ON A GRAMMAR-BASED DESIGN LANGUAGE THAT SUPPORTS AUTOMATED DESIGN GENERATION AND CREATIVITY

Rolf Alber and Stephan Rudolph
Institute for Statics and Dynamics of Aerospace Structures, University of Stuttgart, Germany
alber@isd.uni-stuttgart.de, rudolph@isd.uni-stuttgart.de

Abstract: One of the major obstacles towards the realization of automated creative design lies in the restricting design frames explicitly or implicitly imposed by common means for design representation. In this paper it is proposed to extend these frames by improving the design representation with a description technique that is more suitable to capture conceptual decisions taking place in engineering during early design phases. In this context especially the description of topological arrangements or functional dependencies is emphasized. It is shown that the application of grammar techniques known from formal languages is well suited for the development of a compact description language for engineering design objects. This is illustrated in an example for a formal language of truss designs. Furthermore it is investigated how this new design representation approach can support "creative" engineering and where possible future research along these lines can be done.

Key words: Design generation, design representation, design grammars, design synthesis, L-Systems

1. INTRODUCTION

Many different technologies show interesting parallels concerning the amount of creativity which was introduced into the design during their historical evolution. An often observed scenario is that the biggest part of the overall creativity occurs during the early design phases, whereas the following development process mainly concentrates on optimization in certain details of the already chosen solution concept. Examples for this observation are not only found in the evolution process of single products but even more for the historical development of whole technology branches with a long engineering tradition (e.g. aircrafts, automobiles, computers). Here the amount of creativity which was present at a certain stage exemplifies itself in the variety of basically different solution approaches within the same problem-domain/technology.

To maintain a high level of creativity in the later phases of the design cycle, it would be necessary to go beyond the restricting frames of reference (Akin 1998) which are set up by the conscience of already existing solutions. In respect to the paradigm that "design is search" which relies on a finite and closed-world assumption, a necessary condition for creative design lies therefore in the extension of search space.

In the following it is investigated how this idea of extending design space in order to achieve creativity could be adopted for models of automated computational design. For this purpose, the means for the formal representation of design space applied in conventional computational design tools (like computer-aided design, CAD, differential-algebraic network eqations, DAE) have to be examined in a first step. In a second step, it is investigated how this formal design space could be extended.

1.1 Continuous vs. Discontinuous Design Space Representation

The representation of the whole design space goes fundamentally together with the method used for the representation of one single design specimen, which stands for a specific point in that particular design space.

In general, starting from this point, a manifold of other designs which together form a part of the design space can be accessed by introducing additional degrees of freedom to the representation of this special design specimen. The most common approach for this extension lies in the parameterization of certain numeric values which were necessary for the unique determination of a design. This approach for a partial capturing of design space is straightforward and generally applicable for the common types of formal design languages like the mathematical description in formulas or a geometrical description in technical drawings.

With the common types of design representation the basic idea of achieving creativity by a further extension of design space reaches consequently a limit when all dimensional constraints have been released through parameterization. The next step in order to increase design freedom affects aspects like the basic functional topology or the underlying design principles. In general, there exist in current design tools no suitable means to represent these more abstract aspects of the design.

In the current state, the functional topology and the rough outlines of the desired design solution have to be already in the designers mind when it comes to model the design for further evaluation. The early design phase when these basic design principles are manifested is what we call the "sketch-level". Although this part of the design process has probably the most important influence on the final design and also the biggest opportunity of including innovation and creativity, it is the least supported by current computational design tools. The reason why a modeling of the sketch-level is not yet possible lies in both the lack of understanding of topology manipulation and of a formal representation of the cognitive processes during this phase. These are some of the main obstacles towards the automation of creative design. The aim of extending the design space calls therefore for a design language which is able to provide a formal model in order to express the functional and/or spatial topology and the dependency between different parts and aspects of the design. Typically these characteristics are expressed on a more verbal or descriptive level when dealt with by humans and therefore are hard to imitate by mathematical or algorithmic expressions as needed for a computational implementation.

In the domain of artificial intelligence rule-based information processing concepts have been developed which together with methods from formal languages and language processing are proposed to provide a promising approach for an extended design representation and thus for the desired extension of accessible design space. This paper shows the working principle behind rule-based design grammars for the example of biological growth modeling and investigates the implications of the transfer of this approach for the description of designs in technical domains.

