PHD DISSERTATION

ARCHITECTURAL EXPLORATIONS

A FORMAL REPRESENTATION FOR THE GENERATION AND TRANSFORMATION OF DESIGN GEOMETRY

HODA MOUSTAPHA

CARNEGIE MELLON UNIVERSITY
SCHOOL OF ARCHITECTURE
COMPUTATIONAL DESIGN PROGRAM

THESIS COMMITTEE:

OMER AKIN
RAMESH KRISHNAMURTI
ULRICH FLEMMING
BERND BRUEGGE

ARCHITECTURAL EXPLORATION

A FORMAL REPRESENTATION FOR THE GENERATION AND TRANSFORMATION OF DESIGN GEOMETRY

HODA MOUSTAPHA

CARNEGIE MELLON UNIVERSITY
COLLEGE OF FINE ARTS
SCHOOL OF ARCHITECTURE
COMPUTATIONAL DESIGN PROGRAM
SEPTEMBER 2005

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

ARCHITECTURAL EXPLORATIONS

A FORMAL REPRESENTATION FOR THE GENERATION AND TRANSFORMATION OF DESIGN GEOMETRY

HODA MOUSTAPHA

CARNEGIE MELLON UNIVERSITY
COLLEGE OF FINE ARTS
SCHOOL OF ARCHITECTURE
COMPUTATIONAL DESIGN PROGRAM

SEPTEMBER 2005

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

OMER AKIN - PRINCIPAL ADVISOR

RAMESH KRISHNAMURTI - ADVISOR

ULRICH FLEMMING - ADVISOR

BERND BRUEGGE - ADVISOR

ACKNOWLEDGEMENTS

I would like to acknowledge all those who contributed to my intellectual journey at Carnegie Mellon University.

I begin by offering my sincere thanks to my advisor Professor Ömer Akin, who provided me with exceptional guidance throughout the years, and who always encouraged me to pursue my interests to their full potential. I would like to thank my committee members: Professor Ramesh Krishnamurti for his continued inspiring suggestions, as well as Professor Ulrich Flemming and Professor Bernd Bruegge for their crucial feedback.

I offer my thanks to Darlene Covington-Davis for her constant moral support, especially during difficult times. I thank Judith Kampert, and Elizabeth Fox for their friendship and support. I thank my friends and colleagues in the Graduate Programs at the School of Architecture for their constant encouragement and intellectual exchanges; in particular Dr. Safwan Ali and Dr. Magd Donia for helping me adjust to life in Pittsburgh. I also would like to thank Professors Kristen Kurland, Susan Finger, Mark Gross, and Ellen Do for their friendship and encouragement.

I address my thanks to Professors Volker Hartkopf, Vivian Loftness, Azizan Abdul-Aziz, and Khee-Poh Lam for their financial support during my Ph.D. studies.

I am greatly indebted to His Royal Highness Prince Bandar Bin Sultan and to Her Royal Highness Princess Haifa Al-Faisal for financing the initial phases of my pursuit of knowledge at Carnegie Mellon University.

And last but not least I would like to thank my father, Dr. Ahmed Moustapha, who has always been a role model and a constant source of motivation. As for my mother, Dr. Djehane Hassouna, there will never be enough words to describe all she has done and is still doing for me, and to express my never-ending gratitude.

ABSTRACT

Design is often described as an exploration: a search for an adequate solution amongst a space of alternatives (Simon 1969). It involves the development and transformation of alternatives. Composition of spatial relations such as symmetry, hierarchy, grid-alignment, and proportion, termed "design structures," help in the development of these architectural alternatives, and are used as compositional principles during their transformation.

During the exploratory, early phases of design, configurations continually evolve. Both configuration elements and their relational structure are subject to change. Defining structures, exploring variations within structures, and redefining structures are the common means of transforming alternatives. Such transformations are necessary for developing design configurations and improving their quality.

Transformations of design structures, which often yield intellectually stimulating results, are labor intensive; they require individual modification of related elements. Such repetitive interaction considerably slows down the exploration, and often discourages it completely, particularly when configurations are complex and interrelations are numerous.

My research is motivated by the following factors: (i) the necessity of flexible geometry for early design exploration; (ii) the intellectual stimulation provided by the exploration of structure; (iii) the difficulty involved in transforming design structures; (iv) the lack of computational support for design exploration; and (v) the lack of comprehensive representations for architecturally significant design structures.

I have developed a framework of strategies that allows designers to explore complex configurations by manipulating their organizational structure. This framework, named Interactive Configuration Exploration (ICE), consist of two parallel endeavors: a notation and a computer implementation. The ICE notation is a formalism for describing shapes

and configurations, by means of their generative and relational structures. The ICE implementation is a 3D modeling system that supports the exploration of such shapes and configurations through the transformation of their structures.

The approach used is to separate the structures from configuration elements. In this manner, we can use structures to summarize configurations in the ICE notation, and use structures as manipulation handles to control the configuration in the ICE system.

The principal vehicle in ICE is the regulator, which is an abstraction that captures a single unit of structure (i.e., a single relationship within a configuration). For instance, a grid structure is captured by alignment lines; a symmetry structure is captured by a reflection axis or center of rotation. Regulators, which are inspired by regulating lines, encapsulate a mathematical formula that determines the relationship between elements.

The ICE notation enumerates and classifies the various types of regulators. It defines composition strategies and generation methods in order to represent the widest possible range of configurations. Furthermore, it captures a method for generation as well as a set of applicable transformations for any given configuration, based on its organizational structure. The ICE notation is not merely a geometric descriptor. It allows the derivation of additional geometric information, such as subshapes, boundaries, lengths, areas, volumes, and midpoints by means of simple computations on the notation strings. Additionally, it is possible to derive steps for transforming one configuration into another, by means of a simple algorithm.

The ICE system, offers a higher-level of interaction with design configurations though regulators. These regulators maintain control over configuration elements, thus imposing relational constraints and propagating changes within the configuration. The parameters of regulators are manipulation handles; therefore, a user can transform the configuration, either completely or partially, by applying simple changes to the regulator. Such explorations yield significant transformations with relatively short paths, since manipulating a spatial relation results in the simultaneous transformation of multiple elements.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The early phases of architectural design are characterized by exploratory activities and by iterative development. During these phases, concepts evolve, and ideas are explored through cycles of decisions, evaluations and transformations.

Among the numerous descriptive models of design, the most prominent is Simon's model of exploration. Simon describes the design process as a complex form of problem-solving, and categorizes design problems as ill-structured (Simon, 1969). Design is considered as a search for an adequate solution within a large space of alternative states. Designers navigate though this space by means of transitions that convert one alternative state into another, until the desired goal state is reached.

Akin (1987, p5) explains that problem structuring is a prerequisite to problem solving. It is the phase where the vague, ill-defined description of the problem is converted into a precisely defined, well-structured one. Archea (1987) refers to design structuring as puzzle making. The process of exploration is an important source for understanding the design problem (Harada 1997). The act of repeatedly generating and evaluating alternatives lead to the definition and refinement of requirement specifications, design relations, and constraints that need to be satisfied within the design solution.

Failure to satisfy these conditions leads to a restructuring of the problem, which takes the form of modifying relations and redefining specifications, thus transforming the problem's parameters (Akin 1987). Structuring defines the search space, while solving operates within its boundaries; restructuring, on the other hand, breaks the boundaries and redefines the search space. Restructuring represents a major iteration in design.

Architectural design products are complex geometric configurations, serving multiple functions simultaneously. These configurations are assemblies of parts, organized by

means of spatial relations—such as symmetries, grids, hierarchies, and adjacencies—to create unified coherent wholes. Compositions of such spatial relations define "design structures" that describe complex geometries, elaborate hierarchies, and intricate topologies within architectural configurations. Consider the Pantheon as an example (Figure 1.1, source: Ching 1996, p.288). A radial grid, a rectangular grid, rotational symmetry and mirror symmetry are composed together in harmony to define the design structure of the Pantheon. In Figure 1.1, the structure is indicated by regulating lines.

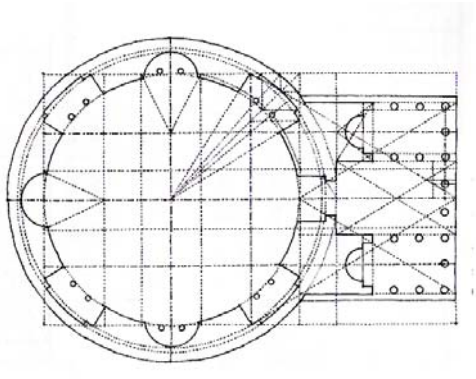


FIGURE 1.1- The Design Structure of the Pantheon

Design structures are not only fundamental to design products, they are also fundamental to the design process. The spatial relations, forming these structures, which are referred to as ordering principles (Ching 1996) and formative ideas (Clark 1985), are an essential part of architectural design toolkit. Designers deal with structures implicitly, by organizing the elements of the configuration, or explicitly, through regulating lines. Le Corbusier also prescribed the use of regulating lines to structure architectural configurations.

During structuring, designers compose the structures of architectural configurations. They define spatial relations and determine the boundaries of the search space. During solving, designers investigate variations within their structures. Their transformations are influenced by the spatial relations of the structure and remain within the limits of the search space. They manipulate the configuration while preserving the relationships and accentuating their structure. During restructuring, designers transform the structure of their configuration; their transformations reformulate the spatial relations of the structure, thus completely redefining the search space.

1.1. MOTIVATION

During the early phases of design, architectural configurations continually evolve. The conceptual dimension of design is flexible, where elements, as well as spatial relations are repeatedly being updated. However, current representations of design products, whether traditional, in the form of drawings or scaled models, or computational, are not flexible. These representations capture static snapshots of design states. They do not fully capture the transitions that formulate the conversion of one alternative state to another. In traditional media, there is no record of transformations except through inference. In computational media, there is a linear record of generative steps, but these cannot be captured and manipulated as such. Therefore, an essential exploratory component is missing in these representations.

The process of exploration is a method for refining and improving the design solution. It is a means of addressing the design problem from various perspectives. It allows more options to be considered and evaluated. It is a vehicle for innovation and discovery, in particular, exploration with structures. In additional to being a tool for transformation across design alternatives, exploring with structures is an intellectually stimulating experience. It leads to the discovery of new forms and compositions, as well as considering avenues that would not be otherwise considered.

Consider the Floor Plan of Frank Lloyd Wright's Lloyd Lewis House (Figure 1.2a, source: Laseau 1992, p.7). Figure 1.2b, 1.2c, and 1.2d show the effects of changing the directions of underlying grid lines or changing their curvatures. Furthermore, consider the ability to control each of these grid lines individually, and to perceive the effects on a certain group of elements. One can only begin to imagine the possibilities for such exploration, and the range of ideas that these may bring during early conceptualization phases.

Exploring structures is labor intensive in both traditional and computational media. Although design structures are implicit in the arrangement of configuration elements, transforming this implicit structure requires the designer to transform every element in the configuration, while managing, mentally, complex spatial relations between them. Such repetitive interaction considerably slows down the exploration, and often

discourages it completely, particularly when configurations are complex, and inter-relations numerous.

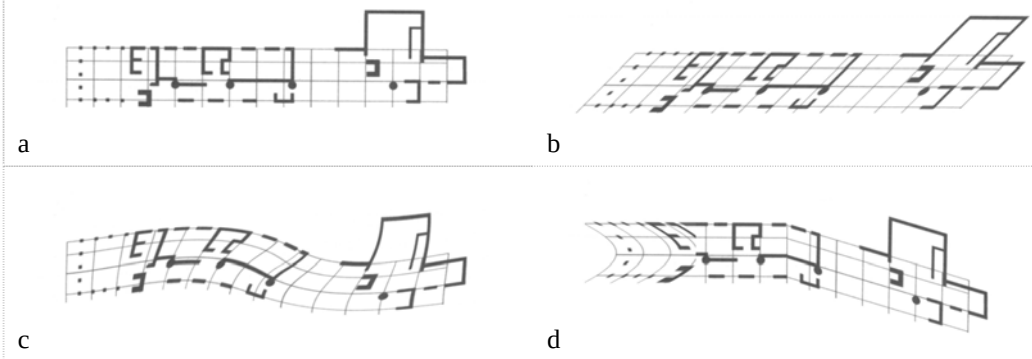


FIGURE 1.2 - Transformation of the Structure of Lloyd Lewis House

Computational design media, such as contemporary CAD systems, are typically conceived for representing final design products, not for exploratory activities. These do not support user-defined design structures and spatial relations. Instead, they capture independent design elements in the form of simple shapes and complex objects. Some computational research prototypes capture topological and hierarchical structures; however, these are system defined and cannot be easily transformed by users. Furthermore, architecturally significant spatial relations, such as symmetry, proportion, grids, etc., have not been sufficiently addressed, as exploration tools, in computational design research.

Despite the active role that structures play in design conceptualization, and although transformation of structures produce intellectually stimulating results, explorations by means of structures are not practically possible. This is due to the time and labor involved in re-organizing all elements within a structure.

Given the aforementioned motivational factors, I believe it is necessary to investigate methods for facilitating the exploration by means of structures, and for eliminating the labor and time involved in the transformation of structures, in order to make the geometric product as flexible as the conceptual dimension of design.

The primary goal of this research is to make the computational design environment a

source for intellectual stimulation. It is important to capture, computationally, the semantic complexity of design structures, and to provide support for transformation of structures, in an instantaneous, real-time manner. I believe that exploration by means of structures will allow the geometric representation to follow an evolving design concept. It will promote the discovery of new forms, new relations, and novel configurations that would not be otherwise explored. Furthermore, it will encourage the designer to explore a much larger space than would be possible with current tools and representations.

1.2. SCOPE

This research addresses computational representations for supporting geometric exploration during early phases of the architectural design process. The emphasis is on the structuring and restructuring activities pertaining to the geometry of architectural configurations. The focus is on using design structures as a primary construct for generation and transformation of alternative configurations.

This multidisciplinary research incorporates topics from mathematics, computation, software engineering and design processes. It addresses internal computational issues such as the flexible representation of structures, and the dependency between design structures and configuration elements. It also addresses front-end issues such as the designer's interaction with structures, the visual display of these structures, and the ability of structures to support exploratory iterative activities.

I do not prescribe specific methods for design. However, I suggest utilizing existing design strategies and augmenting them with computational power, in order to make explorations more effective.

1.3. RESEARCH SYNOPSIS

I have developed a framework of strategies that allows designers to explore complex configurations by manipulating their organizational structure. This framework, named Interactive Configuration Exploration (ICE), consist of two parallel endeavors: a notation and a computer implementation. The ICE notation is a formalism for describing shapes and configurations, by means of their generative and relational structures. The ICE implementation is a 3D modeling system that supports the exploration of such shapes and

configurations through the transformation of their structures.

The approach used is to separate the structures from configuration elements. In this manner, structures are used to summarize configurations in notation, and used as manipulation handles to control the configuration in the implementation.

The principal vehicle in ICE is the regulator, which is an abstraction that captures a single unit of structure (i.e. a single relationship within a configuration). For instance, a grid structure is captured by alignment lines; a symmetry structure is captured by a reflection axis or center of rotation. Regulators, which are inspired by regulating lines, encapsulate a mathematical formula that determines the relationship between elements.

The ICE notation enumerates and classifies various types of regulators. It defines composition strategies and generation methods in order to represent the widest possible range of configurations. Furthermore, it captures a method for generation as well as a set of applicable transformations for any given configuration, based on its organizational structure. The ICE notation is not merely a geometric descriptor. It allows the derivation of additional geometric information, such as subshapes, boundaries, lengths, areas, volumes, and midpoints by means of simple computations on the notation strings. Additionally, it is possible to derive steps for transforming one configuration into another, by means of a simple algorithm.

The ICE system allows a higher-level of interaction with design configurations through regulators. Regulators maintain control over configuration elements, thus imposing relational constraints, and propagating changes within the configuration. The parameters of regulators are manipulation handles; therefore, a user can transform the configuration, either completely or partially, by applying simple changes to the regulator. Such explorations through regulators yield significant transformations with relatively short paths, since manipulating a spatial relation results in the simultaneous transformation of multiple elements.

I envision that this work will impact the way design is carried out in numerous disciplines, particularly in architecture, through the descriptive capacity of the notation and the exploratory capacity of the implementation.

This remaining part of this document is organized as follows:

- Chapter 2: An overview of computational representations and research prototypes addressing various aspects of design exploration.
- Chapter 3: An overview of the various types of design structures observed in architecture, and a description of the regulator framework used for design exploration.
- Chapter 4: A detailed description of the syntax and building blocks and of the ICE notation, as well as an investigation of the various categories of shapes and configurations represented with the ICE notation.
- Chapter 5: a detailed description of derivational, properties and algorithmic additions to the ICE notation.
- Chapter 6: Architectural examples, including an ethnographic example illustrating the evolution of designs as represented by the ICE notation.
- Chapter 7: A complete description of the features of the ICE implementation system.
- Chapter 8: Contributions and speculations about future work.
- Appendices: These include the following: (i) the mathematical background for regulators; (ii) the representation of geometrical pattern and their transformations; (iii) the engineering of the ICE implementation; and (iv) the bibliography.

1.4. THESIS STATEMENT

Regulators are defined as abstractions, which capture the spatial relations organizing design elements into coherent configurations. In their mathematical form, regulators are effective in describing design configurations concisely and accurately. In their computational form, regulators are strategic tools for the iterative exploration of design configurations.

CHAPTER 2

BACKGROUND REVIEW:

COMPUTATIONAL REPRESENTATIONS FOR DESIGN EXPLORATION

In a computational environment, exploration is achieved by means of sharing design tasks between the user and the system, with the user focusing on exploring higher level ideas and the system managing lower level mechanisms. In the context of this research, where design relationships are the primary focus, representations would capture the relations among design entities, and systems would manage these relations. Representation of design relations can take many forms. Relations can be represented as constraints which can be encapsulated in grammar rules. Relations can also be captured in dependency hierarchies. Alternatively relations can be represented through hybrid combination of the aforementioned approaches. Additionally, relations can be represented as mathematical notations.

In this chapter, various approaches to representations used in exploration are surveyed. These include research prototypes (Sections 2.1 to 2.4), mathematical notations (section 2.6), and commercial systems (section 2.5). Research prototypes are further categorized as constraint-based representations, associative representations, design grammars representations and hybrid representations.

2.1. CONSTRAINT-BASED REPRESENTATIONS

In constraint-based representations, design relations as well as design attributes are represented as constraints. In constraint-based systems, users explore by manipulating design entities, while the system maintains the constraints and consequently maintains the integrity of the design.

2.1.1. SKETCHPAD (SUTHERLAND 1963)

SketchPad, a pioneering 2D graphic system, was the first to introduce interactive drawing, direct manipulation, and constraint satisfaction. It supports a drawing process consisting of two steps: (1) drawing simple 2D shapes (lines and circles) by means of a light pen; (2) refining the drawing by applying constraints to it. This process is inspired by the way a designer turns a sketch into a precise drawing. The vision of the SketchPad system includes the incorporation of design analysis and evaluation functionalities. SketchPad's generality allows it to accommodate numerous design domains ranging from artistic drawings to circuit design.

In SketchPad, design relations are defined by constraints, which are maintained upon the manipulation of drawing elements. Changes are recursively propagated among the repetitive subparts of the drawing. SketchPad's uses generic structure hierarchies, which group drawing objects, constraints, and commands according to their types. These allow SketchPad to be extensible, and to accommodate new drawing objects and new constraint types.

