Modelling with forces: grammar-based graphic statics for diverse architectural structures

Juney Lee, Corentin Fivet, Caitlin Mueller, *Massachusetts Institute of Technology, Cambridge, United States of America.*

ABSTRACT

Most architectural modelling software provides the user with geometric freedom in absence of performance, while most engineering software mandates pre-determined forms before it can perform any numerical analysis. This trial-and-error process is not only time intensive, but it also hinders free exploration beyond standard designs. This paper proposes a new structural design methodology that integrates the generative (architectural) and the analytical (engineering) procedures into a simultaneous design process, by combining shape grammars and graphic statics. Design tests presented will demonstrate the applicability of this new methodology to various engineering design problems, and demonstrate how the user can explore diverse and unexpected structural alternatives to conventional solutions.

KEYWORD

Modelling and Design of Processes

1. INTRODUCTION

The institutionalized separation of form (architecture), forces (structure), and material (fabrication/construction) has resulted in a geometric-driven contemporary design culture. During early stages of design, an architect tries to control spaces by "finding a form" among countless possible forms, while an engineer tries to control forces by "form-finding" an optimized solution of that particular form. This modern prioritization of expressive form over material and performance is the platform upon which architecture and structural engineering remain divided as schools in academia and as professions in practice today. The development and logic of most Computer-Aided Design (CAD) software are also based on this separation of form and forces. While the rapidly advancing capabilities of computational tools have enabled architects to digitally model almost any form, and engineers to analyse any structure, digital models generated by architects typically must be re-modelled by engineers in a file format that is compatible with numerical analysis software. More meaningful investment of the computational resources that are available today may be in investigating new structural possibilities, rather than developing better ways of analysing what may be inherently bad forms.

There is a need for computational design tools that can not only generate forms, but simultaneously process structural logic and rules, so that the outcome does not need to be constantly remodelled and checked with numerical analysis software. For architects, introducing real-time performance feedback during the modelling process can help in exploring structurally feasible and diverse designs much more efficiently, and improve the general structural intuition of designers. For engineers, adding generative ability to tools that are traditionally meant to only analyse an already defined structure, can potentially expand the creative capacity of the engineer and lead to undiscovered structural possibilities. In order to explore a wide range of diverse design alternatives, computational power combined with controlled randomness can help the designer in unbiasedly exploring alternative solutions that are unexpected, visually interesting and yet performatively adequate.

2. BACKGROUND

In conventional parametric modelling paradigm, forms are generated and controlled by parameters or variables. Similarly, in computational design and optimization of structures, the objective function is mathematically formulated and numerical parameters are clearly defined. This means that the design space contains all possible solutions to a given problem. While a sophisticated optimization algorithm may help the user in finding the optimal solution within this parameter-based design space, the result is still limited by the design space itself. During earlier stages of design, a parameter-based design space does not contain the wide range of design possibilities that the designer may want to consider (Mueller 2014, pp.79-88).

2.1. Grammar-based design

In order to broaden the design space, a grammar-based approach can be used in place of the conventional parameter-based design paradigm. Grammar-based design, or more commonly known as *shape grammars*, is a set of allowable shape transformations that can be used to define a design language, through which form generations can be automated based on a desired logic, style or objective (Stiny and Gips 1972). It has been used frequently in architectural context to not only analyse design styles and languages, but also generate new ones. William Mitchell hinted at its potential applicability to other fields such as structural design, by incorporating functional attributes and structural criteria to grammar rules in the form of *functional grammars* (Mitchell 1991).

2.2. Structural grammars

Shape grammars have been applied in engineering, most notably by Shea and Cagan, as a method called *shape annealing* (Shea and Cagan 1999). Because this method uses random shape transformations which are entirely geometric, a numerical analysis is required after every operation. Also, the transformations are guided by a stochastic optimization algorithm, which means that unless the transformation improves the overall performance, it will keep iterating until one is found. While successful in generating unexpected solutions, shape annealing is an optimization driven method and seeks to output one solution. Shape annealing is ultimately resource-

intensive, and the resulting diversity is limited. Alternatively, shape grammars can be used to explore trans-typology structures, by randomly mix-and-matching elements of different typologies (Mueller 2014, pp.79-101). A wide range of unexpected yet structurally equilibrated solutions can be found using a small set of grammar rules. However, previous research in this area was limited to typologies that are specific mainly to bridge structures only.

2.3. Graphic statics

Graphic statics is a graphical method of calculating forces for discrete structures under axial loads (Culmann 1864). It is based on construction of two reciprocal diagrams (Cremona 1890): the form diagram representing the actual geometry of the structure, and the force diagram that represents the internal forces as vectors. Because forces are graphically represented, no numerical analysis is required to calculate the forces. When combined with modern day computation, graphic statics can become a powerful design tool by automating the drawing process, and enabling real-time interaction between the reciprocal diagrams. Most notable examples include Active Statics (Greenwold and Allen 2009), eQuilibrium (Van Mele et al. 2014), and Constraint-based Graphic Statics (Fivet and Zastavni 2013).

2.4. Combining grammars and graphic statics

Shape grammars and graphic statics have been explored previously in the field of creative structural design, but never in combination. When shape grammars and graphic statics are combined, several key benefits emerge. First, geometric rules can have direct relationship with corresponding force diagrams so that any geometric transformation results in equilibrium. Because local and global equilibriums are always guaranteed, randomness can be introduced during the generation process to increase diversity of solutions. Second, because force diagrams are constructed for every transformation, there is no need for further numerical analysis. Lastly, the rules have no boundary-specific parameters, which enables the methodology to be applied to a wide range of design problems. By harnessing the intelligent, generative power of shape grammars, and the computational graphic statics that can transform forces into equilibrated forms, architecture and structure can be integrated more seamlessly during conceptual design.

