe 1995) making them difficult to classify as computational synthesis. Rather, topology construction methods build configurations by adding and removing structural members, e.g. using heuristics and structural behavior knowledge for introducing new structural members (Bojczuk and Mroz 1999). Lastly, emphasizing the spatial aspects of structural synthesis, structural morphology methods have been developed for generation of regular, symmetric space structures based on Formex algebra (Nooshin and Disney 2000).

Fewer computational approaches exist for mechanical synthesis. Chakrabarti (2002) generates solution concepts by combining basic functional elements serially where the 3D part models that embody these functional solutions are created afterwards. Lipson and Pollack (2000) use evolutionary methods to simultaneously create structures and controlling software with optimized locomotion ability, demonstrating the validity of their results by rapid prototyping of designs. Finally, Li et al. (2001) demonstrate the use of grammar-based design methods for the conceptual design of innovative epicyclic gear trains.

Performance-Based Parametric Synthesis

A framework for creating performance-based parametric synthesis tools has been developed. The framework is general so as to remain domain independent and promotes:

- a bottom-up synthesis model,
- 3D parametric representations of fundamental design components and their direct manipulation,
- sufficient knowledge within the design representation to allow quantitative performance analysis using integrated software where possible and enabling possibilities for direct prototyping,
- simple, flexible, robust and efficient representations of the design space (or design language),
- generation of both innovative and conventional designs, the latter of which can be used for theoretical validation of the system, and
- development of tools that enhance designers' capabilities for innovation and creativity.

The framework includes four main phases: (1) *investigate* (2) *generate* (3) *evaluate* and (4) *mediate* (fig 1). These phases are interrelated and form an iterative cycle to provide a framework for design synthesis research as well as using synthesis tools to design. This paper will illustrate the advantages of the framework aims through work in discrete structural synthesis and optimization as well as mechanical systems synthesis.

Investigate

Creating a synthesis method starts with investigating a domain and identifying a class of designs within this domain to generate, for example planar truss structures. The range of existing designs, history of design innovation as well as current design methods and practice, including computational support, are explored. The goals of the investigation are to discover what has driven the development of current designs and where synthesis tools could aid in providing new, improved alternatives to enhance design creativity and innovation.

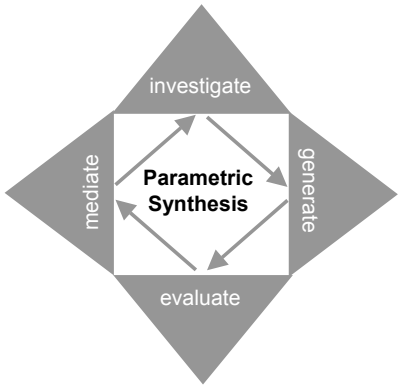

Fig 1: Framework for creating performance-based parametric synthesis tools.

Discrete Structures. Truss structures have provided a good starting point for developing synthesis methods since they are composed of components that all behave in the same way yet offer extensive opportunities for spatial innovation. Functionally, trusses adhere to a simple underlying rule, i.e. Maxwell's rule, that describes valid, or stable, designs. Investigating common truss structures (Schodek 2001), they can be classified by four attributes: base topology (triangle/tetrahedron), symmetries, spatial organization of a system of trusses (one-way/two-way) and the physical design space (2D/3D).

Design of discrete structures, e.g. trusses and frames, is a domain with an immense history going back to Vitruvius' design guidelines set out in the first century BC. Current practice relies heavily on the use of design heuristics, codes and standards that have evolved over the years. Computational tools commonly used in structural design include CAD and structural analysis tools, some of which include code checking. However, new manufacturing capabilities provide great opportunities for moving away from traditional compositions based on symmetry and repetition to explore new geometry. Breaking from convention and navigating these new design spaces is the area where a computational structural synthesis tool is targeted to make the largest impact.