Constraint satisfaction is achieved either by the "one-pass method" or by the "relaxation method". Relaxation involves relaxing one or more constraints in the drawing and solving for the remainder of the constraints. The one-pass method involves the ordering of variables with the following principle: two variables are adjacent if both are affected by a single constraint. A free variable is one, which has so few constraints that it can be solved quite easily. Solving a free variable would eliminate its constraints, thus causing its adjacent variables to become free. Due to the ordering strategy, this method solves all the constraints of the drawing in just one pass. The one-pass method is used for complex cases, where relaxation would be too slow to achieve real-time results.

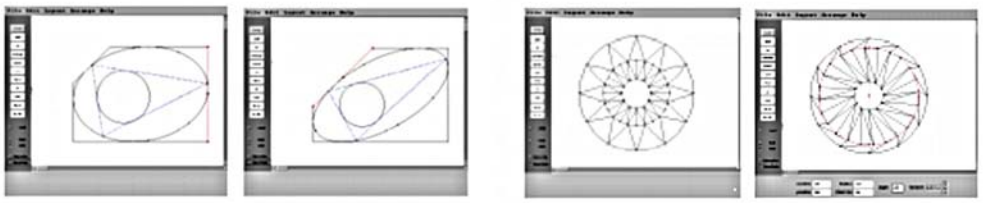
Basic manipulation of drawing objects is achieved by means of a light pen and a few buttons to determine the interaction mode. The graphic display of constraints, which can be toggled on and off, allows users to view, select, and delete constraints.

SketchPad is a seminal work that introduced numerous concepts in graphical interaction that are indispensable for design systems. These include direct manipulations, constraints satisfaction, and augmenting drawing systems with evaluation tools as well as experimenting with motion in drawings.

2.1.2. THE SKETCHER (MEDJDOUB 1999)

Sketcher is an interactive constraint-based prototype that supports the precise construction and exploration of 2D geometric drawings. As users draw and manipulate entities, the system identifies geometric relationships and generates constraints automatically. The system maintains the constraints of the drawing upon further manipulation. Sketcher supports two types of constraints: topological (tangents, perpendiculars, parallels, on and concentric); and metric (fix coordinates, distances, length and angles, and equal subdivisions within a shape). Metric constraints are applied to geometric entities (points, lines, circles, arcs, ellipses and splines). Sketcher also supports the division of a geometric entity into equal parts. Figure 2.1 shows how sketcher maintains constraints when manipulated; 2.1a and 2.1b are variations of the same model, and 2.1c and 2.1d are variations of another model.

The constructive approach is used for satisfying the constraints, which are reduced to quadratic equations and are solved numerically. The drawing is translated into a graph, where nodes are geometric entities and edges are the constraints. A sequence of construction steps is derived, and these steps are carried out to obtain the solution. The graph is created and solved upon every manipulation or creation of constraints. Well-constrained problems are solved by choosing the solution, which best matches the mouse position. Under-constrained problems are guided by the degrees of freedom for geometric attributes. Ruler and compass constructions are solved using quadratic equations, while other types of constructions are solved using numeric methods.



a b c d

FIGURE 2.1 - DRAWINGS GENERATED USING THE SKETCHER

Constraint manipulations vary according to the mouse buttons. Right-mouse interactions create and maintain constraints, while left-mouse interactions ignore and break constraints. User manipulation can cause a constraint to be relaxed or repositioned, resulting in the constraint to change its type (from on to tangent) and, therefore, causes the graph to be redefined. As the constraints are identified, they are expressed by means of construction lines (parallel/perpendicular and tangent) and labels that appear on the drawing. However, constraints that are established in the model are not displayed, thus users cannot differentiate among constrained and non-constrained entities. Conflicts result from modifications, for which the graph has no solutions. In this case the system maintains the last solution and does not attempt to display the conflict situation.

Sketcher's powerful features include the following: automatic specification of constraints; precise feedback as constraints are being identified; automatic sub-division of shapes in equal segments; and incorporation of constrained ellipses and splines. However, Sketcher would benefit from a clear display of established constraints on the drawing, as well as support for users directly manipulating these constraints.

2.1.3. CODRAW (GROSS 1991)

CoDraw is an interactive 2D design exploration environment based on the relational modeling paradigm. It supports symmetric relations between model variables; this enables the bi-directional variation of a model (any variable can be an input from which others are derived). Users specify relations as a model is built, which are maintained by the system as entities are manipulated. Binary and unary relations are expressed as constraints (alignments, tangents, centering, edge offsets, dimension ratio, slopes, and

fixed sizes) which are applied to primitive geometric entities (line segments, circles, arcs, and poly-lines). The representation in CoDraw blurs the boundary between entities and relationships: entities can be considered a set of relations; and relations can be used to define entities.

In CoDraw's grid module, the grid is a design tool that can be selected, composed, superimposed, and used to position design elements and define relations among other elements. The grid module was not integrated to the CoDraw's relational modeling mechanism, so grids are used for positioning but not for transforming design. This is due to the fact that grid management would require discrete manipulation and multiple values that are not supported by CoDraw.

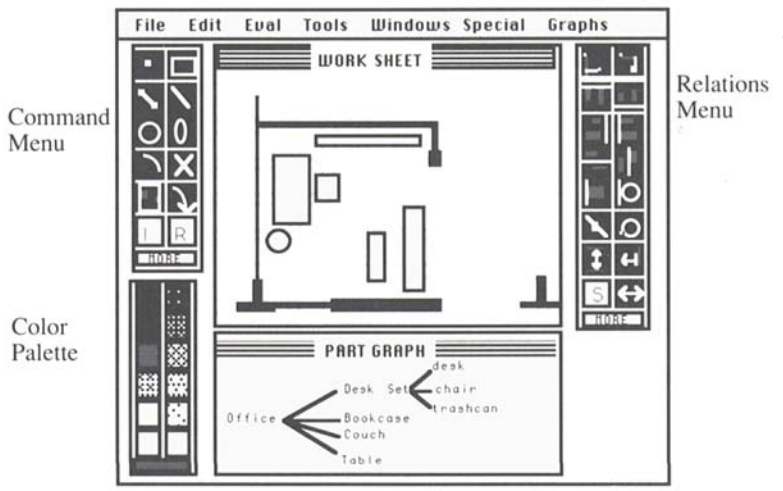


FIGURE 2.2 - THE CODRAW INTERFACE

CoDraw uses CO, a relational modeling language that integrates an object-oriented database that organizes elements into hierarchies, and a reverse spread sheet that provides two-way calculations. A variable is represented as a group of constraints (for example, x=100, 4<x<10, x<5 or x =1\2y) that are stored in a term stack. Constraints also express relations with other variables. Conflicts in CoDraw are resolved by constraint relaxation, which is determined by the object's internal state; rigid or stretchy. CoDraw is extensible: existing relations can be edited and new relations can be specified.

Relations are displayed on the drawing, in order to differentiate between actually

constrained objects and those that just happen to be positioned in a related way, and can be hidden upon request. Users can query internal states of elements to find out whether they are rigid or stretchy. This affects the behavior of the model upon manipulation. However, there is no visual expression for this state, nor is there a visual expression for fixed and free variables. Figure 2.2 shows the interface of CoDraw, with the hierarchic representations for the design and the relations menu.

CoDraw's powerful features include the following: user manipulation of constraints; extensibility in the form of user defined relations; hierarchical representations; manipulations of the internal state of objects, and the grid module.

2.1.4. BRIAR (GLEICHER 1991)

The Briar system is an interactive program that supports the rapid construction and manipulation of precise 2D drawings, and the simulation of kinematic behavior of those drawings. Briar uses augmented snap-dragging to automatically establish persistent constraints in drawings, which are maintained upon manipulation by differential methods.

Briar supports two types of constraints (point-coincident and point-on-objects) that form the basis for a variety of relations. These correspond to two types of snaps: snap to point (center, endpoints, or intersection); and snap to edge (or curve). Special alignment objects (circles of various radii and lines of various orientations) are generated by the system to establish relations among multiple objects; these exist only to be snapped at, and are used to specify distance, orientation, and alignment. An object can be pulled away from a constrained object and then coupled, constrained with another, making it easy for the user to redefine the structure of the model.

Simple geometrical elements are represented by a state vector that contains their parameters and connectors. A physical simulation method is used to maintain constraints, which are solved differentially by reducing the non-linear equations of the constraints into linear equations of their time derivatives. Objects are treated as particles, and user actions as forces; the rate of change in the state of objects, as forces act on it, is computed over time. (Gleicher 1991, p.7).

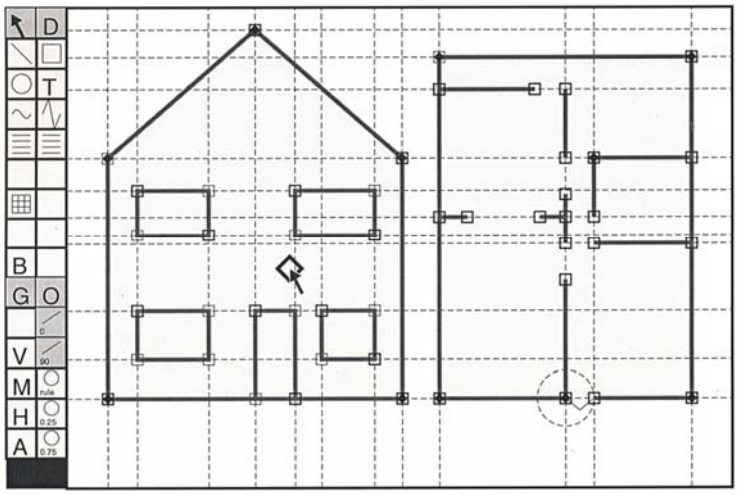


FIGURE 2.3 - THE BRIAR INTERFACE

Simple geometrical elements are represented by a state vector that contains their parameters and both the visual vocabulary for constraints and the feedback regarding constraint identification are clearly expressed on the model. This clarifies the state of the drawing and promotes the predictability of its behavior upon manipulation. Figure 2.3 shows the expression of constraints of a drawing generated in Briar.

Constraints that tend to cluster are placed in an equivalence class, to avoid redundancy in the configuration and visual clutter. As users manipulate drawing objects, the system, guided by its snap-dragging mechanism, makes and breaks the constraints, which can be accepted or rejected by users. Lightweight constraints, such as the tack, give the user additional control to lock entities into position. Alignment objects are controlled indirectly as user manipulates drawing objects.

Briar also supports experimenting with motion. Dynamic differential constraints enable the simulation of kinematic movement in mechanical drawings, while remaining in the interactive mode.

Briar's powerful features include the following: augmented snap-dragging; automatic constraint recognition; a clear visual vocabulary consistent with user feedback; and simulation of dynamic motion. However, the palettes for both forms and constraints are limited, and users cannot directly manipulate constraints.

2.2. ASSOCIATIVE REPRESENTATIONS

In associative representations, design relations constitute dependencies that are defined by the structure of the underlying model. Change is propagated throughout the elements of the structure.

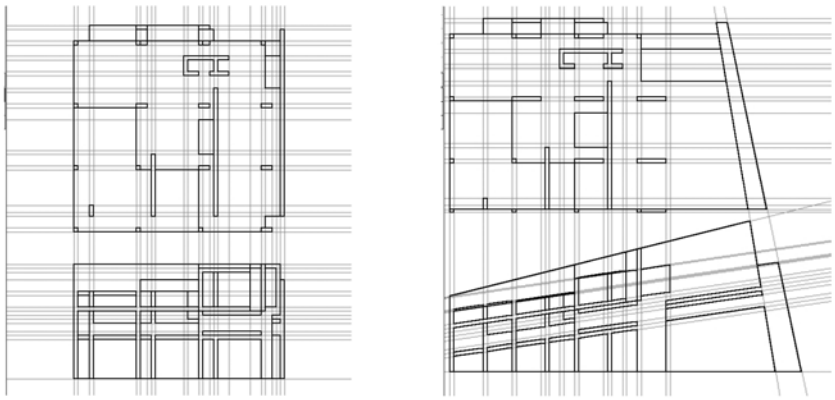
2.2.1. REDRAW (KOLAREVIC 1993)

ReDraw supports the transformation of 2D design compositions by means of manipulating their underlying construction lines. These constitute the framework that establishes formal relations among parts of the design composition. ReDraw's representation is based on the drafting metaphor: ink lines depend on pencil lines (infinite construction lines), which in turn depend on relations (parallel, perpendicular, and connected). In this way, grids, which are patterns of construction lines, and axes, which are construction lines of specific importance, define the hierarchical structure of the design, and regulate its behavior as parts are manipulated. Although both pencil and ink lines are straight in ReDraw's implementation, theoretically, they can be curvilinear (circular, elliptic, parabolic). Designs are composed by making pencil lines and assigning relations to them. Designs are transformed by manipulating (translating and rotating) these pencil lines. ReDraw supports the addition, deletion as well as substituting design decisions.

Every ink line is represented by three pencil lines, one carrier line, and two bounding lines defining its endpoints. ReDraw supports hierarchical uni-directional, or bi-directional, dependencies. Its maintenance mechanism is based on simple direct propagation, through recursive traversal up and down the tree data-structure. Conflicts are resolved by either eliminating a relation, or establishing new ones within the tree structure.

Ink and pencil lines are visually distinct; however, relations among pencil lines are not clearly expressed and must be queried. These visual indicators influence the predictability of the model's behavior upon manipulation. As complexity of drawings increases, pencil lines can become too numerous and overwhelm the drawing, as is illustrated in Figure 2.4, where 2.4b is a variations of 2.4a in which the right construction lines are rotated.

The order of interaction in ReDraw is prescriptive; users must enter pencil lines first, and then ink lines. Similarly, they must specify the type of relation as they are entering the pencil lines; furthermore, these relations cannot be deactivated. These restrictions are partly alleviated by the presence of an undo command.



a b

FIGURE 2.4 - DRAWINGS GENERATED IN REDRAW

ReDraw introduces the notion of construction lines into the computational environment, which is a powerful contribution because it enables the exploration of the structure of the configuration as a whole. However, the sequence of entering pencil lines and relations is not flexible, and the display of pencil lines is visually complex; ReDraw does not support a visual display for relations between pencil lines and nor does it support the deactivation of these relations.

2.3. DESIGN GRAMMAR REPRESENTATIONS

Design grammars, which are used to describe languages of designs, are also used as venues for design exploration, especially in generative design systems. In grammar-based representations, designs are represented by means of a vocabulary of shapes, (defined by lines and labels) and a set of production rules; design relations as well as design transformations are encapsulated in those rules. Configurations are manipulated by the application of these rules in order to change the current state of the design.

2.3.1. DISCOVERFORM (CARLSON 1991)

DiscoverForm is a rule-based program that supports the generation and exploration of self-similar recursive forms, such as branching structures, reptiles, or space filling curves. Forms are developed by recursively cloning a 2D motif according to a single replication rule onto layers that are conceptually ordered along a third dimension.

The structure of these forms is defined by the transformations (translation, rotation, reflection, scale, and shear) that map the motif to its clones. Forms are composed by the recursive cloning of the motif; variations within forms are investigated by changing the details of the motif; and forms are transformed by changing the clones of the motif.

The representation is based on structured grammars, where motifs are represented by their parts and by the transformation that position, and orient, these parts. Recursive structures are described by one motif, one rule, the depth of recursion, and a flag "allgens" that determines whether all generations or the last generation are to be displayed. The recursive form is represented as an ordered display list of motif primitives and clones.

In Carlson's structured grammars, α, is defined by means of an object, a, (positioned at the origin) and the transformation f that positions and orients a with respect to the origin. $\alpha = (a, f)$. A transformation, g, applied to an object, α, is equivalent to the external transformation, g, composed with α's internal transformation, f. The $g(\alpha) = g(a, f) = (a, gf)$. Complex objects are defined by means of the union, intersection and difference of simple structures. New forms are derived from old ones by means of structure rewriting rules. These rules consist of a precedent and a consequent. The precedent is further composed of an inclusive condition and an exclusive condition. $r = (\alpha, \delta) \Rightarrow \beta$. The rule, r, will apply to a form if and only if α is a subset of the form and δ is not in the form.

Motifs and clones are displayed together to give the feeling of a single composite form. Therefore, there is no visual distinction among the various generations of clones. Relations among various parts [clones] are not shown. The transformations are applied directly to the motif and clones of the first generation, by using tacks to fix points or lines

on motifs. These, combined with the mouse direction, determine the transformation that is applicable to the motif (for instance, a fixed point yields a rotation, a fixed line yields reflection).

The resulting forms, as well as the effect of manipulations of motif or clones, are totally unpredictable. A small change to the motif can create an un-correspondingly large change in the form, due to the complexity of the compounding effect of the recursion, as it is illustrated in Figure 2.5.

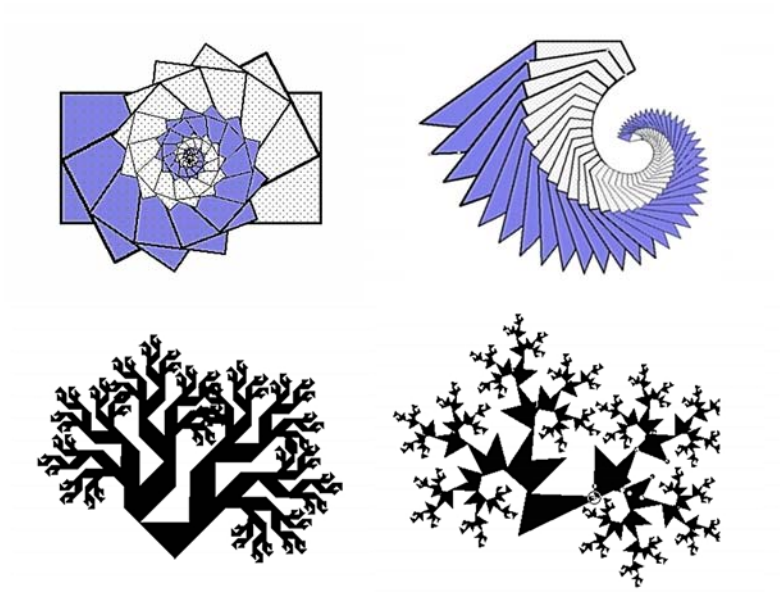


FIGURE 2.5 - TWO PAIRS OF DRAWINGS GENERATED IN DISCOVER FORM

DiscoverForm's is an innovative design system. It has the ability to explore an infinite universe of forms that were not previously possible, with a limited vocabulary (one motif and one rule).It also has the ability to transform structures and to explore variation of structures by direct manipulations and introduces an innovative interface for applying such transformations. However, DiscoverForm does not provide support for altering the recursive structure or exploring parts of the recursive form and does not provide a visual distinction among clones of the various generations.

2.4. HYBRID REPRESENTATIONS

2.4.1. SEED LAYOUT (FLEMMING 1995)

SEED-Layout is a generative system that supports the exploration of 2D schematic layout designs by rapidly generating and evaluating alternatives, using grammar rules in combination with constraint satisfaction. Schematic layouts capture critical architectural information such as circulation patterns, zoning, and the overall massing configuration, as well as numerous performance factors.

