On Shape Grammars, Color Grammars and *Sortal* Grammars

A sortal grammar interpreter for varying shape grammar formalisms

Rudi Stouffs
Delft University of Technology, The Netherlands and National University of Singapore, Singapore.
http://www.tudelft.nl/rmfstouffs and http://www.arch.nus.edu.sg/people/cv/rudi.htm
r.m.f.stouffs@tudelft.nl and stouffs@nus.edu.sg

Abstract. *Grammar formalisms for design come in a large variety, requiring different representations of the entities being generated, and different interpretative mechanisms for this generation. Most examples of shape grammars rely on labeled shapes, a combination of line segments and labeled points. Color grammars extend the shape grammar formalism to allow for a variety of qualitative aspects of design, such as color, to be integrated in the rules of a shape grammar. Sortal grammars consider a compositional approach to the representational structures underlying (augmented) shape grammars, allowing for a variety of grammar formalism to be defined and explored. In this paper, we revisit and explore an exemplar shape grammar from literature to illustrate the use of different grammar formalisms and consider the implementation of rule application within a sortal grammar interpreter.*
Keywords. *Shape grammars; color grammars; sortal grammars; implementation.*

INTRODUCTION

Grammar formalisms for design come in a large variety (e.g., Stiny, 1980; Stiny, 1981; Carlson et al., 1991; Heisserman and Woodbury, 1994; Duarte and Correia, 2006), 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 1981). However, even in the original conception of shape grammars (Stiny and Gips, 1972), an iconic shape (made up of curved lines) serves the role of non-terminal marker rather than labeled points, and a colored infill of the resulting shapes is considered part of the generative specification, though not of the shape grammar.

Next to labels, other non-geometric attributes have been considered for shapes. Stiny (1992) proposes numeric weights as attributes to denote line thicknesses or surface tones. Knight (1989; 1993) considers an extension to the shape grammar formalism that allows for a variety of qualitative aspects of design, such as color, to be integrated in the rules of a shape grammar. Though not specific to colors, the resulting grammar is called a *color grammar* and notions of transparency, opacity and ranking are introduced to regulate the behavior of interacting quality-defined areas or volumes.

In all of these examples, the augmented shapes have been derived from shapes of spatial elements by associating symbols, labels or other qualitative aspects to the elements, under a shape-attribute relationship. In *sortal grammars* (Stouffs and Krishnamurti, 2001), shapes may be either the object or the attribute in the relationship, or both (or neither, though such examples do not constitute spatial grammars as such). *Sortal* grammars utilize *sortal* structures as representational structures, where *sortal* structures are defined as formal compositions of other, primitive, *sortal* structures, termed *sorts*.

In this paper, we revisit and explore an exemplar shape grammar from literature to illustrate the use of different grammar formalisms from among shape grammars, color grammars and *sortal* grammars and consider the implementation of rule application within a *sortal* grammar interpreter.

AN ALGEBRAIC COMPARISON OF SHAPE AUGMENTATION FORMALISMS

Stiny (1991) defines shapes as finite arrangements of *n*-dimensional hyperplane segments of limited but non-zero measure in a *k*-dimensional space, $k \geq n$. The notation $U_{n,k}$ denotes the algebraic set of all such shapes; $U_{1,2}$, also written as U_1 if k=2 is unambiguously understood, refers to an algebra of shapes made up of line segments in two-dimensional space. In three dimensions, a shape grammar could include points, line segments, plane segments or even volumes. If a shape consists of more than one type of spatial element, it belongs to the algebra given by the Cartesian product of the algebras of its spatial element types (Stiny, 1991), e.g., $U_0 \times U_1$ refers to an algebra of points and line segments. The same is said to apply for the specification of labeled points or labeled shapes; given a set L of symbols, which may be empty, we can define an algebra $V_0 = U_0 \times \wp(L)$ of labeled points, where $\wp(L)$ denotes the power set of L, and an algebra $V = U_1 \times V_0$ of *labeled shapes*.