Mechanical Systems: clocks, watches and cameras. The parametric synthesis framework is equally valid for mechanical systems. Many of the implementation issues faced when approaching mechanical and structural problems are similar. Design synthesis of mechanical systems is complex due to the high level of coupling between function, i.e. desired behavior, and structure, i.e. actual components. Further, there is often a higher degree

of function sharing when compared to structural design. Understanding the interactions between function and structure of a mechanical design is an important prerequisite to being able to generate beneficial solutions to design problems. The remit of this work is to develop a general method for generating mechanical systems that can be characterized by having an element of power flow, such as gear mechanisms in mechanical clocks and transmission systems as well as winding and shutter mechanisms in cameras. Currently, this type of design is mostly done by hand. Many examples of mechanisms found in clocks and watches exist in the public domain, e.g. patents, making this an interesting area of study and also providing plenty of examples for redesign and theoretical validation.

Generate

Information from the investigate phase forms the basis for developing a generative mechanism, both written and computational, that is capable of generating both existing and new designs. The generative mechanism is aimed at being a concise description of a large language of designs each composed of fundamental physical components, e.g. truss member, gear, shaft, with requisite geometric and material information such that designs can be both quantitatively evaluated and possibly prototyped directly. The aim is to create a generative representation that is as general as possible, forms an efficient representation of the design space, yet is capable of producing complex and meaningful designs. The design language produced reflects the production system chosen, e.g. expert system, genetic operators, Lindenmayer system or shape grammar. This paper will focus on research in creating engineering design grammars.

Structural grammars. Inspired by work in shape grammars (Stiny 1980), structural grammars have been developed for several classes of discrete truss structures. It is often noted that design grammars can be too complicated to create to be worth the effort. The structural grammars shown were motivated by the simpler, more fractal-like grammars, e.g. Stiny's ice-ray grammar (1977) that uses a concise representation, i.e. four parametric rules, and iterative application to produce complex patterns for Chinese lattice designs.

Rather than create one grammar for generation of all possible truss structures, it has been beneficial to categorize trusses into classes and create grammars for each class. In addition to differentiating structural classes by spatial organization (one-way/two-way) and the physical design space (2D/3D), the grammars are also distinguished by the transformations necessary to generate the designs, i.e. base topology transformation (triangle/tetrahedron) and spatial transformation (2D/2.5D/3D). Each grammar consists of one subset of rules that change the size of any structural member, a second subset of rules that move the location of joints within a design and a third subset of rules that transform the topology, or connectivity, of a design by adding and removing members in a controlled way so that the design always represents a functionally valid truss structure (fig 2). The planar grammar shown generates one-way structures in 2D by transforming designs composed of triangles. Instantiation of all possible sequences of rules leads to an infinite language of designs.

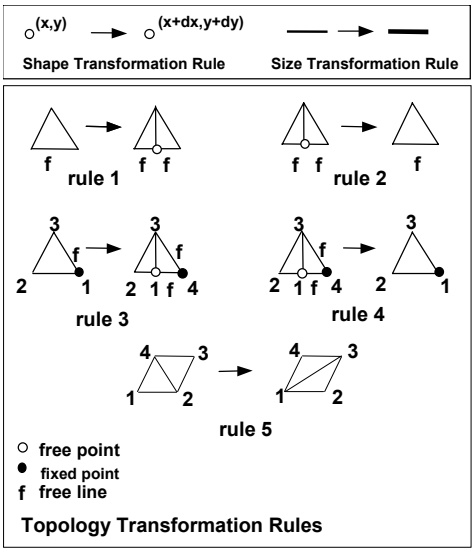

Fig 2: Planar structure grammar

The structural grammars developed are fully parametric and use global constraints that act over all rules such as minimum angle between structural members to restrict the designs produced by the grammar. As shown, the rules are more akin to schemas, for simplicity, and do not illustrate these constraints. This separation between fundamental requirements of the synthesis mechanism and more specific design knowledge, including engineering requirements, has proven essential to the creation of simple yet robust generative mechanisms. The grammar itself is less susceptible to the difficulties of maintaining explicit knowledge within the rules themselves as has been shown to be a difficulty with knowledge-intensive rule-based systems. However, the designer is still left to manage the global constraints, although this is a familiar task for engineers.