2. DESIGN GRAMMARS AS A LANGUAGE FOR FORMAL DESIGN REPRESENTATION

Rule-based information processing concepts which stem originally from the field of artificial intelligence have already demonstrated their promise in several technical areas. Known applications can be found e.g. in architecture for the design of shape patterns (Stiny 1977), in product-shape, function and cost estimates of coffee makers (Agarwal and Cagan 1997 and 1998) as well as in mechanical design applications (Heiserman 2000). Studies of generic XML-based representations of knowledge enhanced engineering design grammars for a DAE (Differential Algebraic Equations) modeling approach have also been undertaken (Noser and Rudolph 2000).

The grammar-based design concept proposed here consists of two phases. The first part takes place in a very formal and abstract domain and works simply with symbolic placeholders far away from the real-world design entities. This part plays the role of evolving a well formed topological structure by arranging a set of symbols through the subsequent application of production rules. The second step translates this topology to a real-world design by interpretation of the symbols as material entities.

The principle of this approach is illustrated by a string grammar called Lindenmayer-Systems (or L-systems for short) which was developed for the description of biological plant and tree growth. L-systems turned out to be a compact and powerful means for the representation of hundreds of topologically different complex natural objects (Lindenmayer and Prusinkiewicz 1996).

2.1 Basic Concept

L-systems represent a special case of the grammar formalisms which were developed and classified within the scope of formal language theory (Chomsky 1957), in order to describe the structure of a string as a systematic application of substitution rules. The main idea behind L-systems is to consider the creation of natural structures as the result of a developmental program which could be understood as a recursive process (Lindenmayer 1968).

2.1.1 Rule-based structuring

In L-system grammars the abstract symbols are represented by characters from an alphabet which manifest a topological structure through their arrangement patterns in strings.

The basic concept for structuring this set of symbols relies on the principle of rule-based substitution.

```
Initiator -> Generator
```

Figure 1. Syntax of a production rule

A substitution rule consists of an initiator on the left side and a generator on the right side as illustrated in Figure 1. If the initiator is identified anywhere in the object at the current stage of recursion, the corresponding rule is applied by replacing this part of the object by the part defined in the generator. The example illustrated in Figure 2 shows how a complex structured string can be unfolded from a single character (i.e. the axiom ω)

within only three iterations by subsequent application of two substitution rules (p_1, p_2).

(Axiom ω)	B
(Rule p_1)	B→F [-B] +B
(Rule p_2)	F→FF
(Recursion)	(Expanded String)
(i =0)	B
(i =1)	F [-B]+B
(i =2)	FF [-F [-B]+B]+B]+F [-B]+B
(i =3)	FFFF[-FF[-F [-B]+B]+F [-B]+B]+FF [F[B]+B]+F [-B]+B

Figure 2. String expansion by recursive application of rewriting rules

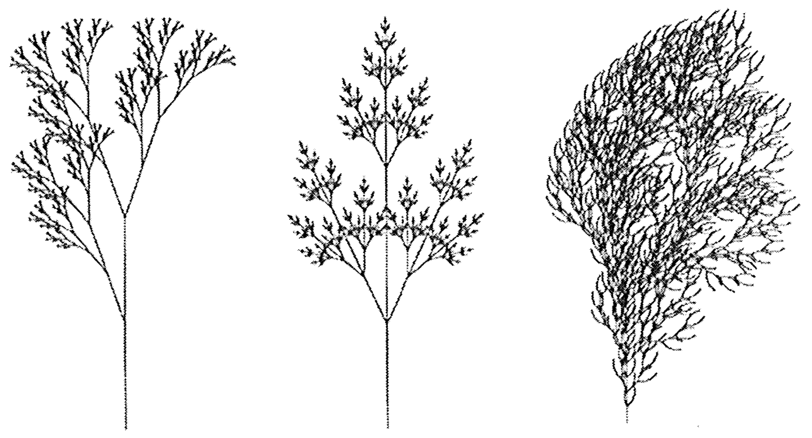
2.1.2 From strings to plants

Since classical L-systems were introduced to represent biological growth processes, a physical interpretation of the topological structure encoded in the abstract character set is necessary to obtain botanic objects. For this purpose, the concept of turtle graphics (Abelson and diSessa 1984) was adopted and modified to translate the single characters in the expanded string as material components and construction operations. Table 1 shows a possible mapping between the characters of the alphabet and the material components.