While it is attractive to consider each of these examples, formally, as a Cartesian product of algebras, whether composed of two shape algebras or of a shape algebra with a non-spatial algebra, the latter as suggested for labeled points, there is a fundamental difference between how these Cartesian products behave in both cases. An algebra of points and line segments, $U_0 \times U_1$, is not significantly different from an algebra of line segments and points, $U_1 \times U_0$, that is, the Cartesian product over shape algebras could be considered commutative, $U_0 \times U_1 \equiv U_1 \times U_0$. However, an algebra of labeled points, $U_0^1 \times \wp^1(L)$, cannot be considered equivalent to an algebra of "pointed labels," $\wp(L) \times U_0$. Firstly, the association of labels, or other qualitative aspects of design, to shapes under a shape-attribute relationship is not of a symmetric kind. Secondly, the operations of shape computation do not necessarily distribute over both algebras in the way they do over a Cartesian product of spatial algebras only. In the latter case, given two shapes each consisting of a line segment and a plane segment, the sum of both shapes is the Cartesian product of the sum of both line segments with the sum of both plane segments. In the case of colored shapes, the sum of two line segments with different colors that spatially overlap cannot be considered to be the sum of both line segments with a color that is the sum of both individual colors. This only applies to the common segment; any other segment that belongs to only one of both shapes has to retain its original color under a proper algebraic model.

Stouffs (1994), instead, suggests a different mathematical formalism for shapes augmented with qualitative aspects, considering a characteristic function to a shape, similar to the definition of half-spaces in constructive solid geometry. The range of the characteristic function is then dependent on the aspect considered to augment the shape. For example, in the case of weights, the range may constitute the set of positive real numbers, $\mathbb{R}^+$, in the case of colors this may be a 3-dimensional additive color space and, in the case of labeled shapes, for a given set of labels L, the range of the characteristic function is $\wp(L)$, the power set of L. Summing qualitative aspects then reduces to adding characteristic functions over the same range together and shape computation distributes once again over the Cartesian product.

Sortal structures (Stouffs, 2008), as underlying *sortal* grammars, are defined as formal compositions of other, primitive, *sortal* structures, termed *sorts*. Each *sort* defines an algebra over its elements. Similarly to shapes and shape attributes in the context of shape grammars, the algebra of a *sorts is* specified through a part relationship on the elements of this *sort*, with the algebraic operations of addition, subtraction, and product defined in accordance to this part relationship. The part relationship also explicates the match relation (or interpretative mechanism) underlying a *sortal* algebra and grammar. Composite *sortal* structures derive their part relationship from their component *sorts* through the formal compositional operators defined over *sorts*. These formal compositional operators constitute a co-ordinate, disjunctive relationship, as in the Cartesian product of two spatial algebras, and a subordinate, semi-conjunctive relationship, as in the Cartesian product of a spatial algebra and a qualitative aspect algebra.

The central problem in implementing grammars is the matching problem, that of determining the transformation under which the left-hand-side of the rule forms a part of the shape/entity under consideration. Since the part relationship of a *sortal* structure is derived from its component *sorts*, most technical difficulties of implementing the matching problem only apply once for each (simple) *sort*. As the part relationship can be applied to various kinds of data types, recognition algorithms can be extended to deal with quite arbitrary data representations, on condition that what constitutes a transformation can be properly defined. Considering the application of a grammar-based approach to a generative problem, a *sortal* structure can be composed and, where necessary, component *sorts* developed, the corresponding grammar formalism explored for the given problem, and the *sortal* structure and grammar adapted to fit the specifics of the problem.

REVISITING AN EXEMPLAR SHAPE GRAMMAR

To illustrate this process, let us consider an example of a shape grammar that is reminiscent of Stiny and Gips' (1972) original generative specification of including a material specification in the form of painting rules. The example is taken from Stiny's "Computing with Form and Meaning in Architecture" (Stiny, 1985) and concerns a grammar composed of three rules: the first rule creates a square from an initial marker, the second creates a rotated square inscribed within the original square, and the third rule removes the marker (Figure 1). The marker, a point, moves from one square to the next to guide the generation. The painting rules, though not explicated, consider an alternating infill of the squares in black and white.

