A FORMAL REPRESENTATION FOR GENERATION AND TRANSFORMATION IN DESIGN

HODA MOUSTAPHA,
Carnegie Mellon University

Abstract. This paper presents a formal notation for describing geometrical configurations. The notation is based on generative and relational structures, and is developed in conjunction with a 3D modelling system. This paper focuses on the syntax and features of the notation. The notation in its capacity to specify the generation and transformation of shapes and complex configurations is illustrated through two architectural examples.

1. Introduction

ICE is a representation to capture higher level constructs for creating and manipulating geometric configurations. ICE comprises two components: a formal notation to specify a configuration through generative and relational constructs, and a generative-interactive 3D modeling system that utilizes these constructs as handles for generating and transforming configurations. These higher level manipulations distinguish ICE from other systems and provide a simple, yet powerful, means for computational design exploration.

This paper introduces the syntax of the ICE notation. The basic generative constructs and properties of ICE are illustrated by means of examples. The notation, as a whole, is illustrated by two architectural graphic examples. The paper concludes with a discussion of the notation and its future development.

1.1. MOTIVATION

The need to describe the structure of complex architectural design configurations was a major motivational factor in developing the ICE notation, which satisfies the following goals: (i) to describe geometric configurations in a clear, succinct, and complete manner; (ii) to maximize the universe of possibilities for geometric descriptions; (iii) to capture the most parsimonious process (or processes) for generation based on the structure; and (iv) to describe the possible transformations which are applicable to a configuration, and therefore, can be used for exploration.

2. The ICE notation

The ICE notation specifies a geometric configuration in terms of the minimal steps required for its generation, and the meaningful relationships for its organization. These generative and relational constructs form the basic units of the configuration's structure, referred to as "regulators." Regulators were introduced in Moustapha and Krishnamurti (2001) to demonstrate an early version of the ICE system for exploring calligraphic compositions. Regulators can control a variety of design elements, and therefore, can be used as handles to manipulate configurations. In short, regulators are higher level entities that "regulate" the behavior of lower level design elements; hence the name.

A regulator encapsulates a formula, by which it computes the position (or some other attribute) of the element(s) it regulates. There are five kinds of regulators: transformations, constraints, hierarchies, variations, and operations. Transformational regulators, such as translations, rotations, reflections, and curves, constitute the primary constructs for generating shapes and configurations. Constraint regulators, such as alignment and adjacency, are constructs for relating parts of shapes or configurations. Hierarchical regulators, such as containment, define hierarchies between shapes. Variational regulators, such as rhythm and gradation, create variations within the generative structure. Operational regulators, such as union and difference, generate complex shapes from simpler ones.

The basic elements of the representation are regulators and points. Notationally, *points* are indicated in lowercase, for instance, *p*, and *regulators* in (bold) uppercase, for instance, **T** for translation. *Shapes,* denoted as lowercase words, are composite objects defined by points and regulators. A prefix, depicted in uppercase Greek, indicates its regulator category, for example, Δ: transformations, Φ: constraints, Ψ: hierarchies, Ξ: variations, and Ω: operations. Superscripted suffixes indicate regulator subtype, for instance, $\Delta\mathbf{C}^{\mathbf{p}}$ and $\Delta\mathbf{C}^{\mathbf{e}}$ respectively specify parabolic and elliptical curve regulators with each having its own formula. Numerical suffixes denote the dimension of the regulator, for instance, $\Delta\mathbf{M}^{\mathbf{0}}, \Delta\mathbf{M}^{\mathbf{1}}$, and $\Delta\mathbf{M}^{\mathbf{2}}$, respectively represent a mirror point (0-dimensions), a mirror line (1-dimension) and a mirror plane (2-dimensions). Subscripted suffixes for regulators, shapes, or points are used to index the different elements of these types, for instance, $\Delta\mathbf{T}_{\mathbf{1}}$, p_2, and shape_2.

The ICE notation can be expressed in either short or expanded forms, the latter essential for formal implementation. The short form expresses the regulator and regulated object(s), for instance $\Delta\mathbf{T}(\text{shape})$. The expanded form, additionally, includes the parameters of the regulator; these are enclosed within curly braces with vectors depicted by an overline, for example, $\Delta\mathbf{T}^{\mathbf{1}}[\ \{\overline{p}, \overline{t}, d, n\}\ (\text{shape})\ ]$. Parameters contribute to a regulator's formula and include geometric parameters such as translation vectors, rotation points/lines, reflection axes, and generative attributes, such as translation distance, rotation degrees, number of generated objects, etc.

Regulators regulate points thereby creating shapes to create configurations, and other regulators to create complex schema. The conjunction $\wedge$ joins two related ICE notation strings, such as $\Delta\mathbf{T}(\text{shape}_1) \wedge \Delta\mathbf{T}(\text{shape}_2)$ or $\Delta\mathbf{T}(\text{shape}_1) \wedge \Delta\mathbf{R}(\text{shape}_1)$. The ICE notation supports the following distributive property, $\Delta\mathbf{T}(\text{shape}_1, \text{shape}_2) = \Delta\mathbf{T}(\text{shape}_1) \wedge \Delta\mathbf{T}(\text{shape}_2)$.

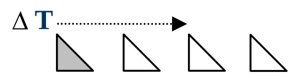
3. Primary generation constructs: Transformational regulators

3.1. TRANSFORMATIONAL REGULATORS

Transformational regulators are generative regulators that take as input a shape or a point, and generate "*n*" output shapes or points. The input shape is assigned index 0 and the output shapes are assigned indices 1-*n*. The outputs are positioned according to the transformation. Transformational regulators are indicated by the Δ prefix. Lines and planes are represented internally by vectors. Table 1 gives the list of transformational regulators that are currently represented by ICE. Note that although most illustrations are in 2D, the regulators are designed to apply to 3D.

