# An Algebraic Approach to Implementing a Shape Grammar Interpreter

Stouffs, Rudi

**Publication date**
2016

**Document Version**
Final published version

**Published in**
Proceedings of the 34th eCAADe Conference

**Citation (APA)**
Stouffs, R. (2016). An Algebraic Approach to Implementing a Shape Grammar Interpreter. In A. Herneoja, T. Österlund, & P. Markkanen (Eds.), *Proceedings of the 34th eCAADe Conference: Complexity & Simplicity* (Vol. 2, pp. 329-338). eCAADe.

# An Algebraic Approach to Implementing a Shape Grammar Interpreter

*Rudi Stouffs*[1]
[1]*National University of Singapore; Delft University of Technology*
[1]*stouffs@nus.edu.sg*

*Shape grammars come in a variety of forms. Algebras of shapes have been defined for spatial elements of different kinds, as well as for shapes augmented with varying attributes, allowing for grammar forms to be expressed in terms of a direct product of basic algebras. This algebraic approach is extended here to the algebraic derivation of combinations of basic shape algebras with attribute algebras. This algebraic abstraction at the same time serves as a procedural abstraction, giving insights into the modular implementation of a general shape grammar interpreter for different grammar forms.*

**Keywords:** *shape grammars, shape algebras, parallel grammars, compound shapes, implementation*

## INTRODUCTION

A shape grammar is a formal rewriting system for producing languages of shapes (Stiny and Gips 1972; Stiny 1980). A shape grammar is typically specified to consist of a set of productions, or shape rules, operating over a vocabulary of (terminal) spatial elements and a vocabulary of (non-terminal) symbols or markers, e.g., labels, and to include an initial shape as the starting point in the productive (generative) process (Stiny 1980; Yue and Krishnamurti 2013). Then, a shape is defined as any composition of spatial elements and, optionally, symbols from the respective vocabularies. A shape rule is commonly expressed in the form $a \rightarrow b$, with both $a$ and $b$ constituting shapes, such that the application of the rule to a shape $s$ under a (similarity) transformation $t$ yields the shape $s - t(a) + t(b)$, with the condition that $t(a) \leq s$. The language defined by a shape grammar is the set of shapes generated by the grammar that do not contain any (non-terminal) symbols.

Grammar formalisms for design come in a large variety, requiring different representations of the entities being generated, and different interpretative mechanisms for this generation. Shape grammars also come in a variety of forms, even if less broadly. Most examples of shape grammars rely on labeled shapes, a combination of line segments and labeled points (in two dimensions) (Stiny 1980). Stiny (1992) proposes numeric weights as attributes to denote line thicknesses or surface tones. Knight (1989; 1993) considers a variety of qualitative aspects of design, such as color, as shape attributes. Stiny (1981) also proposes to augment a shape grammar with a description function in order to enable the construction of (intended) descriptions of designs.

Implementing a shape grammar interpreter requires implementing the part relationship for shapes—with or without attributes—, the operations of sum and difference on shapes, and solving the matching problem, that is, identifying under which transformation a

rule may apply to a given shape. Beirão (2012, 228-236) offers a survey of (implementations of) shape grammar interpreters so far, and must conclude that they have common limitations. Many of them compute only on two-dimensional shapes; and most do not apply subshape detection and therefore do not support emergence. Also, "very few shape grammar interpreters allow for the implementation of rules operating with symbols" (Beirão 2012, 235), never mind other attributes, or a description function.

In this paper, we will address the problem of developing an implementation of a shape grammar interpreter supporting varying shape grammar formalisms, by focusing on the implementation of parallel and compound shape grammars. We will review the literature on parallel and compound shape grammars and propose an algebraic treatment facilitating a modular approach, based on a similarity between algebraic abstraction and procedural abstraction (Frank 1999).

## PARALLEL AND COMPOUND SHAPE GRAMMARS

Originally, a parallel shape grammar was defined by Stiny (1975, 37) and Gips (1975, 7) as a shape grammar intended to be used in the parallel generation of shapes, that is, "whenever a shape rule is used, it is applied simultaneously to every part of the shape to which it is applicable" (Gips 1975, 7; Stiny 1975, 37). This is in contrast to the more common serial application of shape rules, where at each step of the generation a shape rule is applied to only one part of a shape. However, more recently, the term parallel (shape) grammar has been adopted in the context of parallel computations on multiple descriptions.

Li (1999) defines a parallel grammar as a grammar operating on different descriptions with the objective to resolve parametric dependencies. For example, consider a shape grammar generating a plan where the number of rooms is depended on the selection of rules applied. In this process, it is almost impossible to constrain the boundaries of the plan at the same time, as this would require the room sizes to be made dependent on the total number of rooms, which is only known at the end of the production process. Instead, Li (1999) suggests adopting (at least) two descriptions, the first one a diagram with the number of rooms as independent parameter(s), and the second one the plan with the room sizes as dependent parameters (of the number of rooms). By staging the production of the diagram before the production of the plan, the assignment of values to the dependent parameters can be postponed until after the assignment of values to the independent parameters.