Table 1. Character assignment of material components and growth operations

(Character)	(Interpretation)
F	add stalk segment
-	rotate growth direction by a discrete angle counterclockwise
+	rotate growth direction by a discrete angle clockwise
[	introduce stalk segment with branching
]	jump back to the last branching point

In the following examples the character-interpretation given in table 1 was applied. Figure 3 shows several L-system grammars together with their corresponding two dimensional geometric interpretation. In the figures shown the notation i is the necessary recursion depth during the expansion phase and ω denotes the angle applied for the interpretation in table 1.



ω : X	ω : X	ω : F
p_1 : X $\rightarrow$ F[+X]F[-X]+X	p_1 : X $\rightarrow$ XF[+X][-X]FX	p_1 : X $\rightarrow$ F-[[X]+X]+F[+FX]-X
p_2 : F $\rightarrow$ FF	p_2 : F $\rightarrow$ FF	p_2 : F $\rightarrow$ FF
$i=7, \beta=20°$	$i=7, \beta=20°$	$i=5, \beta=22.5°$

Figure 3. L-Systems with Interpretations (Lindenmayer and Prusinkiewicz, 1996)

3. GRAMMARS FOR TECHNICAL OBJECTS

The illustrated examples have shown that it was possible to describe a variety of objects with a basically different topological structure within the same formal representation system. This raises the question for a possible application of the grammar formalism for the description of designs in technical domains. For this reason, the circumstances that make L-systems so compact and powerful should be identified and considered for incorporation into the development of a future engineering design language based on similar grammar formalisms:

– The biological objects shown here were assembled out of a few basic components and/or operations. For the L-Systems shown in Figure 3 the components "add stalk segment" and "turn growth direction" as well as the stack operations were sufficient for the complete assembly of some rather complex objects.
– For the development of a technical grammar it will therefore be a first objective to search for often repeated basic building-blocks out of which the desired object can be composed. The so defined set of components is therefore directly assigned to the abstract symbols on the formal level during the interpretation phase.
– All of the generated biological objects show a high amount of symmetry, self-similarity and repetition. Therefore production rules that were used to control the structure at one point were also applicable in many other

locations. This multiple application of design rules lead to the impressive compactness featured by the shown L-system grammars. Besides the definition of the basic components it will therefore always be important to search for repeatedly used construction principles which could be represented by the very same production rule.

However, a simple analogy from L-system grammars to fields outside the classical areas in biology (Lindenmayer and Prusinkiewicz 1996) and for fractal modeling (Kaandorp 1994) is confronted with severe difficulties. The information processing of the classical L-system grammar is focused on plant-like structures, in particular, objects with a tree-like "branched" topology. Objects where branches join together again and form closed loops are not straightforward realizable with the approach described here. Since these structures are quite common in technical applications, this has been probably the main obstacle for the use of L-Systems in other domains.

This way of encoding information in symbol strings stems from formal systems developed in mathematics in the early 20th century. For representation of logical or algebraic expressions this one-dimensional structures were well suited. Interpreting the string topology as functional dependency or a spatial relation of material components or to find an isomorphic mapping between a linear arranged character chain and more complex topological patterns however can become a hard task. For L-systems such a mapping was achieved by the introduction of the stack-operators "[" and "]" in order to gain tree-like topologies. A similar treatment of looped structures might also be possible by special symbols representing connection points. However, it is clear that such mappings lead to an increasing complexity concerning the interpretation of the structured symbol set and can thus make the creation of a design grammar very difficult. Therefore it might be advantageous to search for a more suitable method for representing the structural information.

3.1 Proposed Modifications for Technical Grammars

Following the suggestions and implications pointed out in the last section, the grammar approach for technical designs has to undergo several changes and extensions which are proposed in the following:

In order to facilitate the isomorphism between the symbol arrangement on the abstract layer and the topology shown in the desired real world objects, it is attempted to work with an abstract structural representation that is still accessible with a rule-based approach but needs less complicated interpretations later. This can be achieved by formatting the dependencies on the abstract symbol layer in undirected graphs instead of strings.

The corresponding grammar formalism for a rule-based construction of such a graph is therefore also called graph grammar (Ehrig and Kreowski 1990). The analogy to string grammars with their substitution rules presented in the last section is straightforward. The initial state denoted by the axiom which was a string in the former case is now represented as an initial graph.

The production rules still resemble the initiator-generator form of the rewriting rules in Figure 1, with the simple difference that now the initiator as well as the generator are represented by partial graphs. The substitution procedure is again applied by finding the initiator-graph as subgraph in the current object and replacing it by the graph defined in the generator. A more detailed examination will show that for complex rules the substitution procedure needs an additional description which is called embedding-relation. This however lies outside the scope of this work but can be found i.e. in Göttler (1988).