The spatial containment structure in SEED-Layout is predetermined. It is defined by Functional Unit hierarchies consisting of buildings, massing elements, floors, zones, and rooms (Figure 2.6). When the Functional Units are allocated in a layout, these form a hierarchy of spatially nested sub-layouts. Each Functional Unit, allocated in a layout, has a corresponding Design Unit, which captures its geometry. The underlying 'topological' structure is strictly orthogonal, and captures the left-right/above below relations between rectangular Design Units within a layout. Variations in layouts can be investigated by generating layouts with alternative structures. SEED-Layout offers three modes of layout generation to the designer: under the designer's control, semi-automated generation, and complete automated generation. All three modes use grammar rules to introduce new Functional Units (or remove existing ones) in the layouts by expanding or contracting the underlying structure. After each modification, the Design Units are re-dimensioned and repositioned, taking the changed structure into account. The hierarchical structures in SEED-Layout cannot be transformed.

SEED-Layout uses subdivision and pinwheel rules to generate designs. SEED-Layout supports complex requirements and prescriptive constraints. The latter are either value constraints (upper and lower bounds on dimensions, areas, and aspect ratios) or relational constraints (required adjacency, minimum or maximum distance, and preferred directions). Prescriptive constraints, which are satisfied automatically by constraint solvers, are formulated explicitly at the Functional Unit and Design Unit levels. Functional Unit constraints generate Design Unit constraints as corresponding Design Units are generated. Additional constraints are computed directly from the layout to guarantee non-overlap between Design Units. The resulting constraints--taken together--

form a system of simultaneous equations and inequalities, which are solved by constraint propagation in numerical intervals, and a disjunctive constraints mechanism, adapted from Baykan (1997).

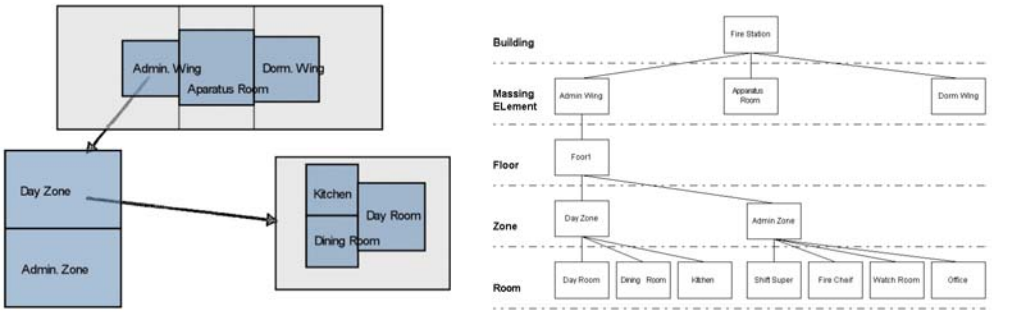


FIGURE 2.6 -THE HIERARCHAL STRUCTURE OF SEED-LAYOUT ILLUSTRATED BY A FIRE STATION

SEED-Layout's elaborate interface supports navigation in a complex design space with numerous alternatives. The layout's structure is clearly displayed on the main Layout view. It is also displayed in the constituent hierarchy and the design space windows.

SEED-Layout is unique in its generative capabilities. It has the capacity to generate alternative configuration under the designer’s control and a mechanism for managing alternatives in a large search space. It also provides a higher-level description of constraints from which lower-level descriptions are generated and maintained. However, there is no support for direct manipulation of Design Units or their structures.

2.4.2. FLOOR LAYOUT AND MASSING STUDY PROGRAMS (HARADA 1997)

Harada's research integrates continuous and discrete transformations in constrained design spaces. Physically-based modeling is used for continuous transformation, while design grammars are used for the discrete transformation within a design search space. The user directly manipulates the design model (by moving and dragging) causing it to undergo continuous transformations until a conflict is reached. This triggers a discrete search for a solution that accommodates both the user's desired changes and the

constraint specification. During this search, the model is transformed by swapping, moving, rotating elements in order to generate alternatives, which are then evaluated in order to select the best one.

The system displays the transformation from the model's state prior to the search, to the resulting state, by a smooth animation that shows elements moving into their new position (Figure 2.7). Continuous and discrete transformations are seamlessly integrated in the system, and the details of the discrete search are hidden from the user. To maintain interactivity, large combinatorial searches are avoided by limiting the discrete search to a couple of elements (the ones adjacent to the selected element) and to minimal transformation steps. Topological and hierarchical structures are determined by the representation of the system; an additional structure is defined by constraints; the user can explore variations within the structure, but only the system can transform it.

Harada implemented two separate software prototypes, one for 2D floor layout and one for 3D massing studies. Both the layout and the massing programs support orthogonal configurations. In the floor-planning program, the rectilinear configuration is represented by a sub-region tree. The geometry is represented by means of the length and width of the top-most rectangle. Child rectangles are represented as a fraction of the parent's length. In the volumetric study program, the configuration is represented by an adjacency graph, which encapsulates both primary attachment relations and secondary (relative coordinate position) relations. Geometry is represented by means of (x, y, z), the block's center, and (lx, ly, lz), half of the blocks width and height. The constraints are represented as forces acting on the design objects and are satisfied by differential numerical methods. In the floor-planning program, constraints are minimum or maximum width, height, area, aspect ratio, and the nail constraint. In the volumetric program, constraints are all the above, plus the non-interpenetration constraint.

In both programs, length, area and volume constraints are visually clear. These appear when the user pushes the spaces to their limits. When the discrete search is taking place, alternatives are counted and the number is displayed to the user. The migration from the old state to the new state is shown by smooth animation, which promotes the predictability of the models behavior. However, since the system chooses among a set of hidden alternatives, the user may not get an expected result.

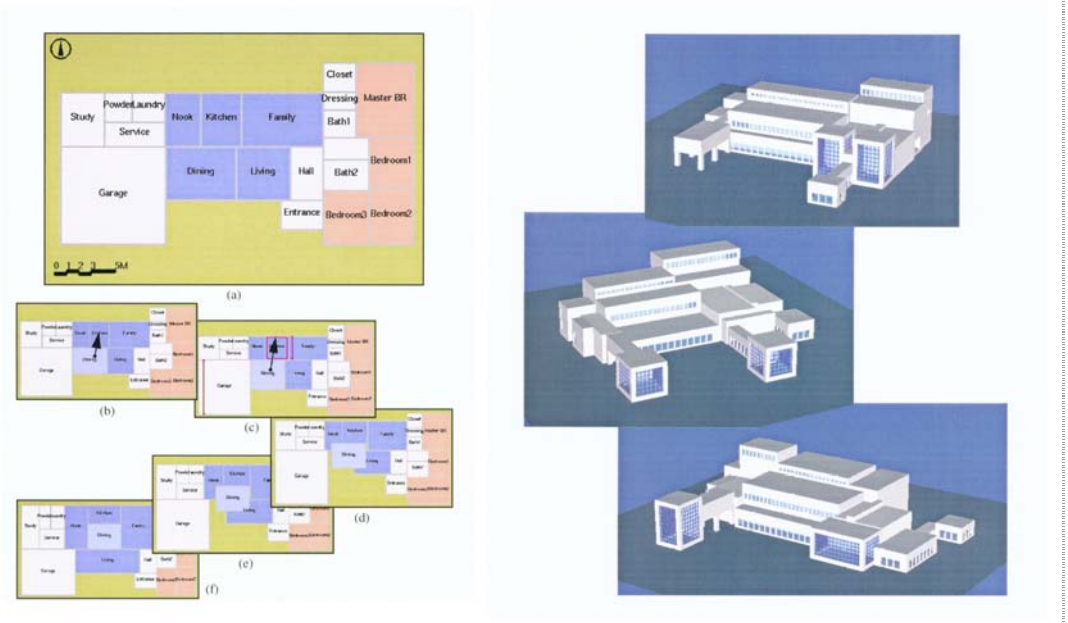


FIGURE 2.7- RE-ARRANGEMENT OF FUNCTIONS UPON MANIPULATION IN HARADA'S PROTOTYPES

Both implementations present an innovative approach to conflict resolution by integrating physically-based modeling and design grammars. Powerful features include a clear expression of constraints on the model, and a smooth animation to express the discrete transformation. However, in both programs, only one scenario for manipulation is supported, and users cannot perform discrete manipulations.

2.4.3. PERFORMANCE SIMULATION INTERFACE (SUTER 1999)

Suter's prototype supports the interactive generation and manipulation of building models in the performance simulation/analysis environment, SEMPER. It incorporates grammar-based generation, dimensional constraints and change propagation. The prototype assumes a subdivision approach in design. A designer builds a model by subdividing an initial cube, and then explores variations in the model by moving/rotating or adding/removing the partitions. Adjusting dimensions of the model, results in redefining the positions of partitions. Relationships between various parts of the model are established by means of link structures.

The prototype supports orthogonal and slanted geometric entities as well as topological,

hierarchical and geometric structures. The former two are automatically defined by the hybrid representation, while the latter is defined by link styles. Changing the link styles transforms the geometric structure.

The representation combines space-based, grammar-based, and sub-region representations. Therefore, spaces are considered voids surrounded by surfaces, and these are subdivided into subspaces by means of grammar rules. Entities are partitioned though a set of partitioning rules and articulated through refinement rules. Each entity has a maximum and minimum bounding partition and a partitioning direction. Volumes are partitioned by surfaces, which are partitioned by lines, which are in turn partitioned by points, resulting in separate hierarchies for volumes, surfaces, and lines.

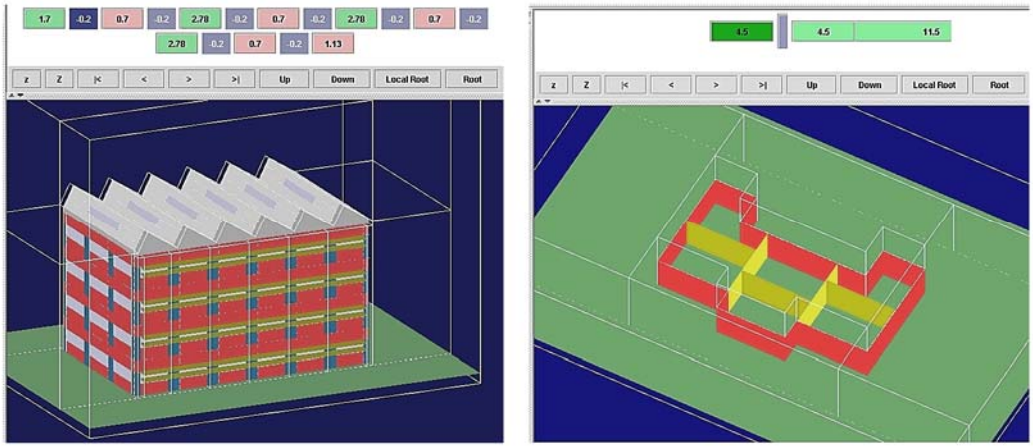


FIGURE 2.8 - THE PERFORMANCE SIMULATION INTERFACE

Refinements, for which there are many types, produce non-orthogonal surfaces such as shed and gable roofs. Constraints for dimensions, offsets, and partitioning directions are controlled by users. Constraint satisfaction consists of conflict detection and change propagation, provided there is no conflict. Conflict resolution is not supported. Changes are propagated in a top-down manner starting from the selected entity.

Links relate distinct entities in the model such that they become identical and are modified (manipulated or partitioned) in the same way. Links are applicable to entities

across various levels of the hierarchies. Link styles provide patterns of links in the model, such as symmetric or rhythmic patterns.

The interface is elaborate. There are several methods to interact with model abstraction and navigate through the model's entities: a component view and a tree view. However, there is no direct manipulation with the model 3D view (Figure 2.8).

Suter’s program provides a coherent representation that supports design subdivision and manipulation of configurations and integrates with performance simulation. It also provides link structures that support the definition of higher level design notions of symmetry and rhythm. However, the complex interaction scheme limits exploration, and the fixed hierarchy prevents parts from being manipulated as units, these cannot be pulled away or rotated. Furthermore, the system does not support direct manipulation and does not provide a clear expression of constraints and links on the model.

2.5. COMMERCIAL SYSTEMS

Most industry-standard CAD systems have numerous powerful features for making precise 3D models. However, among the many systems surveyed, only two provide adequate support for design exploration: GenerativeComponents (Bentley systems) and Revit (AutoDesk).

2.5.1. REVIT, A PARAMETRIC BUILDING MODELER

AutoDesk's Revit is a parametric modeler for architectural design. It supports design phases ranging from early conceptual design to construction scheduling.

Revit supports the investigation of variations within architectural configurations by combining change propagation with constraint satisfaction on two levels of abstraction: (i) for relating elements in the design and (ii) for relating the various drawings of the design. Revit's shared building database ensures the propagation of changes among all the drawings in a given design by means of automatic bi-directional associativity. For instance, if the roof type is changed from shed to gable, the connection to the walls is automatically updated to admit the new configuration, in all drawings.

Revit has a component-based representation, which enables the designer to work directly with walls, floors, doors, windows, roofs etc. Each building component encapsulates its own parametric information as well as the information for integrating with other components. Revit supports both straight and curved geometries. However, spaces, functions, and other abstract design entities are not represented in Revit.

Revit automatically recognizes design constraints, such as connectedness, dimensions, and alignments. These can be accepted or rejected by users. Constraints are preserved upon manipulation; and conflicts are reported to users. Revit's recognition and manipulation of constraints is better suited for rectilinear configurations, including rotated rectilinear grids. Although it supports other angular configurations, it does not recognize some of their special constraints, such as 60-degree relationships or corners where two coordinate systems join. Therefore, the behavior of such configurations tends to be less effective as rectilinear configuration.

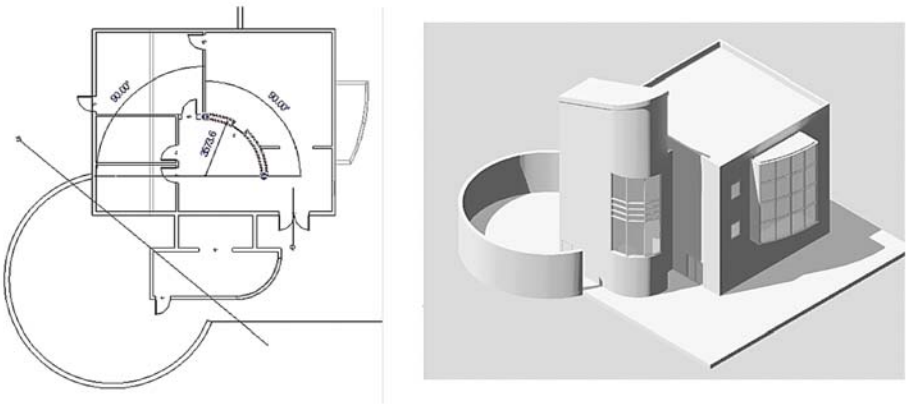


FIGURE 2.9 - AN EXAMPLE OF A PLAN AND 3D MODEL GENERATED BY REVIT

As model entities are being created, precise angle, dimension and dotted construction lines are directly displayed on the model. The construction lines show perpendicular, parallel, tangent and endpoint relationships. This information, as well as persistent constraint information, appears on the model view when an entity is selected, (Figure 2.9) and disappears when the entity is deselected. In this way, visual clutter is avoided. However, manipulating multiple constraints is not possible. Revit also supports reference planes, which are shown as dotted lines in plan view, used for visual as well as dynamic alignment. When walls are constrained to reference lines, moving reference lines will move walls. However, rotating reference lines will often create conflicts.

Revit uses a sketching metaphor to support conceptual design activities such as massing. It has a massing sketch mode and allows the switching between massing and modeling modes. A mass is first sketched then incorporated into the main model. There are three modes for sketching masses: by extrusion, by revolution, or by extension. Massing prior to planning is preferable in Revit, because masses can be readily converted into their wall components, but walls cannot be converted to masses.

Overall, Revit is a powerful system that supports the design drawing and documentation process as a whole. It has numerous powerful exploratory features such as the built-in architectural logic, the automatic recognition of constraints and a clear visual vocabulary for constraints expressed on the model without visual clutter. Revit, however, does not

support the ability to control multiple constraints or to specify new constraints on a model. Furthermore, creating new component is cumbersome.

2.5.2. GENERATIVECOMPONENTS (AISH, 2005)

GenerativeComponents supports the creation and exploration of user-defined complex 3D building elements and integrates interactive manipulation with visual programming techniques.

As the user creates the model, the system keeps track of input parameters and identifies their dependencies, which form a directed and acyclic dependency graph, displayed in a separate window. Geometrical entities are represented by input parameters. Design structures are represented by dependencies. Variations in design structures are investigated by manipulating model entities. Structures are transformed by manipulating the dependency graph, and by locking or freeing values in the graph.

GenerativeComponents is based on a change propagation mechanism. Since the graph is directed, change propagates only in one direction, from upstream components to downstream components. The graph captures the history or sequential development of the modeling operations, with earlier operations forming the upstream components and later operations forming the downstream components. This sequence of operations is re-executed when an input is changed.

In GenerativeComponents design relations are expressed visually in the dependency graph, and numerically in the spreadsheet. All visual representations can be manipulated. However, dependencies and constraints are not directly apparent on the model. GenerativeComponents models are flexible, but the allowable manipulations depend on the input parameters. For instance, a circle entered by its center and radius will be manipulated differently from a circle entered by three points on the circumference.

GenerativeComponents includes a library of abstract geometrical components, point, line, Arc, B-splines and as well as Booleans operation and is further extensible to accommodate new complex user-defined components without programming from the part of the user, such as in Figure 2.10. GenerativeComponents automatically generates programmatic modules that capture the novel components and add them to the object

library. Users can compose, for example, a banana truss, and then use it in repetition to compose a roof structure. This makes two nested levels of dependency structures, one for manipulating the truss, and one for manipulating the roof.

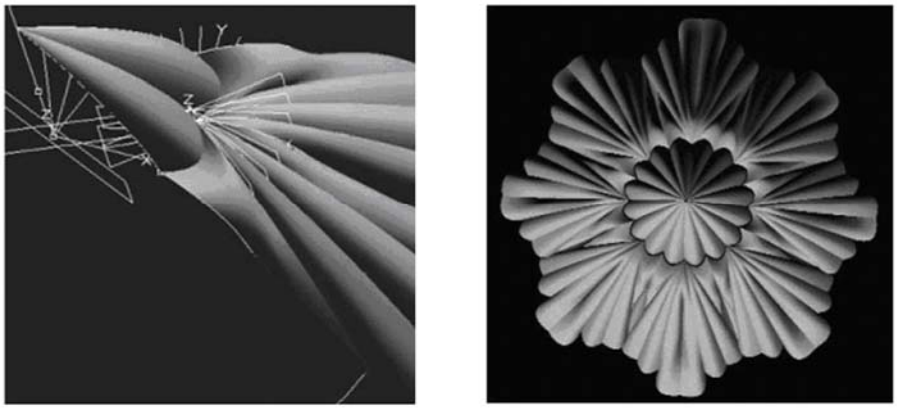


FIGURE 2.10 THE DEFINITION OF A RADIAL ARRAY OF A GENERATIVE COMPONENT

Models created in generative components can be further analyzed with typical Bentley tools such as Bentley Architecture and Bentley Structures.

GenerativeComponents presents a novel approach to CAD systems: It encourages strategic design exploration. Strategic planning is required to define the complex dependency structure, and to determine the desired set of desired manipulations and their consequences. Its powerful features include the ability to create very complex curvilinear geometry from simple components, the capture of operation sequence, and the extensibility to create custom components. However, the complexity of the dependency graph is significant and it is directly proportional to the number of elements in the model. GenerativeComponents lacks architectural components, the support for re-definition of input parameters and bi-directional association.