Considering a grammar formalism that allows for colored plane segments, next to (labeled) points and line segments, a grammar can be constructed that incorporates the painting rules in the generation of the overall shape. Let us start by considering a color grammar for this purpose. Each square in the rule set is specified as a plane segment (also denoted region (Knight, 1993) or field (Knight, 1989)) rather than a collection of four line segments. A color is associated to each plane segment; the possible color values are limited to black and white. In color

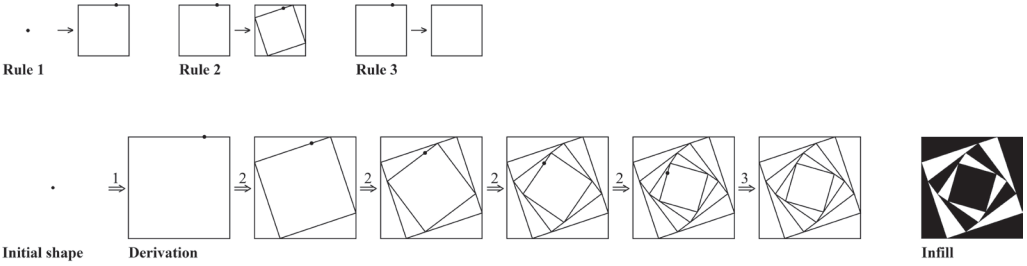

Figure 1
A grammar composed of three rules, generating recursively inscribed squares (redrawn from Stiny, 1985).

grammars, overlapping colored plane segments are handled formally with rankings (Knight, 1993); here, an "opaque" ranking is suggested where any colored plane segment that is added in a rule application covers any part of a colored plane segment already in the design (Figure 2). The first rule creates an initial black square. The second rule, inscribing a rotated square into an existing square, is distinguished into two different rules: the first applies to a black square, inscribing a white rotated square, the second applies to a white square, inscribing a black rotated square. The last rule, removing the marker, is also modified: the square is removed from both the left-hand-side and right-hand-side of the rule, in order to avoid having to specify two separate rules, one for a black square and one for a white square. However, this modifies the grammar (and its resulting language of designs) in that the last rule now directly applies to the initial shape, without the need for Rule 1 to apply first. Instead, splitting the last rule into two as mentioned above, will ensure that Rule 1 must always apply before the marker can be removed (when considering an initial shape consisting only of one or more markers).

Instead of using a color grammar, the same language of designs can be achieved using a shape grammar of weighted plane segments, next to (labeled) points and line segments, where the (numerical) weight is interpreted to denote a surface tone (Stiny, 1992). However, the rules as specified in Figure 2 cannot be considered to apply without modification. Overlapping weighted plane segments are handled formally by considering a partial-order relationship on weights, corresponding to the less-than-or-equal relation on numeric values: assuming higher numeric values for darker surface tones, coincident plane segments with different tones combine into a plane segment with the darkest tone, even though it assumes the same plane segment with other, lighter tones. As such, only Rule 3 (from Figure 2) would apply to a white square but both Rule 2 and Rule 3 would apply to a black square, as a black square assumes the same square with a lighter, e.g., white, tone. This problem may be resolved by also considering a tone for the marker point, more specifically, the opposite tone of the respective square (Figure 3). While a black square assumes the same square with a white tone, a white point will not assume the same point with a black tone, and vice versa. As a consequence, the last rule, removing the marker, necessarily, also needs to be split into two rules, one considering a white maker point, the other a black marker point. Adding a black, respectively, white square, to both the left-hand-side and the right-hand-side of the rule ensures once again that Rule 1 must be applied to an initial shape consisting of one or more marker points before any marker can be removed.

Sortal grammars, as a grammar formalism, encompasses both shape grammars (including weights) and color grammars and, thus, both versions (Figures 2 and 3) of the grammar generating recursively inscribed squares with alternating infill can be defined as a *sortal* grammar. However, any

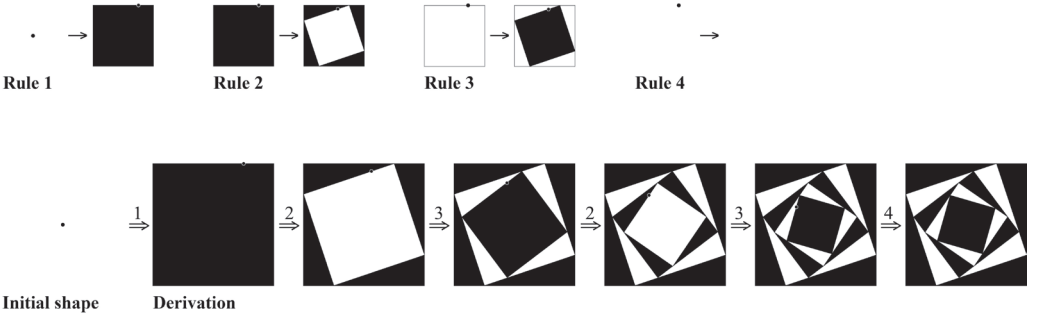

Figure 2
A color grammar generating recursively inscribed squares with alternating infill. A white segment is indicated by a lightly drawn outline in order to distinguish it from the background.