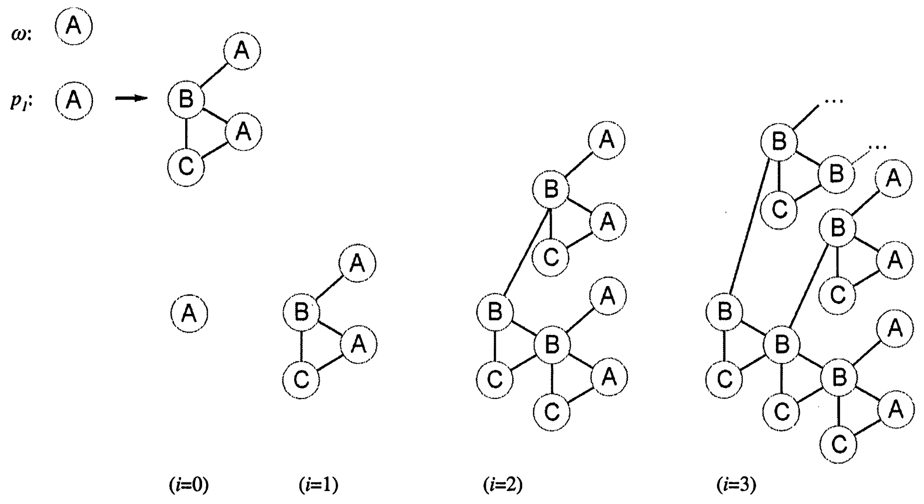


Figure 4. Graph evolution by recursive application of graph grammar rules

Figure 4 presents an example for the syntax of a graph grammar and its application and can be seen as a direct analogy to the string grammar formalism illustrated in Figure 2.

4. A GRAMMAR FOR TRUSS-LIKE STRUCTURES

As an example for the grammar design proposed in the last paragraph an engineering design grammar for truss-like structures as they appear for example in pylons for powerlines, in civil engineering domains or in space-structures has been developed. Since such pylons show at least a certain amount of symmetry, self-similarity and repetition, it is expected that this technical object serves as a good example for how a compact design representation can be achieved by means of a rule-based approach.

4.1 Basic Concept

Following the approach proposed in the last paragraph, basic building blocks for the most common truss-like structures have been identified in order to be replaced by abstract symbols for an easy-to-use rule-based implementation.

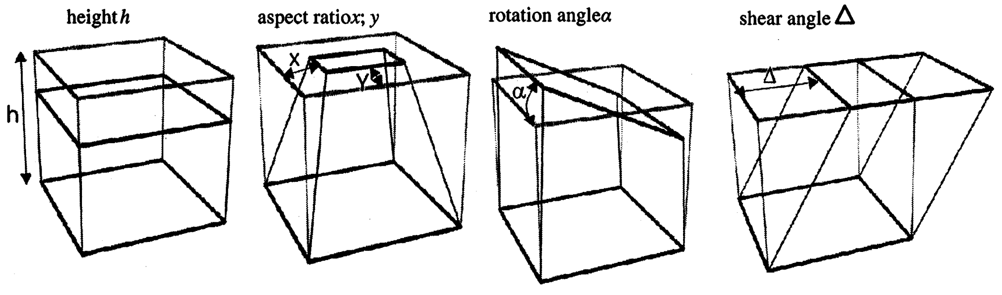


Figure 5. Some basic parameters for the shape of a primitive

Figure 5 shows such a primitive (here a polyhedron) where the parameterization effects the aspects height, length ratios, twist angles and the rod topology inside the polyhedron (not shown). In this way a high variety of primitives can be inherited from the same building block and therefore a high amount of self similarity and repetition is achieved as will be illustrated in later examples.

By defining such polyhedrons most of the common truss-structures can be represented by a graph that describes how these components are instantiated and connected. The combination between different primitives is done through the side faces of the polyhedron which serve as interfaces. For geometrical consistency, it is necessary that the geometry of a subsequently added primitive is adopted such that the corresponding interfaces match each other. Therefore the geometric parameters are always defined relative to the corresponding parameters of the parent primitive.