Table 1. Regulators based on Isometry transformations

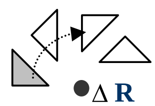
Translation $\Delta\mathbf{T}^{\mathbf{1}}[\ \{\bar{p},\bar{t},d,n\}\ \text{(shape)}\ ]$



Translation generates n output shapes (d distance apart) along the line (indicated by the superscript 1) specified by the starting point $\bar{p}$ and the direction vector $\bar{t}$.

Rotation $\Delta\mathbf{R}^{\mathbf{0}}[\ \{\bar{p},\alpha,n\}\ \text{(shape)}\ ]\ )\ ]$

$\Delta\mathbf{R}^{\mathbf{1}}[\ \{\bar{p},\bar{t},\alpha,n\}\ \text{(shape)}\ ]$

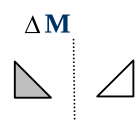


Rotation generates n output shapes, each rotated α degrees apart, in 2D, about about a point p, or in 3D, about the line specified by the starting point $\bar{p}$ and the direction vector $\bar{t}$.

Mirror $\Delta\mathbf{M}^{\mathbf{0}}[\ \{\bar{p},n\}\ \text{(shape)}\ ]$

$\Delta\mathbf{M}^{\mathbf{1}}[\ \{\bar{p},\bar{t},n\}\ \text{(shape)}\ ]$

$\Delta\mathbf{M}^{\mathbf{2}}[\ \{\bar{p},\bar{t},\bar{v},n\}\ \text{(shape)}\ ]$



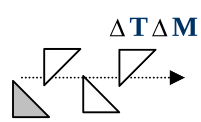
ICE supports three subtypes of mirrors: an inversion about a mirror point p; a mirror about the line defined by point $\bar{p}$ and direction vector $\bar{t}$; and a mirror in the plane defined by point $\bar{p}$ and direction vectors $\bar{t}$ and $\bar{v}$.

Screw Rotation $\Delta\mathbf{R}^{\mathbf{1}}\Delta\mathbf{T}^{\mathbf{1}}[\ \{\bar{p},\bar{t},d,\alpha,n\}\ \text{(shape)}\ ]$

The screw rotation is a composition of a rotation and a translation [see table 3].

Glide $\Delta\mathbf{T}^{\mathbf{1}}\Delta\mathbf{M}^{\mathbf{1}}[\ \{\bar{p},\bar{t},d,n\}\ \text{(shape)}\ ]$

$\Delta\mathbf{T}^{\mathbf{2}}\Delta\mathbf{M}^{\mathbf{2}}[\ \{\bar{p},\bar{t},\bar{v},d,e,n\}\ \text{(shape)}\ ]$

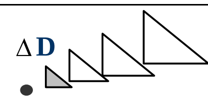


Glide is a composition of a mirror and translation [see table 3]. Sub-types include a glide, in 2D, along a glide line, and in 3D, in a glide plane.

Table 2. Regulator based on affine and non-linear transformations.

Dilation (scale)

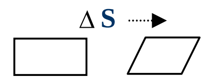
$$\Delta \mathbf{D}^{\mathbf{0}}[\ \{\bar{p},\bar{k},n\}\ (\text{shape})\ ]$$



Dilation scales successive output shapes by a factor *k*, represented by a vector. Scaling can be isotropic (equal in the xyz-directions) or anisotropic.

Shear

$$\Delta \mathbf{S}\ [\ \{\bar{k},n\}\ (\text{shape})\ ]$$

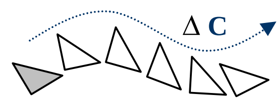


This regulator shears successive output shapes by a factor *k*, represented by a vector that indicates the shearing in each of three principal directions.

Curve

$$\Delta \mathbf{C}^{\mathbf{e}}[\ \{\bar{p},\bar{t},\alpha,n\}\ (\text{shape})\ ]$$

$$\Delta \mathbf{C}^{\mathbf{h}}[\ \{\bar{p},\bar{t},\alpha,n\}\ (\text{shape})\ ]$$



These regulators organize the output shapes along curves or surfaces. Curve regulators have numerous subtypes, for example, elliptical $\Delta \mathbf{C}^{\mathbf{e}}$ and hyperbolic $\Delta \mathbf{C}^{\mathbf{h}}$. Cubic curves are being developed.

To further augment the universe of possible generated shapes, a category of regulators based on non-linear topological transformations (deformations), symbolized by $\Delta \mathbf{F}$, is currently being investigated.

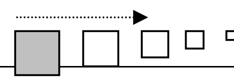
3.2 COMPOSITION OF REGULATORS

To represent the diverse types of structures observed in geometric configurations, ICE supports various ways of composing with regulators. Table 3 illustrates the three main types and their notation.

Table 3. Composition of Regulators

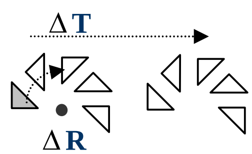
Simultaneous composition. Here, several regulator formulae are applied to the same set of elements; for example, glide and screw rotation use simultaneous composition. Parameters for the composite are the union of the individual regulator parameters, with duplicates differentiated by subscripts.

$$\Delta \mathbf{T}^{1} \Delta \mathbf{D}^{0}[\ \{\bar{p}_T, \bar{p}_D, \bar{t}, \bar{k}, d, n\}\ (\text{shape})\ ]$$



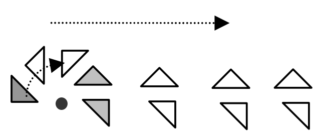
Successive composition. Here, a regulator is applied to the output shapes of another regulator, forming a nested relationship. Notionally, successive compositions correspond to nested parenthesized strings in which inner regulators are applied before outer regulators.

$$\Delta \mathbf{T}^{1}[\ \{\bar{p}, \bar{t}, d, n\}\ ($$
$$\Delta \mathbf{R}^{1}[\ \{\bar{p}, \bar{t}, \alpha, n\}\ (\text{shape})]\ \ )]$$



Partial composition. Here, a regulator is applied to a subset of a previously generated output. Notionally, this is indicated by a subscripted string comprising the # symbol followed by the output shape indices.

$$\Delta \mathbf{T}^{1}[\ \{\bar{p}, \bar{t}, d, n\}\ ($$
$$\Delta \mathbf{R}^{1}[\ \{\bar{p}, \bar{t}, \alpha, n\}\ (\text{shape})]_{\#3,4}\ \ )]$$



3.3 GENERATION METHODS

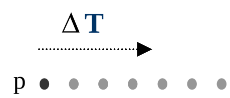
ICE supports several methods of generating shapes as complex subparts of simple shapes: continuous, discrete, combination, subset, and pattern generation. This feature is indicated by superscripts, for instance, $\Delta \mathbf{T}(\mathrm{p})^{<0-3><6-9>}$. Angle brackets group continuous parts together, and the dash indicates that all shapes/points within the range are generated.

