USER DEFINED DESIGN GENERATORS - A GRAMMAR EXTENSION OF CAAD SYSTEMS

AUTHORS
Michael Hellgardt
Prinsengracht 151
1015 DR Amsterdam
The Netherlands
106555.1412@compuserve.com

ABSTRACT
A command system for CAAD, +G, is proposed for the definition and application of design grammars. With an emphasis on Aalto and Scharoun is discussed how design observation can be transcribed into rules of a grammar, SAG (Spatium Array Grammar), that reflects space-between structures ubiquitous in the built environment.
§1 A same shape grammar formalism (Stiny, 1980a) has been applied to model a variety of phenomenae, such as the Fröbel Baukasten, Palladio villas, F. L. Wright's prairie houses etc. This fascinating formal machinery (Stiny) can be used to describe particular languages and grammars, machines, that can generate expressions - by hand or computer implementation. But it cannot understand different languages and grammars in relation to each other. This formal machinery is language theory, frozen at the level of description. What is missing is the link to some kind of theory, language theory, given as a system of grammar rules describing not just blind mechanisms of unrelated languages, but a "speakers" competence in various languages and grammars to generate expressions in actual situations of language performance. To establish hierarchies of rules, to compare rules and rule sets of different languages, to investigate rule acquisition by language performance - these are operations of understanding language, their functioning in space and history, their blindness, their openness, etc.

Can a formal machinery, or a machine, understand anything? The paper joins those who say: no. But these people use to add, in one way or another: any new machinery is a challenge to understanding. On that ground competence in architecture - as in any other domain of specialism, except for the formal sciences - should not be confused with competence in formal design modeling and computer implementation. But the latter, the technology of AI ultimately, can provide tools for a computer supported experimentation with shape and design rules and grammars - not only for the specialist in formal modeling, but also the ordinary architect without special formal knowledge.

§2 “Words are like pistols, ready for use,” one of the great French writers has declared. The same applies to shapes. A word is “fired” in a linear context, as Chomsky's most general form of a grammar or rewriting rule resumes:

(1) A → Z/X—Y, where A is realised as Z in a string between X and Y.
Shapes are “fired” in a spatial, 2- or 3-dimensional context. Architectural design and environmental growth applies two main types of shapes. Accordingly we get
(2) Sh → FA Sh, where Sh is a fixed shape, or a parametric shape with one or more parameters bound to functional restrictions, inserted into a context by a function FA, Find Allocation, or
(3) Sh → FA GS, where GS is Generate a parametric Shape with all parameters free.
Shape-rules and shapes are related by indices and parameters. This makes, within restrictions, grammars, shape rules and shapes independant - prior condition of the kind of experimentation claimed above. A shape rule can be called for any shape of a specified category region in different grammar contexts. It evaluates an evolving design, a shape and its parameters.

§3 The furnitured wings of figure 1 result from as many applications of a rule of type (2) as there are functional units prescribed by a brief. Such units may be part of building component taxonomies. Architectural design is largely a
configuration of functional units, occasionally even exclusively - which does not yet mean that ‘form follows function’.

§4 Shapes in (3) are geometric units or mental images, called leitbild. For Scharoun, in comparison with Aalto, crystalline leitbilder, as those of the “Gläserne Kette”, were fundamental. Two leitbilder are fundamental in the work of Aalto. One cultural: the amphitheatre, one natural: irregular boundaries, remaining of a pool/trunk/cloud/etc. Figure 3, bottom right, shows that GS in (3) can destruct a leitbild. It marks an approach that underlies many of Aalto’s assembly halls. The curved contours of figure 7 are applications of the pool/trunk/cloud/etc leitbild. Such examples open a discussion of how FA in (3) could be formalised and prescribed by parameters of shapes that can be passed to one or more shape rules.

§5 Design can be modelled as a concatenation of shape rules. Search for successful shape rule concatenation can be organised by networks, called Tree Adjoining Design Grammars , that adjoin trees of the form of (4). The root arcs of this tree address terminal shape rules or shape categories. Shape categories are preterminal if they denote terminal shapes. Otherwise they are nonterminals. The nature of architectural design, and grammar in general, can only be grasped if we include nonterminal shapes. Nonterminals call subnetworks for embedded design components which can be of the same kind as a shape rule category (a node) at a higher level. This makes a network recursive. Recursive networks require registers of an evolving design that can be modified and store temporary information for evaluation in various main- and subnetworks. These registers are the context argument, in structured numerical and symbolical form, for all arcs.

