Design innovation through constraint modeling
Axel Kilian
Design innovation through constraint modeling
Axel Kilian

In this paper we describe how constraint modeling can support design innovation. Furthermore, we lay out how constraints are employed in the construction and exploration of a model's design space. We place the approach within the context of design exploration using computational and conceptual representations of design. A review of the literature reveals that geometric, topologic, functional, and quantitative constraints are those most commonly used. For each constraint type, an example is presented drawing from several workshops and research conducted by the author. The examples range from product design, to structural design, to fabrication issues in freeform geometry. Based on the case studies, we describe how the different types of constraints can be used as design drivers and help in the exploration of solution spaces. In conclusion, we identify the need for bidirectional exercising of constraints as the next challenge in design exploration and discuss how it is relevant in particular for cross domain design.

Keywords: design exploration, constraint modeling, parametric modeling
1. Introduction

Constraints are generally viewed as limiting factors in design. But there is evidence in research [1, 2, 3] and architectural practice [4, 5] that constraints can trigger the development of innovative design solutions and are a powerful way to drive solution space. Constraints help to focus design exploration and to work within the boundaries of available resources. There are different types of constraints and they can be applied to a range of aspects of the design problem. We suggest that initially, a constraint may prove to be a limitation, but over the course of a design process it may evolve to become a driver for innovative design solutions to the problem. For design exploration, models that are not over-constrained are necessary. How, then, can a constraint be modeled in a favorable way for design exploration? The type of constraint is less important than how it is modeled. For design exploration, models that are not over-constrained are necessary.

To start, we can identify a number of core types of constraints and the models that can use them. In general, four types of constraints can be identified from the literature [2, 3] relevant to the examples presented here:

- Functional constraints: Requirements for what a design solution must accomplish functionally.
- Topologic constraints: Relationships between entities that form a topology.
- Geometric constraints: Geometric dimensions as well as relationships.
- Quantitative constraints: Quantifiable measures such as volume, thickness or length.

We give an example for each of the constraint types in research conducted by the author.

First, an example is presented where functional constraints are used as design drivers in a product design study looking at writing devices. The particular study was conducted in a concept car design studio headed by William J. Mitchell in 2004 at the MIT Media Lab. Second, an example of topologic constraints in a structural form-finding implementation is presented, based on force equilibrium using particles and spring systems. This model enforces axial forces in structural members for a given topology and it was developed in a workshop co-taught by John Ochsendorf, Axel Kilian, Barbara Cutler, Eric Demaine, Marty Demaine, and Simon Greenwald in 2004 at MIT. Third, an example of geometric constraints is presented. This research study was conducted by the author for enforcing developable surface properties for the fabrication of double-curved surface approximations. Finally, an example of quantitative constraints is presented. Two projects are shown, both based on searching the solution space using...
genetic algorithms and three dimensional constraint solvers to match parametric models to a target value. One deals with an eight-degrees-of-freedom articulated car design, the other with a family of vase designs.

Although isolated constraint types can be identified, real world design problems are never isolated instances of one constraint or another. However, for the scope of this paper, it is helpful to first look at individual constraint types and to suggest possible strategies in modeling them, before moving into how they might interact and overlap. The larger challenge is to implement robust constraint models that allow for linking different constraint types and develop computational models for digitally supporting design exploration.

Constraints in the context of a design exploration can become design drivers. A design driver is a prominent constraint in design exploration that is not easily changed and provides the strongest influence for directing the exploration. We claim in this paper that constraints can play a key role as design drivers in triggering design innovation.

The paper is structured as follows: First, we discuss the computational modeling of constraints and how these are identified in a design problem, how their analysis takes place, how constraints are translated into design drivers, how constraints are applied to a design domain, and how one may explore a solution space for a given set of constraints.

Second, the use of constraint modeling in design exploration is discussed. Different types of constraints are distinguished, bidirectionality is introduced, the translation between design representations, and the relationship between design domain and representation. Further we examine translation models, bidirectional models, the circular design dependencies, the role of geometry and the use of programming in design exploration.