3. SETUP

3.1. Conceptual overview

The proposed methodology automatically and randomly generates designs by iteratively applying a series of rules. The conceptual overview of the computational setup is illustrated in Fig. 1. Generally, the *grammar engine* is responsible for choosing rules, deciding where to apply them and updating the geometry. *Graphical computation engine* functions as the structural blueprint for the all procedures to be performed by the grammar engine.

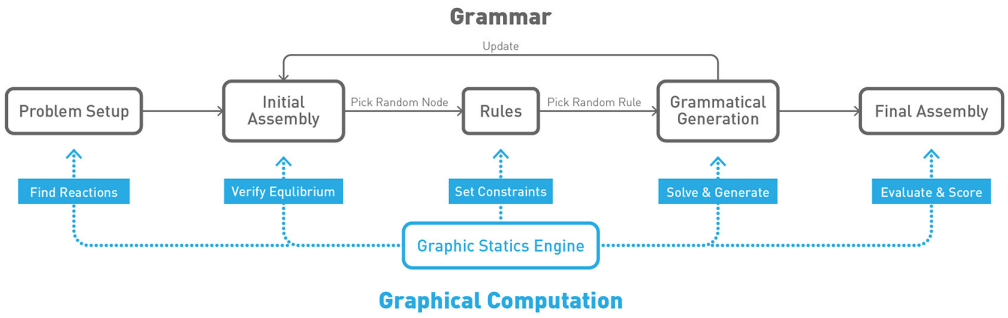


Figure 1: Conceptual overview of integrating shape grammars and graphic statics.

3.2. Elements

The proposed methodology operates on three types of computational classes: 1) a *Force* class that is a 2D vector, with a type parameter, direction and amplitude; 2) a *Node* class that includes a coordinate, state parameter (active or not-active), type parameter, and a list of forces; 3) a *Member* class that is a line, with information about its internal forces; and 4) an Assembly class that includes a list of *Node* classes, list of members, the overall system state, and other information about the entire structure. The four elements are summarized in Fig. 2.

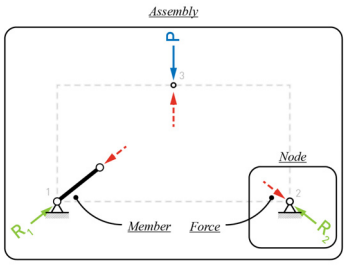


Figure 2: Four basic elements of the methodology.

3.3. Rules

The eight rules used for generating designs in this paper, are summarized in Fig. 3. All parameters incorporate structural logic and knowledge, and are always verified by construction of force diagrams. While the parameters are randomly determined, it is constrained by user-defined lower and upper bounds.

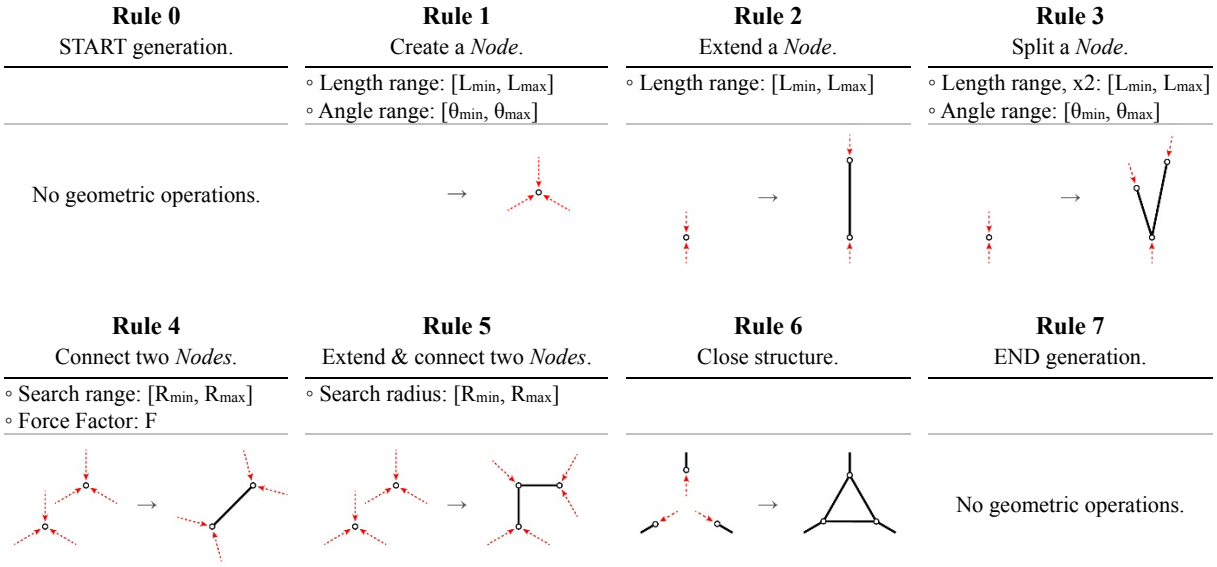


Figure 3: Summary of rules and parameters.

3.4. Constraints

Generative grammars can be a powerful tool in discovering new structural forms. However, without intelligent constraints, the rules may be too broad and generate forms that have limited practical feasibility. In addition, the grammar rules can potentially be applied recursively, or repeated without an end. The following strategies are used to set constraints.

- *Reasonable range angles for the initial reactions.*
- *Setting reasonable local bounds, such as minimum and maximum angles or lengths.*
- *Global termination conditions, such as generation count and recursion control mechanisms.*

3.5. Generation algorithm

Fig. 4 summarizes the automatic random generation algorithm framework. Steps 1a, 1b and 1c are where user input is required to setup the design problem. Then, the initial *Assembly* is constructed. From the parameters defined in steps 1b and 1c, the algorithm randomly chooses a *Node* to apply a random rule. Steps 3 through 8 are repeated until the system reaches a terminating condition defined by the user.