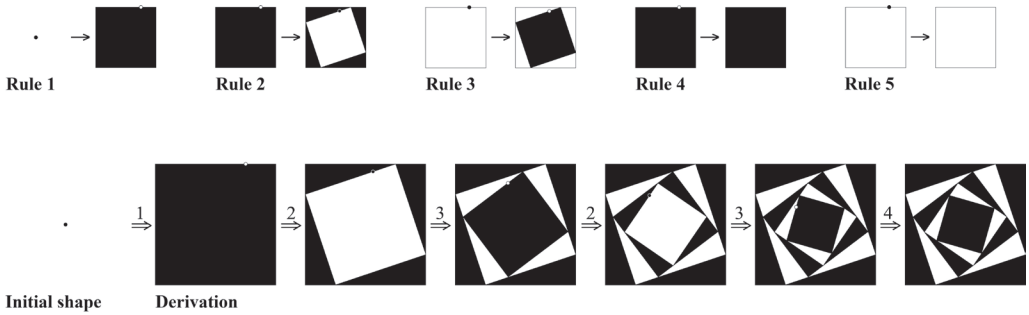

Figure 3
A (weighted) shape grammar generating recursively inscribed squares with alternating infill. A white segment, or point, is indicated by a lightly drawn outline in order to distinguish it from the background.

implementation of a grammar interpreter necessarily introduces additional constraints with respect to rule application, requiring further modifications of the rules constituting the grammar. For example, the first rule (creating an initial square) may apply over and over again in a single derivation as the same marker (with the exception of the possible differentiation in tone) moves from one square to the next, inscribing, square. In addition, the first rule is non-deterministic as a single point maps with another point in an infinite number of ways, considering both variations in rotation and scaling. An implementation must allow for indeterminate rule applications in order to allow Rule 1 to apply. The same may be said about Rule 2 (in both versions); the combination of a point and a co-planar plane segment is also an indeterminate case for *subshape* recognition (Krishnamurti and Stouffs, 1997). Specifically, if a match is found for the left-hand-side of Rule 2, any reduction in scaling (considering the same rotation and the same translation with respect to the marker point) yields a potential match. In two dimensions, a determinate case requires either two distinct points, a point and a non-collinear line, or three distinct lines not all concurrent in one point. Therefore, in order to make the rules considered above deterministic, either an extra (marker) point, or an extra non-collinear (marker) line segment should be added to each rule. Alternatively, the existing marker point may be replaced by three (or four) marker line segments. However, in this case, symmetry should be avoided in order to ensure that the derivation always proceeds in the same direction (angle of rotation).

A *SORTAL* GRAMMAR INTERPRETER

In order to test these ideas, an implementation of a *sortal* grammar interpreter is being developed for use within the Processing programming environment [1]. While various shape grammar interpreters have been developed over the years, most are limited to *labeled shapes* and/or do not fully support *subshape* recognition. The SortalGI *sortal* grammar interpreter library [2] developed for the Processing environment currently allows for points and line segments (with associated stroke tone and stroke/line thickness), plane segments (with associated fill tone), labeled points (the label can have an associated stroke tone), and labeled line and plane segments (similar to line or plane segments but with additional associated label). Fill tones can either be specified as a numeric weight or as an enumerative value with ranking (conform the specification of a color grammar). Only determinate cases of rule application are considered so far.