Computationally, generated structures are represented as directed graphs based on a solid modeling approach that uses a winged edge data structure (Heisserman and Woodbury 1994). This choice of representation has made it very flexible and easily extensible to all of the structural classes considered so far. For a planar structure, the spatial graph is flattened making the exterior boundary of the structure, or structural envelope, detectable by moving counterclockwise around the 'free' edges of the graph (fig 3). This is useful for instance to automatically apply a load to the top chord only of a truss structure and is being used in current work investigating the synthesis of systems of trusses (two-way), e.g. the structural system in a stadium roof.

Based on an investigation of the structural patterns used in geodesic dome design, a new structural grammar that

uses the 3D version of Maxwell's law was developed based on the planar version (Shea and Cagan 1997). Using the same computational representation shown, 2.5D structures were produced simply by taking a planar structure and projecting it onto a prescribed surface in 3-space, e.g. a 'squashed' hemisphere (fig 4), or by free projection normal to the plane of the planar structure (fig 10). If no members are allowed to overlap in the planar representation, then the resulting 2.5D structure will create a faceted surface that encloses a volume with no internal members. If intersections are allowed in the planar representation, a simple removal of one global constraint, more complex interweaving 3D truss designs are produced.

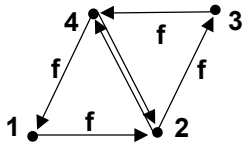

Fig 3: Computational structural grammar representation

An augmented solid modeling representation was chosen originally to facilitate the extension to generating 3D truss structures. However, through a recent investigation of existing 3D truss designs and conversations with structural designers it was discovered that designers want a synthesis tool that generates designs which balance regularity and symmetry with spatial innovation. To achieve this effect without defining different types of symmetry constraints to maintain, a hierarchical composition model is used. Rather than generating 3D trusses using tetrahedrons, as originally thought, it is more effective and robust to generate a planar design (truss A), copy and geometrically transform it to produce the other half of a 3D truss (truss B) and use a bracing algorithm to tie the two halves together (fig 4b). Essentially, the representation consists of three sub-graphs, truss A, truss B and bracing where truss B and bracing are both derived from the original truss, truss A. Using the same planar grammar (fig 2) used to create fig 4a and enabling several possible geometric transformations of the design copy, e.g. translation, rotation, reflection, scale and their compositions, as well as using several bracing algorithms, a fairly comprehensive language of 3D truss structures can be produced from a very simple structural grammar. The example shown (fig 4b) was generated by reflecting the copied truss about a vertical plane defined by the lower truss chord.

A Parallel mechanical grammar. Moving to synthesis of mechanical clocks and watches, a Function-Behavior-Structure model (Umeda et al. 1990) was used as the basis of the generative representation (Starling and Shea 2002). A parallel grammar has been developed (fig 5) that more explicitly distinguishes between a function grammar and a structure grammar, compared to the previously presented structural grammars.

A production system is required for both parts of the parallel representation to both create valid designs and perturb them. A function representation holds information in a directed graph about the power flow and functional connectivity of the modeled design components (spindle, gear, escapement, power source, base plate). Function rules that add and remove labeled nodes in a graph are used to build valid watch and clock designs (fig 5). This differs from the structural grammars that start with a functionally valid design and perturb it. Detailed 3D form information based on the component type, e.g. dimensions and material, as well as constraints maintaining structural connectivity, i.e. gear joints, is held in the structure representation to ensure a valid structure. In parallel to applying the function rules, corresponding structure rules, called C-Rules or Create-Rules, are applied such that the two representations are always synchronized to ensure that the combined representations correspond to a valid design (Starling and Shea 2002).