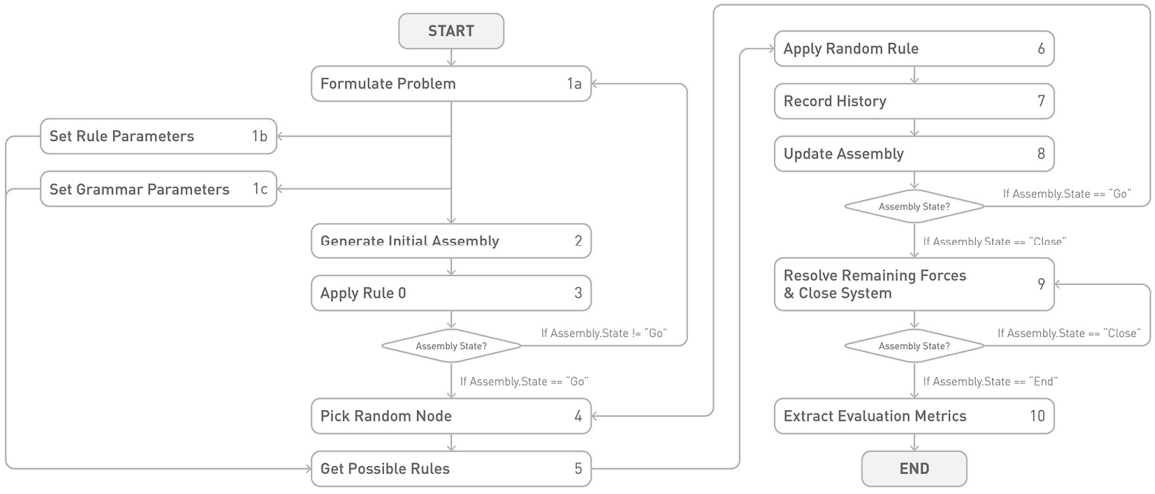


Figure 4: Framework of the automatic random generation algorithm.

3.6. Sample generation

Fig. 5 shows a step-by-step generation sequence of one possible design for a simple problem.

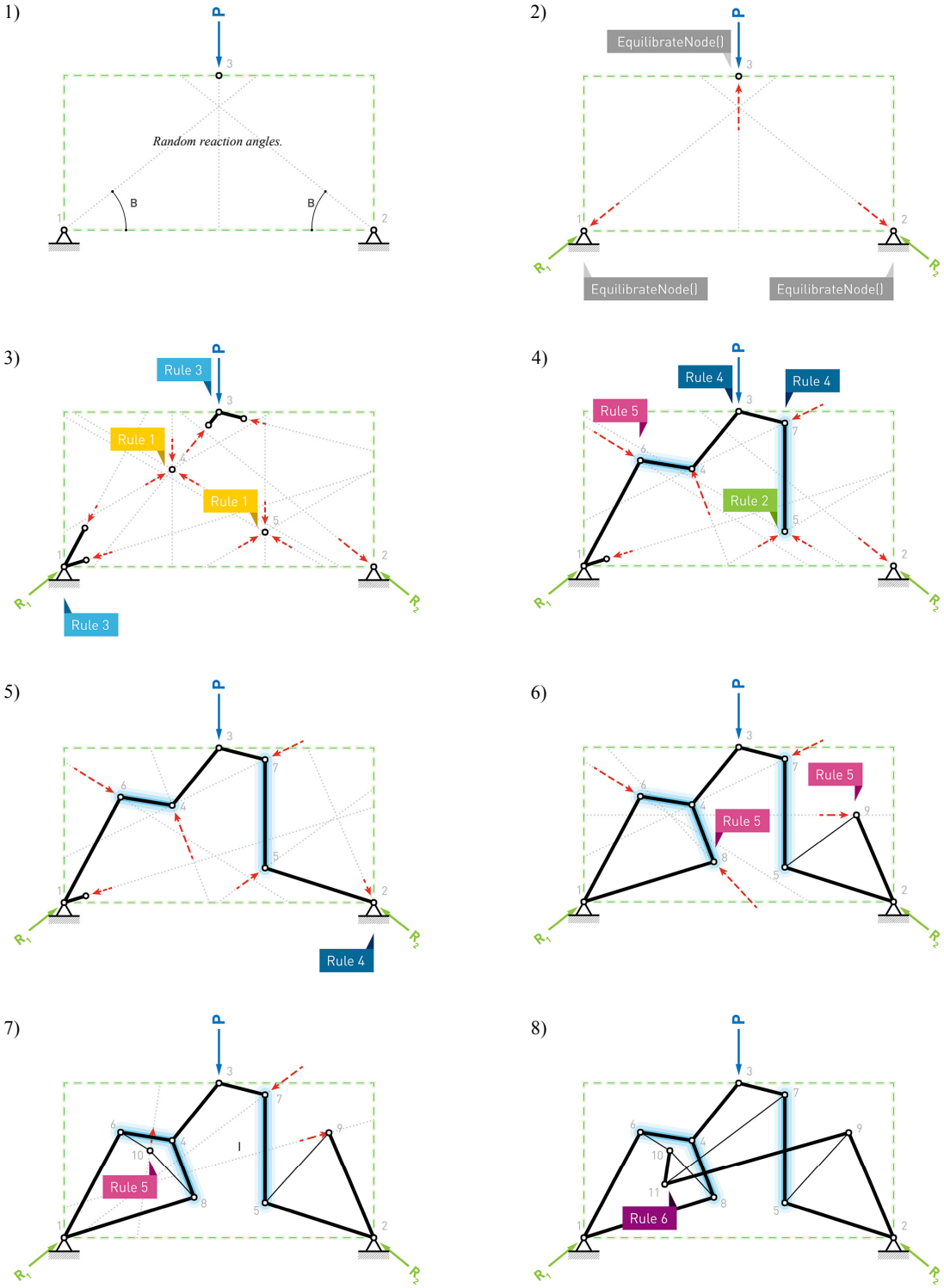
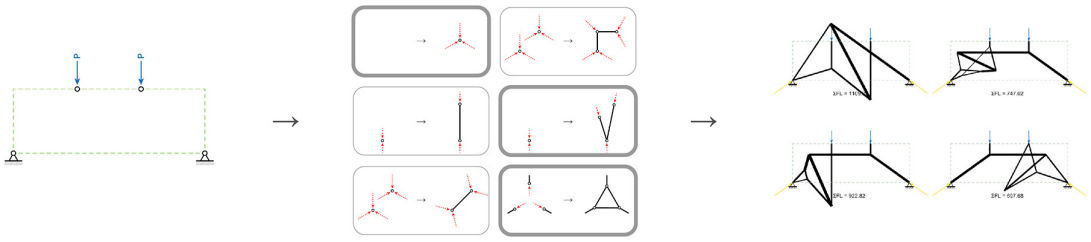