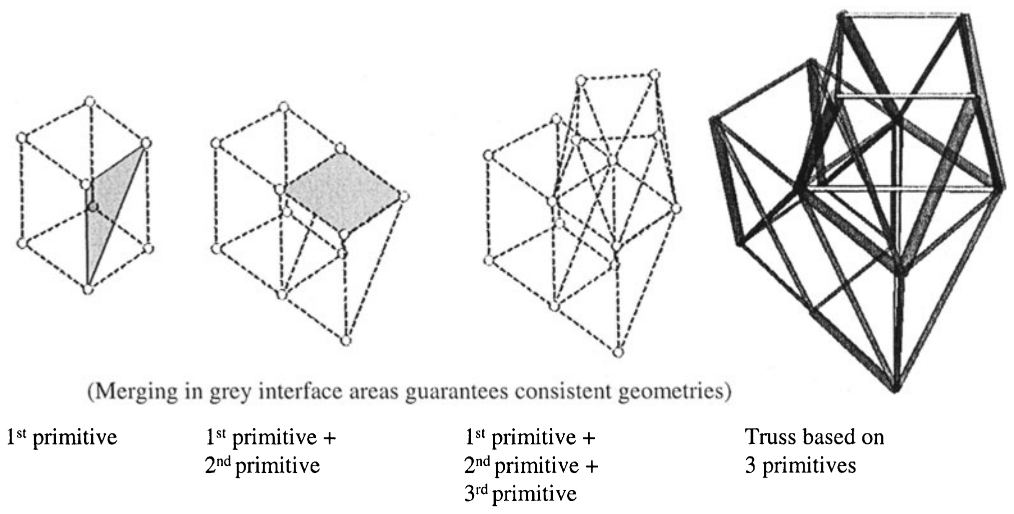

(Merging in grey interface areas guarantees consistent geometries)
1st primitive
1st primitive + 2nd primitive
1st primitive + 2nd primitive + 3rd primitive
Truss based on 3 primitives

Figure 6. Truss based on three connected primitives

Figure 6 shows in an example how a truss-structure can be assembled from three connected geometrical primitives. Each of the components may hereby be modified according to Figure 5. The shaded areas indicate the interfaces where the connection of components occurs.

4.2 Implementation

The graph grammar implementation which has been chosen for the truss grammar operates on an abstract node set, whereby each node has a letter of the latin alphabet assigned as label for a unique identification.

After each rule application an immediate interpretation step follows which corresponds to the assignment of a geometric primitive to the nodes. Thereby the graph-relations define the connection between the single primitives. Furthermore an evaluation step can be performed after each rule application. For our purpose an evaluation of basic geometric properties is sufficient. The gained geometric information (like position, size, angles) derived in this step can be fed back into the design graph and thereby interrogated by additional constraints in the graph rule formalism.

4.3 Example

In the following an example for the truss grammar is presented. Figure 7 shows a set of 5 rules which describe the building principles behind the desired truss.

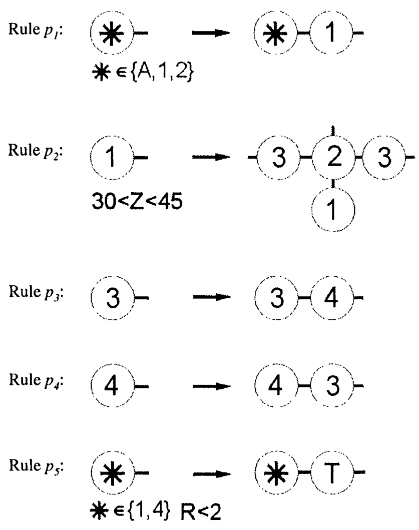


Figure 7. Rule set for the high voltage pylon topology

As a new feature wildcards have been introduced to the rule syntax which are denoted by the symbol "*". In the rule-set shown here, two additional constraints occur, interrogating the height above ground (Z) and the size (R) of the primitives interface.

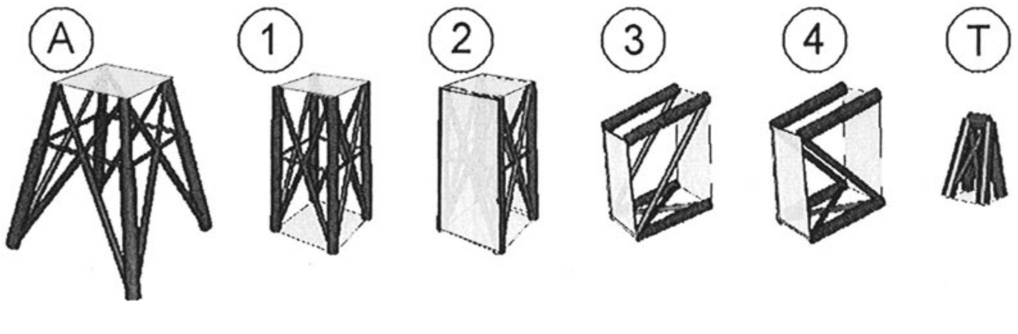


Figure 8. Material component interpretation of the rule symbols

The interpretation of the derived symbol graph is done with the assignment table given in Figure 8. The 6 shown primitives were all derived from the same underlying polyhedron with different instantiations for the geometric and topological parameters. The shaded faces indicate possible connection interfaces.