Duarte (2001) defines a parallel grammar as separating different representations or aspects of a design into different computations that interact with each other. Specifically, Duarte (2001; 2005) considers a discursive (parallel) grammar incorporating a shape grammar and a (textual) description grammar—as well as a set of heuristics, where the latter is intended to constrain the rules that are applicable at each step of the design generation. While the shape grammar operates on shapes and the description grammar on textual (including numeric) descriptions, their rules are commonly coupled, with the description rule part constraining the shape rule part. This combination of a shape grammar and a description grammar follows Stiny's (1981) definition of a description function to augment a shape grammar in order to construct design descriptions. Where Stiny (1981) considers a description function as made up of functions, with each function assigned to a shape rule and computing in parallel to the shape rule, Duarte instead denotes the functions as description rules and the description function as a description grammar. Otherwise, these operate in exactly the same way and with the same intention. In fact, although Stiny nowhere adopts the term parallel grammar in this sense, Knight (1999; 2003a) explicitly attributes the concept to Stiny (1981). Knight (2003a) also offers a definition of a parallel grammar as "a network of two or more grammars that operate simultaneously."

Admittedly, Li (1999) and Knight's (2003a; also,

Duarte 2001) interpretation are not unequivocal. Following Stiny's (1990) definition of a design as "an element in an *n*-ary relation among drawings, other kinds of descriptions, and correlative devices as needed," Li (2001; 2004) considers seven drawings (from plan diagram to plan, section and elevation) and nine descriptions (specifying measures of width, depth, and height, among others), in his specification of a shape grammar for (teaching) the architectural style of the *Yingzao fashi*. However, he considers only four grammar components to define his parallel grammar, corresponding to four stages in the production (Li 1999). However, in Knight's (2003a) parallel grammar interpretation of Stiny's (1981) example, the two grammar components of the parallel grammar—the shape grammar and the description function—apply hand in hand: "the rules of a parallel grammar may be linked so that the application of a rule in one grammar triggers the application of one or more rules in other grammars" (Knight 2003a). In fact, Knight (2003a) suggests the same interpretation for Li's (2001; 2004) grammar, with each drawing and description specifying a component grammar in the parallel grammar, for each stage. Duarte (2001) takes a similar position when describing his discursive grammar applied to the houses designed by the architect Alvaro Siza at Malagueira. Even though, strictly speaking, he only refers to two grammars—a shape grammar and a description grammar—, he specifically acknowledges that both grammars include several sub-grammars. "These sub-grammars correspond to viewpoints in the shape grammar (e.g. first floor plan), and to features in the description grammar (e.g. morphology)" (Duarte 2001). Viewpoints define separate drawings (sketches, plans, elevation, envelope, etc.) and features individual descriptions.

When the rules of a parallel grammar are linked, the linked rules can be expressed as one compound rule (Knight 2003a). Therefore, some authors consider compound grammars as an alternative term to parallel grammars, though others adopt the term compound grammars also to denote compositions of grammars that do not operate in parallel. For example, Beirão (2012) uses the term compound grammar to denote a composition of several discursive grammars, where each discursive grammar formalizes a so-called Urban Induction Pattern (UIP), encoding a typical urban design operation or design move. Knight (2004), instead, suggests a distinction between synchronized and a-synchronized parallel grammars, where the latter allows for sequential production stages as proposed by Li (1999; 2001).

## SHAPE ALGEBRAS

While Stiny avoids the term parallel grammar when referring to parallel computations, he does emphasize parallel computations in multiple algebras. For example, when a rule applies to a shape composed of points and line segments, though the rule may require both one or more points and one or more line segments to be present within a prescribed spatial relationship, the rule computes with points and line segments in parallel. The overall shape rule computation actually combines two computations—one with shapes of line segments and one with shapes of points—"that are carried out in parallel and influence one another mutually" (Stiny 1990).

Points and line segments, and by extension other spatial elements, can be considered to adhere to an algebra (specifically, a generalized Boolean algebra (Krstic 1999)), that is ordered by a part relation ('≤') and closed under the operations of sum ('+'), product ('·'), and difference ('−'), as well as relevant transformations. For example, points may belong to the algebra $U_{02}$ and line segments to the algebra $U_{12}$ (in two dimensions) (Stiny 1992); $U_{ij}$ denotes the algebra of spatial elements of dimension *i*, e.g., 0 for points, 1 for line segments, 2 for plane segments, embedded in a space with dimension *j*, e.g., 1-D, 2-D, 3-D. Stiny (1992) extends the notion of algebras to shapes with attributes: labeled points belong to the algebra $V_{02}$, while weighted line segments belong to the algebra $W_{12}$. Then, shapes of line segments and labeled points can be said to belong to an algebra that is the direct product of the algebras $U_{12}$ and

$V_{02}, U_{12} \times V_{02}$. Consequently, a shape rule applying to shapes of line segments and labeled points can be considered to combine two shape rules, one in $U_{12}$ applying to line segments and one in $V_{02}$ applying to labeled points.