2. Constraints in the context of computational modeling

Modeling constraints computationally requires both a robust theoretical model as well as a working digital implementation. We show some examples of successful implementations and several diagrammatic models. Solvers play a crucial role in constraint resolution and more specifically in the recent development of realistic cloth simulation in animations. The need for cloth simulation has pushed the development of robust and efficient solver techniques [4]. These solvers work well for problems that can be expressed as ordinary differential equations (ODE). However, not all design constraint problems translate into ODE’s. Constraints exist in many forms, and constraint solvers are developed domain specific. For instance in contrast to ODE based solvers many constraint solvers in artificial intelligence are based on unification algorithms, a cornerstone of automated theorem proving [2], a class of constraints not addressed in this paper. Therefore an
important first step is identifying the design problem and its constraints and find an appropriate constraint implementation. We suggest that a useful technique is mapping out the components involved and identifying adjacencies between them. The next step is to identify the constraints and their relationships in order to turn the analytical approach into one that leads to novel design solutions.

2.1. Identifying the design problem by mapping the domain

The definition of the design problem itself is a major step in setting up any constraint analysis. In order to understand the contributing factors and the cross dependencies a first step is to map the domain. An example for the mapping of a design domain is given in the diagram example below (Figure 1). The task is to identify the potential for new networking standards for the home networking market from analyzing the existing home infrastructure. Rather than using a matrix-based approach, the common devices in the home were positioned in relative networking proximity to each other to serve as a basis for identifying emerging network proximities. Proximity in the context of the diagram refers to both physical proximity of the devices in the home as well as proximity in terms of their being connected to the same or adjacent existing networks such as the power grid, an Ethernet or a wireless network for example. In addition emerging proximities in usage are taken into account, such as the increasing intermeshing of entertainment and computer equipment. Here the graphing of the design problem reveals
emerging properties on a visual level. The mapping of design-relevant relationships, features, and properties can reveal proximities between elements of the design problem that were previously overlooked. It is a first step to externalizing the design problem and visualizing it. The specific diagramming technique evolves with the design problem that is being pictured and is part of the design approach. This example (Figure 1) reflects the recording process of the gathered information in a spatial and visual manner in the course of many iterations throughout the gathering of information.

The accumulation of information on the design problem helps to form the design space and possible approaches for exploring it. Formatting the information in the diagram potentially reveals more than what was put into it by exposing patterns and tendencies on the visual level that would not have been apparent in the data alone. Diagramming the design problem is a form of evaluation and helps in formulating position for design additions. The example of mapping a domain demonstrates the first step in structuring a design problem and in identifying the constraints.

2.2. Analysis of the constraints in the design problem

The description of a design problem involves analyzing the problem for existing constraints and degrees of freedom. One possible approach to do so is to identify the commonalities between a sample of existing design solutions to the problem. The example shown in Figure 2 follows this approach for the task of a writing instrument.

Rather than choosing an approach based on the decomposition of components, a function-based approach was chosen. The function-based analysis allows for more flexibility in the description and a more general description across solutions with very different form factors. In contrast, a grammatical subdivision based on the parts of existing designs would limit the exploration to dimensional variations of existing designs, or at best combinatorial variations of parametric parts. In the example the examples range from a basic piece of chalk to a mechanized retractable pencil. We suggest going beyond part decomposition and instead abstract the higher level common functional description of the set of design solutions. The functional description then is the starting point for a component independent design implementation that expands the studied set of precedents.

The pen diagram (Figure 2) shows the different writing device functions coded by color. The first column shows the analysis of the existing devices, from a piece of chalk to a spray can. The analysis identifies key functions in the writing process across the example set. For instance, leaving a mark with a substance is a core function of all devices, at least the physical ones studied here. Given the function of leaving a mark with a substance and a target surface, a choice for delivering the substance is necessary. From the
example set there seems to be great variety in how the substance is deposited, calling for a deposit function as a separate node from delivery. Last, the handling question triggers a design response for packaging. The substitution of functions with implementation choices creates functional demands. The difference between a function and the functional demand is that once a function has been substituted with an implementation choice that choice triggers a functional demand for its neighbors. A match between functions and components is not guaranteed for all the devices, which is a key difference of a functional analysis to a component-based analysis. But the absence of a one-to-one mapping provides the looseness in the design goal description that allows for novel designs. In fact, an implementation choice might bundle several functions into a single component such as it is the case for the chalk piece that implements leaving a mark, delivery and handling all in one step. Such bundling is one approach to lead to innovative designs not seen in the sample set.