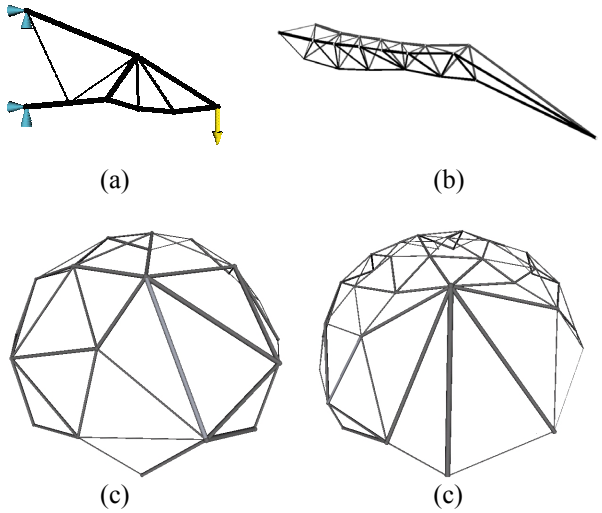

Fig 4: Three classes of truss structures generated: (a) planar, (b) 3D, (c) and (d) 2.5D truss structures

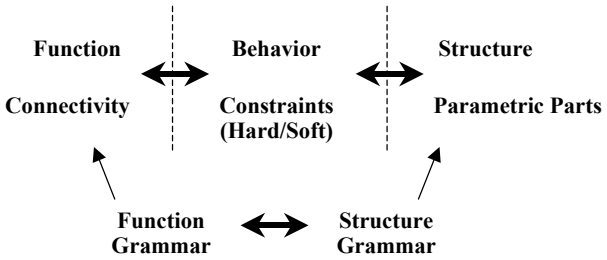

Fig 5: A parallel grammar based on a FBS model

A simple mechanical watch design is now presented. Figure 6b shows the function representation; figure 7b shows the structure representation. Thin tubes are the spindles, or axles, on which dark gray gear disks are mounted. The light grey disk is the power source, and the semi-transparent thin disks are the base plates for the whole assembly. While there is a 1:1 relation between a function graph and its corresponding structural

instantiation, complex spatial relations among parts are possible. For example, application of structure rules in Fig 7a illustrates three different valid structure representations for the same function graph, i.e. the use of two concentric spindles versus two spindles spread apart. The higher possibility for complex interactions among parts make this formal split in representations necessary to keep the design representation as simple, understandable and flexible as possible. Allowing this generality of spatial interaction is aimed at enabling function sharing by individual components in future work.

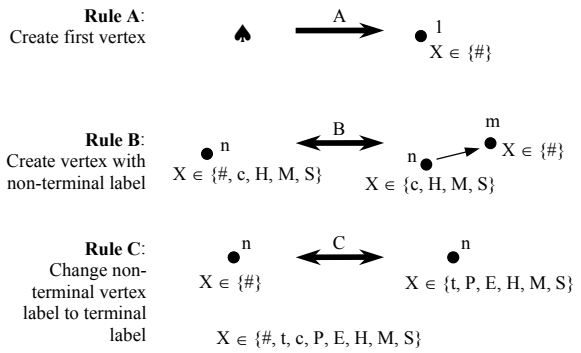

Fig 6a: Function rules

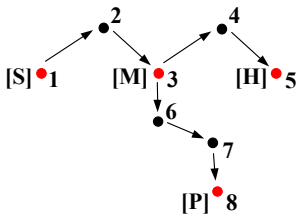

Fig 6b: A sample function graph for a wristwatch. S represents the second hand spindle, M represents the minute hand spindle, H represents the hour hand spindle and P represents the power source. Edges are gear pairs.

Starling and Shea (2003) describe a second class of rules, termed P-Rules, or Perturb-Rules, that, while changing the structure of the design, maintain its overall function and adhere to spatial constraints that govern valid behavior, e.g. maintaining the required gear ratios between particular spindles in the design. C-Rules add, or, if applied in reverse, remove from a design while P-Rules change parameters of existing components in a design. The production rules shown are tailored towards their design domain, however, the emphasis is still on generality. Work is underway to apply the same parallel grammar to the generation of a winding and shutter mechanism in a camera.