Any selection of shape algebras, including labeled and weighted shape algebras, can be combined using the direct product into a compound algebra of compound shapes that are made up of a mix of various spatial elements, and optionally augmented with labels or weights. Chase (1999) notes that "this is common in maps, as map features may be distinguished by different element types (for example, lines representing roads, points representing cities), or labels used to distinguish elements with the same basic geometry but different semantics (for example, lines can represent roads and rivers)." But, examples in architectural representation abound as well. Compound shapes may be expressed in unions of the sets that form shapes from different algebras, with the understanding that basic and augmented spatial elements only interact if they are of the same kind (same dimension and attribute kind, if any), and are independent otherwise (Stiny 1992).

This notion of compound shapes, as resulting from compound algebras defined by the direct product of basic shape algebras, is conducive to a modular (or procedural) implementation of a shape grammar interpreter (e.g., Frank 1999). Each basic shape algebra can define a single module (or a class in an object-oriented programming paradigm), and modules can be combined to define compound algebras, facilitating a variety of shape grammar formalisms.

The algebra's *signature* specifies the operations of the algebra, at a minimum, the operations of sum ('+'), product ('·'), and difference ('−'). The allowable transformations can be considered external to the algebra, as part of the signature or as part of the algebra's *carrier*, i.e., the set of elements of the algebra (Krstic 1999; 2012). The part relation ('≤') is not an operation of the algebra but can be expressed in terms of the operation of product: $t(a) \leq s \Leftrightarrow t(a) \cdot s = t(a)$. From a modular implementation point of view, the algebra's signature contributes to the module's interface (the class methods in object-oriented programming), but the interface can be extended to include, among others, the part relation. Nevertheless, having a (even partially) uniform interface, i.e., sharing the same class methods, greatly eases the implementation of a general shape grammar interpreter.

Unfortunately, as Knight (2003b) acknowledges, "the algebras that designers use, informally or formally, are many." Beyond labels and weights, shapes can be augmented with attributes of any kind: "aesthetic, formal, functional, structural, and so on. For example, points can have diameters, lines can have thicknesses, planes can have colors, solids can have materials" (Knight 2003b). "The only condition is that the operators of any shape algebra are defined on all its elementary objects, are recursively applicable, and are closed" (Yue and Krishnamurti 2013). Fortunately, the notion of an algebra as derived from existing algebras can be extended to augmented shapes, e.g., of labeled points (Stouffs 2008). Defining an algebra for labels is straightforward. Similar to points, a label can be said to be part of another label only if these are identical. Then, the operations of sum, product, and difference correspond to the set operations of union, intersection and difference. Labels do not exhibit any allowable transformations, unless we consider case transformations. However, the operation of direct product on algebras will not support an attribute behavior, and it is not straightforward to consider an alternative operation on algebras. Krstic (1999) offers an unintentional hint.

## ALGEBRAIC ABSTRACTIONS

Krstic (1999) notes the difference between an algebra of (maximal) spatial elements and an algebra of shapes. The former is a partial algebra as the operations of sum, product, and difference are not closed. The sum of two spatial elements is a spatial element only if the two elements overlap or are both part of another spatial element and have boundaries that overlap. In general, the operations of sum, product, and difference on spatial elements are only defined

if the spatial elements exist in the same subspace, where the dimension of the subspace equals the dimension of the spatial elements. For example, two line segments must be part of the same infinite line, two plane segments of the same infinite plane, and two volumes of the same 3D hyperplane. This subspace is denoted the *carrier* of the spatial element, but in order to avoid any confusion with the carrier of an algebra, we will instead refer to the *co-descriptor* of the subspace. Then, the operations of sum, product, and difference on spatial elements are defined only if the spatial elements have the same co-descriptor, that is, they are *co-equal*. Note from above that, in and of itself, this is not a sufficient condition for the operations to be closed.

Having established the difference, Krstic (1999) goes on to focus solely on shape algebras. However, it is possible to derive shape algebras from partial algebras of spatial or other elements in a general way. First, let us assume a two-sorted partial algebra with carrier {*A*, $\mathcal{P}(A)$} and signature including the operations of *combine*, *common* and *complement* on members of *A*. We consider a two-sorted algebra because we want the operations of combine, common, and complement to extend upon the respective operations of sum, product, and difference, and at the same time be closed for co-equal (spatial) elements. For example, the combine of two co-equal spatial elements that do not have overlapping boundaries will be the set of the two spatial elements. Thus, each of the operations of combine, common, and complement takes as argument two elements of *A* and return an element of $\mathcal{P}(A)$, the set of all subsets of *A*.

## Deriving Algebras from Two-sorted Partial Algebras

To derive a shape algebra from this two-sorted partial algebra, we need to distinguish the desired behavior. Each behavior will result in a different derivation. Fortunately, we can reuse behaviors for different kinds of spatial or other elements. The simplest behavior is a discrete behavior, applying to both points and labels.