The second column describes the devices in a tree-like fashion with increasing subdivision and complexity of the parts. The part hierarchy is appropriate for a descriptive approach but less so when aiming for design innovation.

The third column does exactly that. It provides a functional description of the family of writing devices and is based on five core functions of the set of studied writing instruments. Those functions are chained together based on interdependencies into a so-called function chain. The function chain works much like a design checklist prompting design choices without suggesting explicitly any of the studied precedents. When using a function
chain its tokens are replaced with design choices one by one. For instance, the substance token may be replaced with ink. This choice triggers neighboring tokens to be implemented. Step by step, the abstract function chain is turned into a topology of functional demands. These demands form the basis for the next step, which is form implementation. Functional demands are translated into geometrically variable components. Based on their neighboring conditions, they have to conform to adjacent tokens both in dimensions and propagation of functions. When following through this process a new design instance is created.

The example is kept abstract. We emphasize the importance of the analysis of the design problem before setting up a constraint network in terms of parametric topologies or functions in order to allow for novel solutions. Parametric variations alone can only cover a very small spectrum of possible designs within dimensional and compositional variation of parts, rather than providing conceptual variety. These limitations of parametric descriptions are often neglected in light of the current interest in parametric modeling in design.

Diagrams help in the exploration of function chain constraints. An analysis leads to a formal description of a solution space. That formal description is translated into a function chain that can be used to generate new designs within the writing device family. The functional constraints define the functional demands which in turn give rise to the implementation constraints. Because they apply to an earlier design stage, they have the potential to influence the design much more dramatically than the parametric variation of parts alone.

2.3 Translating the constraints into design drivers

For design exploration it is necessary to translate constraints into design drivers. This can be accomplished by setting up an exploration system that incorporates the constraints but allows for exploration of the unconstrained aspects of a possible design. The hanging model implementation is an example of this approach and demonstrates the use of a solver-driven architecture based on particles and springs for tension-only form finding [7, 8]. The hanging model (Figure 3) is the exploration of geometry within the constraints of a given topology and structural behavior. In addition to adjusting the geometry to meet the structural constraints, the environment can visualize the approximate material envelope in proportion to the forces in the geometric members. This ties in an additional constraint, the load bearing capacity of a given material. The material factor defines the minimal allowable cross section for the beam members under load according to local buckling length and in the implementation it provides the designer with a live feedback of member volumes in the current design configuration under the given material constraint.

In the implementation, an interactive model uses solvers to negotiate
between a fixed topology and a slightly adjustable geometry based on simulated spring and point masses. Properties are embedded in the behavioral model through those primitives. The network of primitives is solved for equilibrium of forces through a solver. The digital implementation allows the user to shift the design driver. One way is through editing the topology, another through editing the geometry by changing the lengths of the members, and last by adjusting the material parameters which control the allowable cross sections of the beam extrusions based on the forces present in the members.

2.4. Applying constraints to a problem domain

New constraints may also be applied to an existing design domain. For instance, this is the case with the increasing use of digital generative design in combination with CNC fabrication techniques in architectural design [9, 10, 11, 12, 13]. The fabrication techniques as well as the digital generation of the modeling information have specific constraints that apply to the design artifact. Examples are fabrication techniques based on flat sheet material. Flat-sheet fabrication and prototyping has increased due to the availability of relatively inexpensive CNC machinery for cutting, such as laser and plasma cutters. These machines are now available in design studios as well. The developability constraint imposed by the flat sheet material and the fabrication process has led to the development of a fabrication-specific design language in the construction industry similar to developments decades earlier in product, ship, and airplane design [4].