Table 4. Generation Methods

Discrete generation. Here, output shapes are not connected. The examples in tables 1 and 2 are generated discretely.

$\Delta\mathbf{T}(\mathrm{p})^{<0><1><2><3><4><5><6>}$ or

$\Delta\mathbf{T}(\mathrm{p})^{<0>-<6>}$



Continuous generation. Here, output shapes are connected and the loci of points in-between the output-shapes are also generated. Continuous generation is used for creating shapes from connected vertices. The shape examples in table 5 are generated continuously.

$\Delta\mathbf{T}(\mathrm{p})^{<0,1,2,3,4,5,6>}$ or

$\Delta\mathbf{T}(\mathrm{p})^{<0-6>}$

Combined generation. Combined generation includes both continuous and discrete parts. It is used for generating shapes that have disconnected parts.

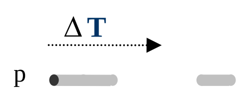
$\Delta\mathbf{T}(\mathrm{p})^{<0,1,2,3><4><5,6>}$ or

$\Delta\mathbf{T}(\mathrm{p})^{<0-3><4><5-6>}$

Subset generation. In subset generation, only some indices (from 0-*n*) are generated; thus, gaps are created, not by discontinuity as in the previous method, but by absence of an output shape/point.

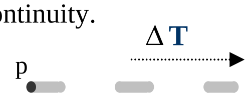
$\Delta\mathbf{T}(\mathrm{p})^{<0,1,2><5,6>}$ or

$\Delta\mathbf{T}(\mathrm{p})^{<0-2><5-6>}$



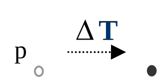
Pattern generation. Pattern generation is intended to describe repetitive patterns, for instance a dashed line, in a concise manner. The symbol ∴ indicates the start of the pattern, *i* denotes a generated index, and the ϕ indicates an absent index, and the brackets indicate continuity.

$\Delta\mathbf{T}(\mathrm{p})^{\therefore <ii>\phi}$



Non-generative Transformation. Regulators can be used to work non-generatively, i.e., these transform the input shape. This is indicated in the subscript by only including the generated index.

$\Delta\mathbf{T}(\mathrm{p})^{<i>}$



3.4 SHAPE GENERATION

By using the continuous generation method, a variety of simple shapes (Table 5) and complex shapes (Table 6) can be generated by means of regulators.

Table 5. Examples of simple shape generation

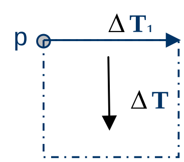
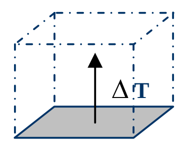
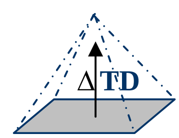
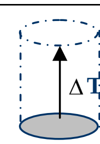
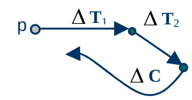
Straight line: $\Delta\mathbf{T}(\mathrm{p})$ The translation regulator $\Delta\mathbf{T}$ sweeps the starting point *p* to generate a straight line.	
Curved line: $\Delta\mathbf{C}(\mathrm{p})$ The curve regulator $\Delta\mathbf{C}$ sweeps *p* to create a curved line.	
Plane: $\Delta\mathbf{T}_2(\Delta\mathbf{T}_1(\mathrm{p}))$ A plane is generated by a composition of two translation regulators. The second translation inputs all the output points of the first.	
Prism: $\Delta\mathbf{T}_3(\mathrm{base}) \wedge \mathrm{base} = \Delta\mathbf{T}_2(\Delta\mathbf{T}_1(\mathrm{p}))$ A prism is generated by sweeping a square base along a translation regulator $\Delta\mathbf{T}$.	
Pyramid: $\Delta\mathbf{TD}(\mathrm{base})$ A pyramid is generated the simultaneous composition of a translation regulator $\Delta\mathbf{T}$ and a dilation regulator $\Delta\mathbf{D}$. If the scale factor is increased, the result is a frustum.	
Sphere: $\Delta\mathbf{R}_2(\mathrm{circle}) \wedge \mathrm{circle} = \Delta\mathbf{R}_1(\Delta\mathbf{T}(\mathrm{p}))$ A sphere is generated by sweep rotating a circle.	
Cylinder: $\Delta\mathbf{T}(\mathrm{circle})$ A cylinder is generated by sweeping a circular base along a translation regulator $\Delta\mathbf{T}$.	
Cone: $\Delta\mathbf{TD}(\mathrm{circle})$ A cone is generated by sweeping a circle along a translation regulator $\Delta\mathbf{T}$ composed with a dilation regulator $\Delta\mathbf{D}$.	

Table 6. Complex shape examples

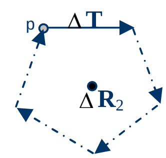
Polyline: $\Delta\mathbf{C}(\Delta\mathbf{T_1}(\Delta\mathbf{T_1}(\mathrm{p})_{\#n})_{\#n})$

A polyline is generated by successive compositions; each regulator inputs the last point of the preceding regulator.



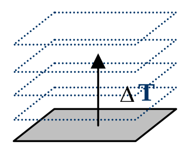
Regular polygon: $\Delta\mathbf{R}(\Delta\mathbf{T}((\mathrm{p})^{<0-n>})^{<0>-<4>})$

An outline polygon is generated by translating a point to construct a line, then by rotating the line, discretely, to construct the remaining sides.