Figure 9 shows snapshots from the stepwise evolution of a transmission towe following the ruleset in Figure 7. Similar as for L-systems, the rules are executed in an abstract space and then interpreted using the material components in Figure 8.

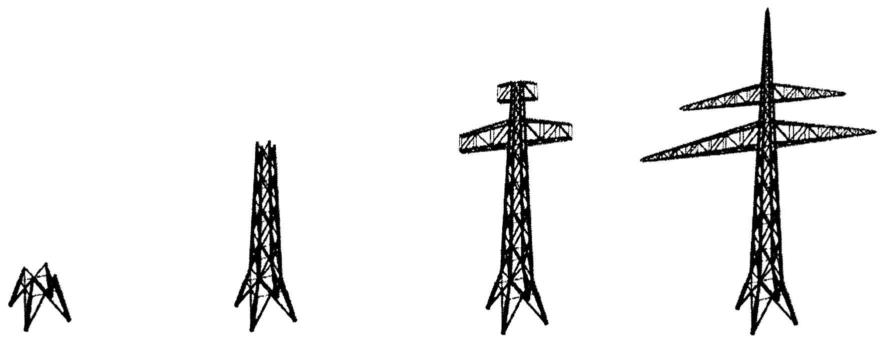


Figure 9. Rule-based evolution of a high voltage pylon (Alber, 2000)

Comparing the complexity of the derived object with the information needed for its construction shows a high packing density for this design grammar similar to that of the L-systems from which they were inspired.

The formal approach for covering design knowledge concerning basic construction principles shows especially its usefulness when it comes to principal changes in the model. Figure 10 shows the object resulting from a simple change in the design rules. For the shown pylon the additional constraint in rule 2 of Figure 7 was changed from 30<Z<45 to 25<Z<55 which leads to a fundamentally different topology.

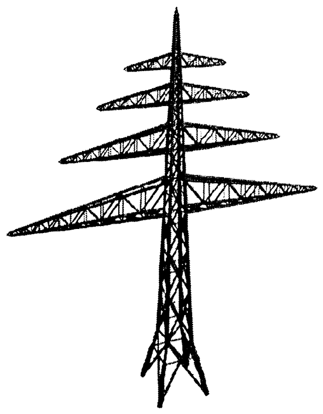


Figure 10. Transmission tower as a result of altered rule set

5. SEARCHING THE DESIGN SPACE

The goal which was aspired with the grammar approach was to capture large portions of the design space within one single formalism and especially to increase the design freedom, such that the automatic generation of new designs becomes possible. As another point which seems important in the context of this work is the question whether and why one might be willing to attribute to the output of such a generative design language the property of being "creative". For investigating these questions, a closer examination of the covered design space is done in the following.