Figure 4 illustrates the specification and application of a *sortal* grammar generating recursively inscribed squares with alternating infill. It uses a square outline of four marker line segments to ensure determinate rule application. Except for the first rule's left-hand-side, which matches the initial square shape, one of the marker line segments is shortened to inhibit symmetry so as to ensure that rule application always proceeds in the same way (always rota-

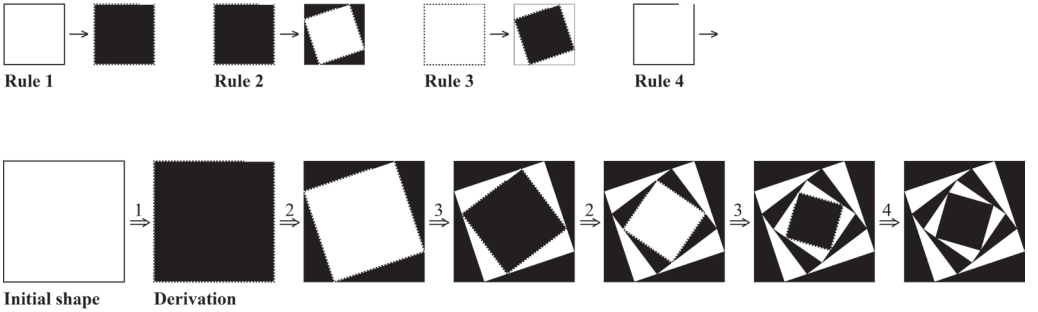

Figure 4
A sortal *grammar using marker line segments and enumerative tones, generating recursively inscribed squares with alternating infill. Line segments are drawn dashed where they coincide with the boundary of a plane segment. A white plane segment is distinguished by a lightly drawn outline.*

ting the inscribed square in the same direction). The final rule (to remove the marker lines) still applies to the initial shape, though if applied would leave a small line segment. Table 1 (top) presents the *sortal* structure underlying this grammar, consisting of line segments with associated stroke thicknesses and stroke tones ("strokedLines"), and plane segments with associated enumerative colors "black" and "white" with "opaque" ranking ("filledShapes").

Table 2 (left) provides an extract from the Processing code, illustrating the initialization of the SortalGI engine (with the specification of the enumerative color values) and the specification of Rule 2. The left-hand-side of the rule specifies the four marker line segments as well as the corresponding black plane segment. The right-hand-side of the rule replaces the four marker line segments and adds the inscribed and rotated, white plane segment. Stroke values are specified conform to the Processing environment: 0 represents black and 255 represents white. Within the SortalGI library these are converted to 255 and 1, respectively, in order to adhere to the expected partial-order relationship on tones (darker tones containing lighter tones). Note that the ranking of enumerative colors, or other qualitative de-

```
sort strokeWeights : [Weight](10);
sort strokes : [Weight](255);
sort lines : [LineSegment];
sort strokedLines : lines ^ strokeWeights ^ strokes;
sort fills : [Enumerative]({"black", "white"});
sort shapes : [PlaneSegment];
sort filledShapes : shapes ^ fills;
sort processingShapes : strokedLines + filledShapes;
```

```
sort strokeWeights : [Weight](10);
sort strokes : [Weight](255);
sort points : [Points];
sort strokedPoints : points ^ strokeWeights ^ strokes;
sort lines : [LineSegment];
sort strokedLines : lines ^ strokeWeights ^ strokes;
sort fills : [Weight](255);
sort filledShapes : shapes ^ fills;
sort processingShapes : strokedPoints + strokedLines + filledShapes;
```

Table 1
Definition of the sortal *structures for the* sortal *grammars illustrated in Figure 4 (top) and Figure 5 (bottom). Sorts are specified by a characteristic individual (enclosed within square brackets) with zero, one or more arguments (enclosed within parentheses). Sorts are composed with the '+' operator (specifying a co-ordinate, disjunctive relationship) and the '^' operator (specifying a subordinate, semi-conjunctive relationship).*

sign aspects (in color grammars), on the other hand, does not adhere to a partial-order relationship. Similarly, the requirement for any *sort* to define an algebra does not strictly apply to enumerative *sorts*; while it is a sufficient condition for the composition of *sortal* structures, it is not a necessary condition.

Figure 5 illustrates an alternative specification of a *sortal* grammar generating recursively inscribed squares with alternating infill. It uses two marker points to ensure determinate rule application. Only the initial shape (and the left-hand-side of Rule 1) remains composed of four line segments.