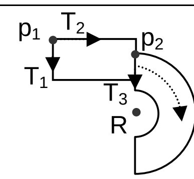
Parallel planes = $\Delta\mathbf{T}(\mathrm{square})^{<0><4>}$

A series of parallel planes are generated by the discrete method.



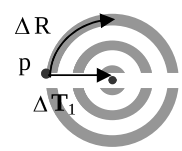
complex_shape = $\Delta\mathbf{T_2}(\Delta\mathbf{T_1}(\mathrm{p}_1)) \;\wedge\; \Delta\mathbf{R}(\Delta\mathbf{T_3}(\mathrm{p}_2))$

A complex shape composed of several independent parts.



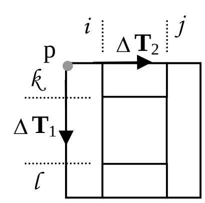
sub_shape = $\Delta\mathbf{R}(\Delta\mathbf{T}\,(\mathrm{p})^{\therefore<iii>\phi})^{<5-175><185-365>}$

A complex subshape of a simple known shape is generated by means of the subset generation method.



Square donut: $\Delta\mathbf{T}_2\begin{bmatrix}\left(\mathbf{T}_1(p)^{<0-n_1>}\right)^{<0-i><j-n_2>}\\ \left(\mathbf{T}_1(p)^{<0-k>}\right)^{<i-j>}\\ \left(\mathbf{T}_1(p)^{<l-n_1>}\right)^{<i-j>}\end{bmatrix}$

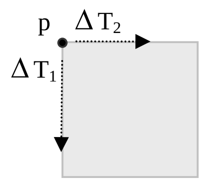
A square donut is described by using the subset generation method. The outputs of $\Delta\mathbf{T}_1$ are treated differently in $\Delta\mathbf{T}_2$. ICE supports several ways of generating this shape.

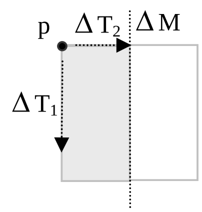


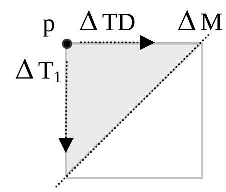
3.5 MULTIPLE REPRESENTATIONS

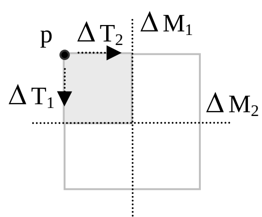
Although the ICE notation is not ambiguous, it is also not unique: a configuration can be represented in several ways. For instance, Table 7 shows a few of the many ways that a simple square can be generated in ICE. This feature affects the transformations applicable to a shape. The ICE notation describes linear shapes, planar shapes, and solid shapes. Solid shapes can be described by means of their vertices, edges, or surfaces. The subset generation method enables any subshape, including boundaries, to be derived from the solid representation.

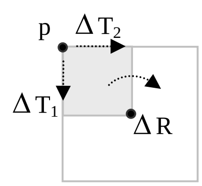
Table 7. Multiple Representations

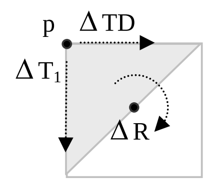
<table>
<tr>
<td>

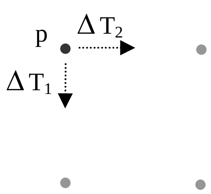
$\Delta\mathbf{T_2}(\Delta\mathbf{T_1}(\mathrm{p}))$</td>
<td>

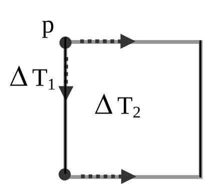
$\Delta\mathbf{M}(\Delta\mathbf{T_2}(\Delta\mathbf{T_1}(\mathrm{p})))$</td>
<td>

$\Delta\mathbf{M}(\Delta\mathbf{TD}(\Delta\mathbf{T_1}(\mathrm{p})))$</td>
</tr>
<tr>
<td>

$\Delta\mathbf{M}(\Delta\mathbf{M}(\Delta\mathbf{T_2}(\Delta\mathbf{T_1}(\mathrm{p}))))$</td>
<td>

$\Delta\mathbf{R}(\Delta\mathbf{T_2}(\Delta\mathbf{T_1}(\mathrm{p})))$</td>
<td>

$\Delta\mathbf{R}(\Delta\mathbf{TD}(\Delta\mathbf{T_1}(\mathrm{p})))$</td>
</tr>
<tr>
<td>

$\Delta\mathbf{T_2}(\Delta\mathbf{T_1}(\mathrm{p})^{<0><n>})^{<0><n>}$
Discrete generation derives the bounding vertices.</td>
<td colspan="2">

$$\Delta\mathbf{T}_2\begin{bmatrix}\left(\mathbf{T}_1(p)^{<0><n>}\right)^{<0-n>}\\ \left(\mathbf{T}_1(p)^{<0-n>}\right)^{<0><n>}\end{bmatrix}$$

Subset generation derives the bounding edges by (1) sweeping the first and last points of $\Delta\mathbf{T}_1$ though $\Delta\mathbf{T}_2$, and (2) copying the line generated by $\Delta\mathbf{T}_1$ along $\Delta\mathbf{T}_2$.</td>
</tr>
</table>

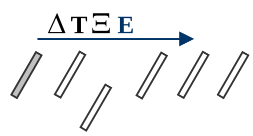
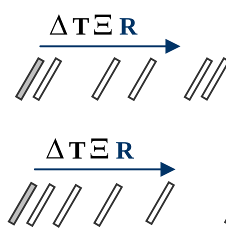
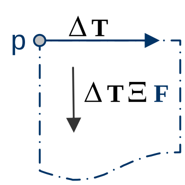
4. Secondary generation and relational constructs: variational, constraint, and operational regulators

The previous section described the primary generative building blocks of the ICE notation. This section describes the secondary regulators that are dependent on the primary ones for their functionality: these are either composed with the primary regulators, or applied to shapes (defined by primary regulators) and consequently affect the primary regulators. These include variational, constraint, hierarchical, and operational regulators.