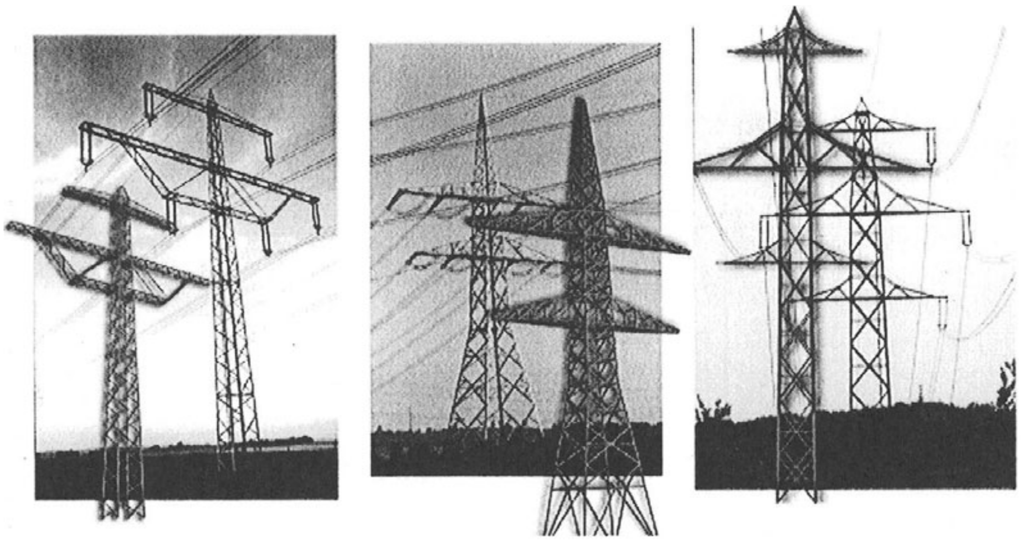


Figure 11. Real-world transmission towers imitated by the design grammar

By manual formulation of grammar rules it was possible to find a truss-language description for several real-world objects. Examples are illustrated in Figure 11 (Alber 2001). Each of the three high voltage pylons was generated by a design grammar within the truss grammar formalism and thus corresponds to a point in the captured part of design space. The entirety of these points represents the space of current-state designs.

As expected, the language captures a design space which extends these points by far. Therefore it is interesting which new trusses correspond to these additional points beyond the current-state designs. For this purpose, a simple generative procedure was written in order to randomly generate a set of rules and thus randomly pick out points from our design space. Some examples for the trusses corresponding to these points are illustrated in Figure 12.

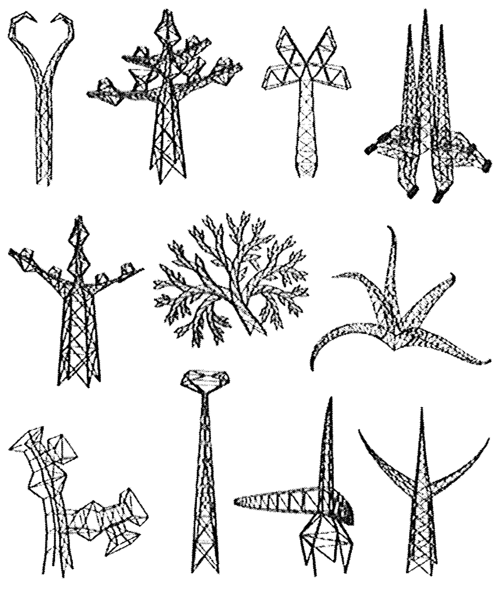


Figure 12. Random-generated points in transmission tower design space

Whether these new trusses deserve the attribute of being creative might still lie in the eye of the beholder and is of course strongly connected with the question how one defines creativity. For the following, one possible definition of creativity as found in (Akin 1998) is adopted: "Creativity is the process that leads to the creation of products that are novel and valuable." In this respect, the required novelty might be created by random mutation as shown in Figure 12, while design evaluation must be satisfied by an appropriate design evaluation model (Rudolph 1996).

For the question of novelty it can be stated that a variety of truss topologies has been generated by a completely formal approach which most probably wouldn't have come into the mind of a human designer. As far as the assessment of value is concerned, this is less distinct and depends more on the goals and design requirements opposed to a certain design task. This means that the decision whether a point in design space is only new or even creative is a question of evaluation.

At this point it shall be pointed out that the rule-based approach with design grammars was not intended to replace the common computerized design tools but rather to extend their abilities. In this way our design language parser plays the role of a front-end application with the possibilities to support subsequent evaluation tools for analysis and detail optimization from areas like FEM, CAD, DAE, etc.

Figure 13 shows as an example a model which was generated by the truss language and was exported for further analysis into a commercial FEM package (NASTRAN 2001). This means that a direct connection between an automated generation and the automated evaluation of design objects is possible. Thus the question for creative designs can be answered by an intelligent search through the design space, in order to find designs which are not only new but also valuable at the same time. In this respect the unanswered question what "creativity" really is can be transformed into the question of determining a value-function for design evaluation.

However, this approach still needs further investigation. Especially the question for an in-advance description of the characteristics of the captured design space (like size, borders, seclusiveness, finiteness, consistency) in dependency of the type of grammar formalism would bring much insight in the question for automated creative design and would also deepen our understanding of the cognitive mechanisms taking place in the early design phases of creative engineering.