At the core of this example is the translation of free-form shapes into developable surfaces using the developable primitive of the cone. The application of this primitive can produce interesting new approaches to low cost fabrication of free-form surfaces. The example below (Figure 4) is that of a cone-based translation of a free-form surface into developable cone-based parts.

In a more sophisticated version, the degree of curvature could control the spacing of the cones to achieve a consistent approximation error. Alternatively, a growth-based approach could be taken similar to the process in MoS5 [14] where an L-system guides the growth of a surface. In the example shown here, a simple close-packing-circle approach is used to
produce the basis of the circular cones and the circle center points form the tip. We are interested in the circular cone base as a spatial curve that coincides with the curvature of the surface while the resulting cone surface is still strictly developable. The fabrication constraint is therefore embedded in the geometric property of the chosen primitive, which makes it a very robust approach.

The next example of turning constraints into design drivers is taken from the Concept Car design studio mentioned above, headed by William J. Mitchell. The studio explored novel car architectures using a fairly open approach. One of the outcomes was the “athlete”, a car that has many more degrees of freedom for steering than the two degrees of freedom required to steer on a planar surface. Articulation became the design driver for this concept car design study. The additional degrees of freedom provide the opportunity to explore alternatives to rigid car chassis designs. The challenge was to solve the constraints for a particular movement and to determine the relationships between the different degrees of freedom for a particular movement. To do so, the three-dimensional constraint solver in the CATIA© engineering package was used. The frame structure was modeled three-dimensionally and the components were jointed through geometric constraints. Relationships of movement between the different joints were then defined through equations. Finally one of the joints was animated, which propagated the movement through the entire constraint.

Figure 4. Cone-based approximation of a double curved surface using developable surfaces only.
system of the frame (Figure 5). This constraint solving study was used to validate the complex interdependencies of the six degrees of freedom. The study led to adjustments in the relationships of the constraints and to the addition of two constraints to compensate for non-planar wheel positions in turns. The problem had previously been studied in physical models, but only the constraint-solving approach provided results precise enough to detect the necessary adjustments in the joint layout.

Here constraint solving helps to find a working parameter configuration for driving the vehicle. In general it offers a direct way of resolving competing constraints in three dimensions.

2.5. Exploring the solution space for a given set of constraints

Setting up a design problem through modeling its constraints is a first step to finding possible solutions. However, once a possible solution is found it is not guaranteed to be the only one or necessarily the best one. Genetic algorithms have been extensively used to optimize parametric objects for a given fitness function [15]. The fitness function can be viewed as an implementation for enforcing a constraint in form of the fitness measure (Figure 6). The resulting objects conform to the constraint by selection although strictly speaking the fitness function itself is not a constraint.
For the design exploration of lower dimensional parametric constructs, genetic algorithms often prove to be too cumbersome to set up or provide too little opportunity for user intervention in order to guide the process. An alternative, at least for lower dimensional parametric searches, is mapping parametric values to geometric control objects. These objects make it possible to capture desirable parametric settings and explore the neighborhood for small variations of the chosen values without losing the context of the previous result.

Besides their ability to support dimensional variations, parametric models also model topologic constraints through embedded part association. While navigating the solution of a parameterized object, one can record the different settings accordingly. This approach allows for interpolation of the intermitting parameters. A vase controlled by six parameters (Figure 7) demonstrates a geometric control object to provide parameter interpolation and the possibility to record and memorize states of the parametric settings for the object. Designing parametrically poses the challenge of evaluating the range of possible outcomes presented by a parametric construct set up for the exploration. Higher numbers of parameters make it less intuitive to interact with the design construct. The parametric control object also allows for another instance of a bidirectional link as the control object is both generated by the sample design but also can by direct manipulation drive the set of designs.

A further example (Figure 8) shows a family of objects whose parameters are mapped to one point’s XYZ value in a grid that samples a surface based on parametric UV spacing. Moving the points of the UV-based...
grid changes the points XYZ coordinates which are mapped to the parameters and therefore regenerates the object family. By increasing or decreasing the sampling rate around the points of interest, one can explore parametric variations in more detail where needed. As an alternative to a design surface, one can use a design volume for the exploration. Each point is one sample within the exploration range. In order to increase the resolution of variations around a particular sample, the sampling can be bundled around one parameter (Figure 8). The mapping of preferable design instances in the solution space creates spatial traces, which can suggest additional members of the design family by interpolation of or through spatial proximity to the existing samples in the solution space.