![Diagram](4)

- X (root) pre- or nonterminal shape category level
- 1 2 3... terminal shape-rule OR pre- or nonterminal subnetwork level
- A B C... next root level

§6 A design grammar can be defined by a number of compulsory or selectable shape categories (CSC, SSC) and sequential restrictions, called design rules. Design rule parameters can prescribe restrictions for particular shape categories. Node transition in (4) is blocked by test labels if illegal shape category sequences are encountered. To the extent that these test labels are uniform a TADG can be nondeterministic. In case a not viable combination is encountered search can go back and try another arc: a path option in (4) stored in a path memory when the nodes of a TADG are encoded.

§7 Design performance is more than the application of just one grammar. It experiments with shape vocabularies and syntactic devices that can be part of various grammars and “languages”. Architectural design performance, moreover, often evaluates existing context situations as initial shape, or design. The concept of parametric shape grammar (Stiny, 1980a) ignores these aspects. If we want to account for design performance a device is needed that applies grammar knowledge, that is design competence, in a local

---

* Tree Adjoining Grammars are applied in Natural Language Processing. Not all references and inspirations are specified in this paper.
context of requirements, intentions and context restrictions. Such a device is called a design generator.

If a grammar contains no SSC's the distinction between design grammar and generator is senseless. Otherwise a design generator must be defined and encoded as a network for the CSC's of a grammar selected and a user specified subset of its SSC's.

A design generator is run with the compulsory argument of a context as initial shape (which cannot be part of a grammar as it is in (Stiny, 1980a)) and optional grammar and operational arguments for non-default values. Grammar arguments are shape parameters and design rule parameters. Operational arguments are search and representation parameters. Search parameters (c.f. §11 and 12) prescribe a search mode, exhaustive or some kind of directed search, and the required additional parameters, for instance goodness criteria. Representation parameters assign a CAAD system and possibly related prescriptions for graphic representation.

§8 Grammar and operational devices are the main sections of a design reasoning system, +G, that can be used as a command system for CAAD systems for design representation. Design results from a constant exchange between design performance in practice and observation and design competence. +G is design competence that supports such an exchange in computer aided form for users without special formal knowledge. Provided a shape definition macro is given that controls the connection between shapes and shape rules, a shape library, possibly related to existing component libraries, can be extended by such users. The same applies to grammars, provided a body of shapes and shape rules is given that allows experimentation outside existing grammars. The development of operational devices and, presumably, shape rules is the domain of formal modeling. +G can only result from a permanent exchange between design in practice and theory, AI research and software development.

+G = (Grammar Knowledge, Operational Devices)
GK = (Design Grammars, shape library, shape rule library)
DG = (Compulsory Shape Categories, Selectable ShapeCategories, Design Rules)
OD = (devices for shape rule concatenation, search, user definitions and the transcription of data resulting from design reasoning into CAAD commands)

§9 We proceed with some observation of design performance, searching for a grammar frame which is applicable to divergent phenomenae. All expressions in the shape rules (5) to (28) are nonterminal shape categories, with the exception of AFMI in (16), MDI in (20) and XI in (21) which are preterminals. (4) and all grammar rules below, except for (6), (7), (18) and (22) to (28), are in principle tested by computer implementation.

Imagine n settlers, each of them building a house such that a courtyard results between the new and the previous house. Such courtyards are called spatium and such houses extensio. Spatium, in philosophical terms, is more than a measurable extensio, it is the space of movement, the unknown, the potential etc. An array of spatium’s can be resumed as:

(5) SAD → E S SAD, where SAD is Spatium Array Design, an expression of a SAG (Spatium Array Grammar), S Spatium and E Extensio. If no n is specified SAD is endless. Otherwise it is recalled n times with the brief of a settler in life or symbolically represented form.
§10 Settler_1 has to look if some object in a context given can substitute a previous house (a). If none is found, he must build more than one houses to get a courtyard, a spatium, between them (b), or a house like a shell (like Fiesler's *Endless House* etc.), that is a more or less hollow extensio (c). For (a) and (c) we can write for E_1 in (5):

(6) E_1 → R, where R is Read natural context (c.f. §16) and
(7) E_1 → HE, where HE is Hollow Extensio.