2.6. MATHEMATICAL REPRESENTATION OF SHAPES AND PATTERNS

In this section configurations are represented by means of their mathematical properties. Design relations are considered as transformations that map one part to another. Although the goals of these representations are not design exploration, but rather design description and design analysis, the approaches presented here focus on relationships which are a fundamental ingredient in design exploration.

2.6.1. SHAPE PATTERN REPRESENTATION (CHA 2004)

Cha's and Gero's "Shape Pattern Representation" is a notation for describing patterns by capturing relations between repetitive parts. These relations are organized in a hierarchical tree structure. The goal of this representation is to provide a language for style learning, shape analogies and shape complexity measure.

This representation, which is based on predicate calculus, captures formative ideas in design. It captures the implicit design knowledge of shape organizations, by means of explicit predicates and arguments.

A shape S is represented by lines P_1 to P_n. The shape pattern representation syntax supports transformation based relations, depicted by τ, and topological relations, depicted by σ. For transformation-based relations: τ_1 = translation, τ_2 = rotation, τ_3 = mirror and τ_4 =scale. For topological relations: σ_1 = over, σ_2 = under, σ_3 = right, σ_4 = left, and σ_5 = between σ_6 = inside, σ_7 = outside, and σ_8 = center.

A shape pattern consisting of two shapes is represented by the following string: $e_2 = \tau_1\{e_1,(a_1,a_3)\}$. The referent shape e_1 and the parameters a_1 and a_3 are arguments for the predicate τ_1, which is the translation relation between the two shapes (Figure 2.11a). A composite pattern defined by two relationships such as the translation τ_1 and scale τ_4 is represented by the following composite predicates: $e_2 = \tau_1\{\tau_4[e_1,a_4],(a_1,a_3)\}$ (Figure 2.11b). A shape pattern consisting of multiple shapes uses the nesting operator $\subseteq$ to indicate recursive nature of the pattern, for instance: $S = \subseteq_{i=1}^{n} \tau_1\{e_i,(a_1,a_3)\}$ (Figure 2.11c). A multiple level pattern, such as the pattern

consisting of two translation relations acting on the same referent shape, is represented by multiple nesting predicates: $e_2 = \subseteq_{i=1}^{n} \tau_1\{\subseteq_{i=1}^{m} \tau_1[e_i,(a_1,a_3)],(a_1,a_3)\}$ (Figure 2.11d). The notation also supports super relationship acting on sub relationships, independently of shapes; each sub-relation has a different referent shape such as $S = \subseteq_{i=1}^{6} \tau_2\{S_i,(60,a_5)\}$ and $S_i = \subseteq_{i=1}^{4} \tau_1\{e_i,(a_1,a_3)\}$ illustrated in (Figure 2.11e).

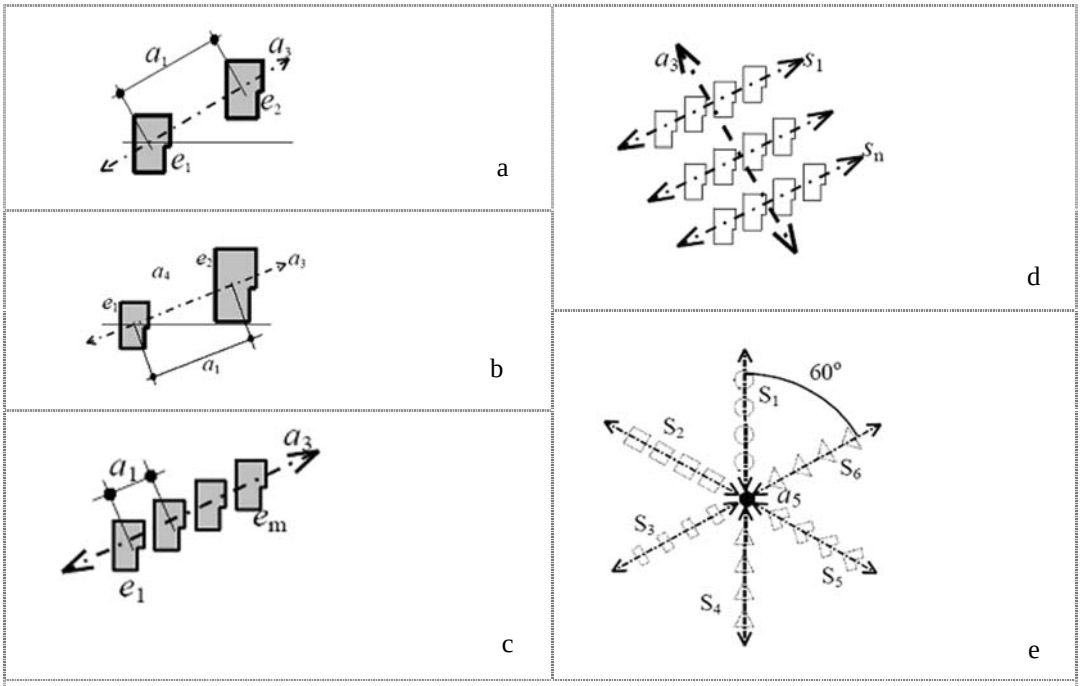
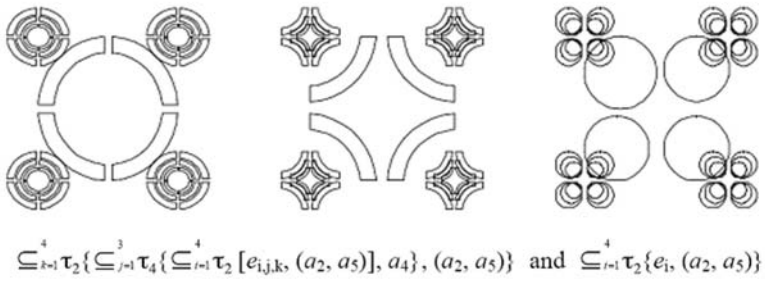


FIGURE 2.11 SUMMARY OF THE SHAPE PATTERN REPRESENTATION FEATURES

Shape pattern schemas represent the spatial characteristics for a set of shape patterns. Each shape pattern is an instance of a shape schema. Shape schemas are used for shape analysis where the following are key concepts:

- Analogy: Shape analogy is achieved by comparing the predicates defining each shape. Analogous shapes have identical pattern schema but not the same referent shape as is illustrated in Figure 2.12.
- Embedding: One scheme can be embedded in another one, making a sub-schema/super-schema relationship.
- Sharing: two schemas can share the same sub-schema.

- Recursion: A recursive scheme embeds within itself.
- Complexity: Shape complexity is proportional to the composites levels of the representation tree.
- Style: Styles are identified by identifying specific patterns in the tree representation.
- Multiple representations: A single pattern can be represented in several ways. Such multiple representations provide various interpretations for a single shape pattern as is shown in Figure 2.13.



$\subseteq^{4}_{i=1}\tau_2\{\subseteq^{3}_{j=1}\tau_4\{\subseteq^{4}_{i=1}\tau_2[e_{i,j,k}, (a_2, a_5)], a_4\}, (a_2, a_5)\}$ and $\subseteq^{4}_{i=1}\tau_2\{e_i, (a_2, a_5)\}$

FIGURE 2.12 - ANALOGOUS SHAPES

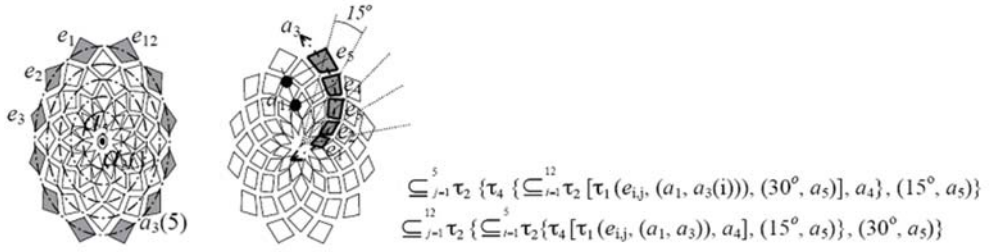


FIGURE 2.13 - MULTIPLE REPRESENTATION

Cha's shape pattern representation introduced several significant issues pertaining to shape pattern analysis and their application in architectural designs. The powerful concepts in the representation include shape description through relational knowledge, shape pattern schema, and the various strategies for shape analysis. Although this representation is used for 2D repetitive patterns, and the scope of relations is limited to transformation and topological relations, the syntax is difficult to read. This is due to the following factors: (i) the inconsistent use of brackets and parentheses in nested shapes;

and (ii) the naming convention for relations and other objects is not mnemonic, with the same symbol representing numerous objects.

2.6.2. A GENERATIVE THEORY OF SHAPE (BASED ON GROUP THEORY) (LEYTON 2001)

Leyton's generative theory of shape describes simple and complex shapes by means of group structures. The approach is to maximize transfer and the goal is to maximize recoverability. In Leyton's theory, there are no objects; every shape is described using actions (Leyton 2001, p.29). Transfer is the ability to re-use previous actions as part of subsequent actions; recoverability is the ability to recover the sequence of actions that lead to generating the shape. The applicability of the theory ranges from Architectural Design, to Mechanical Engineering to Robotic Manipulation, and many others.

The theory is based on the assumption that complex shapes are generated by means of symmetry breaking or more precisely asymmetry building (Leyton 2001, p.40). The shape creation (or shape designing) process begins with a series of actions defining the symmetries of the shape, and proceeds with a set of actions that break these symmetries. The symmetry breaking actions are recorded in the shapes themselves. Consider the asymmetric shape in Figure 2.14a: one can infer that a rectangular piece was subtracted from the symmetric square; however, one cannot infer from a square all possible asymmetric actions that may have been negated to obtain the symmetric square. Consider the tilted parallelogram in Figure 2.14b: According to Leyton, its generation steps are as follows (1) creating the square, (2) stretching the square to form a rectangle, (3) applying shear to the rectangle to form a horizontal parallelogram, and finally (4) rotating the horizontal parallelogram to obtain the tilted parallelogram. Therefore, the sequence from symmetry to asymmetry allows shapes to record generative actions and, therefore, maximizing recoverability.

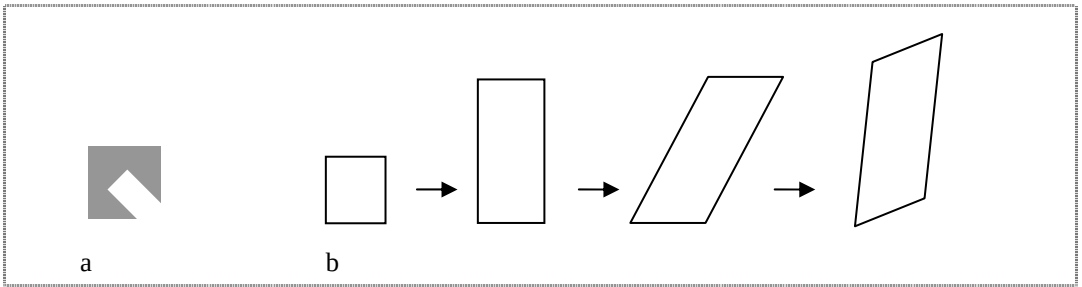


FIGURE 2.14 - (A) MODIFIED SQUARE (B) GENERATING A ROTATED PARALLELOGRAM

In Leyton's theory, structures are mathematical groups, which are combined by means wreath products w. A line, for instance, is denoted by the group of real numbers $\mathbb{R}$, and the occupancy group $\mathbb{Z}_2$. The square is denoted by $\mathbb{Z}_2 \, w \, \mathbb{R}$ and by the 4-fold rotation group $\mathbb{Z}_4$, which includes the rotation transformations {*e, r90, r180, r270*} which are subgroups of the symmetries of the square (Leyton 2001, p.9). The parallelogram is generated by applying the general linear group $GL(2,\mathbb{R})$ to the wreath structure of the square.

- Line: $\mathbb{Z}_2 \, w \, \mathbb{R}$
- Square: $\mathbb{Z}_2 \, w \, \mathbb{R} \, w \, \mathbb{Z}_4$
- Parallelogram: $\mathbb{Z}_2 \, w \, \mathbb{R} \, w \, \mathbb{Z}_4 w \, GL(2,\mathbb{R})$

In Leyton's syntax, $\mathbb{R}$ is used to describe continuous generative actions, while $\mathbb{Z}$ is used to describe discrete actions. Typically, the first group in the wreath product is the fiber group and the subsequent group is the control group, where control groups move (preceding) the fiber groups around.

Leyton (2001, pp.229-238) represents the basic 3D surface shapes as follows. All other shapes are generated by using these as primitives, either by means of spatial combination or Boolean operations. $\mathbb{S}O(2)$ is the continuous group of rotation in the plane.

- Plane: $\mathbb{R} \, w \, \mathbb{R}$
- Sphere and Torus: $\mathbb{S}O(2) \, w \, \mathbb{S}O(2) \, w \, \mathbb{R}$
- Cylinder and Cone: $\mathbb{R} \, w \, \mathbb{S}O(2) w \, \mathbb{S}O(2)$
- Cube: $\mathbb{R} \, w \, \mathbb{Z}_4 \, w \, \mathbb{R}$

Both the Sphere and the Torus are represented by the same wreath structure $\mathbb{S}O(2) \, w \, \mathbb{S}O(2)$. However, the difference is in the distance between the rotation axes with respect to each other. This is referred to as the control radius depicted by $\mathbb{R}$; changing it breaks the symmetry of the sphere and generates the Torus. Similarly, the Cylinder and the Cone are represented by the wreath structure $\mathbb{R} \, w \, \mathbb{S}O(2)$; with the main difference

being the angle between initial line and the rotation axis, which is the control angle, depicted by $\mathbb{S}O(2)$.

Typically, when shapes are combined, the symmetry of each one is broken and only combined symmetries remain. In Leyton's approach (p.241), the symmetries of each shape are stored in their group definition as well as the affine transformation that relate them. $AGL(3,\mathbb{R})$ is the affine group on the 3D real space and the initial set of primitives are referred to the alignment kernel.

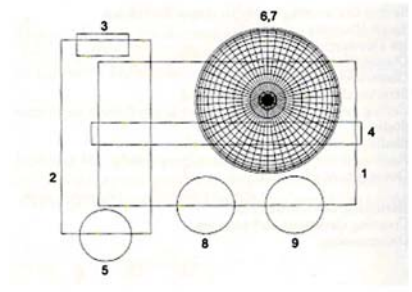
- Combined cylinder and cube: $\lfloor G\,cylinder \times G\,cube \rfloor\ w\ AGL(3,\mathbb{R})$

Boolean operations are represented in a similar manner (Leyton 2001, p.252). Union, intersection, and difference share the same spatial group structure as spatial combination; however, these are further distinguished by means of the occupancy group $\mathbb{Z}_2$.

Unfolding groups constitute the mechanism that allows control groups to treat various shapes within a fiber group differently. (Leyton 2001, p.250)

- Unfolding group: $\lfloor G_1 \times G_2 ... \times G_n \rfloor_T\ w\ C(G_1)\ w\ C(G_2)\ w\ C(G_{n-1})$

Leyton (2001, p.365) claims that Architectural Design is a process of asymmetry building and can be represented by means of unfolding groups. In the massing study in Figure 2.15, masses are created by unfolding transfer structure of three basic primitives, the sphere, the cylinder and the cube. Each primitive being copied and each copy being treated differently by affine transformations to obtain the desired mass, as is illustrated in Figure 2.15.

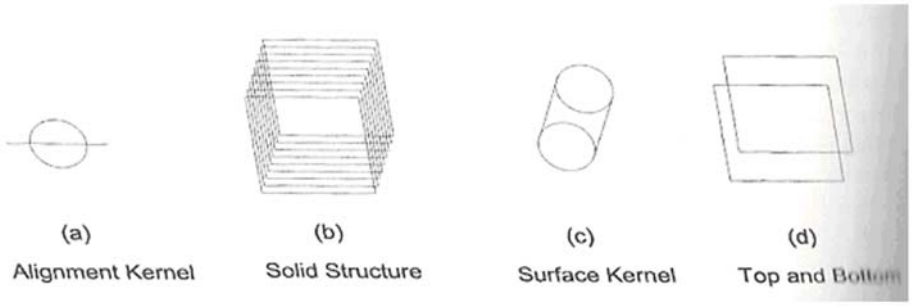


$$[[G_{sphere}]_{P_3} \times [G_{cylinder}]_{P_2} \times [G_{cube}]_{P_1}]_{\mathcal{U}}$$
$$\text{Ⓦ}\ [{}_{daP_1}AGL(3,\mathbb{R})^{aP_1}$$
$$\times\ [[{}_{wvuP_2}AGL(3,\mathbb{R})^{vuP_2} \times {}_{cbaP_1}AGL(3,\mathbb{R})^{baP_1} \times {}_{baP_1}\{e\}^{baP_1}]$$
$$\text{Ⓦ}\ {}_{*vuP_2\ ,\ baP_1}AGL(3,\mathbb{R})^{\ uP_2\ ,\ aP_1}]$$
$$\times\ {}_{muP_3\ ,\ muP_2}\ AGL(3,\mathbb{R})^{\ uP_3\ ,\ uP_2}$$
$$\times\ {}_{aP_1}\{e\}^{aP_1}]$$
$$\text{Ⓦ}\ {}_{*uP_3\ ,\ *uP_2\ ,\ aP_1}AGL(3,\mathbb{R})^{\ P_3\ ,\ P_2\ ,\ P_1} \times {}_{P_1}\{e\}^{P_1}\ .$$

FIGURE 2.15 (A) A MASSING CONFIGURATION (B) ITS CORRESPONDING NOTATION

Material choices are represented by color groups, while stairs, column grids and gable roofs are conveniently represented by transfer group structures.

In Leyton's theory (2001, p.397) solid structures, which are more complex than their surface counterparts, are presented by means of hyper-octahedral wreath Hyperplane groups, which combine the infinite translations in space to produce a solid with the surface definition of a shape. The cylinder and its corresponding notation are illustrated in Figure 2.16.



$$\overbrace{\left[\left[\mathbb{S}O(2)\right]_{P2}\times\mathbb{R}_{P1}\right]}^{alignment\ \ker nel}u\ \ w\overbrace{\left[\mathbb{R}\ w\mathbb{R}\right]^{P1}}^{solid\ \ structure}w\ \overbrace{\mathbb{R}^{P2}}^{surface\ \ker nel}\ \ w\ \overbrace{\mathbb{Z}_2^{\left[\mathbb{R}\ w\mathbb{R}\right]_{1/2}}}^{top\ and\ bottom}$$

FIGURE 2.16 - THE REPRESENTATION FOR A SOLID CYLINDER

Sweep structures are conveniently represented by the fiber control group structure: $sweep = profile\ w\ path$ (2001, p.430). The profile and path are splines denoted by $\mathbb{R}^0\ w\ \mathbb{R}\ w\left[\mathbb{Z}_2 w\Sigma_1\right]^{\mathbb{R}^0_{1/2}}w\mathbb{H}$ where $\mathbb{H}$ is the group representation of the Hermite cubic spline.

Leyton's theory is applicable to many domains; Robotic motion can be described using these fiber groups. For instance the motion of cutting machines can be described as follows.