4.1 VARIATIONAL REGULATORS

Variational regulators, (Table 8) symbolized by Ξ, are composed with generative regulators to create a variation in the output shapes by controlling shape attributes or regulator parameters.

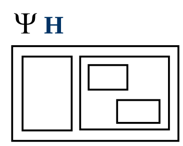
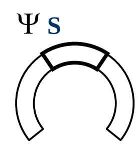
Table 8. Variation regulators

Exception: $\Xi\mathbf{E}\ [\ \{\bar{a},\bar{v}\}\ (\text{shape}_0 - \text{shape}_n)\ ]$ This regulator sets one or more shapes to be an exception to the output set by overriding an attribute *a*, (for instance, position) with a value *v*.	
Rhythm/Gradation: $\Xi\mathbf{R}\ [\ \{a, \boldsymbol{f}, c\}\ (\text{shape}_0 - \text{shape}_n)\ ]$ This regulator creates a rhythm/gradation effect within the output shapes, by applying a formula **f**, and coefficient *c*, to attribute a of the generative regulator (for instance, the translation distance) as it is applied to the output shapes.	
Differential: $\Xi\mathbf{F}\ [\ \{a, \boldsymbol{f}, c\}\ (\text{shape}_0 - \text{shape}_n)\ ]$ This regulator creates a variation in the output, by sweeping/copying the elements of input set differently. It applies the formula **f,** and coefficient *c***,** to attribute *a* of the generative regulator as it is applied to the input shapes. This regulator is effective only for many input shapes.	

4.2 HIERARCHICAL REGULATORS

Hierarchical regulators symbolized by Ψ, define hierarchies of shapes; these can be defined independently, or in composition with generative regulators. Table 9 illustrates the two hierarchical regulators supported in ICE.

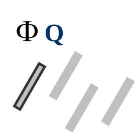
Table 9. Hierarchical Regulators

Containment: $\Psi\mathbf{H}\ [\ \{\}\ (\text{container}, \text{constituent}_0 - \text{constituent}_n)\ ]$ The containment regulator creates a container-constituent relationship, independent of geometry. Typically, containment inputs the container and constituents, however it can input the container and generate the constituents, or vice versa.	
Subshape: $\Psi\mathbf{S}\ [\ \{\}\ (\text{supershape}, \text{subshape}_0 - \text{subshape}_n)\ ]$ The subshape regulator creates a geometric dependency between shapes (or more precisely between their generative regulators).	

4.3 CONSTRAINT REGULATORS

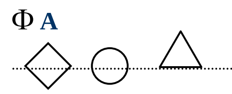
Constraint regulators, symbolized by Φ, bound shapes or define relationships between shapes. Since shapes in the ICE notation are defined by regulators, the constraints are ultimately applied to parameters of the generative regulators. Constraints can be defined independently or in composition with generative regulators. Although some compositions involving constraint regulators may cause conflicts, there will be no restrictions placed on composition until all possibilities are thoroughly explored. Table 10 illustrates the constraints supported in ICE.

Table 10. Constraint Regulator

Attribute equivalence: $\Phi\mathbf{Q}\ [\ \{a, v\}\ (\text{shape}_0 - \text{shape}_n)\ ]$ The attribute equivalence regulator assigns a value *v* to an attribute *a* (for instance color) of a shape.	

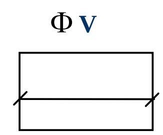
Alignment: $\Phi\mathbf{A}^2[\ \{\bar{p},\bar{t},\bar{v}\}\ (\text{shape}_0 - \text{shape}_k)\ ]$

There are three subtypes of alignments: $\Phi\mathbf{A}^0$ restricts an object to a point, $\Phi\mathbf{A}^1$ aligns an object to a line, and $\Phi\mathbf{A}^2$ aligns an object to a plane.



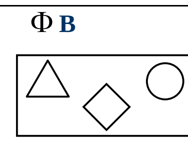
Dimension: $\Phi\mathbf{V}\ [\ \{min, max, mod\}\ (\text{shape})\ ]$

There are three subtypes for this regulator: $\Phi\mathbf{V}^1$ restricts length, $\Phi\mathbf{V}^2$ restricts area**,** and $\Phi\mathbf{V}^3$ restricts volume. The parameters are minimum value, maximum value and an incremental module.



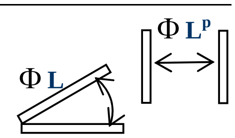
Boundary: $\Phi\mathbf{B}^2[\ \{o\}\ (\text{shape}_{\text{bound}}, \text{shape}_1 - \text{shape}_k)]$

The boundary regulator defines a legal region for a shape, within an offset o.



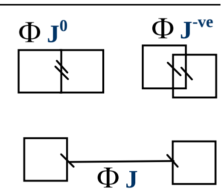
Angle: $\Phi\mathbf{L}\ [\ \{min, max, mod\}\ (\text{shape}_1 - \text{shape}_k)\ ]$

This regulator sets the angle between two shapes. A variant $\Phi\mathbf{L^p}$ sets shapes as being parallel.



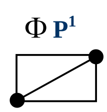
Distance: $\Phi\mathbf{J}\ [\ \{min, max, mod\}\ (\text{shape}_1, \text{shape}_k)\ ]$

This regulator defines the proximity between shapes. A zero distance is an adjacency $\Phi\mathbf{J}^0$, and a negative distance is an overlap $\Phi\mathbf{J}^{-ve}$



Proportion: $\Phi\mathbf{P}^1[\ \{\bar{p},\bar{t},d\}\ (\text{shape})\ ]$

This regulator controls the aspect ratio of a shape by means of a diagonal line.