The initial design shown in figure 7b was generated using information from the function graph in figure 6b and gear ratio data specified by the user. The structure representation is then created using a generate-and-test algorithm, applying C-Rules and P-Rules as necessary to create a valid structure. Verification of the designs generated by the grammar was carried out by inspection of a virtual prototype clock, as opposed to the watch shown here, and comparison with an existing clock design (Starling and Shea 2002).

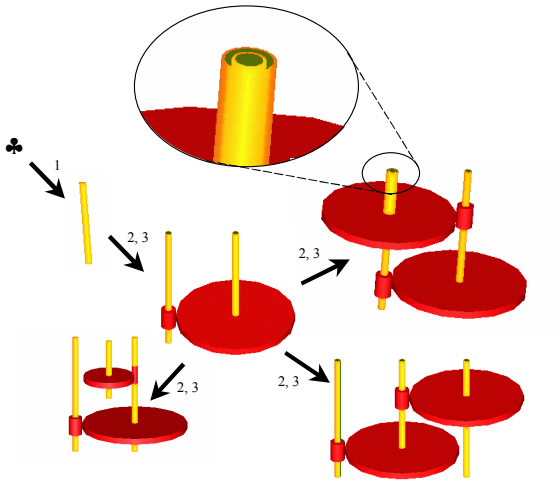

Fig 7a: Structure rule applications

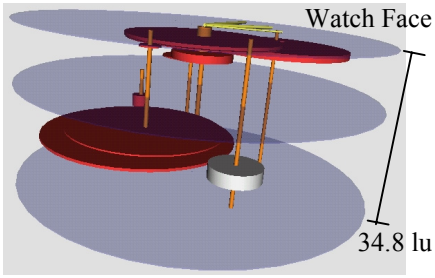

Fig 7b: A structural representation for the function graph in Fig 6b.

This parallel grammar can be unified with the previous structural grammars (fig 2) by considering topology rules as 'function' rules and shape and sizing rules that act on the parametrics of a structural design as 'structure' rules, more specifically P-Rules. In the structural grammar the C-Rules are part of the topology grammar itself. For example, Rule 1 in fig 2 adds one truss member to the design and sets its requisite parameters. So far, there has not been a need to make a formal separation between the function and structure aspects of the rules due to the simplicity of the function in the designs. However, moving to the synthesis of mixed-mode structures, e.g. generating tents that require both compression beams and tensile fabric components, may require a parallel grammar. Also in structures but working in architectural design, Li (1999) has used a parallel representation to solve an architectural design problem where the detailed form design relies on knowledge of the eventual overall generic, functional, design.

Another primary difference between the structural and mechanical grammars is the nature of the spatial connectivity constraints. Spatial connectivity of a truss

structure is maintained implicitly by the function graph (structural topology) that also contains 3D spatial information, i.e. if a node in the graph is moved then all edges (structural members) move as well. Since in the parallel grammar the function graph does not contain explicit spatial knowledge, spatial connectivity constraints are needed in the structure rules to ensure that gears that are assumed to transfer motion in the function graph can fulfill this requirement in the structure representation. Again, making these distinctions in the synthesis representation for simple examples is aimed at moving to functionally more complicated designs in the future.

Evaluate

If the generative mechanism does not explicitly contain enough knowledge necessary to compare designs and the design language described is sufficiently large to require automated search techniques (next section), additional evaluation models are needed to interpret and assess the performance of designs within the language described. The exact nature of evaluation models depends on the domain of the synthesis tool and can include both behavioral evaluation, e.g. structural analysis, as well as evaluation of further important design criteria, both quantitative and qualitative. Developing effective models of evaluation so that a computer algorithm can reason effectively about the relative merits of designs within a design language is a difficult task.

Both the structural and mechanical grammars implemented are fully parametric. Unconstrained they are capable of producing infinite languages of designs. In the structural grammars, while some design constraints are placed within the grammar itself to restrict design generation most evaluation, or performance assessment, takes place outside of the generative mechanism. Structural performance has been modeled in terms of many competing factors including structural efficiency, economy, usage and aesthetics (Shea and Cagan 1999). For example, the domes generated in figs 4c and 4d illustrate a tradeoff between minimum mass, maximum enclosed space, minimum surface area and uniformity of member length. A different tradeoff can be seen by allowing a maximum of 50 (fig 4c) or 100 (fig 4d) members to be generated for the same design scenario.