An algebra with carrier $\mathcal{P}(A)$ and signature including sum ('+'), product ('·'), difference ('−'), and reduce ('r') can be defined for a discrete behavior as follows:

$$\begin{aligned} &\forall\, X, Y \in \mathcal{P}(A) \Rightarrow \\ &\quad X + Y = X \cup Y \\ &\quad X \cdot Y \;= X \cap Y \\ &\quad X - Y = X \setminus Y \\ &\quad \mathsf{r}(X) = X \end{aligned} \tag{1}$$

In a discrete behavior, the operations of sum, product, and difference correspond to the normal set operations of union, intersection and difference. The reduce operation reduces any set to a set of maximal elements. Under a discrete behavior, any set is maximal because any duplicates are automatically removed. The algebras $U_0$ (shapes of points; we omit the dimension of the embedding space for simplicity) and $L$, sets of labels, can be defined in this way. Note that descriptions, from an algebraic point of view, behave exactly as labels and, thus, the algebra $D$ of sets of descriptions can be defined in this way as well.

Before we address other spatial elements, let us first consider a behavior for weights (e.g., line thicknesses or surface tones), as is apparent from drawings on paper—a single line drawn multiple times, each time with a different thickness, appears as if it were drawn once with the largest thickness, even though it assumes the same line with other thicknesses (Stiny 1992).

An algebra with carrier $\mathcal{P}_1(A)$, the set of all singleton subsets of *A*, and signature including sum ('+'), product ('·'), difference ('−'), and reduce ('r') can be defined for an ordinal behavior, applying to weights, in terms of the two-sorted partial algebra with carrier {*A*, $\mathcal{P}(A)$} and signature including the operations of combine, common and complement, as follows:

$$\begin{aligned} &\forall\, \{x\}, \{y\} \in \mathcal{P}_1(A) \Rightarrow \\ &\quad \{x\} + \{y\} = \mathsf{combine}(x, y) \\ &\quad \{x\} \cdot \{y\} = \mathsf{common}(x, y) \\ &\quad \{x\} - \{y\} = \mathsf{complement}(x, y) \\ &\quad \mathsf{r}(\{x\}) = \{x\} \end{aligned} \tag{2}$$

For weights, we know that the result of the operations of combine, common, and complement on

two singleton weights is always a singleton weight. We use this knowledge to define the operations of sum, product, and difference in terms of the operations of combine, common, and complement. Again, the reduce operation results in the argument (singleton) set itself. The algebra $N$ of singletons of numeric weights can be defined in this way.

Deriving a shape algebra for spatial elements other than points from a two-sorted partial algebra is a little bit more complicated because of the need to consider co-equal shapes. We take a two-step approach. First, we derive a sub-algebra for co-equal shapes of spatial elements, next we define a shape algebra for a single type of spatial elements from this sub-algebra.

A sub-algebra with carrier $\mathcal{P}(A)$ and signature including co-combine, co-common, co-complement, and co-reduce can be defined for an areal behavior as follows:

$$
\begin{array}{l}
\forall\, X,\, Y \in \mathcal{P}(A) : \\
\quad \forall\, x \in X,\, \forall\, y \in Y, \mathsf{co}(x) = \mathsf{co}(y) \Rightarrow \\
\mathsf{co\text{-}combine}(X, Y) = \\
\quad \mathsf{construct}\left( \bigcup \left\{ \begin{array}{c} \mathsf{outside}(\mathsf{b}(X), Y) \\ \mathsf{outside}(\mathsf{b}(Y), X) \\ \mathsf{same\text{-}side}(\mathsf{b}(X), \mathsf{b}(Y)) \end{array} \right. \right) \\
\mathsf{co\text{-}common}(X, Y) = \\
\quad \mathsf{construct}\left( \bigcup \left\{ \begin{array}{c} \mathsf{outside}(\mathsf{b}(X), Y) \\ \mathsf{inside}(\mathsf{b}(Y), X) \\ \mathsf{same\text{-}side}(\mathsf{b}(X), \mathsf{b}(Y)) \end{array} \right. \right) \\
\mathsf{co\text{-}compliment}(X, Y) = \\
\quad \mathsf{construct}\left( \bigcup \left\{ \begin{array}{c} \mathsf{outside}(\mathsf{b}(X), Y) \\ \mathsf{inside}(\mathsf{b}(Y), X) \\ \mathsf{opposite\text{-}side}(\mathsf{b}(X), \mathsf{b}(Y)) \end{array} \right. \right) \\
\mathsf{co\text{-}reduce}(X) = \\
\quad \left\{ \begin{array}{ll} \mathsf{co\text{-}combine}(\{x\}, \mathsf{co\text{-}reduce}(X \setminus x)) & \exists\, x \in X \\ \emptyset & \mathsf{otherwise} \end{array} \right.
\end{array}
\tag{3}
$$