If we ignore the aspect of being *head units* in the Salute context, the bottom variants of figure 1 could be produced by a settler of type (b), with the result of n = 2 for (5) and a not empty S for n = 2.

§11 The wings of figure 1 result, among other, from:

(8) E_1 → MBV SAD, where MBV is Make Brief Variants. This means that a number of brief variants is generated first - topological variants of the kind demonstrated by figure 1 - and SAD is restarted with one or more of these variants, depending on the search mode prescribed as a parameter of a design generator. In case exhaustive design is prescribed we get

(9) E_1 → NBV SAD, where NBV is Next Brief Variant and SAD is recalled until all brief variants are addressed.

§12 In case of GA (Genetic Algorithm) directed search we get:

(10) E_1 → RSAD SAD, where RSAD is a Random SAD for a random brief variant and random parameters. This form generates an initial "population" of a specified length with which SAD is restarted as

(11) SAD → IIP, where IIP is Improve Initial Population (an algorithm based on (Kundu, 1996)). The variants with hatched regions (spatium's) in figure 1 result from such an improvement that couples, to put it simplified, brief variants with right origins, O_R's. O_R shifts in a field prescribed by (14) and distance- and angle-parameters set as search parameters with the definition of the design generator. Other search parameters are goodness criteria, here the spatium surface and 2 distance values.

§13 With the exception of E_1 as discussed in §10 any E_i in (5) is of the form

(12) E_i → (2) SAD*, where SAD* is a toploop call of SAD discussed in §14, or
(13) E_i → (3) SAD*.

The wings of the variants of figure 1 result from (15). Here FA in (3) is of the form of

(14) FA = FOO, where FOO is Find Origin Outside any extensio in the current register. FOO works on the arguments of a reference position and a brief. In §14 and §18 a 'Find Origin Inside' is raised.

The upper right corners, O_R, of the variants of figure 1 are such origins, defined in relation to a reference origin, here O_l of a wing already given and an angle, \(\alpha_R\). They result from a procedure which is perhaps typical for Scharoun: to organise wings around a central space, simulated as *Spatium* here, which Scharoun called "Raum der Mitte". Though organic architecture and Scharoun design are not necessarily oblique-angled, such wings are generally swung in relation to eachother. Perhaps it is only the way of swinging that is Scharoun specific.

GS in (3) in (13) can result from another SAD, another SAG design generator, or from another grammar generating an extensio. The wings of figure 1 result from a Furniture Mat Array (FMA) grammar:

(15) GS → FMA
(16) FMA → AFM_1 ... AFM_n, where AFM is Attach Furniture Mat, a rule of type (2).
The “kindergarten grammar” underlying the Fröbel (and Lego) building blocks (Stiny, 1980b) is an example of an array system that allows the attachment of blocks to blocks (labelled shapes) in various ways. We can call such rules attach rules. In that view they are not rules of Spatium Array Grammars, but of Array Grammars. They organise the array of units attached to each other with no space in-between. FMA is another array grammar. It evaluates

- attach points resulting from the distribution of objects (for example matjes, furniture mats, a Dutch convention) in boundaries and

- accessibility restrictions defined as parameters (slots) of shape categories.

Figure 1: east part of Hans Scharoun’s ‘Salute’ design (top left); SAG generated variants for a fixed boundary (top centre, Autocad representation); SAG generated variants for head unit (bitmap representation): initial FMA (top right), rejected member of initial population for GA directed search (bottom left), GA improved population (bottom right)

§14 Figure 1 top right, shows an initial FMA-generated configuration for left and right wings with overlaps. In both (12) and (13) the top SAD operator is reactivated in the form of SAD* in order to detect and to adjust such destructive results of new Ei’s. The *-index says that the mechanism is essentially of the same form as (5), but does not create new extensions or spatium’s, but tests and probably re-designs given spatios and extensios by operators E* and S*. As a design toploop, the SAD* operator runs back through a design which means it takes extensios and spatios in reversed order. It is restarted as long as a previous SAD* has been successful, that is a destruction has been encountered and the design register edited:

(17) SAD* → S* E* SAD*

Efficiency oriented architects avoid such toploops. But they can be inevitable. For economic reasons and similar, or - as it appears to be in the work of
architects like Aalto or Scharoun - for the mere sake of design. For Aalto, as will be demonstrated in §18, we can even say that such toploops were “programmed” by another option for ‘Find Allocation’ in order to trigger solutions that cannot be achieved otherwise:

(18) \( FA = \text{FOI} \), where FOI is Find Origin Inside an extensio or part of it.