Table 1 (bottom) presents the *sortal* structure underlying this grammar, consisting of both points and line segments with associated stroke thicknesses and stroke tones ("strokedPoints" and "stroked-

Table 2
Examples of rule specification using the SortalGI library in the Processing environment: (left): using marker line segments and enumerative tones; (right) using marker points and numeric weights for tones. The declaration of SortalGI functions mimics as much as possible the declaration of similar Processing functions.

```
final String BLACK = "black";
final String WHITE = "white";
final String[] names = {BLACK, WHITE};
final float[] values = {0, 255};
// initialize the SortalGI engine
sgi = SortalGI.initialize(this, names, values);

// specify Rule 2
SortalRule r2 = new SortalRule("r2", "Black
rule");
r2.beginLHS();
sgi.stroke(0);
sgi.strokeWeight(1);
sgi.line(0, 0, 75, 0);
sgi.line(0, 0, 0, 100);
sgi.line(0, 100, 100, 100);
sgi.line(100, 100, 100, 0);
sgi.noStroke();
sgi.fill(BLACK);
sgi.quad(0, 0, 100, 0, 100, 100, 0, 100);
r2.endLHS();
r2.beginRHS();
sgi.stroke(255);
sgi.strokeWeight(1);
sgi.line(0, 25, 56.25, 6.25);
sgi.line(0, 25, 25, 100);
sgi.line(25, 100, 100, 75);
sgi.line(100, 75, 75, 0);
sgi.noStroke();
sgi.fill(BLACK);
sgi.quad(0, 0, 100, 0, 100, 100, 0, 100);
sgi.fill(WHITE);
sgi.quad(75, 0, 100, 75, 25, 100, 0, 25);
r2.endRHS();
```

```
// initialize the SortalGI engine
sgi = SortalGI.initialize(this);

// specify Rule 2
SortalRule r2 = new SortalRule("r2", "Black
rule");
r2.beginLHS();
sgi.stroke(255);
sgi.strokeWeight(3);
sgi.point(75, 0);
sgi.point(100, 0);
sgi.noStroke();
sgi.fill(0);
sgi.quad(75, 0, 100, 75, 25, 100, 0, 25);
r2.endLHS();
r2.beginRHS();
sgi.stroke(0);
sgi.strokeWeight(3);
sgi.point(56.25, 6.25);
sgi.point(75, 0);
sgi.noStroke();
sgi.fill(255);
sgi.quad(75, 0, 100, 75, 25, 100, 0, 25);
r2.endRHS();
```

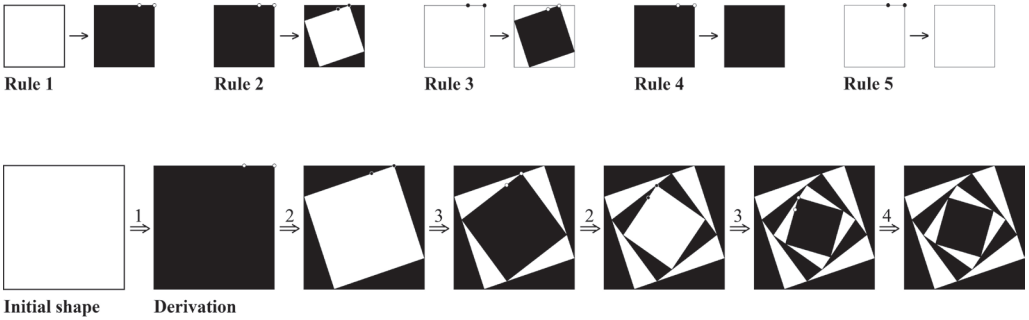

Figure 5
A sortal *grammar using marker points and numeric weight tones, generating recursively inscribed squares with alternating infill. A white plane segment, or point, is indicated by a lightly drawn outline.*

Lines"), and plane segments with associated (numeric) surface tones ("filledShapes"). Table 2 (right) shows the corresponding initialization of the SortalGI engine and the specification of Rule 2 within the Processing environment. The left-hand-side of the rule specifies the two marker points as well as the (inscribed, rotated) part of the black plane segment that will be replaced with a white segment. The right-hand-side of the rule replaces the two marker points and adds the inscribed and rotated, white plane segment. The stroke tone of the marker points is always opposite to the fill tone of the plane segment, in order to ensure that rules only match as expected, notwithstanding the fact that a black point, or plane segment, assumes a white point, or plane segment, respectively.