4.4 OPERATIONAL REGULATORS

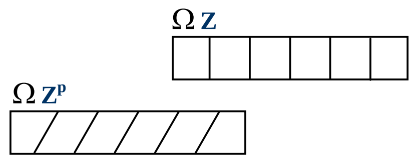
Operational regulators, symbolized by Ω, define complex shapes from simpler ones, as illustrated in table 11. Like constraints, operations are ultimately applied to a shape's generative regulator(s).

Table 11. Operations Regulators

Subdivision:

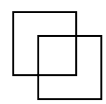
$\Omega\mathbf{Z}\ [\ \{s,n\}\ (\text{shape})\ ]$

$\Omega\mathbf{Z}^{\mathbf{P}}[\ \{s,n\}\ (\text{shape},\text{plane})\ ]$



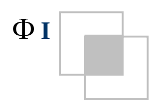
The subdivision regulator inputs a shape and subdivides it n times and allocates a spacing s between the sudivisions. There are two variants of subdivision: $\Phi\mathbf{Z}$'s subdivisions are normal to the shape's direction (i.e., to the generative regulator's direction); $\Phi\mathbf{Z}^{\mathbf{P}}$ subdivides according to a splitting plane.

The Boolean operations regulators input two or more objects, and generate their union, intersection, or difference. Boolean regulators have no parameters, but control the generative properties of the shape's primary regulators.



Union: $\Omega\mathbf{U}\ [\ \{\}\ (\text{shape}_0 - \text{shape}_k)\ ]$

Intersection: $\Omega\mathbf{I}\ [\ \{\}\ (\text{shape}_0 - \text{shape}_k)\ ]$



Difference: $\Omega\mathbf{D}\ [\ \{\}\ (\text{shape}_0 - \text{shape}_k)\ ]$

Symmetric Difference: $\Omega\mathbf{M}\ [\ \{\}\ (\text{shape}_0 - \text{shape}_k)\ ]$

5. Transformations in ICE.

The ICE notation is also designed to capture transformations that are applicable to configurations. The ICE notation supports transformations at four levels: (i) regulated element; (ii) generation method; (iii) regulator parameters; and (iv) regulator composition. These are illustrated in Table 12.

Notice that simple notation changes represent significant changes in the geometry.

Transforming the regulated element modifies the configuration while maintaining its geometric structure. Transformations in this category include moving the point, and moving-rotating-scaling the shape.

Transforming the generation method creates variations and subshapes, but maintains the geometric structure of the configuration. Transformations in this category include changing the number of elements generated, changing the discrete continuous properties, changing the generated subset, and changing the generation pattern.

Transforming the parameters regulator modifies the configuration's geometric structure but not the notation's structure. Transformations in this category include changing the regulator's geometry by moving it or rotating it, and changing the major parameter such as rotation degree or minimum-maximum value.

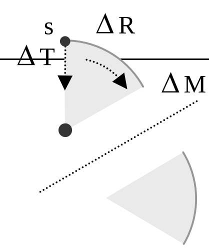
Transforming the regulator composition redefines the notation string and completely alters the configuration's structure. Transformations in this category include adding, composing, inserting, deleting, replacing, or reordering regulators in a sequence.

The notation string captures all applicable transformations on each of these four levels. The various symbols, parameters, and indices of the notation represent manipulation handles for the ICE system. The notation string can also capture the most parsimonious process for generating a configuration. When the string is reconfigured, the set of applicable transformation handles and the generation process can be redefined.

The notation however, does not capture an exploratory process, i.e., one that proceeds by means of transforming the notation string until a satisfactory configuration is achieved.

The multiple representation property of the notation allows for different strings to represent the same configuration. Therefore, it supports different processes to generate it and different transformations for it. Thus different exploration paths can be achieved for the same configuration.

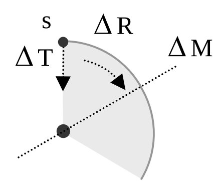
Table 12. Transformations captured in ICE

Initial configuration:	

$$s = (0 \quad 3 \quad 0)$$
$$\Delta\mathbf{M}\ [\ \{\overline{p},\bar{t},d,n\}\ ($$
$$\Delta\mathbf{R}\ [\ \{\overline{p},\bar{t},\alpha,n\}\ ($$
$$\Delta\mathbf{T}[\ \{\overline{p},\bar{t},d,n\}\ (s)^{<0-n>}\ ]\ )^{<0-n>}\ ]\ )^{<0><1>}\ ]$$

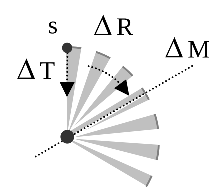
Transformation of regulated elements:
Moving the input shape:

$$s = (0 \quad 3 \quad 0) \Rightarrow s = (0 \quad 2 \quad 0)$$
$$\mathbf{R}.p = (0 \quad 1 \quad 0) \Rightarrow \mathbf{R}.p = (0 \quad 0 \quad 0)$$
$$\Delta\mathbf{M}\ [\ \{\overline{p},\bar{t},d,n\}\ ($$
$$\Delta\mathbf{R}\ [\ \{\overline{p},\bar{t},\alpha,n\}\ ($$
$$\Delta\mathbf{T}[\ \{\overline{p},\bar{t},d,n\}\ (s)^{<0-n>}\ ]\ )^{<0-n>}\ ]\ )^{<0><1>}\ ]$$



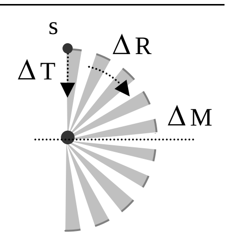
Transformation of generation method
Changing the generation method of $\Delta\mathbf{R}$ to pattern

$$\Delta\mathbf{M}\ [\ \{\overline{p},\bar{t},d,n\}\ ($$
$$\Delta\mathbf{R}\ [\ \{\overline{p},\bar{t},\alpha,n\}\ ($$
$$\Delta\mathbf{T_1}[\ \{\overline{p},\bar{t},d,n\}\ (s)^{<0-n>}\ ]\ )^{\therefore<0-3>\phi}\ ]\ )^{<0><1>}\ ]$$