A design toploop is also called as long as a design contains only preliminarily designed sections. A preliminary elaboration can be prescribed by external conditions, such as stages of realisation. Or it can result from the logic of design itself too: certain solutions emerge only gradually in step by step refined design contexts. A preliminary solution is entered into the register of an evolving design associated with temporary information that prestructures later recursion and refinement.

Finally, design toploops can also serve as global tests without a particular reason. Certain judgements are only possible when a design is visible as a whole. Such toploops are important for architects like Scharoun, often to the lament of customers. “La forme cout cher,” another French writer has declared.

§15 §15 S in (5) proceeds as follows:

(19) \( S \rightarrow D F \), where F is a Fill operator for a fill-brief,
(20) \( D \rightarrow \text{MD}_1 \ldots \text{MD}_n \), where MD is Make Delimitation, a rule of type (3), and
(21) \( F \rightarrow X_1 \ldots X_n \text{TS} \), where \( X_i \) is an object to be allocated in S and TS Test Spatium

The spatiums of figure 1 result from two types of very simple MD rules. Figure 7, mentioned in §4, shows a more elaborated kind of an ‘attached’ delimitation or boundary. Shape rule evaluation can be prepared by auxiliary shape rule functions, possibly to be applied repeatedly by shape rules with different shape arguments. For example, such an auxiliary function can elaborate a set of reference points for the tentative allocation of various shape categories in a boundary. (21) can work with such an auxiliary function. Auxiliary functions can work on parameters, a resolution degree for example for the distribution of reference points, which must be set globally for special shape rule types for a whole state space of shape rules to be evaluated.

Spatium is parasitic: it results from extensio. (19) to (21) are rather tools for search - whatever that may be. No design toploop, SAD*, is called at the end of (19), because only an extensio can be destructive.

§16 Only (a) of the three settler types in §10 performs context sensitive design, starting with (6) as initial operation, \( E_1 \). Context sensitive design is important for entire schools of architecture. Treib (Aalto, 1998) says:

As physical compositions, Aalto’s non-urban buildings tended to divide into two basic groups: concave or convex. The concave schemes reiterated the contours of fissures and valleys- The convex schemes complemented or reinforced rising landforms. And for those sites that lacked potent natural features Aalto constructed his own architectural landscape.

(22) read_context(R)
    
    construct_context(CSC) complete_context(CC)
We can resume this as a read rule, \( R \), read-context, in tree form. This tree says that a fan of leitbilder is inquired, if some leitbild can be applied in order to elaborate a design as a completion of a natural context. If no leitbild is found to be applicable, or if the initial context register is empty (lacks natural features ...), a first extensio, \( E_1 \), is constructed as an artificial initial context (CSC) and the read-context operator is restarted with the result as initial context register.

Figure 11 shows a beautiful example of a design resulting from a convex context reading. The building applies vertically the shape of the hill it stands on. An example of the same species shows figure 12. Here the shape of the high rise part follows the isohypses of the site. This horizontal approach is continued in the third dimension in form of a back wall which is inclined in the direction of the slope.

Figure 4 and 5 testimony a concave context reading: a mould. The programme is subdivided in various sections which in case of figure 4 are more or less embedded into the mould figure read: an irregular open arc of the leitbild pool/trunk/cloud/etc. Figure 5 is different. It demonstrates that Aalto has also been experimenting with another approach which is rather a contrast- than an apply-approach. In figure 5, which is in line with the executed design, the museum is allocated in essentially only one single block at the bottom of the mould and isolated from the rising slopes that form the mould. It is interesting to observe another contrast case: figure 6. Here the contrasting block is allocated not at the bottom but at the top of a mould context.