The operations of co-combine, co-common, co-complement, and co-reduce only apply to co-equal shapes. Instead of comparing the shapes for co-equality, it is checked that all spatial elements have the same co-descriptor ('co'). Then, the operations can be expressed in terms of the boundaries ('b') of each co-equal shape (Krishnamurti and Stouffs 2004; Stouffs and Krishnamurti 2006), here termed *co-shape*. Specifically, 'outside(b(*X*), *Y*)' returns the collection of boundaries of co-shape *X* that lie outside of co-shape *Y*. Similarly, 'inside(b(*Y*), *X*)' returns the collection of boundaries of co-shape *Y* that lie outside of co-shape *X*. 'same-side(b(*X*), b(*Y*))' denotes the collection of boundaries of both co-shapes X and Y where the interiors of the respective co-shapes lie on the same side of the boundary, and 'opposite-side(b(*X*), b(*Y*))' the collection of boundaries of both co-shapes X and Y where the interiors of the respective co-shapes lie on opposite sides of the boundary. Then, the boundary of the co-shape resulting from the co-combine operation is formed by the 'outside(b(*X*), *Y*)', 'outside(b(*Y*), *X*)', and 'same-side(b(*X*), b(*Y*))' collections, and the co-shape can be constructed from the union of these collections. The co-common and co-complement operations are similarly defined. In the case of the co-reduce operation, each spatial element in the co-shape is co-combined with the co-reduced remainder of the co-shape. From an implementation point of view, this recursive definition of co-reduce may not be the most efficient; actually, the same can be said about the other operations—the classification of boundary segments with respect to another co-shape can be computed once for all of the classes inside, outside, same-side and opposite-side. Obviously, these definitions only serve as abstractions of the actual procedures, we refer to Stouffs and Krishnamurti (2006) for actual, and efficient, algorithms.

Then, a shape algebra with carrier $\mathcal{P}(A)$ and signature including sum ('+'), product ('·'), difference ('−'), and reduce ('r') can be defined for an areal behavior, applying to line segments, plane segments and volumes, in terms of the sub-algebra with carrier $\mathcal{P}(A)$ and signature co-combine, co-common, co-complement, and co-reduce, as follows:

$$
\begin{array}{l}
\forall X,\, Y \in \mathcal{P}(A) \Rightarrow \\
\quad X + Y = \mathsf{r}(X \cup Y) \\
\quad X \cdot Y = \bigcup_c \mathsf{co\text{-}common}\left( \begin{array}{c} \{x \in X : \mathsf{co}(x) = c\}, \\ \{y \in Y : \mathsf{co}(y) = c\} \end{array} \right) \\
\quad X - Y = \\
\qquad \cup \left\{ \begin{array}{l} \bigcup_c \{x \in X : \mathsf{co}(x) = c \wedge \neg \exists y \in Y : \mathsf{co}(y) = c\} \\ \bigcup_c \mathsf{co\text{-}complement}\left( \begin{array}{c} \{x \in X : \mathsf{co}(x) = c\}, \\ \{y \in Y : \mathsf{co}(y) = c\} \end{array} \right) \end{array} \right. \\
\quad \mathsf{r}(X) = \bigcup_c \mathsf{co\text{-}reduce}(\{x \in X : \mathsf{co}(x) = c\})
\end{array}
\tag{4}
$$

The operations of product, difference and reduce are expressed directly in terms of the operations of

co-common, co-complement and co-reduce on the respective co-shapes. In the case of complement, co-shapes from *X* for which there exists no (co-equal) co-shape in *Y* also form part of the result. A similar approach can be taken for the operation of sum, however, for simplicity, we prefer to express the operation of sum in terms of the operation of reduce on the combined sets of spatial elements. Note that the algebras $U_1$ (shapes of line segments), $U_2$ (shapes of plane segments) and $U_3$ (shapes of volumes) can all be defined in this way.

We should note that while an areal behavior applies to shapes of line segments as well, from an implementation point of view, it would be more efficient to define an interval behavior for shapes of line segments. Additionally, while these behaviors cover shapes of different kinds of spatial elements, and sets of labels and singletons of weights, other behaviors can be identified to apply to other kinds of attributes, for example, for material rankings (Knight 1993). Instead, we will now continue to derive compositions of algebras under the direct product.

### *Deriving Algebras with the Direct Product*

The direct product applies to all algebras that share the same signature, specifically, the algebras $U_0$, $U_1$, $U_2$, $U_3$, $L$, $D$, and $N$, we have previously defined. We will address some implications of this in the discussion below. Here, we define a shape algebra with carrier $\mathcal{P}(A) \times \mathcal{P}(B)$ and signature including sum ('+'), product ('·'), difference ('−'), and reduce ('r') in terms of the shape algebras with carriers $\mathcal{P}(A)$ and $\mathcal{P}(B)$ and identical signatures, as follows:

$$\begin{aligned}
&\forall\,(A_1, B_1),\ (A_2, B_2) \in \mathcal{P}(A) \times \mathcal{P}(B) \Rightarrow \\
&\quad (A_1, B_1) + (A_2, B_2) = (A_1 + A_2, B_1 + B_2) \\
&\quad (A_1, B_1) \cdot (A_2, B_2) = (A_1 \cdot A_2, B_1 \cdot B_2) \\
&\quad (A_1, B_1) - (A_2, B_2) = (A_1 - A_2, B_1 - B_2) \\
&\quad \mathsf{r}(A_1, B_1) = (\mathsf{r}(A_1), \mathsf{r}(B_1))
\end{aligned} \tag{5}$$

An algebra of shapes of points and line segments, $U_0 \times U_1$, and an algebra of shapes of line segments and sets of descriptions, $U_1 \times D$, among others, can be defined in this way.