By embedding as little design specific knowledge in the grammar itself, the evaluation module can be used with multiple structural grammars for different classes. As an aim of the synthesis framework is to extend capabilities of current analysis tools, where possible integration of external behavioral analysis is desired. The structural grammars use integrated finite element analysis software, FElt, for behavioral evaluation that gives flexibility in the types of analysis that can be requested as well as the type of structural model used, e.g. truss or frame. The work in mechanical systems is experimenting with the integration of Dymola, a dynamic simulation language, for analysis of a camera shutter mechanism during synthesis to evaluate efficiency via battery power consumption.

The mechanical grammar shown contains more design knowledge in the generative mechanism itself in the form of constraints to restrict the design language to only contain designs that meet behavioral constraints. Nevertheless, valid designs that meet behavioral constraints are not necessarily useful: evaluation of designs within the language is still needed to compare design performance according to further criteria, for example cost, mass and compactness.

Mediate

The fourth phase of the framework, mediate, aims to provide mechanisms for reasoning about the evaluated design language, so as to explore, search, compare and choose among alternatives. Reasoning mechanisms can include optimization and search, case-based reasoning and machine learning. In limited cases where the design language is small, for instance less than 20 designs, it may be possible to search the evaluated language by hand. The appropriate search mechanism is dependent on the design language, nature of the evaluation metrics and intended impact of the resulting tool on the design process. For instance, stochastic processes are often combined with generative representations that are less knowledge intensive while deterministic search processes combine better with more knowledge-based representations. Examples using simulated annealing optimization and simple search to mediate among designs will be given.

Structural topology and shape annealing (STSA). STSA is one instantiation of the parametric synthesis framework. It is a generate-and-test type method that combines the structural grammars presented with simulated annealing, a stochastic optimization method (Kirkpatrick et al. 1993), to produce optimally directed geometric designs (Cagan and Michell 1993). Adding structural evaluation to the existing geometric evaluation has resulted in STSA (Shea and Cagan 1997), a robust and practical method suited to exploring both conventional structures and performance-driven yet novel structural forms (fig 4). The result is the generation of complex geometric forms via a stochastic process that are not only functionally feasible but also reflect further design goals.

Integrating automatic finite element analysis requires attention to avoid lengthy analysis times that can result in a synthesis tool with no potential for designer interaction. There is a tradeoff between the model complexity required to produce realistic designs and analysis time necessary. A recent application to full-scale transmission towers for an energy company illustrated a strong potential for its use in the re-design of current towers (fig 8) to reduce life-cycle costs (Shea and Smith 1999). This was only possibly by determining the number of dominant load cases, which was seven, and checking the remaining 11 load cases after design generation. If the design was generated for only the primary load case, large changes in the member sizes to meet the full set of load cases were needed.

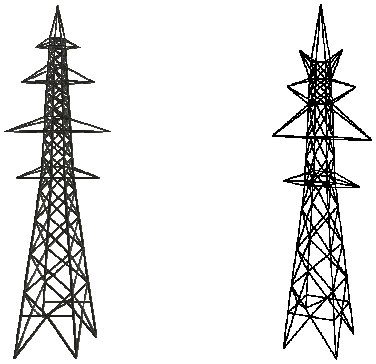

Fig 8: Current (left) and novel (right) transmission tower designs generated by STSA

This case study was also a test of the capacity of the method for re-design of a highly constrained design and scalability. The initial design contained 322 members and a maximum of 600 elements were allowed. The new design generated has 16 fewer joints and 80 fewer members with comparable structural performance. In highly constrained situations where the design performance space is non-convex, using soft constraints within stochastic search produces designs that are very difficult or sometimes impossible to achieve using hard constraints alone.