Figure 5: An example of an automatic random generation sequence.

5. DESIGN TESTS

5.1. Workflow

Unlike most conventional engineering tools, this methodology begins without a starting geometry. First, the user sets up the problem by defining the applied loads, reactions and the locations of those forces and supports (Fig. 6a). The user then chooses the rules to apply, and weights for each rule which defines how likely it is that a rule will be randomly selected to be applied (Fig. 6b). Finally, the user defines how many options to produce. Results can be further diversified by modifying the following global parameters: 1) minimum number of rule applications for each generation; 2) rule sensitivity towards the beginning, the middle or the end of the generation cycle; 3) termination conditions; and 4) random seed (Fig. 6c).



a) **Step 1**: Setup problem, create initial Assembly.

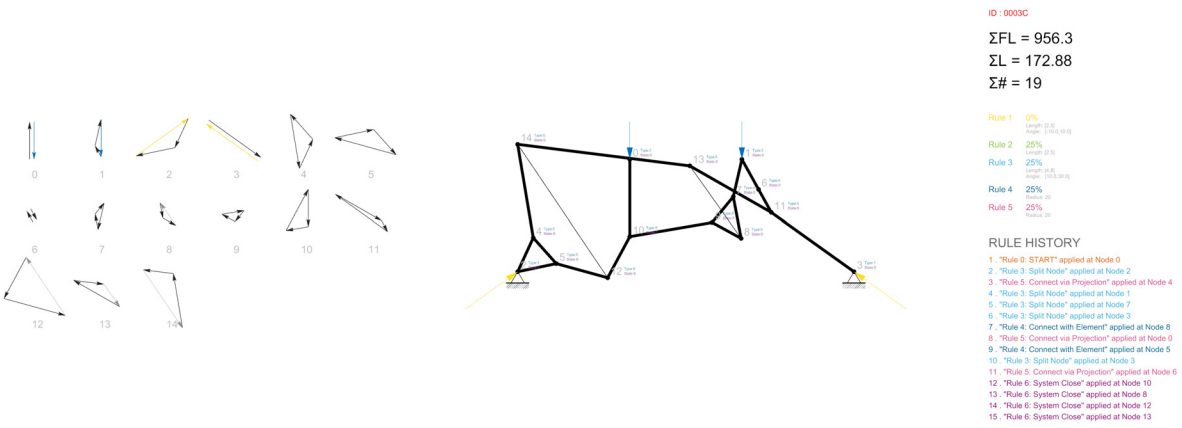
b) **Step 2**: Pick rules to apply, and set rule weights.

c) **Step 3**: Generate iterations, and sort based on a desired performance metric.

Figure 6: User workflow of the proposed methodology.

5.2. User interface

The proposed methodology was implemented as an interactive tool, using IronPython, Rhinoceros and Grasshopper. Each iteration instantaneously generates: 1) corresponding force diagrams for every node (Fig. 7a), 2) a form diagram with clear labels (Fig. 7b), and 3) rule history, information and evaluation metrics for the current solution (Fig. 7c). Visual representation of the forces, the evaluation metric, and the rule history which summarizes how the structure was derived, enables clearer understanding of the structure and informs better design decisions more quickly. The rule history, which records all the parameters that were used to generate the current iteration, is an important feature that enables reproducibility of the same iteration during later stages in design, when more information about the boundary conditions and the project in general may be available. This blueprint can also be used to develop more detailed versions of the design, and allow creative breeding using genetic algorithms.



a) b) c)

Figure 7: Screenshot of an example problem in Rhinoceros.

5.3. Results

Figs. 8-13 show the application of the tool on six different design tests. Designs shown are top eight performing solutions out of 80 iterations. All designs have performances that are approximately within 20% of the solution that can be derived using the fewest number of members, which is given a normalized score of 1.00 in each design test. When a simple, benchmark solution cannot be easily found as in the case of design test 6, the best performing solution out of the 8 designs is used as the benchmark. The performance is based on the total volume of structural material, or equivalently the total load path. Assuming constant internal stress at its optimal or final iteration state, the total volume or load path can be calculated by the simple formula: $\Sigma|F|\cdot L$, where F is the internal force of a member, and L is the length of that member (Baker et al. 2013). Using graphic statics, $\Sigma|F|\cdot L$ can be computed easily by multiplying the length of the member in the form diagram, and the length of the corresponding force vector which is provided by the force diagrams. The designs randomly produced through grammatical exploration exhibit significant diversity, which may often be desirable even at the sacrifice of a small amount of efficiency.