- Rotating blade to create a hole is described by a rotation movement and the movement in the z axis. $\mathbb{S}O(2)\,w\,\mathbb{R}$
- Rotating blade to create a slot is described by a rotation movement and the

movement in the x and z axes. $\mathbb{S}O(2)\ w\ \mathbb{R}\ w\ \mathbb{R}$

- Rotating blade to create a pocket is described by a rotation movement and the movement in the x, y, and z axes. $\mathbb{S}O(2)\ w\ \mathbb{R}\ w\ \mathbb{R}\ w\ \mathbb{R}$

Leyton's notation gets very complex as configurations become elaborate. Furthermore, the syntax does not capture parameters for shape generation, for instance translation distance; neither does the group notation show which transformation are part of the group. $\mathbb{Z}_4$, for instance, may be representing a 4-fold discrete rotation or a 4-fold discrete translation. Similarly, $\mathbb{Z}_2$ can be a reflection or a 2-fold discrete rotation.

Leyton's description of design (Leyton 2001, p.365) equates the design process with the drafting process of using AutoCAD or ProEngineer. This surface view of design is not comprehensive and does not include essential exploratory activities of the process. Although the theory is indented as an underlying computational representation, not for exploration, its basic assumptions cannot be applied to an iterative design process. Leyton's approach is based on maximizing transfer in generating all parts of the configuration. This does not necessarily match with the designer's intentions. A designer may want to relate some parts, while keeping other parts independent. Leyton imposes a specific order of symmetry, then asymmetry, on the generative sequence with the purpose of maximizing recoverability. Although, this prescriptive approach organizes the representation, it limits the options for generating configurations, and does not necessarily match the designer's methods for defining and exploring shapes.

2.7. COMPARATIVE ANALYSIS

The representations reviewed in this section include seminal as well as novel computational approaches to design exploration. The representations have in common the encapsulation and management of relations among design entities and expressing these either theoretically in the form of a notation or practically in the form of an interactive system. Each has its focus, contributions and opportunities for further research. The key issues that are relevant to describing and exploring configurations through their relational structures are described below.

Design structures: Flemming and Harada investigated hierarchical and topological structures. Sutherland, Gleicher, Medjdoub and Gross investigated topological and (some) geometric structures in the form of unary and binary constraints. Carlson dealt with hierarchical and geometric structures. Kolarevic worked with a subset of geometric structures (grid structures). Suter investigated hierarchical structures and a limited version of bilateral symmetry and rhythm in subdivisions by defining them in link styles. However, his investigations were restricted by the nearly orthogonal universe of configurations defined by his representation. Cha worked with topological and symmetrical structures, while Leyton's approach dealt with isometric and affine group structures.

Transformation of structures: Kolarevic investigated the transformation of grid structures, defined by alignments, parallel, and perpendicular relations. However, these structures were not flexible, and could not be redefined. Furthermore, Kolarevic's explorations were restricted to the universe of straight-line configurations. His sequential approach of interaction, influenced by the drafting metaphor, greatly limits the exploration. Carlson investigated transformations of structures within the universe of self-similar recursive forms, defined by one motif and one rule. The structure was regenerated upon the manipulations of motifs. However, transformations of recursive structures are not intuitive and the universe of self-similar forms is quite limited. Harada's systems supported discrete transformation of topological structures, but supported only one scenario of exploration.

Dynamic structures: Dynamic constraints, used in Briar, are useful for designing movable building components. This approach is valuable for reconfigurable architectural

configurations and mechanical parts. Leyton also addressed the representation for the motion of robotic components in his theory.

Extensibility and Customizable structures: In most systems, the structure can be defined from a predefined set of relations. However, CoDraw and GenerativeComponents provide extensibility by supporting users defined types of structures.

Discrete explorations within structures: Discrete exploration is essential to design, since the design search space is a hybrid (continuous and discrete) space. Harada's prototypes used system-initiated discrete transformations to complement user-initiated continuous transformations.

Automatic identification of structures: Automatic constraint generation, used in Briar, Sketcher and Revit, relieves the user from the additional burden of specifying constraints. However, this mechanism relies on the interpretation of user intent, which, in case of a misinterpretation, can generate unintended constraints leading to undesirable behavior. Alternatively, Seed-Layout, as well as Harada's and Suter's prototypes, have predefined structures that are automatically established as users define their models.

Conflict resolution: Sutherland addresses conflict resolution by relaxing constraints. Medjdoub addresses conflict resolution by a constructive method. Gleicher's model always satisfies the constraints; consequently, there is no need for conflict resolution. GenerativeComponents uses uni-directional associativity, therefore avoiding conflicts. Harada addresses conflict resolution in an innovative way by using discrete search mechanism to find an alternative solution that avoids the conflict. Other systems, such as Revit, and Suter's system, notify users as conflicts are detected.

Interaction with structures: Some systems (SEED-Layout, Harada's system, Suter's system, and CoDraw) support user manipulation of structures; this is usually achieved through creating and breaking constraints or relations. Others manage the making and breaking of structures solely by the system (Briar, Sketcher). Revit, on the other hand, combines system generation of constraints with users breaking constraints. Although the user manipulation of constraints is essential for redefining structures (Akin 1987), it becomes more cumbersome as configurations increase in complexity, and constraints increase in number.

Visual display of structures: The clear display of structure is a crucial usability issue that affects predictability among other factors. Some of the reviewed systems provide a clear display of the structure directly on the model (Briar, SEED-Layout, Harada's systems, and CoDraw); while others provide separate displays (GenerativeComponents, Suter). Revit provides a selective display that shows constraints bound to selected objects. In addition, the systems that provide automatic identification of constraints (Gleicher, Medjdoub, and Revit) provide a clear feedback to users regarding the creation of constraints.

Predictability of structural manipulations: Each of the system surveyed has a different level of predictability, depending on the system's intent. Carlson's recursive structures, for instance, are very unpredictable, due to the non-intuitive characteristic of recursive structures. Carlson's intended DiscoverForm as a tool for the discovery of unanticipated avenues of inquiry (Carlson 1991). The predictability of Kolarevic's models depends on their complexity; Kolarevic's also intended ReDraw to produce surprising results, which can trigger innovation and creativity (Kolarevic 1997). Harada, on the other hand, effectively used smooth animation to provide predictability.

Complexity of the syntax: In both Leyton and Cha's representation, the notation is quite complex. Cha's notation, based on predicates and arguments, captures a lot of relations in the design; however, the facts that the naming convention is not mnemonic, the inconsistency of the brackets and the extensive use of subscripts, make the notation difficult to read. The complexity of Leyton's representation is due to the necessity to describe every shape by means of group constructs. The use of unfolding increases the complexity and therefore decreases the readability of the notation. However, essential parameters in the description of the shapes are not captured in the notation string, but are subsumed in the group definition.

Capturing history of generation: On the theoretical level Leyton's theory is motivated by capturing a unique generative process for any shape. However, because symmetry actions need to precede asymmetry actions in order to maximize recoverability, this approach imposes a sequential application on the design process, thus limiting the options for design generation. This prescriptive sequence does not necessarily match the designer's approach. On the systems level, GenerativeComponents captures all actions

leading to creating a form; this sequence is later used to modify and explore the form.

2.7.1. OPPORTUNITIES

Each of the aforementioned representations focused on a particular aspect of exploration, however, none of them investigated exploration in a comprehensive way in order that determine ingredients, strategies and techniques for maximizing exploration.

- None of these representations addressed the complete set of higher-level geometric relations such as symmetry, hierarchy, proportion, and rhythm, which are essential compositional tools in architecture, and which play a fundamental role in architectural design exploration.
- None of these representations had a scheme for organizing lower level dependencies at higher levels of abstractions.
- Some of these representations addressed transformation of structures at a limited level, but none addressed the redefinition, deactivation, replacement of structures that is necessary for achieving the maximum flexibility or interaction and for discovering new structures.
- Although Harada addressed discrete transformations, none of the representations supported users making their own discrete transformations such as swapping or inserting.

CHAPTER 3

APPROACH: A FRAMEWORK FOR DESIGN EXPLORATION

The representations surveyed in the previous chapter, "Background: Representations for Design Exploration," addressed several aspects of design exploration independently. Each aspect focused on a specific method of exploration within a specific class of configurations. However, there has been no comprehensive framework investigating the variety of exploratory activities for the complete range of configurations that exist in architectural design.

In this chapter, I present the premise of this dissertation from a normative perspective, as well as from an empirical perspective. I describe the approach for developing a framework for design exploration, from the computational perspective. In section 3.1, I discuss the classification, representation, and use of design structures as observed in notable architectural endeavors, and as explained in architectural theory. In section 3.2, I illustrate the representation and strategic use of structure in early design processes, through the results of an empirical study. In section 3.3, I describe a framework for representing and exploring architectural configurations. The framework, named Interactive Configuration Exploration (ICE), uses relational and generative structures, as a means for concisely describing, rapidly generating, and interactively transforming design configuration.

3.1. DESIGN STRUCTURES

Design structures are abstractions encapsulating compositions of spatial relations, which organize simple architectural elements into coherent complex configurations. Architectural compositions consist of numerous elements, which can be spatial entities (spaces or zones) or building components (walls, windows, columns, etc). A structure organizes such individual elements by determining formal properties such as position, orientation, sizes and form. Design structures are of special significance in architecture, because they utilize ordering principles to relate elements within complex configurations as well as elements within various resolutions.

Design structures play an important role in design conceptualization. These are the implicit vehicles used to compose and refine complex configurations of spatial entities. For every design problem there are numerous possible solutions and, therefore, several possible structural organizations. Hence the need for exploring structures. Through design structures, one can explore design configurations globally, rather than manipulate numerous individual elements, locally.

In this section, I present a classification for design structures, and discuss the basic units for each of these classes. I also review visual representations of structures, and describe the concept of "transformation of structures" as an exploration tool.

3.1.1. CATEGORIES OF DESIGN STRUCTURES

There are numerous classifications of architectural configurations; Most rely on style; others base their categories on plan-shape (Curtis 1935, and Krier 1988a,), spatial organization (Ching 1996), and formative ideas (Clark 1985). Curtis (1935, pp.189-195) and Krier's (1988a, pp.43-67) classifications, based on plan shape include square, round, octagonal, cross, L-shaped, U-shaped, T-shaped, and many others. Ching's (1996, p.189), classification, based on spatial organizations, include centralized, linear, radial, clustered, grid organizations. Each of these can be further classified according to their main ordering principles of symmetry, hierarchy, rhythm, datum (Ching 1996, p.320). Clark (1985, p.137) presents an elaborate classification based on formative ideas. These include plan to section, unit to whole, repetitive to unique, additive to subtractive, symmetry, balance, geometry, configuration patterns, progressions, and reduction. Each of these is

further classified into its own subcategories. For instance, configuration patterns are classified as being linear, central, double-center, cluster, nested, and concentric.

Wong (1993, p.59), on the other hand, classified abstract 2D design configurations according to their structure. He identifies formal, semiformal, informal structures, as well as active, inactive, visible and invisible structures. Within formal structures, Wong includes repetition, gradation, and radiation.

In this context, *architecturally significant* structures are used to classify architectural design configurations. These structures, which are applicable to any level of abstraction in design configurations (such as plan organization, 3D massing, 2D façade, etc.) can be categorized as topological, hierarchical or geometric.

Topological structures define networks of relations between spatial entities. These are typically used in adjacency diagrams of early architectural conceptualizations to determine the location of spaces with respect to each other. March (1974) studied graph networks as an interpretation of space arrangement. Figure 3.1 shows three of Frank Lloyd Wright's houses, Life (1938), Jester (1938), and Sundt (1940) that differ in geometry, but share the underlying topological structure (March 1974, p.27). They also share a similar geometric coherence among their parts expressed by regular forms.

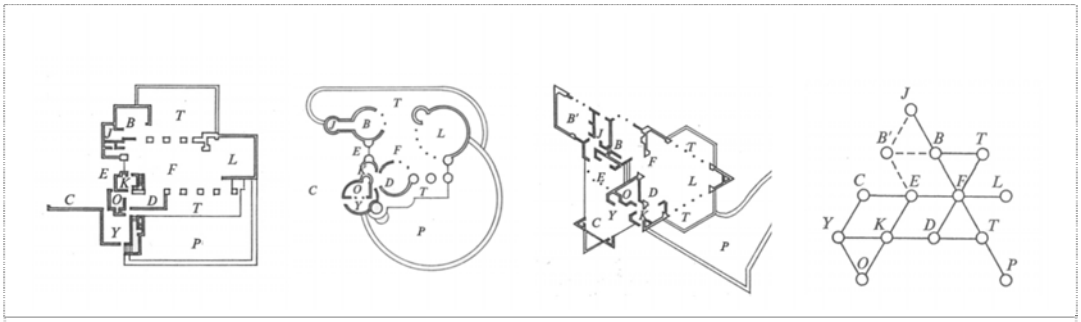


FIGURE 3.1 - FRANK LLOYD WRIGHT'S LIFE, JESTER, AND SUNDT HOUSES, AND THEIR SHARED TOPOLOGICAL STRUCTURE

Hierarchical structures define grouping and organizational hierarchies of spatial entities. Figure 3.2 shows the hierarchical decomposition of a fire station into wings, zones, and spaces.

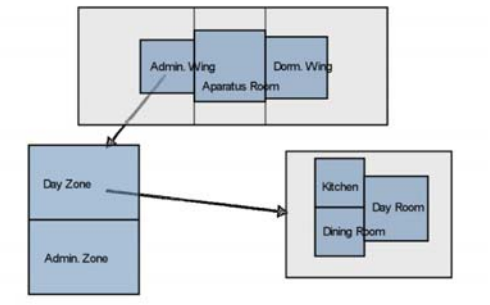


FIGURE 3.2 - THE HIERARCHICAL STRUCTURE FOR A FIRE STATION

Geometric structures define formal relationships between elements. These are the most versatile types of structures. Even with the same topology and hierarchy, a configuration can have many geometrical alternatives as was illustrated by the three Frank Lloyd Wright Houses in Figure 3.1. These can be further categorized as grid structures, repetitive structures, variational structures, and non-regular structures.

Symmetry structures define symmetrical mappings between elements. Grid structures organize proportions, alignments, and angles between configuration elements. Figure 3.3 shows the reflectional symmetry and grid structures of Andrea Palladio's Villa Capra (source: Ching 1996, p.195). Variational structures form rhythm and gradation pattern within elements. Non-regular structures define subtractive and additive configurations achieved by operations such as adding, cutting or slicing, thereby breaking the regularity of a configuration. Figure 3.4 illustrates the variational structure of the Guggenheim museum in its gradation, and its non-regular structure in the union of its distinct parts (source: Laseau 1992, pp.121-127).

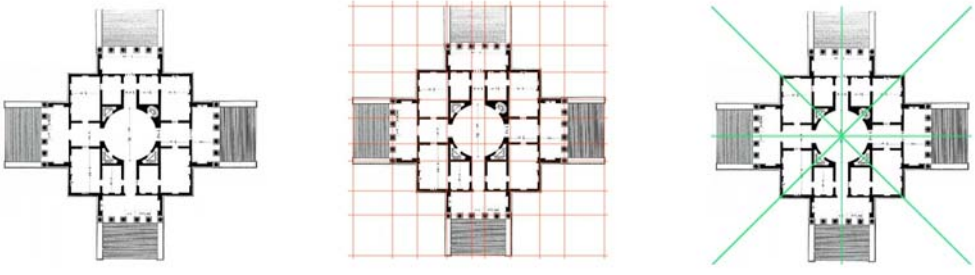


FIGURE 3.3 - THE GRID AND SYMMETRY STRUCTURE OF PALLADIO'S VILLA CAPRA

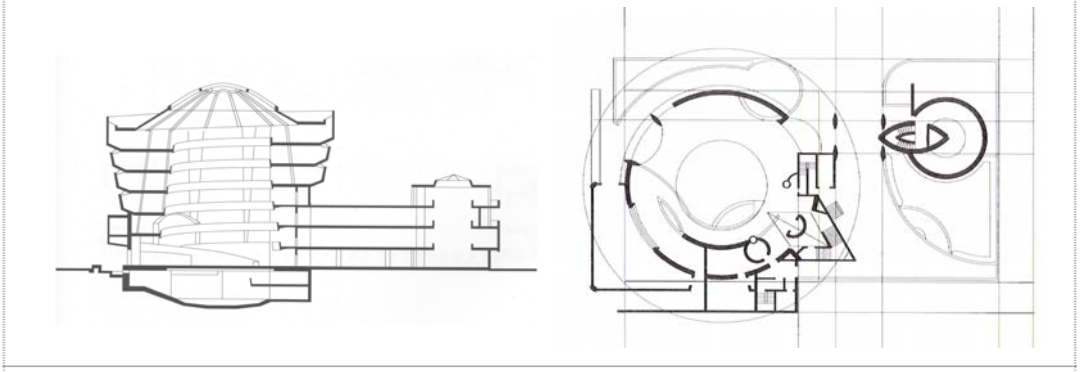


FIGURE 3.4 - VARIATIONAL AND NON-REGULAR STRUCTURE OF THE GUGGENHEIM MUSEUM

Typically the total structure of a building is composed of several superimposed substructures, consisting of topological, hierarchical and geometric abstractions. One can consider early design conceptualization as a process in which these multiple layers of structures are developed either simultaneously or separately then integrated.

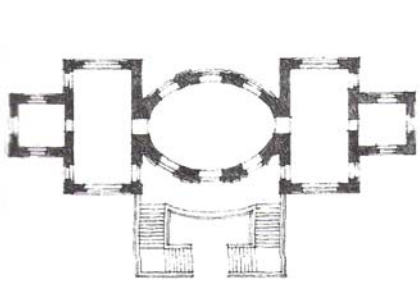
3.1.2. UNITS OF DESIGN STRUCTURES

Complex design structures are compositions of simple spatial relations. The following relations, which constitute the smallest units defining topological, hierarchical, and geometric structures, are classified according to their mathematical properties.

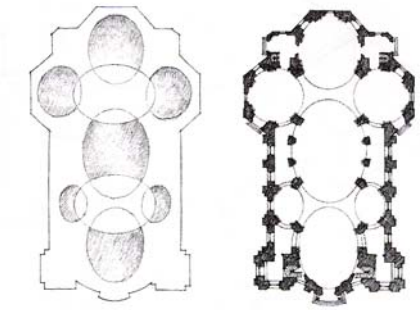
3.1.2.1 TOPOLOGICAL RELATIONS

Topological relations, determined by design requirements, define the proximity between two design elements. The most common is the ***adjacency*** relation, followed by the ***overlap***, ***inside***, and ***distance*** (or separated) relations. Some spaces need to be adjacent due to circulation, whilst others need to be far apart due to thermal or acoustic circumstances. The inside relation defines a boundary: For instance, the site and its setbacks form a boundary that defines the region where a building can be located.

Figure 3.5 illustrates examples of adjacent spaces in Fisher von Erlach's Pavilion Design, and overlapping spaces in Balthazar Neumann's' Pilgrimage church (source: Ching 1996, pp.185-187).



A - Adjacency



B - Overlaps

FIGURE 3.5 - TOPOLOGICAL RELATIONS

3.1.2.2 HIERARCHICAL RELATIONS

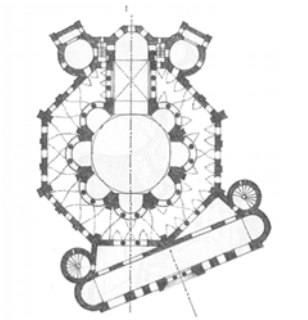
Hierarchical relations relate elements in an organizational hierarchy. These consist of the ***containment***, or ***subshape*** relations. A container can have many constituents, and containment hierarchies can have indefinite depths. For example, a building contains several zones, which in turn contain many spaces. A space can be further decomposed into floor, wall, and ceiling components. Similarly, windows and doors are contained within the walls. In its abstract form, the containment relation is independent of geometry; however, when spaces become actual architectural elements, topologies and geometries become significant factors. The subshape relation, on the other hand, is purely geometric. It defines shapes that are "sub-part" of others, for instance, a façade articulation is a subshape of the façade.