### *Deriving Augmented Shape Algebras*

Next, we can tackle the issue of an algebra for augmented shapes. Rather than referring to a shape algebra and an attribute algebra (e.g., $L$ or $N$), we will instead refer to the partial algebra for the spatial elements, just as we define the shape algebra from the partial algebra of its spatial elements. The behavior of an augmented shape algebra, after all, mimics the behavior of the underlying shape algebra, with a few differences. For example, consider two overlapping line segments. Without attributes, these combine. With attributes, they combine only if they share the same attributes or, otherwise, of the segments are identical. Otherwise, different segments (or parts thereof) will necessarily have different attributes and will need to be represented separately. The behavior of the attribute shapes, for each of the different line segments (or parts thereof), however remains the same.

An algebra with carrier $\mathcal{P}(A \times \mathcal{P}(B))$ and signature including sum ('+'), product ('·'), difference ('−'), and reduce ('r') can be defined for a discrete behavior, in terms of the (attribute) algebra $\mathcal{P}(B)$ with signature including sum ('+'), product ('·'), difference ('−'), and reduce ('r'), as follows:

$$\begin{aligned}
&\forall\, X,\ Y \in \mathcal{P}(A \times \mathcal{P}(B)) \Rightarrow \\
&\quad X + Y = \\
&\qquad \bigcup \begin{cases} \{(x, B_x + B_y) : (x, B_x) \in X \wedge (x, B_y) \in Y\} \\ \{(x, B_x) : (x, B_x) \in X \wedge \neg\exists (x, B_y) \in Y\} \\ \{(x, B_y) : (x, B_y) \in Y \wedge \neg\exists\, (x, B_x) \in X\} \end{cases} \\
&\quad X \cdot Y = \\
&\qquad \{(x, B_x \cdot B_y) : (x, B_x) \in X \wedge (x, B_y) \in Y\} \\
&\quad X - Y = \\
&\qquad \bigcup \begin{cases} \{(x, B_x - B_y) : (x, B_x) \in X \wedge (x, B_y) \in Y\} \\ \{(x, B_x) : (x, B_x) \in X \wedge \neg\exists\, (x, B_y) \in Y\} \end{cases} \\
&\quad \mathsf{r}(X) = \\
&\qquad \begin{cases} \{(x, B_x)\} + \mathsf{r}(X \setminus (x, B_x)) & \exists\, (x, B_x) \in X \\ \emptyset & \text{otherwise} \end{cases}
\end{aligned} \tag{6}$$

Comparing this to the discrete behavior for a non-augmented shape algebra, we observe that for the operations of sum, product and difference, if a spatial element is shared between both (augmented) shapes, we combine both attributes of the spatial element under the same operation. In the case of the operations of sum and difference, we may need to add spatial elements, with their original attribute,

that belong to one augmented shape but not the other. We express the operation of reduce in terms of the operation of sum for the same algebra. An algebra of shapes of labeled points, $V_0$, or shapes of weighted points, $W_0$, with the weights representing grey-scales or diameters, can be defined in this way.

Having established an augmented shape algebra for points, a demonstration of an augmented shape algebra for other spatial elements, using the areal behavior, remains. In fact, where the shape algebra for an areal behavior is expressed in terms of the sub-algebra for co-equal shapes, we can define a sub-algebra for co-equal augmented shapes and retain the definition for the shape algebra as a definition for the augmented shape algebra as well. The only difficulty is the use of the co-descriptor function ('co') on elements of the shape algebra. However, if we overload the co-descriptor function to accept augmented spatial elements, i.e., elements with attributes, then there is no issue.

A sub-algebra with carrier $\mathcal{P}(A \times \mathcal{P}(B))$ and signature including co-combine, co-common, co-complement, and co-reduce can be defined for an areal behavior, in terms of the two-sorted partial algebra with carrier $\{A, \mathcal{P}(A)\}$ and signature including the operations of combine, common and complement, and the (attribute) algebra $\mathcal{P}(B)$ with signature including sum ('+'), product ('·'), difference ('−'), and reduce ('r'), as follows:

$$
\begin{array}{l}
\forall\, X,\, Y \in \mathcal{P}(A \times \mathcal{P}(B)) : \\
\forall (x, B_x) \in X,\, \forall (y, B_y) \in Y,\, \mathsf{co}(x) = \mathsf{co}(y) \Rightarrow \\
\quad \mathsf{co\text{-}combine}(X, Y) = \\
\qquad \mathsf{co\text{-}r}\left( \bigcup \left\{ \begin{array}{c} \mathsf{common\text{-}sum}(X, Y) \\ \mathsf{complement}(X, Y) \\ \mathsf{complement}(Y, X) \end{array} \right) \right. \\
\quad \mathsf{co\text{-}common}(X, Y) = \\
\qquad \mathsf{co\text{-}r}(\mathsf{common\text{-}product}(X, Y)) \\
\quad \mathsf{co\text{-}complement}(X, Y) = \\
\qquad \mathsf{co\text{-}r}\left( \bigcup \left\{ \begin{array}{c} \mathsf{common\text{-}difference}(X, Y) \\ \mathsf{complement}(X, Y) \end{array} \right) \right. \\
\quad \mathsf{co\text{-}reduce}(X) = \\
\qquad \left\{ \begin{array}{ll} \mathsf{co\text{-}combine}\left( \begin{array}{c} \{(x, B_x)\}, \\ \mathsf{co\text{-}reduce}(X \setminus (x, B_x)) \end{array} \right) & \exists\, (x, B_x) \in X \\ \emptyset & \text{otherwise} \end{array} \right.
\end{array}
\tag{7}
$$