Implementation issues

Developing a *sortal* grammar interpreter requires the matching problem to be solved independently of the specific *sortal* structure over which the grammar is specified. Different *sorts* may allow for different transformations, such as similarity transformations for spatial information and case transformations for text-based information. In order to avoid an exhaustive search over all *sorts* for potential matches, both transformations and *sorts* are ranked by pertinence. Transformations can be ranked according to their degrees of freedom (e.g., seven for a similarity transformation: three translational, three rotational and one (uniform) scaling; zero or one for case transformations as only a discrete number of case transformations can be distinguished). At the same time, *sorts* can be ranked according to their dimensionality, as either discrete, linear, planar or spatial. As such, a greedy algorithm can be developed that will focus its attention first to *sorts* with the lowest combination of dimensionality and transformational degrees of freedom and on adjacent component *sorts* under the (subordinate, semi-conjunctive) attribute relationship. For example, in the *sortal* equivalent to the algebra $V = U_1 \times V_0$ of *labeled shapes*, labels will be considered first, followed by the points they are associated to. The matching of these points will naturally be restricted by this association. Only if (labels and) points are insufficient to determine the matching transformations, then line segments will also be considered.

CONCLUSION

Sortal grammars support varying grammar formalisms, allowing the user to explore alternative formulations of the same grammar, yielding the same design language. The SortalGI *sortal* grammar interpreter supports such exploration within the Processing environment, though requires some programming (or scripting) experience from the user. Additional support for ellipses, arcs, volume segments, textures for plane segments and various other compositions, such as labeled line and plane segments may still be added to expand the exploration space. The SortalGI library can also be used outside of the Processing environment, allowing for the development of graphical user interfaces to sup-

port grammar development and exploration using the *sortal* grammar formalism schema. The ability to explore different grammar formalisms to achieve the same design language may yield new research questions about advantages and disadvantages thereof and the appropriateness of a particular grammar formalism for a design problem or, even, a family of design problems.

REFERENCES

Carlson, C, McKelvey, R and Woodbury, RF 1991, 'An introduction to structure and structure grammars', *Environment and Planning B: Planning and Design*, 18(4), pp. 417–426.

Duarte, JP 2005, 'A discursive grammar for customizing mass housing: the case of Siza's houses at Malagueira', *Automation in Construction*, 14(2), pp. 265–275.

Heisserman, J and Woodbury, R 1994, 'Geometric design with boundary solid grammars' in JS Gero and E Tyugu (eds), *Formal Design Methods for CAD: Proceedings of the IFIP TC5/WG5.2 Workshop on Formal Design Methods for CAD*, Tallinn, Estonia, pp. 85–105.

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

Knight, TW 1993, 'Color grammars: the representation of form and color in design', *Leonardo* 26(2), pp. 117–124.

Krishnamurti, R and Stouffs, R 1997, 'Spatial change: continuity, reversibility and emergent shapes', *Environment and Planning B: Planning and Design*, 24(3), pp. 359–384.

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), pp. 257–267.

Stiny, G 1985, 'Computing with form and meaning in architecture', *Journal of Architectural Education*, 39(1), pp. 7–19.

Stiny, G 1991, 'The algebras of design,' *Research in Engineering Design*, 2(3), pp. 171–181.

Stiny, G 1992, 'Weights', *Environment and Planning B: Planning and Design*, 19(4), pp. 413–430.

Stiny, G and Gips J 1972, 'Shape grammars and the generative specification of painting and sculpture' in CV Freiman (ed), *Proceedings of IFIP Congress71*, North-Holland, Amsterdam, pp. 1460–1465. Republished in OR Petrocelli (ed), *The Best Computer Papers of 1971*, Auerbach, Philadelphia, pp. 125–135.

Stouffs, R 1994, *The Algebra of Shapes*, PhD dissertation, Dept. of Architecture, Carnegie Mellon University, Pittsburgh, Pa.

Stouffs, R 2008, 'Constructing design representations using a sortal approach', *Advanced Engineering Informatics*, 22(1), pp. 71–89.

Stouffs, R and Krishnamurti R 2001, 'Sortal grammars as a framework for exploring grammar formalisms' in M Burry, S Datta, A Dawson and J Rollo (eds), *Mathematics and Design 2001*, Geelong, Australia, pp. 261–269.

[1] www.processing.org

[2] www.sortal.org