3.1.2.3 GEOMETRIC RELATIONS FORMING SYMMETRY STRUCTURES

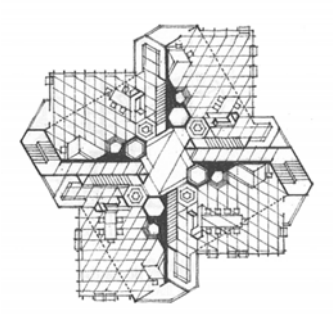
Symmetry relations organize repetitive elements in a configuration. Symmetry is a very common formal organization tool in architecture. It can be observed in floor plans, façades, gardens, and decorative articulations. These include ***translational***, ***reflectional***, and ***rotational*** symmetry. Other forms of repetitive structures can be defined by ***curvilinear*** relations.

Figure 3.6 illustrates examples of symmetry and repetitive relations used in plan organizations. 3.6a - Two major axes of reflection join to form the structure in the church in S. Vitale Ravenna (source: Ching 1996, p.247). 3.6b - Rotational symmetry globally defines the structure in St. Mark's Towers (source: Ching 1996, p.76). 3.6c - The

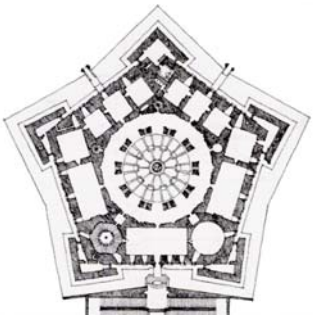
rotational and reflectional symmetry define the pentagonal plan of Vignola's Pallazo Farnese (source: Ching 1996, p.194). 3.6d - The curvilinear repetition defining the structure of Alvar Alto's Baker House (source: Ching 1996, p.207).



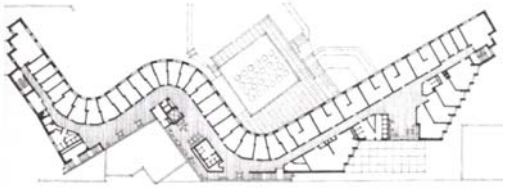
a - Reflectional Symmetry



b - Rotational Symmetry



c - Rotational Symmetry



d - Curve Repetition

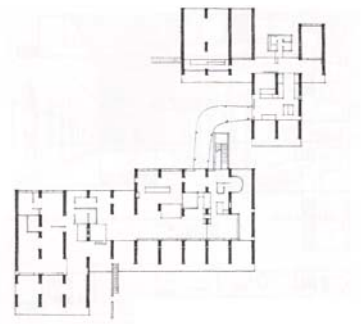
FIGURE 3.6 - SYMMETRY AND REPETITIVE RELATIONS

3.1.2.4 GEOMETRIC RELATIONS FORMING GRID STRUCTURES

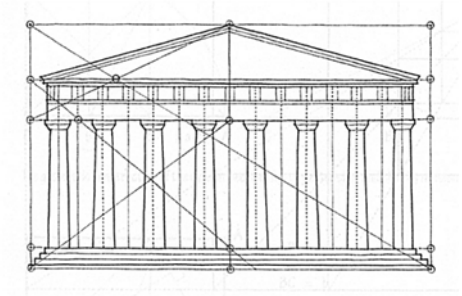
Grid structures are the most common ordering strategies in architecture. Relations defining grid structures can be considered as positional, directional, and dimensional constraints. Positional constraints, such as alignments, define positions of elements in the configuration. Directional constraints define the orientation of elements and angles between them. Dimensional constraints, such as proportion, restrict the sizes of elements. These are used to define proportion systems.

Alignment defines reference lines and planes to align elements. Typically buildings are aligned with their surroundings; windows are aligned on façades, etc. Many architectural schools of thought are based primarily on ***proportion*** systems, for instance, the Golden

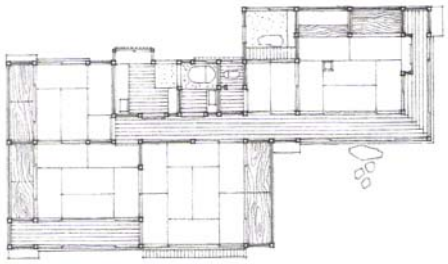
Section, Ken, Modulo, just to name a few. Proportion restricts the relation between two sides of an element or group of elements. The ***Size*** constraint**,** which restricts the length, area or volume of elements, is particularly significant for area requirements in space allocation. The ***angle*** relation, which restricts the angle between two elements, can be used to define minimum roof slopes, for instance.



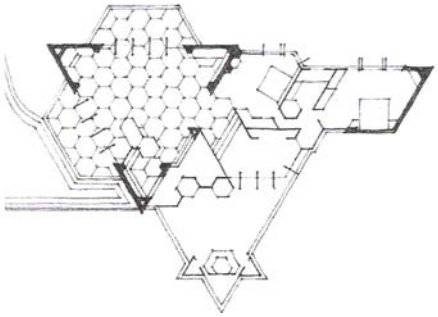
a - Alignment



b - Proportion



c - Size



d - Angle

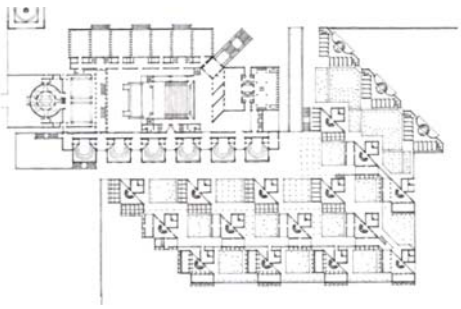
FIGURE 3.7 - RELATIONS DEFINING GRID STRUCTURES

Figure 3.7 illustrates the use of constraint relations. 3.7a - Parallel planes are aligned to form le Corbusier's Sarabhai House (source: Ching 1996, p.144). 3.7b - The Parthenon's Golden proportions are indicated by diagonal lines (source: Ching 1996, p.288). Notice the reflectional and translational symmetry in the Parthenon as well. 3.7c - The modular size constraint is maintained by the Ken proportion in the Japanese residence (source: Ching 1996, p.308). 3.7d - The 60º angle relations are consistently maintained in Frank Lloyd Wright's Sundt house (source: Ching 1996, p.40).

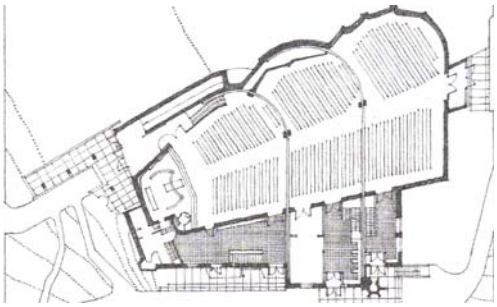
3.1.2.5 GEOMETRIC RELATIONS FORMING VARIATIONAL STRUCTURES

Variational relations such as ***rhythm*** and ***gradation*** introduce variations into repetitive structures. These bring perceived dynamism in configurations that would otherwise be

considered static. Figure 3.8 shows the rhythm in Luis Khan's Indian Institute of Management, and the gradation in Alvar Alto's Church at Vuoksennisk (source: Ching 1996, p.318 and p.369). Notice the overlapping relation in Alvar Alto's Church as well.



a - Rhythm

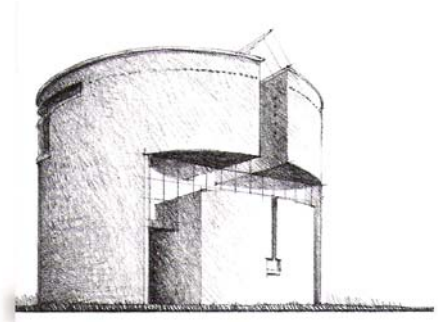


b - Gradation

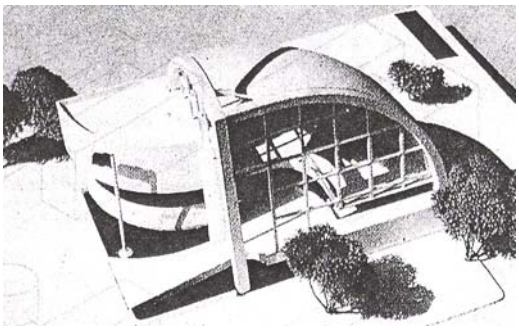
FIGURE 3.8 - VARIATION STRUCTURES

3.1.2.6 GEOMETRIC RELATIONS FORMING NON-REGULAR STRUCTURES

Relations forming non-regular structures are realized through operations. Additive structures are defined by union operation of overlapping elements, while subtractive structures are generated by a subtraction of minor elements from a major element. Other non-regular forms can be defined by subdividing a form or cutting it. Figure 3.9 shows the subtractive structure in Mario Botta's House at Stabio (source: Ching 1996, p.53) and the structure of the cut sphere in the proposal of the Turkish pavilion (source: Onat 1995, p.59).



a - Subtraction



b- Cutting

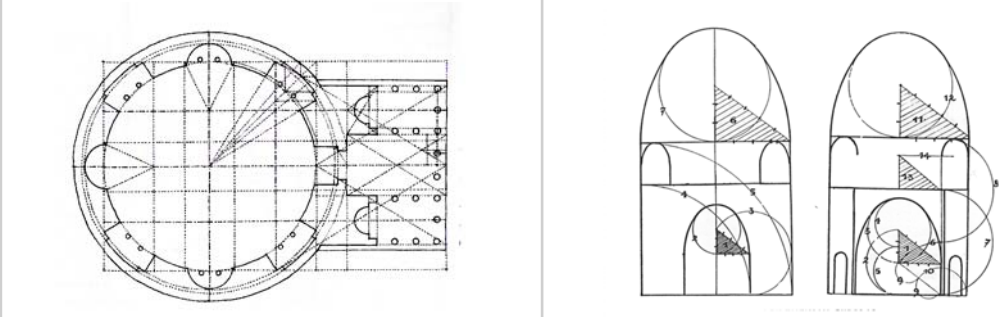
FIGURE 3.9 - NON REGULAR STRUCTURES DEFINED BY OPERATIONS

The diversity of the examples illustrating structures and spatial relations suggest that the concept of structure exist in architecture across the boundaries of time and culture.

3.1.3. REPRESENTATION OF DESIGN STRUCTURES

It is common practice in architecture to visually express geometric relations among elements by using regulating lines. According to Le Corbusier, “A regulating line is an inevitable element of Architecture …. It is an assurance against capriciousness … it confers on the work the quality of rhythm …. The regulating line is a satisfaction of a spiritual order, which leads to the pursuit of ingenious and harmonious relations. … The choice of regulating line fixes the fundamental geometry of the work.” (Le Corbusier 1960, p.71).

Regulating lines are visual abstractions used to represent relations. These are used as guidelines that determine the basic geometric structure in an architectural composition. Regulating lines are used to control proportions and indicate common alignments of elements. Regulating lines can be straight or curved; in fact, regulating arcs and circles have been widely used to represent relations among elements in plans as well as in facades. Figure 3.10 illustrates the use of regulating lines to organize the circular plan of the Pantheon (source: Ching 1996, p.288), and the use of circular lines to regulate the Achaemenian cupolas (source: Le Corbusier 1960, p.72).



a - Straight regulating lines

b - Circular regulating lines

FIGURE 3.10 - REGULATING LINES

From Le Corbusier 's description of regulating lines, it is evident that he promotes their use in design. This can be interpreted as a prescriptive approach toward establishing a

geometric structure to define an order within architectural compositions. He also refers to "the choice of the regulating line," implying that it defines the geometry of the work and the character of the architectural composition.

JNL Durand, in an effort to systemize the architectural design process, introduced a "method to follow in the composition of any project" (Madrazo 1994, p.16). Although his method was widely criticized, it is worthy of mention, because it relies on establishing the structure as a primary design step.

Durand's method begins by establishing the basic structure by regulating lines. It then proceeds by further developing the structure at a higher resolution (Figure 3.11). "Durand has actually described a transformation of a rough scheme into a detailed representation of a building, a transformation of geometry into architecture," (source: Madrazo 1994, p.17).

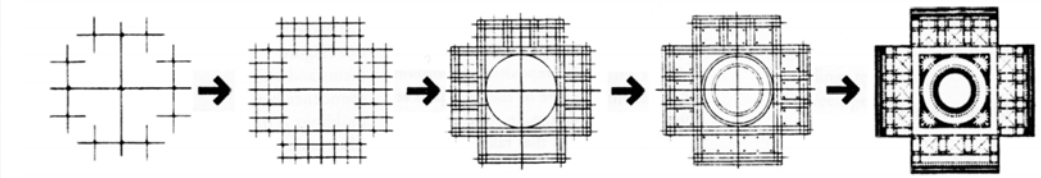


FIGURE 3.11 - DURANT'S PRESCRIPTIVE METHOD FOR DESIGN

Durand's approach highlights the importance of the design structure in the course of design development. However, he uses this structure to restrict the design process, not only to organize the composition. His method overlooks the fact that structures can be identified anytime during the course of design and could be derived from other elements or even discovered as emergent forms.

3.1.4. TRANSFORMATION OF STRUCTURES

In a design space where each configuration is considered a state, transformations are the primary vehicles used to navigate through these states. Transformations can be continuous or discrete. Continuous transformations, which include isometries (translation, rotation, mirror, glide), affinities (scaling, stretching, shearing) and

projections (perspective), preserve the essential properties of elements to which they are applied. Discrete transformations change fundamental properties of their elements and often vary the number of elements in a configuration. Discrete transformations include instantiation, deletion, subdivision, replacement, and Boolean operations. Subdivision takes one element and converts them to many; Boolean operations, which include union, intersection, difference and symmetric difference, take many elements and convert them into one. Replacement, which is the most versatile transformation, takes away one/many element/s from a configuration and returns another/others (perhaps with a completely different form).

Although structures are used as composition tools, these have rarely been used as exploration tools. This is due to the complexity involved in exploring compound structures. Transforming a structure is equivalent to transforming the numerous elements forming that structure, thus, it is equivalent to transforming the whole configuration. Nevertheless, the concept of transforming structures was investigated within various domains as analytical or generative tools. Dürer, Thompson (1971), Laseau (1992) investigated continuous transformations of structures, while Steadman (1998) investigated discrete transformations of the structure of an archetypal building to generate other building types.

Dürer described transformations of the human profile by introducing variations in the underlying grid structure (Mitchell 1990, p.116). By varying the coordinates of the grid lines, he varied the proportions of the profile. By varying the angle of the grid lines, he introduced shear or perspective in the profile (Figure 3.12). Therefore, the corresponding profiles were changed according to new grids, and Dürer was able to produce various caricatures (source: Mitchell 1990, p.116).

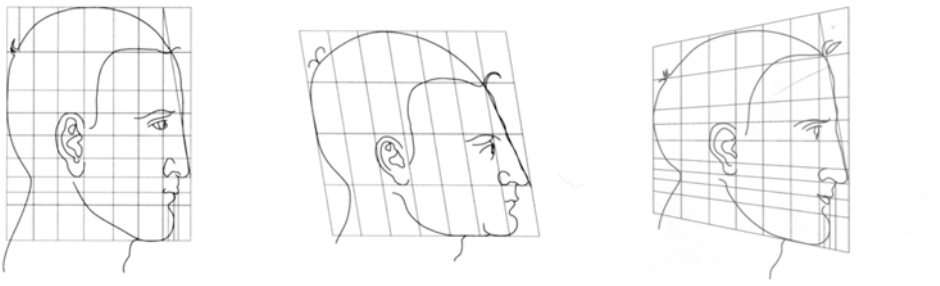


FIGURE 3.12 - DÜRER'S HUMAN PROFILES

D'Arcy Thompson (1971, pp.268-325) extended this concept further by using radial coordinates to vary the curvature of the grid lines. In his study of related forms, Thompson developed a systematic method of transforming grid lines that corresponded with natural growth patterns. He was able to develop a deformation scheme that allowed him to trace the similarities and difference in proportions between species. By varying the curvature of each line of the coordinate system differently, he was able to map one the skeleton or profile of one species into another. Figure 3.13 shows examples using this system to compare fish and crustacean forms. Thompson also applied this technique to transform skulls and bones of various mammals.

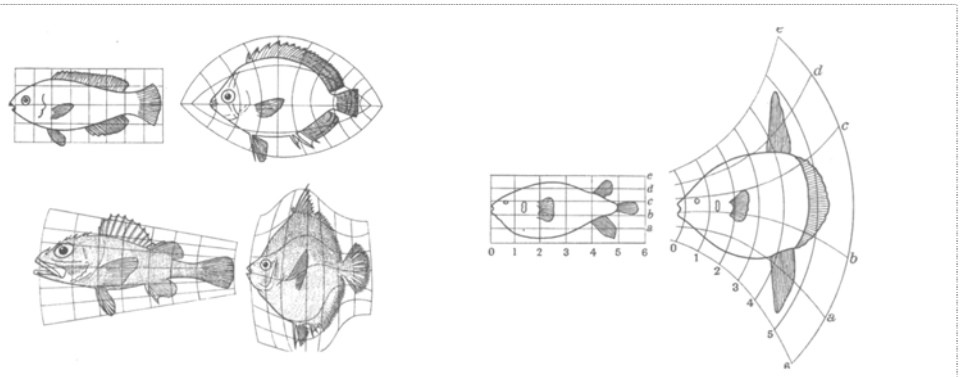


FIGURE 3.13 - D'ARCY THOMPSON'S TRANSFORMATIONS OF FISH SPECIES

Laseau and Tice (1992, p.7) emulated Thompson's approach of varying grid curvatures to determine the transformation mappings for two of Frank Lloyd Wright's Usonian Houses (Figure 3.14).

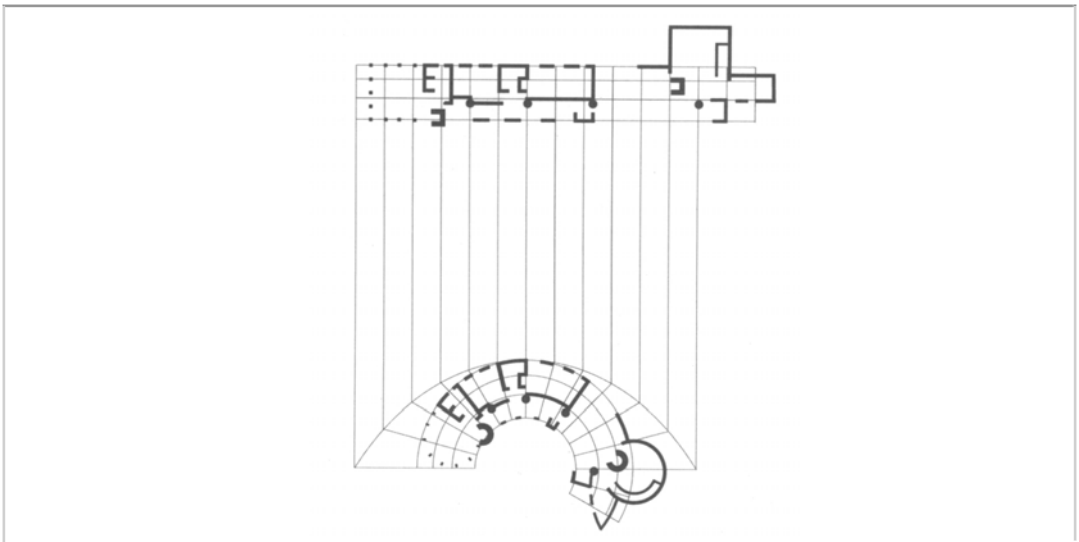


FIGURE 3.14 - LASEAU'S ANALYSIS OF LLOYD LEWIS AND DAVID WRIGHT HOUSE PLANS

Steadman (1994, pp.S7-S30) presents a classification of built form based on lighting (natural, and artificial), and average room size (cellular, open space, and hall). This yields six categories, further classified by the number of stories and by the type of natural lighting (side or top). He also investigated a method of transforming built form from a standard parametric "archetype" into any configuration within these classifications (Steadman 1998, p.98). The method included discrete transformations of suppressing parts, connecting parts, as well as dimensional transformations such as scaling (Figure 3.15).