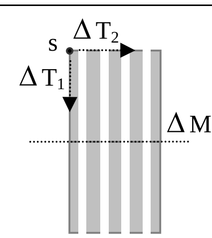
Transformation of regulator parameters
Rotate M: $\Delta\mathbf{M}.t = [2 \quad 1 \quad 0] \Rightarrow \Delta\mathbf{M}.t = [1 \quad 0 \quad 0]$

$$\mathbf{M}\ [\ \{\overline{p},\bar{t},d,n\}\ ($$
$$\mathbf{R}\ [\ \{\overline{p},\bar{t},\alpha,n\}\ ($$
$$\mathbf{T_1}[\ \{\overline{p},\bar{t},d,n\}\ (s)^{<0-n>}\ ]\ )^{\therefore<0-3>\phi}\ ]\ )^{<0><1>}\ ]$$



Transformation of regulator composition
Replacing the regulator $\Delta\mathbf{R}$ by $\Delta\mathbf{T_2}$

$$\mathbf{M}\ [\ \{\overline{p},\bar{t},d,n\}\ ($$
$$\mathbf{T_2}[\ \{\overline{p},\bar{t},d,n\}\ ($$
$$\mathbf{T_1}[\ \{\overline{p},\bar{t},d,n\}\ (s)^{<0-n>}\ ]\ )^{\therefore<0-3>\phi}\ ]\ )^{<0><1>}\ ]$$

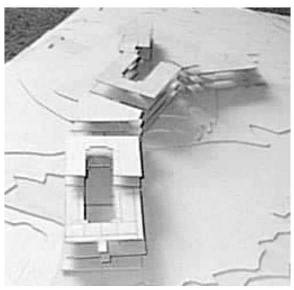
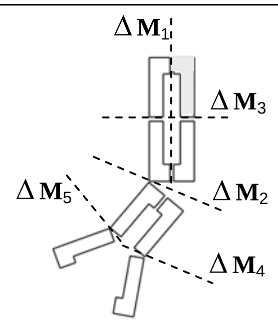
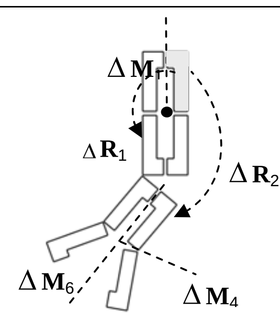


6. Architectural Design Examples

6.1 DESIGN STUDIO EXAMPLE

Akin and Moustapha (forthcoming) have used the ICE notation to codify a complete sequence of drawings from an annotated design studio. Table 13 shows one drawing of a dormitory project submission, as represented by the ICE notation in two distinct ways. The two distinct notations capture two distinct generation processes, as well as two distinct sets of applicable transformations.

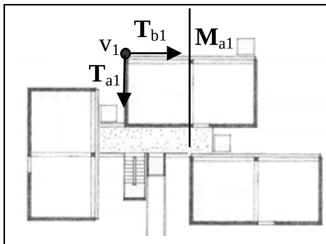
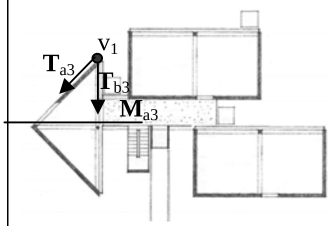
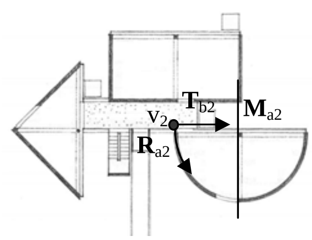
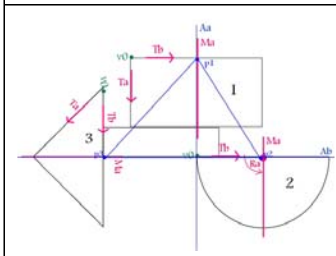
Table 13 An example from the annotated studio

	
Generation alternative 1: $\mathbf{M_3}[\ \{\bar{p},\bar{t},\bar{v},n\}\ ($ $\mathbf{M_2}[\ \{\bar{p},\bar{t},\bar{v},n\}\ ($ $\mathbf{M_1}[\ \{\bar{p},\bar{t},\bar{v},n\}\ (\text{dormUnit}_0)]_{\#1})\ ]\ )]$ $\mathbf{M_4}[\ \{\bar{p},\bar{t},\bar{v},n\}\ (\text{dormUnit}_4)]$ $\mathbf{M_5}[\ \{\bar{p},\bar{t},\bar{v},n\}\ (\text{dormUnit}_5)]$	
Generation alternative 2: $\mathbf{R_1}[\ \{\bar{p},\bar{t},\alpha,n\}\ ($ $\mathbf{M_1}[\ \{\bar{p},\bar{t},\bar{v},n\}\ (\text{dormUnit}_0)]\ )\ ]$ $\mathbf{M_6}[\ \{\bar{p},\bar{t},\bar{v},n\}\ ($ $\mathbf{M_4}[\ \{\bar{p},\bar{t},\bar{v},n\}\ ($ $\mathbf{R_2}[\ \{\bar{p},\bar{t},\alpha,n\}\ (\text{dormUnit}_0)]\ )\ ]\ )]$	

6.2 HYPOTHETICAL EXAMPLE HEJDUK'S HALF HOUSE

Moustapha (2003) encoded the hypothetical generation of Hejduk's half house from a rectangle. Due to the length of this example, snapshots of the half house generation are included as a simplified abstraction (Table 14). The generation was exploratory, i.e., it used transformations to generate alternatives. The three rectangle configuration was transformed into the half house simply by replacing and adjusting regulators.