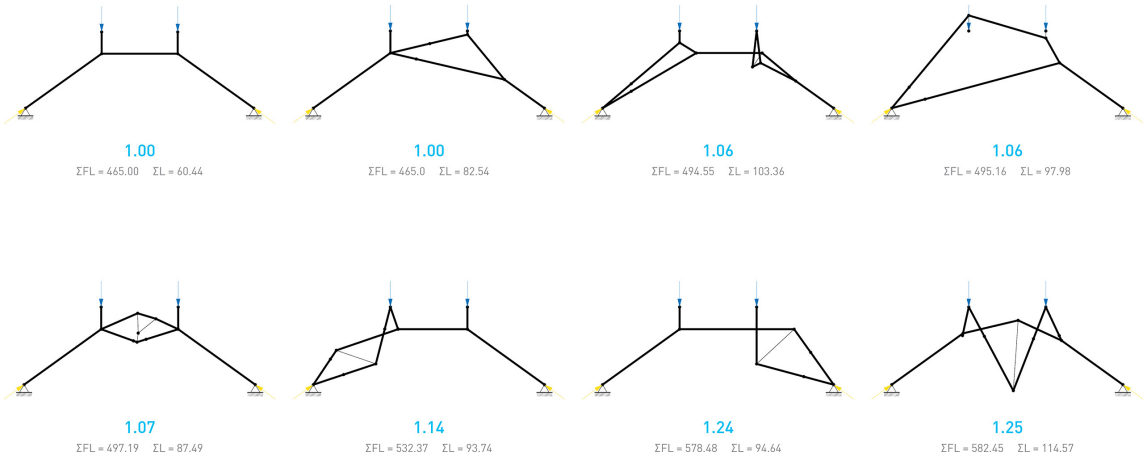


Figure 8: Design test 1, a span structure.

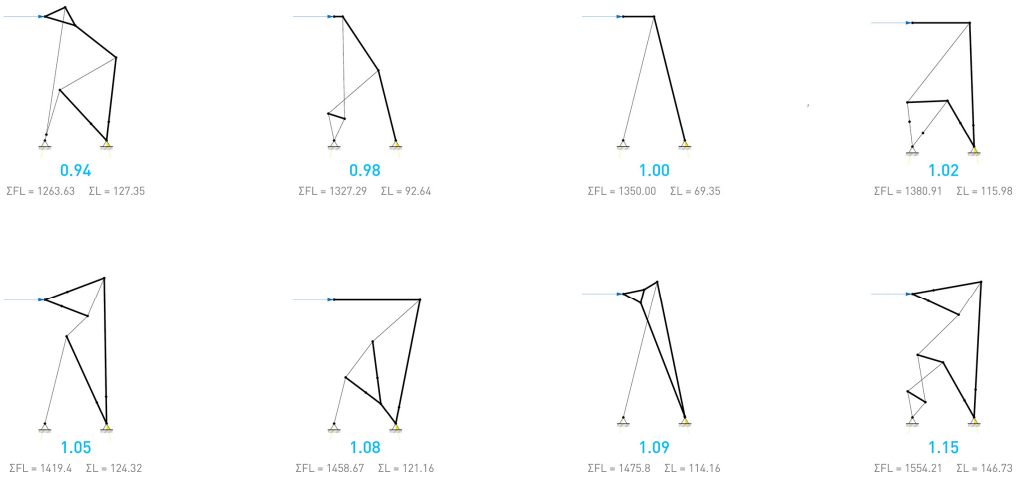


Figure 9: Design test 2, vertically cantilevered structure.

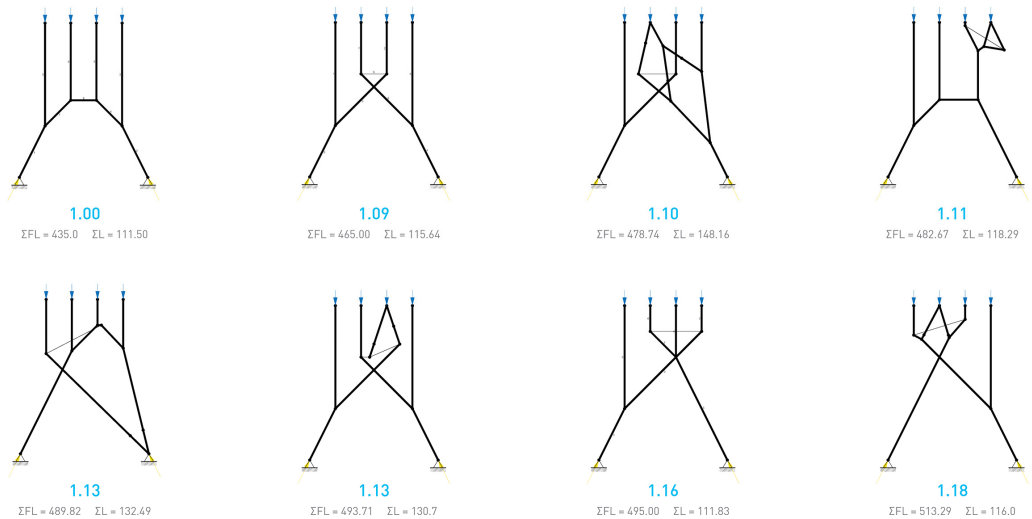


Figure 10: Design test 3, a wall-like compressive structure.