Generating thin watches. Turning back to the mechanical grammar, for a given function graph (fig 6b) a large number of structural embodiments can be produced all of which satisfy behavioral constraints (fig 7b). The initial design shown in figure 7b was generated using gear ratio data specified by the user. Better solutions that meet further design criteria can be found within the design language using direct search. For instance, a thin mechanism is desirable for a wristwatch so that it can be worn comfortably.

To find a suitable solution to this design problem, the main evaluation function is the overall thickness of the design, however, further criteria must also be taken into account. This is because reducing overall thickness alone will not drive some of the changes necessary. For example, the existence of thick gear disks in the design might be blocking the application of the P-Rules that can move base plates closer together. Hence shrinking the size of gear disks does not directly influence the objective function but is a vital indirect component of the process required to find preferred thin designs.

Figure 9 shows the thinnest design generated, measured as the smallest distance between outer base plates, from 20 experiments. This was achieved by minimising mass as a secondary design objective, where the gear disks were taken as brass and the base plates and spindles as steel. The overall thickness was reduced from 34.8 (fig 7b) to 12.7 length units (fig 9).

Prototyping. An important step in the process of creating performance-based synthesis tools is to take novel and conventional generated designs from the mediation phase and compare them to the initial investigation. This provides theoretical validation of the synthesis tool, showing that it is capable of producing designs similar to those currently available. The structural grammars have been theoretically validated versus structural optimization benchmarks (Shea and Cagan 1997) whereas the mechanical grammar was shown capable of generating the mechanism of an existing alarm clock (Starling and Shea 2002).

For innovative designs generated, physical prototyping is essential to their understanding. To highlight this point, the first 1:1 physical prototype of a structure generated by STSA was built recently in Amsterdam (fig 10). The final design was a simply supported canopy connecting the four corners of a courtyard and spans a maximum space of 12.85 m by 23.3 m while wrapping around a tree. It consisted of 132 unique structural member lengths, only two distinct section sizes, 56 different joints and 74 unique panels. While the design is not a dome structure, there exists a relation between the structural patterns produced and geodesic patterns that formed the basis for the structural grammar (Shea and Cagan 1997).

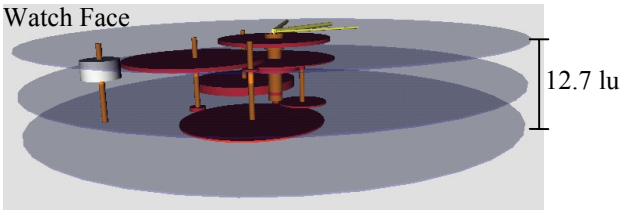

Fig 9: Structure Representation of preferred thin watch design (best of 20)

It is believed to be the first architectural structure built where both the form and related structure were generated by a computer via design parameters and conditions rather than explicitly describing geometry and structure. In the domain of structural and architectural design, physical prototyping is proving to be essential to the understanding and interest in using synthesis tools for designing. It is necessary so that novel designs can be experienced giving much insight into the synthesis method and performance models that produced them.

Conclusions

This paper has presented a general framework for creating performance-based synthesis tools. A key element of the framework is the use of simple, fundamental and flexible grammatical production systems that provide efficient representations of vast design languages using a bottom-up composition model for synthesizing designs from the component level. Research in creating structural grammars has shown great possibility for spatial innovation and was used as a basis for producing a parallel grammar for synthesis of mechanical systems based on an FBS model. In both domains integration of behavior analysis tools is proving essential to the usefulness of the designs

generated. Research issues that must be addressed for these methods to become powerful and useful tools for designers include further practical constraints and evaluation metrics as well as more potential for designer interaction.

Fig 10: Free-form design generated using STSA and full-scale prototype construction

Acknowledgements

Current research support is provided by the EPSRC (UK) and a Philip Leverhulme Prize through the Leverhulme Trust (UK).

References

Antonsson, EK and Cagan, J (2001), *Formal Engineering Design Synthesis*, Cambridge University Press, Cambridge.