We can resume these observations as two options for a Complete-Context rule, CC in (22):

(23) CC \( \rightarrow \) IL\(_{1,n}\) AL, \( \rightarrow \) IL\(_{1,n}\) FCS,
where IL is Insert Leitbild and AL Apply Leitbild and where FCS is Find Contrast Structure.

These rules say that a leitbild first is inserted in the context it is read from in order to serve as a reference for a contrast solution, or to be applied to shape a design in one way or another. Figure 9 and 10 can be read as another illustration of the latter. Figure 9 shows a broken line, an obtuse angle, inserted in a site-plan. This could be the result of a contrast context reading. This figure is applied then for the topological distribution of the main brief sections. The result of this first extensio, \( E_1 \), is not yet a real one, but an auxiliary pattern that prestructures the allocation of extensio’s, \( E_i \)'s, to follow. In this way figure 10 can be read.

We can also imagine that in a first, probably not even graphically represented step the right part of the obtuse angle was entirely or partially filled by a single global, but already approximately dimensioned grid - or rather a kind of terraced grid, following the slope of the site. If this is justified, we have an example of a preliminary extensio, to be subdivided and refined by later transformations.

Figure 10 also demonstrates how a spatium can emerge in various ways between externsio’s. As a simple array order with no spaces between its parts,
the lower left part, however, appears rather to be an exception from that. In terms of (5) this can be expressed by empty spatiums between externsio’s, which do not block the top mechanism. At closer inspection, this part turns out to be garden architecture.

§17 A peculiar approach to allocate shapes or design sections in a design context belongs to the most salient features of Aalto’s legacy and meets what has been marked as collage or heterotopic.

The top right sketch of figure 3 can be read as a kind of head-allocation, the larger sketch to its left as a kind of angle-allocation: the auditorium is embedded into an angle region: the spatium region resulting from a concave context leitbild, as discussed above. This region could have been empty such that no overlap with some unit already given is possible, or not. The bottom left sketch of figure 2 and, perhaps, the larger right sketch too show that convex angle-allocations are also possible. The lecture hall of figure 8 we can read as an instance of a head-allocation or a convex angle-allocation.

§18 The top left sketch of figure 2 demonstrates, perhaps better than other examples, that such forms of allocation are easily destructive. Here we have a lateral allocation such that the not rectangular library section is allocated at the broadside of the rectangular reference section with the result of an overlap, though not a very large one. Such overlaps are detected by the SAD* toploop called in (12) and (13) after any extensio. The general form of an extensio to be tested and possibly edited, E* in (17), is:

\[ (25) \ E^* \rightarrow \text{FOL EE,} \quad \text{where FOL is Find OverLap, a test function that recalls itself as long as no overlap (no intersection) is found, and where EE is Edit Extensio.} \]

In the case of the top left sketch of figure 2 there is no previous spatium S, and the first extensio given, the rectangular part of the sketch, is found destructed, as E\text{found}, overlapped by the new not rectangular shaped extensio E\text{current}. There are two possibilities to reconcile such destructions, two options for EE in (25): to edit the current, overlapping extensio (26), or the previous, overlapped one (27). Resulting from the SAD* toploop, both can be edited too. The transformation of the initial FMA layout of figure 1, top right, into not overlapping configurations of the kind of the bottom row of figure 1 can be an example of the first of these alternatives. It can be resumed as:

\[ (26) \ EE \rightarrow \text{RAU EEC,} \quad \text{where RAU is ReAllocate Unit, the overlapping unit of FOL, and EEC Edit Extensio Current.} \]

The top left sketch of figure 2 can be interpreted in terms of the second of the two alternatives, a second EE option resulting in a spatium cut out from a destructed extensio.

\[ (27) \ EE \rightarrow \text{EEF S; where EEF is Edit Extensio Found as:} \]
\[ (28) \ EEF \rightarrow \text{EE}_{\text{found}}\text{EE}_{\text{current-extended EE}_{\text{destructed}},} \quad \text{which means subtract the not rectangular part of the sketch, E}_{\text{current}}, in Extended shape from the rectangular part, E}_{\text{found-}, \text{and Edit the resulting Extensio}_{\text{destructed}.} \]

References


Figure 2 - 11 are from (Aalto, 1998), with the exception of figure 6 which is from (Fleig, 1997).