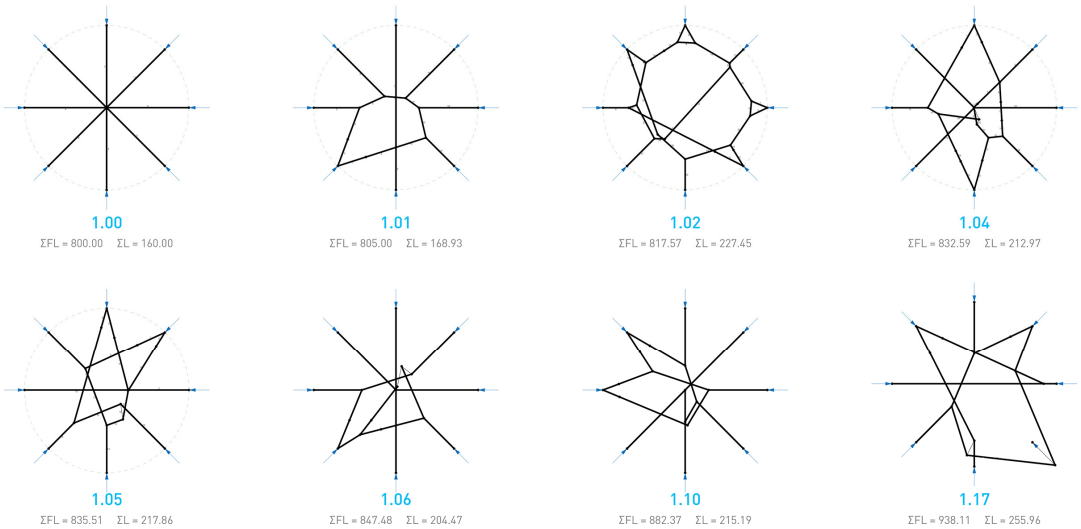


Figure 11: Design test 4, radial compression structure.

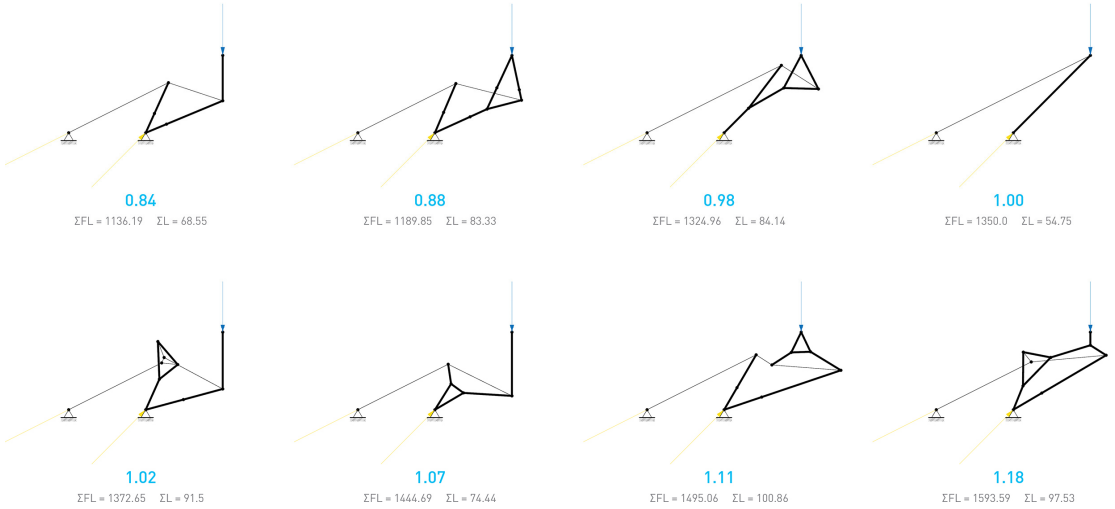


Figure 12: Design test 5, horizontally cantilevered structure.

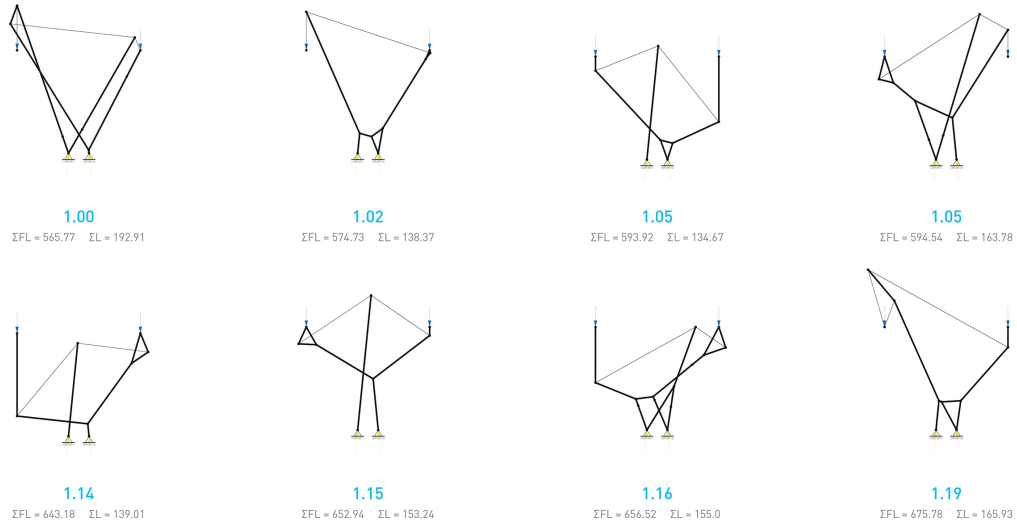


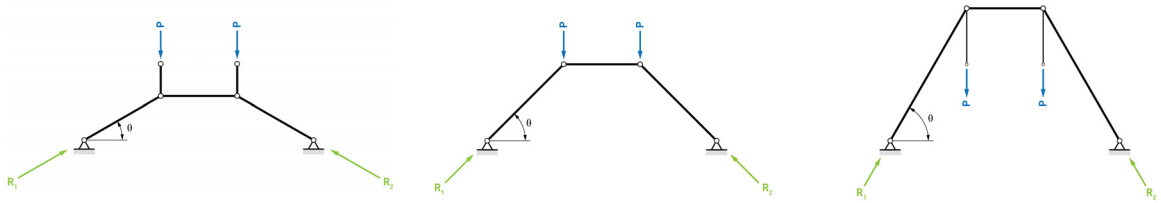
Figure 13: Design test 6, horizontally cantilevered structure in two directions.