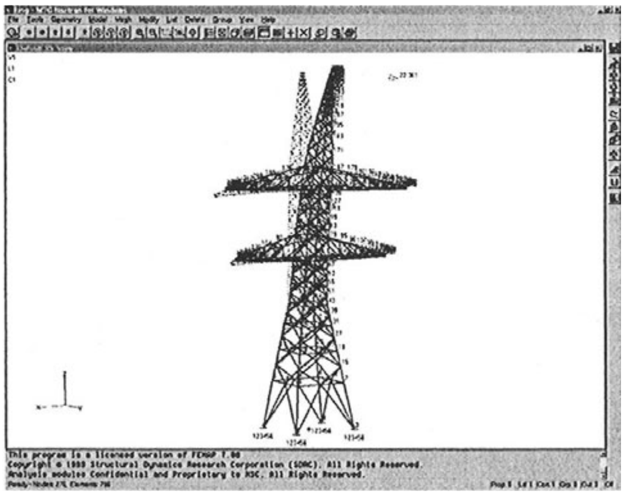


Figure 13. FEM computation with MSC Nastran (Nastran, 2002)

6. SUMMARY

Natural language still seems to be the most powerful means for the description of human thoughts. Although specialized description languages like mathematics result sometimes in a faster or more compact description, they are contained in natural language, whereas the inverse does not hold true. Today's computational design tools depend on specialized modelling languages which are designed for the description of one or more special aspects. The frames of reference which thereby restrict the design representation are one of the main obstacles towards an automated creative design. It is therefore proposed to broaden these frames by extending the design representation with a description technique that is suitable for early design phases.

The possibility to describe complex design topologies with grammars was shown for known examples from biology. Based on the promising results exhibited by this technique a graph-grammar based design language for truss structures was developed. The examination of the design space resulting from this approach showed that a lot of new designs could be captured in one consistent formalism. The conception of the design language consists of a rule-based graph formalism (in stage one) and an interpretation of the resulting graph as a composition of materialized components (in stage two). Based on the assumption that creativity is the generation of objects that are new and valuable, it is observed that an automated generation of creative designs is computational extensive, but possible within this scope.

REFERENCES

Abelson H. and diSessa, A. (1984) Turtle Geometry, The MIT Press, Cambridge

Agarwal M. and Cagan J. (1997) Shape Grammars and their Languages - A Methodology for Product design and Product Representation. Proceedings ASME Design Engineering Technical Conferences.

Akin, O. and Akin, C. (1998) On the Process of Creativity in Puzzles, Inventions, and Designs, Automation and Construction, 7 (123-138)

Alber, R.: (2001) Synthese und Evolution einer technischen Entwurfsgrammatik nach Vorbild biologischer Wachstums- und Entwicklungsprinzipien. MSc Thesis, Universität Stuttgart.

Cagan J. and Agarwal M. (1998), A Blend of Different Tastes: the Language of Coffeemakers. Environment and Planning B: Planning and Design 25, 205-226.

Chomsky N. (1957), Syntactic Structures. Mouton, The Hague.

Ehrig H. and Kreowski H.-J. (1990), Graph Grammars and Their Application to Computer Science. Springer-Verlag, Berlin.

Göttler H. (1988), Graphgrammatiken in der Softwaretechnik. Springer-Verlag, Berlin.

Heisserman J. (2000), A design representation to support automated design generation. Proceedings Artificial Intelligence in Design Conference 2000, Gero, J. (ed), Kluwer Academic Publishers, 545-566.

Kaandorp J. (1994), Fractal Modelling, Growth and Form in Biology. Springer-Verlag, Berlin.

Lindenmayer A. and Prusinkiewicz P. (1996), The Algorithmic Beauty of Plants. Springer.

Lindenmayer A. (1968), Mathematical models for cellular interaction in development, Part 1and 2, Journal of Theoretical Biology, 18, 280-315.

MSC Nastran (2002). http://www.mscsoftware.com/

Rudolph S. (1996), On a Symbolic CAD-Front-End for Design Evaluation Based on the Pi-Theorem. In: Gero, J. and Sudweeks, F. (eds.): Proceedings of the IFIP WG5.2 Workshop on Formal Design Methods for Computer-Aided Design, June 13-16th, 1995, Mexico City, Mexico. Chapman and Hall, 1996, London, 165-179.

Rudolph S. and Noser H. (2000), On engineering Design generation with XML-based Knowledge-enhanced Grammars. Proceedings IFIP WG5.2 Workshop on Knowledge Intensive CAD (KIC-4), Parma, Italy, May 22-24.

Stiny G. (1977), Ice-ray: a note on the generation of Chinese lattice designs. Enviroment and Planning B: 4 89-98.