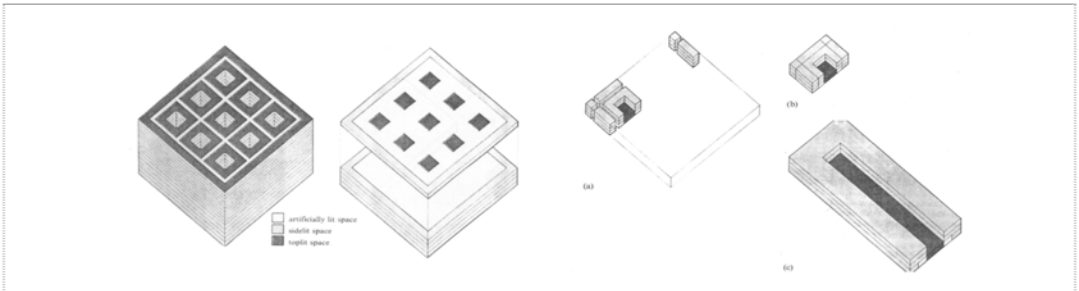


FIGURE 3.15 - STEADMAN'S DISCRETE TRANSFORMATION OF STRUCTURES

Figure 3.16 illustrates a hypothetical example of transforming Life House to Jester House, by using discrete transformations of replacing forms. The process begins with Life house. It then proceeds by replacing the square spaces by circular ones and the

rectilinear masses by curvilinear ones, whilst adjusting their positions and sizes.

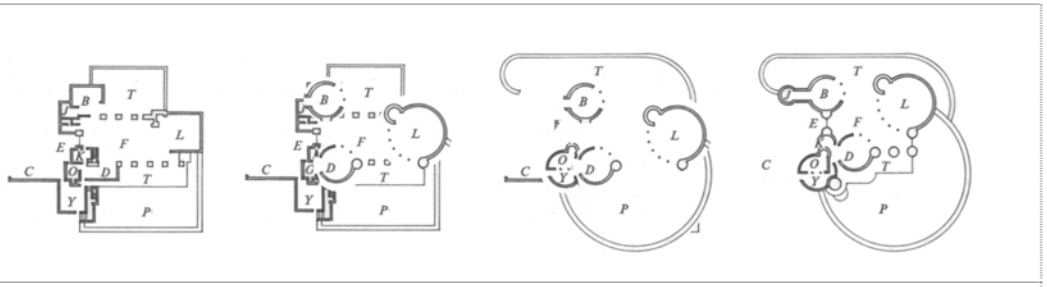


FIGURE 3.16 THE HYPOTHETICAL TRANSFORMATION FROM LIFE HOUSE TO JESTER HOUSE

As these examples illustrate, slight transformations of structures can greatly affect configurations, and are undoubtedly a source for intellectual stimulations and therefore, are powerful venues for design exploration.

3.2. EMPIRICAL OBSERVATION OF DESIGN STRUCTURES

In this section, I describe an empirical observation based on a protocol experiment that illustrates the use of structures in early phases of design. The goal of this study was motivated by the need to discover whether design structures are merely theoretical constructs, or whether they are practical entities playing an active role in design development. It led to evidence of the use of regulating elements, not only to express and compose structures of architectural configurations, but also as strategic devices in guiding the design process. Further analysis of the data indicates that these devices play a significant role in defining strategies for structuring sub-problems, managing part-whole hierarchies, organizing topology-geometry, scaffolding the design process, and restructuring of problem parameters. In this section, I present the abridged version of results; for the complete results the reader is referred to Akin, Ö. and H. Moustapha "Strategic Use of Representation in Architectural Massing" (Design Studies, 25, 1, 2003, pp31-50).

3.2.1. EXPERIMENTAL SETUP

The protocol experiment consisted of observing six architects while they designed a three-dimensional massing model of a dormitory building on the Carnegie Mellon campus. Each session lasted two hours on the average, and was recorded on videotape. All participants, who are professional architects, with experiences ranging from 5 to 25 years, are referred to as P1 to P6. The corresponding protocol sessions are referred to as S1 to S6, where the first three were carried out in the sketch medium (pencil and paper) and the three others in the computing medium (CAD system). Alternatives generated by each participant are referred to as Ai1 to Ain, where 'i' corresponds to the participant ID.

3.2.2. REGULATING ELEMENTS AS DESIGN STRATEGIES

The principal mechanism utilized in structuring massing activities was the use of regulating elements such as axes of symmetry, alignment axes, and bounding lines. All participants maintained geometric order in their designs using such mechanisms. Although they freely manipulated (added and removed) massing elements, through the use of regulating elements they were able to preserve their underlying structures and even

accentuate them.

3.2.2.1 REPRESENTING STRUCTURES

In the protocol sessions, participants used external representations, whether verbalizations, sketches, or computational records, with the apparent purpose of carrying on a design dialogue with themselves. They used regulating elements to express their design organization, either explicitly or implicitly. The explicit expression took either a graphic or verbal form. In the implicit expression, the relation was defined by the position of elements, but no lines were drawn nor discussed, as in the case of P2 (Table 3.1a). Participants also made verbal references to regulating axes as in the case of P4 (Table 3.1b), who defines the geometric structure verbally prior to any drawing activity. Even though the axis is not explicitly depicted, its verbal presence serves, just as successfully, the same function that the explicit axes serve in other protocol episodes.

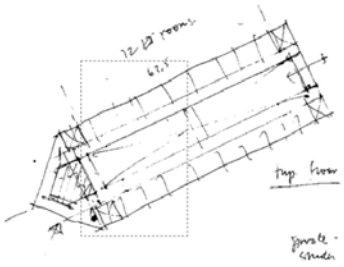
	1.9.1 "One possibility is to make it a continuing line from Margaret Morrison and then design at the corner a transition into the new coordinate system." 1.9.2 "The same way in which the dormitories use the octagon to transition between the Carnegie Mellon grid and the dorms grid, which happens to be 45° such that an octagon can do it very nicely."
a - Implicit representation of axes	b - Verbal representation of axes

TABLE 3.1 - IMPLICIT AND VERBAL REPRESENTATION OF REGULATING AXES

3.2.2.2 STRUCTURING THE SUB-PROBLEM

One of the most straightforward ways of structuring an ill-structured problem is to break it into more and more specialized parts. In the case of massing, the decomposition of the design problem is graphically driven. The participants appear to create local problem sub-structures by adding sub-division lines into the massing representation as for the case of P1 and P3 (Figure 3.17).

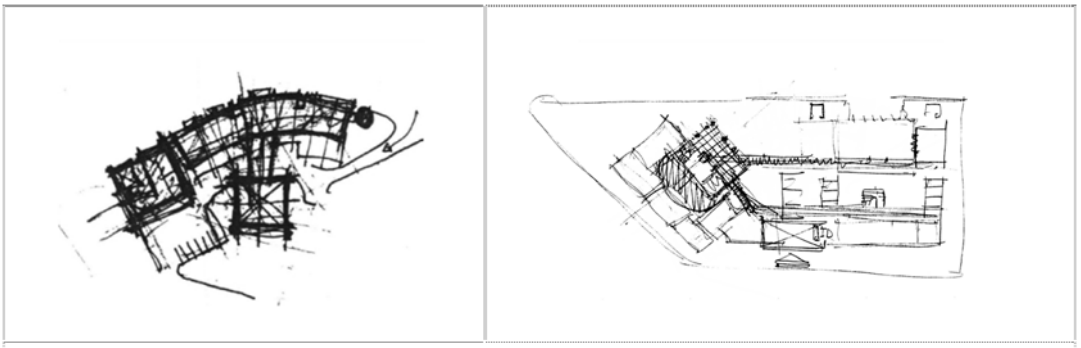


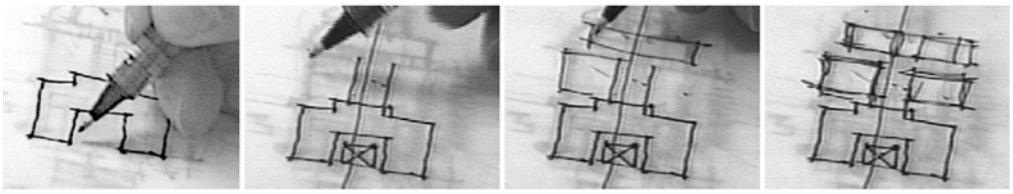
FIGURE 3.17 - P1 AND P3 DEVELOPING SUBSTRUCTURES

3.2.2.3 MANAGING PART -WHOLE HIERARCHY

A popular regulating element, observed during the protocol sessions is the alignment line that aligns individual design elements with respect to it. Aside from the compositional orders that result from such use, the alignment axes represent meta-elements that control the spatial organization of other, lower-level elements. The evidence suggests that the massing strategies defined here, establish a two-tier hierarchy between the regulator (super-node) and the regulated (sub-node). Nested regulating elements of massing can then create indefinitely deep hierarchies.

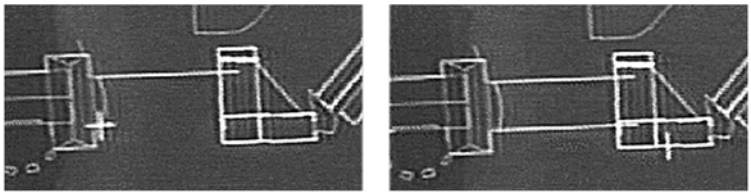
3.2.2.4 SCAFFOLDING THE DESIGN PROCESS

Another view of the protocol data relies on the scaffolding metaphor (Akin et. al. 2003). Just as a scaffold provides a structure for accommodating construction activities, the physical massing activities of design elements relies on the framework created by the regulating elements. There appears to be a two-way interaction between the regulating and massing elements, particularly in the manner in which regulating elements are derived from masses and, inversely, masses are guided by regulating elements. Scaffold creation seems to be based on the extension of alignments in the current design. This is particularly evident in P1 (Table 3.2), and P4 (Table 3.3).



5.1.1 P1 draws the courtyard.
5.1.2 She draws the public (administration & restaurant) around the courtyard.
5.1.3 Then she draws the central axis line.
5.2 P1 develops the dorm zone.
5.2.1 She starts by drawing 3 blocks.
5.2.2 She then divides one of them along the central axis ending with 4 blocks.

TABLE 3.2 - P1 DERIVING AN AXIS FROM A MASSING ELEMENT



P4 uses the strategy of extending two lines from an existing building's external protrusion, and utilizes these as guides to create a protrusion onto her own proposed building. She then discovers a novel relation (Akin 1996) between these lines and her own buildings sub-structure. She says "Ahaaa … I found a very interesting relationship". This causes an adjustment to the proposed massing configuration.

TABLE 3.3 - EXTENDING A SUBDIVISION LINE

3.2.2.5 ORGANIZING TOPOLOGY AND GEOMETRY

Regulating elements of massing also appear to be representing the topology of a given geometric composition. For instance, axes are used to represent associations and alignments of spaces, independent of shape and size. Spaces can be strung along an axis creating a linear topological structure. Alternatively, multiple axes can be used to create much more complex relationships, like grids and urban road patterns. Table 3.4 shows P1's sketches for determining the topological relationships.

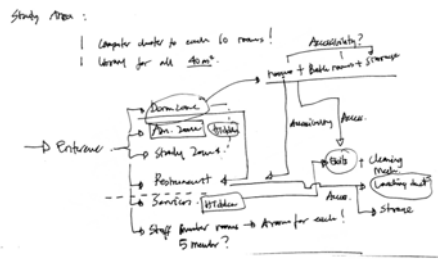
1.2 P1 identifies patterns in the program. 1.3 She divides the program into zones (Entrance, administration zone, study zone, dorm zone, etc.) 1.4 She analyses relationships between the zones by drawing arrows. 1.5 Adjacencies, Accessibility (public, semipublic, private), Exits. 1.6 Start developing concepts based on these relations.	

TABLE 3.4 - P1 DERIVING TOPOLOGY FROM PROGRAM RELATIONS

3.2.2.6 RESTRUCTURING OF PROBLEM PARAMETERS

One of the behavioral characteristics of expert designers is a skill and propensity to restructure design problems (Akin and Moustapha 2003). The protocol data showed several forms of restructuring. A frequent form of restructuring is through the development of alternatives such as in the case of P1 (Figure 3.18). However, restructuring the problem does not always mean a wholesale redesign, or the generation of an entirely new alternative. Occasionally, the participants achieved the same effect by modifying key elements or secondary regulating element in the solution domain, as in the case of P4 (Figure 3.19).

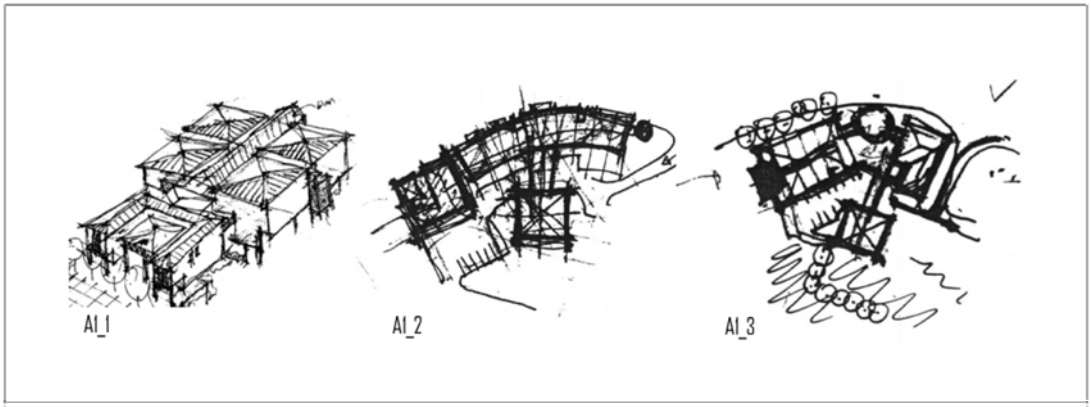


FIGURE 3.18 - P1'S ALTERNATIVES ILLUSTRATING MAJOR CHANGES IN STRUCTURE

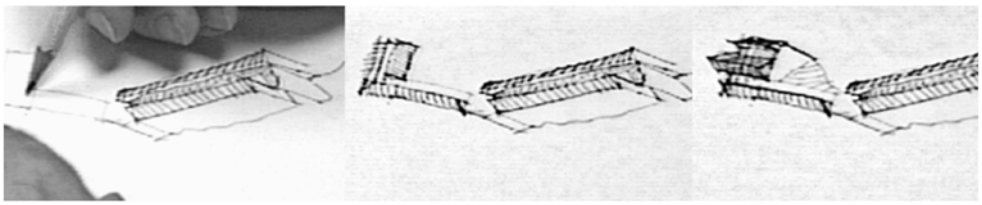


FIGURE 3.19 - P4 CHANGES THE GEOMETRIC STRUCTURE OF THE ROOF CONFIGURATION.

The protocols results illustrated several situations where designers handled the geometric structure of a massing configuration in such a way that they seemed to be doing more than just composing forms. Repeatedly and consistently, the data showed behaviors that structure and manage the design development process. This evidence highlights the importance of structures in early design activities and emphasizes the strategic and active nature of regulating elements for design exploration.

3.3. THE ICE FRAMEWORK FOR EXPLORING WITH DESIGN STRUCTURES

After surveying the types of design structures in notable architectural configurations, and after establishing their strategic role in early design development through empirical observation, I introduce the Interactive Configuration Exploration (ICE) framework. The ICE framework is a comprehensive mechanism that relies on architecturally significant structures for design representation, and relies on transformation of structures for design exploration. The ICE framework supports transformation of structures conceptually, formally, and computationally, with the purpose of maximizing the exploratory potential of design configurations. Conceptually, ICE is an exploratory venue in the domain of architecture. Formally, ICE is a notational representation. Computationally, ICE is an interactive real time experience.

The approach used in the ICE framework is to separate the organizational dimension of design structures from the physical dimension of architectural elements. Therefore, each dimension can be addressed separately. Through design structures, one can explore a design configuration as a whole, while maintaining its integrity. By manipulating individual elements, one can explore variations within the same structure. By manipulating the structure itself, one can all redefine all elements organized by the structure, and thus transform configurations completely.

3.3.1. REGULATORS

The principal vehicles used in the ICE framework, are regulators, which are abstraction that capture the spatial relations of design structures. Each regulator is associated to a specific set of elements in the configuration and is augmented with control over these elements. Regulators, which are inspired by regulating lines, are expressed computationally analogous to the way spatial relations are expressed traditionally. However, these are extended to include regulating points, planes and volumes, which, in combination, express the structure of the architectural configurations in three dimensions.

- Regulating points include intersections of lines and centers of rotations.
- Regulating lines or arcs include intersections of planes, axes of rotations, axes of

symmetry, alignment lines, bounding lines and diagonal proportion lines.

- Regulating planes include axis of symmetry for volumes, alignment planes and bounding planes.
- Regulating volumes include bounding volumes.

In the ICE framework, there is a regulator corresponding to each of the spatial relations described in Section 3.12. Regulators convert relations into dynamic, reconfigurable entities rather than static entities. Regulators control their associated elements according to specific mathematical properties. Each regulator type encapsulates a formula (a polynomial equation), by which it controls the attributes of its associated elements. The parameters of equation can be set and modified by users. These modifications result in changing position, orientation, curvature or other factors of the regulators which, in turn, influence positions, orientations, curvatures, or other factors of regulated elements.

Regulators are not merely visual abstractions, but computational abstractions that represent handles for the structure of the configuration. Therefore, regulators allow the use of the structure as an exploratory venue in order to promote intellectual stimulation.

A designer explores an architectural configuration with regulators in the following manner. He/she composes structures by generating regulators of various types and associating various configuration elements to them. He/she investigates variations within structures by manipulating configuration elements; regulators ensure that relations among elements are preserved. He/she transforms structures of configurations by manipulating regulators and modifying their parameters; regulator transforms the configuration accordingly.

Regulator are conceived with transformation of structure in mind and are designed to capture the most basic relations that, when composed together, define complex structures. In the following section, I describe exploratory patterns for preserving as well as for transforming configurations defined by each regulator type.

3.3.1.1 TOPOLOGICAL REGULATORS

Topological regulators (Figure 3.20) correspond to topological relations. These binary regulators ensure that the two topologically related elements always maintain their relations even when one of them is manipulated. Elements controlled by the ***Adjacency*** regulator are always adjacent; when one is moved, the other one follows. The ***Distance*** regulator maintains a specific minimum, or maximum, distance by preventing the elements from moving beyond that distance. Elements controlled by the ***Overlap*** regulator always remain overlapping; motions that disjoin these elements are prevented. In the case of the ***Boundary*** (or inside) regulator, one element always remains inside the other. The internal element is not allowed to move beyond the boundary element. When the boundary element is moved, the internal element follows. Also the internal element cannot be resized beyond the size of its boundary.