5.4. Exploration of parameters

Several global and local parameters can be modified to explore design alternatives, as well as tradeoffs between various constraints. Parameters described in this section relate specifically to the design test 1 in Fig. 8.

5.4.1. Global parameter 1: reaction angles

Because the support reaction vectors are determined before the automatic random generation begins, a variety of possible solutions with varying shapes and performances can be explored by setting a reasonable bound for this angle parameter, as illustrated in Fig. 14. Conversely, the reaction angles can be altered after a design has been chosen to improve the performance. This parameter is most closely related to the boundary conditions of the supports, and how much horizontal reaction a support can provide.



θ = 30°
ΣFL = 539.71

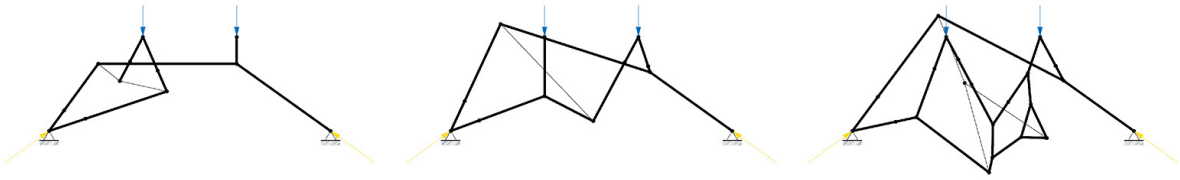
θ = 45°
ΣFL = 375.0

θ = 60°
ΣFL = 499.52.0

Figure 14: Changing the angle of the reactions not only changes the shape of the structure, but also its performance.

5.4.2. Global parameter 2: generation count

Generation count defines the maximum number of rules that can be applied in a generation. Fig. 15 shows three similar structures with varying generation counts. The generation count can be increased in cases where more redundancy may be required, or more geometric variation and expression within a design are desired.



Generation Count: 3
ΣFL = 603.51

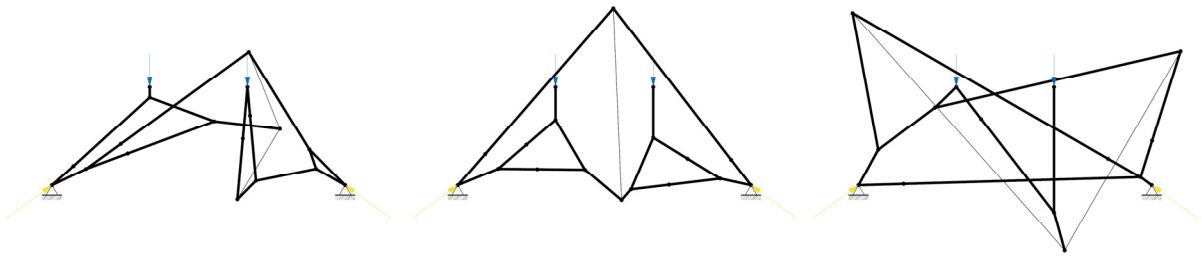
Generation Count: 6
ΣFL = 751.04

Generation Count: 9
ΣFL = 885.73

Figure 15: Changing the generation count controls the number of rules to be applied for each generation.

5.4.3. Rule parameters

Fig. 16 shows the effect that rule parameter variations can have on the results. The only parameter that was changed was the lower and upper bounds for the angle of the split rule. While increasing the range of possible angles does not necessarily improve the performance, the larger angles may be necessary for constructability of joints. Similarly, modifying the parameters for other rules will result in drastically diverse designs.



Angle range: 10°-20°
ΣFL = 603.51

Angle range: 20°-40°
ΣFL = 908.04

Angle range: 40°-60°
ΣFL = 1333.87

Figure 16: Effect of changing rule parameters on the results.

5.5. Practical applications

While the proposed tool allows fast exploration of design possibilities during conceptual design, the results will need to be interpreted by the architect and the engineer in order to develop the design with more detail and rigor during later stages of design. Fig. 17 illustrates how three results selected by the designer can be developed into realistic, yet significantly different and unexpected roof structures.

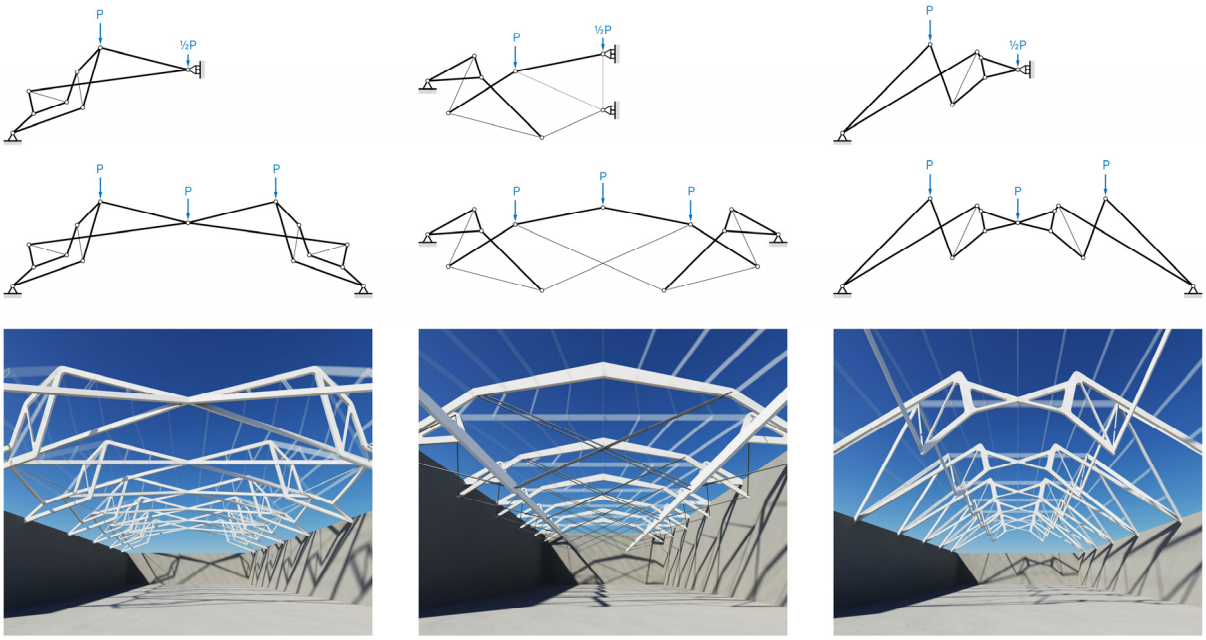


Figure 17: Sample designs with symmetry enforced, resulting in more practically applicable designs.