Bendsøe, MP (1995), *Optimization of Structural Topology, Shape and Material*, Springer-Verlag, Berlin.

Bojczuk, D and Mroz, Z (1999), "Optimal topology and configuration design of trusses with stress and buckling constraints", *Structural Optimization*, **17**, pp. 25-35.

Cagan, J and Michell, WJ (1993), "Optimally Directed Shape Generation by Shape Annealing", *Environment and Planning B*, **20**, pp. 5-12.

Caldas, L (2002), "Evolving Three-Dimensional Architecture Form", *Artificial Intelligence in Design '02*, Cambridge, United Kingdom, pp. 351-370.

Chakrabarti, A (2002), *Engineering Design Synthesis*, Springer-Verlag, London.

Duarte, JP (Forthcoming), "Customizing Mass Housing: The grammar of Alvaro Siza's Houses at Malagueira", *Environment and Planning B*.

Heisserman, J and Woodbury, R (1994), "Geometric Design With Boundary Solid Grammars", *Formal Design Methods for CAD (B-18)*, pp. 85-105.

Kirkpatrick, S, Gelatt Jr, CD and Vecchi, MP (1993), "Optimization by Simulated Annealing", *Science*, **220**(4598), pp. 671-679.

Li, AI-K (1999), "Expressing parametric dependence in shape grammars, with an example from traditional chinese architecture", *Association of Computer Aided Architectural Design Research in Asia (CAADRIA)*, Shanghai, pp. 265-274.

Li, X, Schmidt, L, He, W, Li, L and Qian, Y (2001), "Transformation of an EGT Grammar: New Grammar, New Designs", *ASME Design Technical Conference*, Pittsburgh, Pennsylvania.

Lipson, H and Pollack, JB (2000), "Evolution of Physical Machines: Towards Escape Velocity", *6th International Conference on Artificial Intelligence in Design, AID'00*, Worcester Polytechnic Institute, Worcester, Massachusetts, USA, pp. 269-285.

Michell, AGM (1904), "The Limits of Economy of Material in Frame-Structures", *Philosophical Magazine*, **8**(47), pp. 589-597.

Nooshin, H and Disney, P (2000), "Formex configuration processing I", *Int. J. Space Structures*, **15**(1), pp. 1-52.

Post, E (1943), "Formal reductions of the general combinatorial decision problems", *American Journal of Mathematics*, **65**, pp. 197-268.

Schodek, DL (2001), *Structures*, 4th ed., Prentice Hall.

Shea, K and Cagan, J (1997), "Innovative dome design: Applying geodesic patterns with shape annealing", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **11**, pp. 379-394.

Shea, K and Cagan, J (1999), "Languages and Semantics of Grammatical Discrete Structures", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Special issue on Generative Systems in Design*, **13**(4), pp. 241-251.

Shea, K and Smith, IFC (1999), "Applying shape annealing to full-scale transmission tower re-design", *Proceedings of DETC99: 1999 ASME Design Engineering Technical Conferences*, Las Vegas, NV.

Simon, HA (1973), "The Structure of Ill-Structured Problems", *Artificial Intelligence*, **4**, pp. 181-201.

Starling, AC and Shea, K (2002), "A Clock Grammar: The Use of a Parallel Grammar in Performance-Based Mechanical Design Synthesis", *2002 ASME International Design Engineering Technical Conferences*, Montreal, Canada.

Starling, AC and Shea, K (2003), "A Grammatical Approach to Computational Generation of Mechanical Clock Designs", *International Conference on Engineering Design, ICED'03*, Stockholm, Sweden.

Stiny, G (1977), "Ice-ray: a note on Chinese lattice designs", *Environment and Planning B: Planning and Design*, **4**, pp. 89-98.

Stiny, G (1980), "Introduction to Shape and Shape Grammars", *Environment and Planning B*, **7**, pp. 343-351.

Umeda, Y, Tomiyama, T and Yoshikawa, H (1990), "Function, behaviour, and structure", *Applications of Artificial Intelligence in Engineering V: Design*, Berlin, pp. 195-211.