We cannot simply determine a resulting co-shape—from one of the operations of co-combine, co-common, and co-complement—from the classification of the boundaries of both co-shapes in their entirety. Instead, we need to apply the classification and construction to each pair of spatial elements from the respective co-shapes, in order to be able to assign the appropriate attributes—as defined by the operation of sum, product or difference applied to the respective attributes of the spatial elements. This definition relies on the specification of helper functions 'common-sum', 'common-product', 'common-difference', 'complement', and 'co-r', defined below. The first three helper functions return the common shape of two co-equal spatial elements, with as attribute, respectively, the sum, product and difference of the respective attributes. The 'complement' returns the complement shape of a spatial element with respect to a co-equal shape, with as attribute the original attribute of the spatial element. Finally, 'co-r' is a variant of the operation of co-reduce that assumes that none of the spatial elements overlap, though they may share boundaries. Two spatial elements that share boundary may be combined if they also share the same attribute(s).

$$
\begin{array}{l}
\mathsf{common\text{-}sum}(X, Y) = \\
\quad \bigcup_x \bigcup_y \left\{ (z, B_x + B_y) : \left\{ \begin{array}{c} (x, B_x) \in X \wedge \\ (y, B_y) \in Y \wedge \\ z \in \mathsf{co\text{-}common}(x, y) \end{array} \right\} \right. \\
\mathsf{common\text{-}product}(X, Y) = \\
\quad \bigcup_x \bigcup_y \left\{ (z, B_x \cdot B_y) : \left\{ \begin{array}{c} (x, B_x) \in X \wedge \\ (y, B_y) \in Y \wedge \\ z \in \mathsf{co\text{-}common}(x, y) \end{array} \right\} \right. \\
\mathsf{common\text{-}difference}(X, Y) = \\
\quad \bigcup_x \bigcup_y \left\{ (z, B_x - B_y) : \left\{ \begin{array}{c} (x, B_x) \in X \wedge \\ (y, B_y) \in Y \wedge \\ z \in \mathsf{co\text{-}common}(x, y) \end{array} \right\} \right. \\
\mathsf{complement}(X, Y) = \\
\quad \bigcup_x \left\{ (z, B_x) : \left\{ \begin{array}{c} (x, B_x) \in X \wedge \\ z \in \mathsf{difference}(\{x\}, Y) \end{array} \right\} \right. \\
\mathsf{co\text{-}r}(X) = \\
\quad \left\{ \begin{array}{ll} \mathsf{co\text{-}r}\left( \begin{array}{c} \{(z, B_x)\} \cup \\ X \setminus \\ \{(x, B_x), (y, B_x)\} \end{array} \right) & \left\{ \begin{array}{c} (x, B_x) \in X \wedge \\ (y, B_x) \in X \wedge \\ x \neq y \wedge \\ \{z\} = \\ \mathsf{co\text{-}combine}(x, y) \end{array} \right. \\ \emptyset & \text{otherwise} \end{array} \right.
\end{array}
\tag{8}
$$

An algebra of shapes of weighted line segments, $W_1$, or shapes of labeled plane segments, $V_2$, among others, can be defined in this way.

### *Discussion*

We have adopted a constructive, algebraic approach to defining shape algebras and augmented shape algebras. This allows for a variety of algebras to be defined from an array of basic (partial) algebras of spatial and other elements. For example, we can define an algebra of shapes of line segments and labeled points, $U_1 \times V_0$, from basic partial algebras of points, line segments and labels. Other opportunities arise. Rather than restricting ourselves to labels and weights as attributes, we can consider other attributes, such as color, in the same way. Consider a basic algebra of sets of colors, $C$, and assume we express the combination of a shape algebra with an attribute algebra with the operator '$\wedge$', termed *attribution*, then we can define an algebra of shapes of colored plane segments as $U_2 \wedge C$. Similarly, we can then write $V_0 = U_0 \wedge L$ and $W_1 = U_1 \wedge N$. Additionally, we can define an attribute algebra as a direct product of two basic attribute algebras, e.g., $U_0 \wedge (L \times N)$ defines an algebra of shapes of points with both labels and weights as attributes. We may consider the operation of direct product to distribute over the operation of attribution, resulting in $(U_0 \wedge L) \times (U_0 \wedge N) = V_0 \times W_0$.

While we commonly consider algebras of shapes augmented with algebras of (sets of) non-spatial elements, and the direct product of (augmented) shape algebras, other combinations are possible as well. Consider the algebra $D$ of sets of descriptions. Though it is a non-spatial algebra, most authors consider descriptions parallel to drawings, not as attributes thereof. Nevertheless, Beirão (2012) considers descriptions as attributes to spatial objects, allowing for objects to refer to alternate, though similar, descriptions. Thus, $U_1 \times D$ is as valid as $U_1 \wedge D$. Of course, some limitations in the way we combine algebras may still be preferable, reflecting on the kinds of design descriptions (whether drawings or other descriptions) we tend to use.