6. CONCLUSIONS

6.1. Contributions

This paper introduced a grammar-based design methodology, as an alternative to the conventional parametric design paradigm, which is limited in variety and often lead to expected solutions. The following specific contributions were presented.

6.1.1. More trial and less error

By incorporating forces during the form generation process, the resulting designs are guaranteed to be in equilibrium. Therefore, no further numerical analysis is required. Reduced coordination time between architects and engineers allows exploration of better and more interesting ideas faster. While most numerical analysis tools provide quick feedback on performance, they do not inform the designer with any guidance for improving the design. On the other hand, graphic statics instantly generates clear visualization of forces which enables the designers to get a clearer understanding of the structure's internal forces. As a result, the designer's intuition of the relationship between form and forces is improved, and better decisions will be made more quickly as the project progresses.

6.1.2. Unbiased exploration of diverse design alternatives

With automated generation by the computer, which is guided by the design goals input by the human designer, diverse solutions can be generated that simply would not be conceivable manually by a human designer with a pencil or a mouse. In addition, the automated generation of multiple designs at once not only increases the creative capacity of the designer, but also leads to new insights and better understanding of the design problem itself.

6.1.3. Generative graphic statics: beyond reciprocity

The reciprocal relationship between form and forces in graphic statics, means that one has to be created before the other can be drawn. Therefore, most computational graphic statics tools only work with pre-set problems, functioning mostly as an interactive analysis or visualization tool. By combining graphic statics with shape grammars, the form finding capabilities of graphic statics can be used to *generate* equilibrium structures. Most previous work done on shape grammars require a shape to pre-exist before a rule can be applied. However, the rules presented in this paper are based on *Nodes* or points, and are not dependent on any preceding shapes or conditions. Therefore, the methodology is flexible enough to be applied to a variety of design problems, and is able to generate structures without any prescribed typologies or preferences.

6.2. Future work

Although this paper was a successful first attempt of implementing this new methodology, there are several important directions for future work. First, global parameters could be improved to gain better control of overall generation process, including more intelligent ways in which the rules are chosen and where they are applied. Secondly, more detailed or material-specific constraints, buckling lengths, and minimizing number of overlapping members could be incorporated. Also, because all designs shown in this

paper are also statically equilibrated only for the defined load case, it will be important to develop a procedure to check for possible mechanisms and local instabilities. Lastly, while this paper focused on rules based on the form diagram, rules can also be developed for the force diagram (Akbarzadeh et al. 2014), which will further enrich the design possibilities.

6.3. Closing remark

Overall, this new methodology demonstrates the validity in combining and applying shape grammars and graphic statics together to various engineering design problems. The general versatility and customizability of the tool, and the speed at which it can generate unconventional and yet statically equilibrated structures, greatly improves possibilities for creative yet performance-focused explorations during early stages of conceptual structural design.

REFERENCES

Akbarzadeh M, Van Mele T, Block P (2014) Compression-only form finding through finite subdivision of the force polygon. In: Proceedings of the international association for shell and spatial structures symposium, Brasilia, Brazil, 15-19 September 2014

Allen E, Zalewski W (2009) Form and Forces: Designing Efficient, Expressive Structures. John Wiley and Sons, New York

Baker W, Beghini L, Arkadiusz M, Carrion J, Beghini A (2013) Maxwell's reciprocal diagrams and discrete Michell frames. In: Rozvany G (ed) Structural and multidisciplinary optimization, vol 48. Springer, Heidelberg, pp 267-277

Cremona L (1890) Graphical statics: two treatises on the graphical calculus and reciprocal figures in graphical statics. Claredon Press, Oxford

Culmann K (1864) Die graphische statik. Verlag Meyer & Zeller, Zurich

Fivet C, Zastavni D (2013) Constraint based graphic statics: new paradigm of computer-aided structural equilibrium design. Journal of the international association for shell and spatial structures 54(4):271-280

Greenwold S, Allen E (2003) Active statics. http://acg.media.mit.edu/people/simong/statics/Start.html. Accessed 2 March 2015

Lee, J (2015) Grammatical design with graphic statics: rule-based generation of diverse equilibrium structures. Master thesis, Massachusetts Institute of Technology

Mitchell W (1991) Functional grammars: an introduction. In: Goldman Zdepski (eds) proceedings of association for computer aided design in architecture: reality and virtual reality, Los Angeles, 1991

Mueller CT (2014) Computational exploration of the structural design space. Dissertation, Massachusetts Institute of Technology.

Shea K, Cagan J (1999) Languages and semantics of grammatical discrete structures. Artificial intelligence for engineering design, analysis and manufacturing 13:241-251

Stiny G, Gips J (1972) Shape grammars and generative specification of painting and sculpture. Information processing 71:1460-1465

Van Mele T, Block P, Ernst C, Ballo L (2014) eQUILIBRIUM: An interactive, graphic statics-based learning platform for structural design. http://block.arch.ethz.ch/equilibrium. Accessed 5 May 2015