3. Design exploration

Exploration is a central aspect of design. The process of defining the boundaries and constraints of an exploration helps to define the problem itself. Computational methods to support design exploration potentially enhance the designer's reach and allow for the modeling of complex interdependencies that cannot be easily studied in non-computational ways.

3.1. Types of constraints

As discussed above, constraints play a central role in the definition of most design exploration. Constraints set the limits and the metric of evaluation for an exploration. Although they are a limitation, they do help to
externalize the issues present in a design problem and thereby can turn from constraints into design drivers. The constraints illustrated in the examples above are only a fraction of the constraints present in design problems. The examples are meant to be suggestive of the possibilities that come about with the integration of constraints in computationally supported exploration. This is not necessarily applicable to all design problems and not all constraints can be readily translated into quantifiable representations. Those constraints require more sophisticated computational models to support them and are not addressed in the scope of this paper.

3.2. The role of constraints as design drivers

Constraints in the context of a design exploration can turn into design drivers. Their role as design drivers is determined by their attributed weight in the design problem. A design driver is a constraint that remains unchanged throughout the exploration even if there are competing contradicting constraints present. Design drivers are constraints that are challenging enough to require a design innovation to satisfy them. The developability of the paper cone in the double curved roof study is an example of design driver (Figure 4). One could have explored other ways of approximating a double curved design surface but the developability constraint was chosen as the focus of the exploration. For the vehicle design study the design driver or main constraint was the idea of an articulated body. Most other design development was derived from this constraint.

3.3. Bidirectionality in constraint modeling

To be truly supportive of design exploration, constraint solvers need to be bidirectional [16]. This means the constraint network cannot be implemented as a hierarchically structured dependency tree that allows only propagation of effects towards the tree leaves. A graph requires a more general definition of part relationships since the possibility of cyclic dependencies exists. The circular nature of the network and the possible reversal of the propagation direction along the links requires the use of a bidirectional solver.

Constraint explorations are often only analytical in nature, meaning that a change in a parameter will produce a result but the result can in turn not be used as the driver to continue the exploration. In contrast, a classic example of bidirectional exploration of a constraint network is graphic statics [17], where a force polygon is linked with a form polygon through geometric constraints and change can occur in both the form polygon and in the force polygon. This allows for the exploration of either form or force while each change in one representation affects the results in the other through the graphics statics constraints. When the form and force network
becomes more complex, the mappings between the two representations are not fully determined as multiple solutions may exist for the form polygon for a given state of the force polygon. To explore the possibilities it is necessary to use a bidirectional solver. We modeled several such tests using large graphic statics system modeled in CATIA®'s sketcher environment, which features a bidirectional solver.

3.4. Translation model

The translation model is an externalized intellectual, computational or physical construct that can relate design constraints in one representation to design constraints in the target representation. The majority of engineering models are analytical in nature. Their mapping occurs in a one-directional manner from one design representation to another. A classic example is structural analysis where the analysis of the form of a design produces a representation of the design in terms of forces. The translation is unidirectional. The force representation in isolation does not allow the reconstruction of the form if the process is reversed. The translation only works in one way and makes design exploration involving the different domains difficult. In contrast, the ideal translation model allows for bidirectional mappings between the design representations. We suggest instead defining a translation model as generating a new design representation by carrying over the constraints rather than translating the representation.