Knight (2003b; see also, Krstic 2012) also considers a sum of algebras that requires algebras to be specified in the same space. For example, in the case of a drawing of line segments and labeled points, the segments and points operate in the same (2D or 3D) space and transformations of shapes of line segments and shapes of labeled points necessarily need to go hand in hand. Instead, when shape algebras operate in a different space (or drawing), transformations may differ. The sum of algebras applies in the first case, the (direct) product in the second case. However, as we left transformations out of the picture, the direct product of algebras as we defined it is applicable to both cases. We refer to Krstic (2012) for a treatment of transformations in the context of shape algebras.

## CONCLUSION

We presented an algebraic approach to describing compound shapes that includes the definition of non-spatial algebras and the combination of shape algebras and non-spatial algebras under an operation of attribution. This algebraic abstraction serves as a procedural abstraction for the modular implementation of a general shape grammar interpreter.

## REFERENCES


Beirão, JN 2012, *CItyMaker: designing grammars for urban design*, Ph.D. Thesis, Delft University of Technology

Chase, SC 1999, 'Supporting emergence in geographic information systems', *Environment and Planning B: Planning and Design*, 26(1), p. 33–44

Duarte, JP 2001, *Customizing mass housing: a discursive grammar for Siza's Malagueira houses*, Ph.D. Thesis, MIT

Duarte, JP 2005, 'Towards the mass customization of housing: the grammar of Siza's houses at Malagueira', *Environment and Planning B: Planning and Design*, 32(3), p. 347–380

Frank, AU 1999, 'One step up the abstraction ladder: combining algebras', in Freksa, C and Mark, DM (eds) 1999, *COSIT'99, LNCS 1661*, Springer, Berlin, pp. 95-108

Gips, J 1975, *Shape Grammars and their Uses: Artificial Perception, Shape Generation and Computer Aesthetics*, Birkhäuser, Basel

Knight, TW 1989, 'Color grammars: designing with lines and colors', *Environment and Planning B: Planning and Design*, 16(4), p. 417–449

Knight, TW 1993, 'Color grammars: the representation of form and color in design', *Leonardo*, 26(2), p. 117–124
Knight, TW 1999 'Applications in architectural design, and education and practice', *NSF/MIT Workshop on Shape Computation*, report, Cambridge, MA
Knight, TW 2003a, 'Computing with emergence', *Environment and Planning B: Planning and Design*, 30(1), p. 125–155
Knight, TW 2003b, 'Computing with ambiguity', *Environment and Planning B: Planning and Design*, 30(2), p. 165–180
Knight, TW 2004 'Interaction in visual design computing', *Visual and Spatial Reasoning in Design III*, presentation, Cambridge, MA
Krishnamurti, R and Stouffs, R 2004, 'The boundary of a shape and its classification', *Journal of Design Research*, 4(1)
Krstic, D 1999, 'Constructing algebras of design', *Environment and Planning B: Planning and Design*, 26(1), pp. 45-57
Krstic, D 2012, 'Algebras of shapes revisited', in Gero, JS (eds) 2012, *Design Computing and Cognition '12*, Springer, Dordrecht, p. 361–376
Li, AI 1999 'Expressing parametric dependence in shape grammars, with an example from traditional Chinese architecture', *Proceedings of CAADRIA 1999*, Shanghai, p. 265–274
Li, AI 2001, *A shape grammar for teaching the architectural style of the Yingzao fashi*, Ph.D. Thesis, MIT
Li, AI 2004, 'Styles, grammars, authors, and users', in Gero, JS (eds) 2004, *Design Computing and Cognition '04*, Kluwer, Dordrecht, p. 197–215
Stiny, G 1975, *Pictorial and Formal Aspects of Shape and Shape Grammars*, Springer, Basel
Stiny, G 1980, 'Introduction to shape and shape grammars', *Environment and Planning B: Planning and Design*, 7(3), pp. 343-351
Stiny, G 1981, 'A note on the description of designs', *Environment and Planning B: Planning and Design*, 8(3), p. 257–267
Stiny, G 1990, 'What is a design?', *Environment and Planning B: Planning and Design*, 17(1), p. 97–103
Stiny, G 1992, 'Weights', *Environment and Planning B: Planning and Design*, 19(4), p. 413–430
Stiny, G and Gips, J 1972, 'Shape grammars and the generative specification of painting and sculpture', in Freiman, CV (eds) 1972, *Information Processing 71*, North-Holland, Amsterdam, p. 1460–1465
Stouffs, R 2008, 'Constructing design representations using a sortal approach', *Advanced Engineering Informatics*, 22(1), pp. 71-89
Stouffs, R and Krishnamurti, R 2006, 'Algorithms for classifying and constructing the boundary of a shape', *Journal of Design Research*, 5(1), pp. 54-95
Yue, K and Krishnamurti, R 2013, 'Tractable shape grammars', *Environment and Planning B: Planning and Design*, 40(4), p. 576–594