Table 14. A hypothetical example: Hejduk's Half House

<table>
<tr>
<td>
</td>
<td>Initial configuration: Three rectangles
Rectangle1 =
$\Delta \mathbf{M}_{a1}[\ \{\bar{p},\bar{t},d,n\}\ ($
$\Delta \mathbf{T}_{b1}[\ \{\bar{p},\bar{t},d,n\}\ ($
$\Delta \mathbf{T}_{a1}[\ \{\bar{p},\bar{t},d,n\}\ (\mathrm{v}_1)_{\#1,n}\ ]\ )]\ )]$</td>
</tr>
<tr>
<td>
</td>
<td>Transform Rectangle3 to Triangle3
• Update $\mathbf{T}_b$ in the sequence such that it only applies to v_3 and Rotate $\mathbf{T}_a$ 45°
Triangle3 = $\mathbf{M}_{a3}[\ \{\bar{p},\bar{t},d,n\}\ ($
$\mathbf{T}_{a3}[\ \{\bar{p},\bar{t},d,n\}\ (\mathrm{v}_3)]+$
$\mathbf{T}_{b3}[\ \{\bar{p},\bar{t},d,n\}\ (\mathrm{v}_3)]\ )]$</td>
</tr>
<tr>
<td>
</td>
<td>Transform rectangle 2 to semicircle 2
• Update $\mathbf{T}_b$ in the sequence such that it only applies to v_2 and Replace $\mathbf{T}_a$ by $\mathbf{R}_a$
Semicircle2 = $\mathbf{M}_{a2}[\ \{\bar{p},\bar{t},d,n\}\ ($
$\mathbf{R}_{a2}[\ \{\bar{p},\bar{t},\alpha,n\}\ (\mathrm{v}_2)]+$
$\mathbf{T}_{b2}[\ \{\bar{p},\bar{t},d,n\}\ (\mathrm{v}_2)]\ )]$</td>
</tr>
<tr>
<td>
</td>
<td>Half House =
Rectangle1 $\wedge$ Semicircle2 $\wedge$ Triangle3
$\Delta \mathbf{A}^{\mathbf{0}}[\ \{\bar{p}_1\}\ (\text{Rectangle1}.T_b.\text{mid})\ ]$
$\Delta \mathbf{A}^{\mathbf{0}}[\ \{\bar{p}_3\}\ (\text{Triangle3}.T_b.\text{mid})\ ]$
$\Delta \mathbf{A}^{\mathbf{0}}[\ \{\bar{p}2\}\ (\text{Semicircle2}.T_b.\text{mid})\ ]$</td>
</tr>
</table>

7. Conclusion

The ICE notation can represent a geometric configuration as a string that expresses generative and relational structures. By accommodating various methods of composition and various methods of generation, the ICE notation expands the universe of possible shapes and configurations that it can describe. This representation succinctly captures the complete configuration. It also captures the generative process in the most parsimonious manner. Furthermore, it is possible to determine further information about configurations from notation strings. For instance, given the string of generative regulators, one can compute subshapes, boundaries, lengths, areas, volumes and midpoints, just by evaluating their parameters and indices. It is also possible to determine the transformations applicable to a configuration through its notation string. Whether these transformations are achieved through adjusting parameters or through redefining the notation string, it is possible to designate manipulation handles for these transformations.

Since the ICE notation is developed in parallel to the ICE generative/interactive system, every regulator described in this paper, has been implemented or is currently being implemented in the computer medium. The ICE system supports regulators as manipulation handles, therefore the models it generates are highly flexible. Their parameters as well as their generative sequence can be redefined at runtime and in real time. Describing a complex geometrical configuration through a concise string has the additional computational advantages of minimizing the size of storage, and maximizing the speed of file transfer. For the aforementioned reasons of completeness, information processing, flexibility, and computational efficiency, the ICE system promises to be a valuable tool for design exploration.

It is important to note that ICE is not the only notation in its class. Leyton (2001) developed a generative theory of shape, which uses similar fundamental principles of mathematics as ICE. Leyton also addressed the issue of process-capture, which he refers to as recoverability. Cha and Gero (2001) have developed a shape pattern representation, based on isometry transformations and used it to describe numerous notable buildings. While Leyton's work, focused on theoretical aspects of shape generation, and Cha-Gero's work focused on pattern description, the ICE system/notation focuses on the practical aspects of implementation and interaction, the most important of which is the ability to manipulate the configurations generated by the ICE notation.

In addition to being part of a computational system, the ICE notation extends the aforementioned representations in the following ways:

- ICE is designed to work in 3D. All parameters and operations in ICE are based on 3-dimensional geometry principles.
- The regulator construct in ICE subsumes generative transformations, and encode other functions such as constraints, operations, hierarchies and variations.
- Support for continuous, discrete and sub-part generation methods is a unique feature of ICE. This allows for complex shape description and maximizes the possibilities for shape generation.
- The property of deriving boundary, sub-part information, and other geometric information from a given ICE string, augments the ICE notation from being merely a geometry descriptor to that of a geometry processor.
- Support for different levels of information simplifies the representation. Short and long forms allow the ICE notation string to be viewed at two crucial levels of abstraction: relational and parameterization). The shape encapsulation feature of ICE helps structure the notation string and avoids redundancy of description.

This notation will continue to be refined and improved. In particular, the rigorous mathematical aspects of regulators, regulator relationships, and geometry processing algorithms will be addressed in future work.

Acknowledgements

I would like to acknowledge my advisors Professors Ömer Akin and Ramesh Krishnamurti for their constant guidance and support during the development of this work.

References

Akin, O, Moustapha, H: forthcoming, Formalizing Generation and Transformation in Design, *Proceedings of the Design and Cognition Conference, MIT, Cambridge, MA,* 2004, in print

Cha, M, Gero, J: 2001, Shape Pattern Representation for Design Computation, *Automation in construction, Elsevier, New York.*

Leyton, M: 2001, A Generative Theory of Shape. *Springer Verlag, New York*.

Moustapha, H, Krishnamurti, R: 2001, Arabic Calligraphy: A Computational Exploration, *Mathematics and Design 2001, Third International Conference, Deakin University, Geelong.*

Moustapha, H: 2003, From Rectangle to Half House to House 10. *Unpublished document*.