3.5. Bidirectional models

The condition of bidirectionality is a reappearing challenge throughout the examples in the paper. It is often mistaken as a lesser version of multidirectionality. However, it does not refer to connectivity in a design topology, but rather to the property of the translation model. Bidirectional stands for the translation in both directions without preference for one over the other. The network of interlinked design representations can go beyond that of a pair of domains described above. In fact, most design problems are circular in their cross dependencies, meaning the chain of influences wraps back on itself. There is some

3.6. Circular dependencies in design problems

Circular dependencies are far harder to explore than non-circular ones due to the feedback loops among the different representations. Most designs rely on a number of interlinked representations linked by constraints. Figure 9 shows how the different design representation in a chair design can be split up into design driver, implementation and representation and how these are interlinked though constraints. They create a complex construct even for relatively simple design problems such as the chair shown here.
Externalizing and describing these constructs is a major part of the design process. The exploration of variants is based on this externalized construct. We refer to this construct as a design explorer, which differs from the definition of design explorer given in [3].

3.7. Geometry and design exploration

Geometry is the main vehicle for design representation. In many cases, geometric design representations work successfully across different domains, and the ability to represent design in multiple domains is crucial to allow for design exploration.

Geometry is a very powerful representation format. It covers a wide range of design representation from descriptive geometry to abstract visual shape, but it does not cover all design representations. The translation of other design representations into a geometry format is not lossless. Geometry’s central role is further strengthened by the support of digital design environments that are based on geometry. The underlying math is
robust and straightforward to implement and acts as the backbone of geometric design representation. Geometric procedures in the form of tools in design environments can influence the choices designers make in their design representation. The integration of nongeometric design components into a central geometric design representation requires models of translation. Developing such models by extending the digital design environments is becoming increasingly popular among designers. One reason seems to be that many students and practitioners have reached a certain fluency with the existing tool sets and strive to distinguish themselves through customized design processes. These can be created through scripting or programming extensions of commercial programs. Programming therefore is becoming an important skill to create one’s own, design-specific digital context. Geometry may still serve as the output of the design, but the design representation shifts, at least partially, into the realm of programming.

3.8. Programming for design

By no means is programming an ideal medium to capture design representations. But it does provide a powerful extension to the black box software usually available to a designer. Programming is not very flexible or approachable and it is a skill that most designers need to learn rather painfully and often without much success. However, the threshold can be significantly lowered by using programming environments that cater to design communities such as the processing environment [18]. The programming language within processing is very similar to Java®, but the programming structure is simplified by offering commonly-used procedures for beginners and advanced users alike to express their concepts in programming form.

4. Conclusion

In conclusion, we described a set of case studies in the use of constraints as design drivers giving examples for quantitative, geometric, topologic, and functional constraints. There are many more possible constraints, for instance constraints between multiple media and representations, many of which are defined in the literature [19, 2, 3].

The purpose of the paper is to demonstrate the heterogeneous nature of constraints and how using and modeling them made a difference in the case studies. The case studies show the combination of a number of established computational principles such as genetic algorithms, parametric modeling, and graphs in support of design exploration. Ideally, the modules that implement such computational models would be more easily accessible and combinable in the design environments in use today. Some computational design environments are developing in this direction, most notably processing by Ben Fry and Casey Reas. [18]
A bigger challenge for future work lies in the development of solver architectures that support constraint resolution for non-geometric constraints reflecting the heterogeneous nature of design problems. Constraints exist in many forms and there is no master model that can incorporate them all, but improving the bridges between different isolated constraint models could improve the availability in design.

Acknowledgements

I would like to thank John Ochsendorf, Simon Greenwold, Barbara Cutler, Eric Demaine, and Marty Demaine for their contributions and comments in the context of the hanging chain workshop conducted at MIT. I would like to thank William J. Mitchell for providing the opportunities for the design exploration in the Concept Car studio through the Smart Cities Group of the MIT Media Lab. I would like to thank the Department of Architecture and in particular the Computation Group for supporting my research through the Ph.D. program, which made this paper possible. I am grateful to my Ph.D. committee, Takahiko Nagakura, William J. Mitchell and Una-May O’Reilly, for their insightful comments and their support throughout my thesis.

References

10. Kilian, A., Fabrication of partially double-curved surfaces out of flat sheet material


Axel Kilian
Massachusetts Institute of Technology, Department of Architecture, School of Architecture and Planning, 77 Massachusetts Avenue
akilian@alum.mit.edu