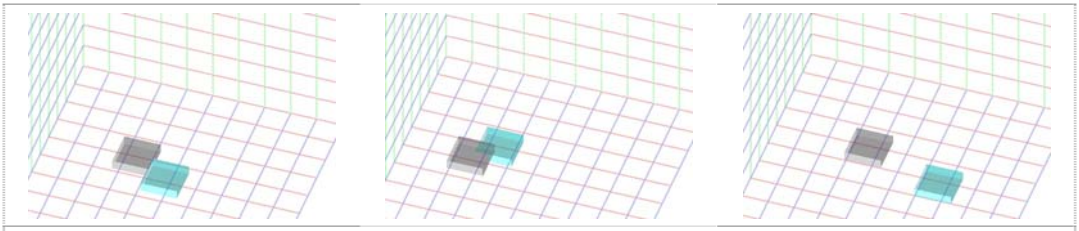


FIGURE 3.20 - ADJACENCY, OVERLAP AND DISTANCE REGULATORS

3.3.1.2 HIERARCHICAL REGULATORS

Hierarchical regulators correspond to hierarchical relationships. The ***Containment*** regulator defines a container and its constituent elements. The Containment regulator in itself has no geometrical implications; however, it can be composed with other regulators to introduce geometrical and topological restrictions to the hierarchy. For example, a ***Boundary*** regulator prevents contained spaces from being positioned outside the container zone. The ***Subshape*** regulator establishes geometrical coherence between shapes, and ensures that the subshape always has the same geometry as the supershape. If the geometry of the supershape, for instance a façade, is modified, its subshape articulations are also updated as well.

3.3.1.3 GEOMETRIC – TRANSFORMATIONAL –REGULATORS

Transformational regulators are inspired by symmetry relations, which are based on isometry transformations in 3-dimensional space such as translation, rotation, and reflection. However, these are extended to include affine transformations, such as scale and shear, as well as equations in space to represent additional repetitive relations, such as curves. Transformational regulators are the primary constructs of the ICE framework because, in addition to controlling relations between symmetrical elements, transformational regulators generate elements based on these relations.

Transformational regulators generate multiple outputs, from a single input element. Regulators control the position/orientation of outputs, with respect to the input. Transformational regulators ensure that the symmetrical relation between these elements is preserved upon their manipulation, and if the regulator's geometry or variables are modified, the relation is redefined.

The ***Translation*** regulator generates outputs translated along a straight line. The variables of the Translation regulator include the distance, the orientation of the line and the number of outputs. Figure 3.21 illustrates the effects of changing the distance and the orientation of the Translation regulator.

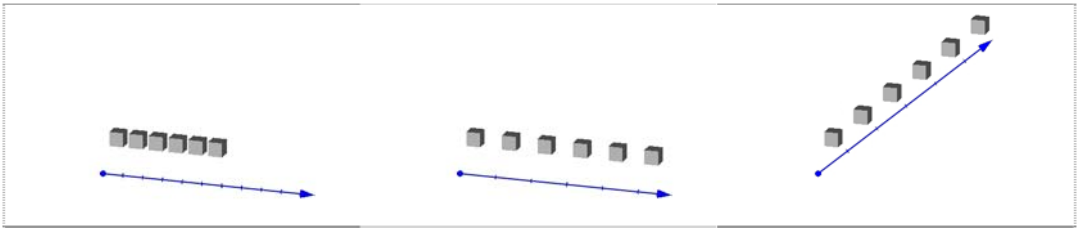


FIGURE 3.21 - TRANSLATION REGULATOR

The ***Rotation*** regulator creates outputs rotated about a point/line. The variables of the Rotation regulator are the rotation degree, the number of outputs, the position, and orientation of the Rotation regulator. Figure 3.22 illustrates the effects of changing the orientation and the position of the Rotation regulator.

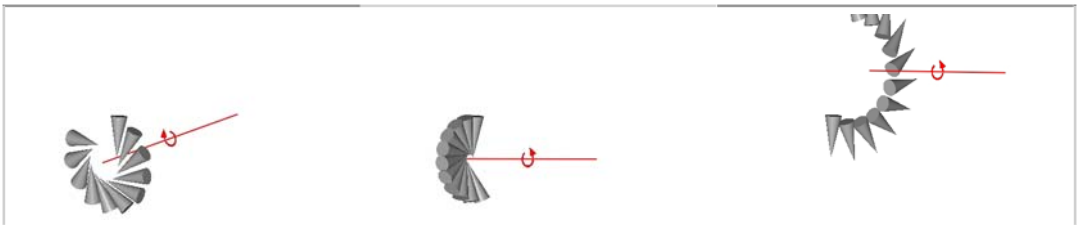


FIGURE 3.22 - ROTATION REGULATOR

The ***Mirror*** regulator generates an output reflected about a line or plane. The variables of the Mirror regulator are position and orientation of the mirror plane. Figure 3.23 illustrates the effects of changing the orientation and the position of the Mirror regulator.

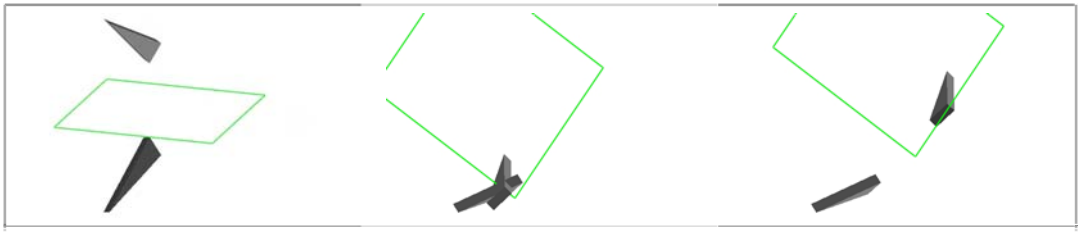


FIGURE 3.23 - MIRROR REGULATOR

The ***Dilation*** regulator scales the outputs about a specific center of scale. The variables of the Dilation regulator are the scale factor, the center of scale, and the number of outputs. The ***Shear*** regulator creates sheared outputs. Its variables are the shear factor, and the direction of orientation of the Shear regulator. Figure 3.24 illustrates the effects of moving the center of scale and rotating the Shear regulator.

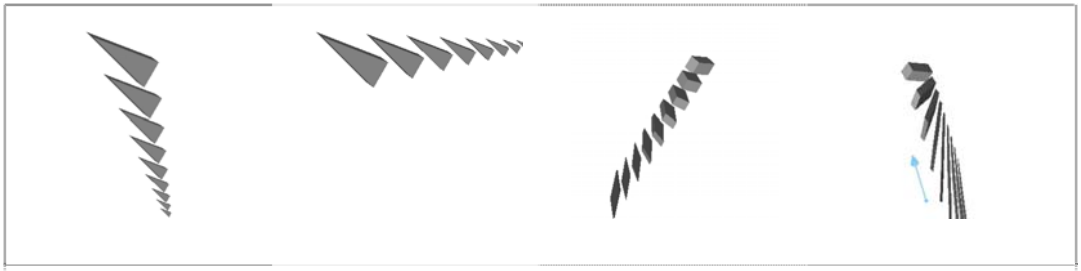


FIGURE 3.24 - DILATION AND SHEAR REGULATORS

The ***Curve*** regulator produces outputs along a curved line. The variables of the Curve

regulator are the distance between outputs, the curvature, and the direction of the curved line. Figure 3.25 illustrates the effects of increasing the distance and rotating the Curve regulator.

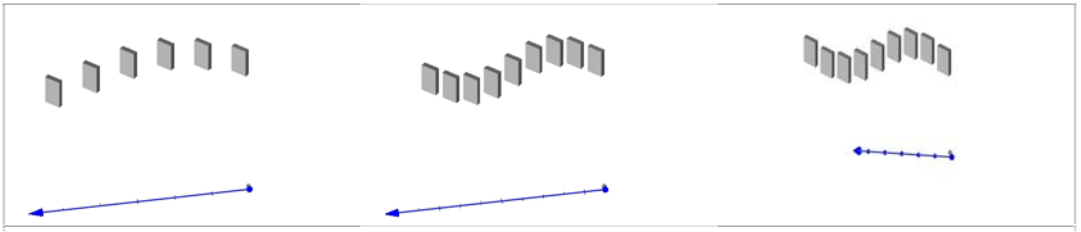


FIGURE 3.25 - CURVE REGULATOR

The basic set of transformation regulators can be composed to form more complex relations. The Glide regulator (Figure 3.26a) is formed by composing Translation and Mirror regulators and the Screw regulator (Figure 3.26b) is formed by the composing Translation and Rotation regulators.

FIGURE 3.26 - GLIDE AND SCREW REGULATORS

3.3.1.4 GEOMETRIC – CONSTRAINT – REGULATORS

Constraint regulators correspond to relations defining grid structures. Constraint regulators restricts positions, define minimum/maximum values that must not be exceeded, or an incremental module that must be satisfied.

The ***Alignment*** regulator restricts elements to a reference point, line, or plane. The elements are allowed to move only along the alignment reference. If the Alignment regulator is moved or rotated the elements are re-aligned with it. Alignment can also restrict element along a circle or curve. Figure 3.27 illustrates the effects of linear and circular alignments as well as rotating the Alignment regulator.

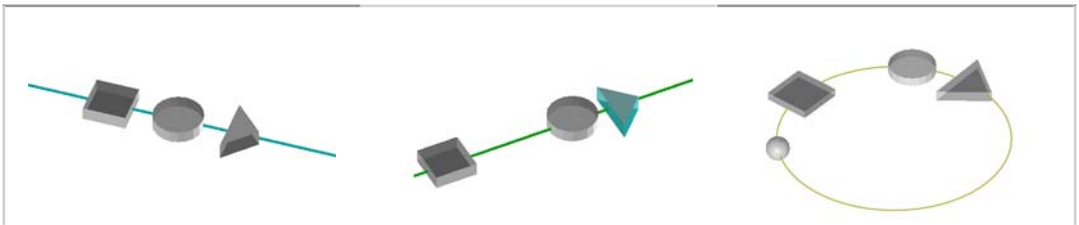


FIGURE 3.27 – ALIGNMENT REGULATOR

The ***Proportion*** regulator restricts the aspect ratio of an element by means of a diagonal line. The element can only be resized within these proportions. Moving or rotating the diagonal line redefines new proportions. Figure 3.28 illustrates the effects of manipulating the Proportion regulator.

FIGURE 3.28 – PROPORTION REGULATOR

The ***Angle*** regulator restricts the angle between two elements. If one element is rotated beyond the minimum/maximum range, the other element is also rotated to preserve the angle. If the angle's values are updated, the shapes are reconfigured. Figure 3.29 illustrates the effects of reconfiguring the angle.

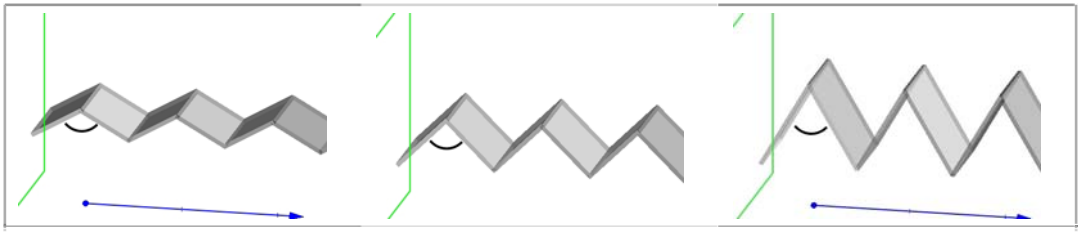


FIGURE 3.29 – ANGLE REGULATOR

The ***Size*** regulator restricts the lengths, areas and volumes of elements to a maximum or minimum value. Elements cannot be resized beyond their minimum/maximum range.

In addition to positional, directional, and dimensional constraint-based regulators, value constraint regulators are introduced. The ***Equivalence*** regulator defines an equivalence relation between attributes across many shapes. When this attribute is changed in one element, it is changed in the others. Color, form, and size are applicable attributes. For example, in Frank Lloyd Wright's Jester House (Figure 3.1), most spaces are equivalent in their circular form, yet their other attributes, including scale and position are not linked in any other way.

3.3.1.5 GEOMETRIC – VARIATIONAL – REGULATORS

Variational regulators correspond to the relations defining variational structures. The ***Rhythm*** regulator creates rhythmic effects with the output set; if the rhythm coefficient and cycle are modified the configuration is updated. The ***Gradation*** regulator creates gradual effects with the output set, as the gradation coefficient is changed, the configuration is updated. Figure 3.30 illustrates the rhythm and gradation regulators composed with a Rotation regulator.

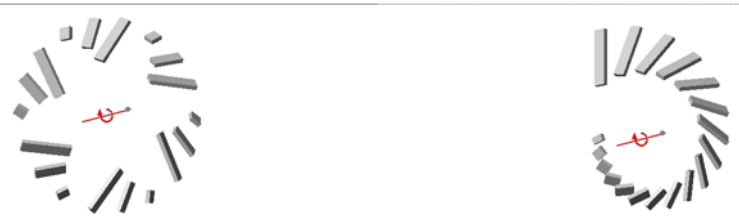


FIGURE 3.30 – RHYTHM AND GRADATION REGULATORS

The variation regulators are extended to include the exception and the differential regulators. The ***Exception*** regulator designates an element as being different from the output set. The ***Differential*** regulator introduces a rhythm or gradation effect as the regulator is applied to various inputs. Figure 3.31 shows two exceptions applied to a Rotation regulator and a Differential regulator applied to the second of two successive Translation regulators.

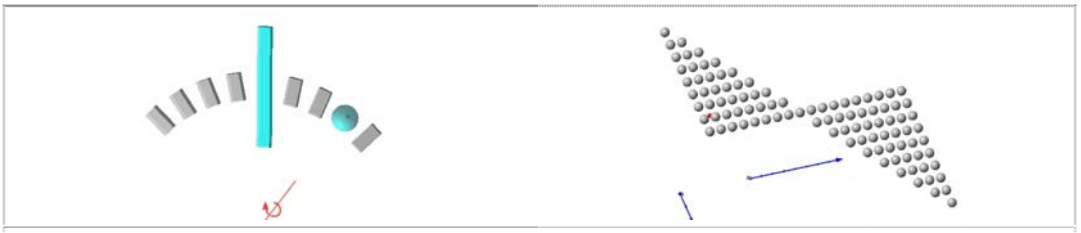


FIGURE 3.31 – EXCEPTION AND DIFFERENTIAL REGULATORS

3.3.1.6 GEOMETRIC – OPERATIONAL – REGULATORS

Operational regulators correspond to operations defining non-regular structures. These include Boolean operations, subdivisions and cutting.

The Boolean operation regulators, inputs several elements and generates the ***Union***, ***Subtraction***, or ***Intersection*** as an output**.** The Boolean regulator ensures that the resultant form is redefined when the input forms are manipulated. Figure 3.32 shows the effects of resizing the subtracted form.

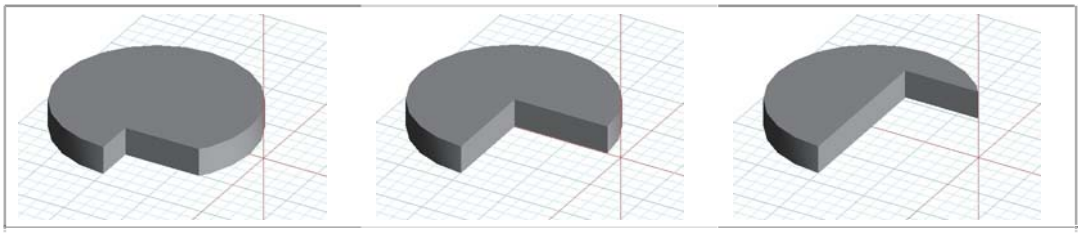


FIGURE 3.32 – BOOLEAN DIFFERENCE REGULATOR

The ***Subdivision*** regulator inputs an element and subdivides it into equal subparts, which are produced as outputs. It can be composed with variational regulators to introduce rhythmic parts, which are gradually increasing or decreasing in size. When the original shape is modified, the subdivisions will be redefined. The ***Cutting*** regulator cuts the shape along a plane (or planes). If the plane is modified, the cutting pieces are redefined. Figure 3.33 shows the effects of rotating subdivision planes and redefining the original shape.

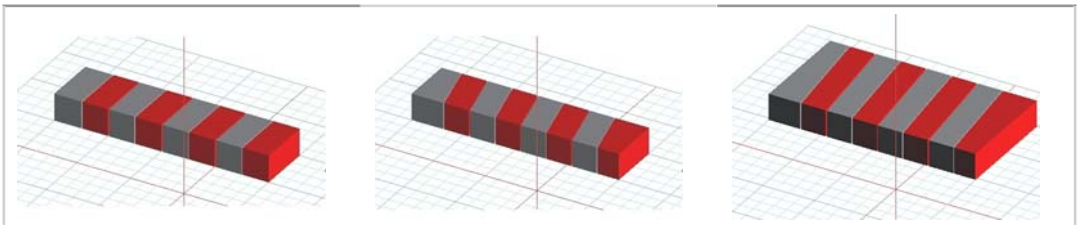


FIGURE 3.33 – SUBDIVISION REGULATORS

3.3.2. THE DYNAMICS OF REGULATORS

The ICE framework is designed with the goal of supporting cyclic/iterative exploration, and, consequently, with flexibility as a major priority. Regulators offer the ability to produce topological, hierarchical, symmetry and grid structures. Regulators introduce constraints and variations in these structures, and produce non-regular forms through operations. Regulators are intended to support transformation of structures at a continuous, as well as at a discrete level.

To define such complex structures, it is necessary to compose regulators. Regulators can be composed simultaneously to create complex relationships from simple ones (Figure 3.26 shows the composite regulators of Glide and Screw). Regulators can also be composed successively to form elaborate patterns, such as in Figure 3.34, which illustrates a linear element with the successive application of Rotation, Mirror, and Translation regulators. Furthermore, as a flexible measure, regulators can be decomposed.

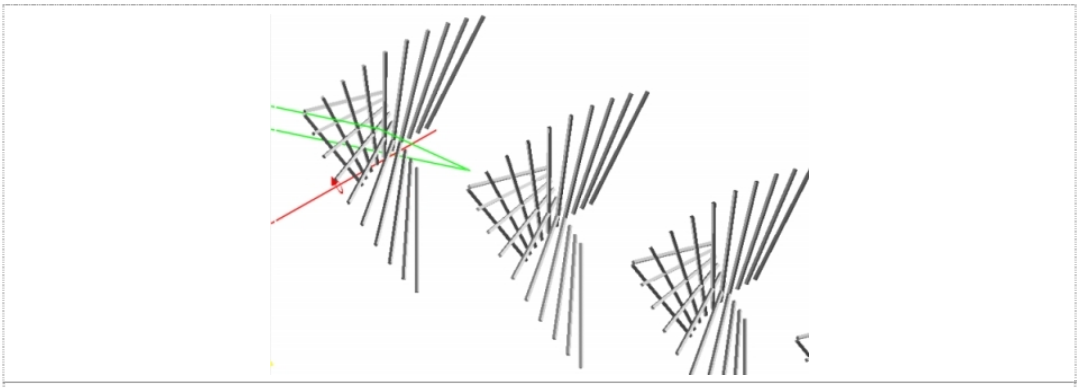


FIGURE 3.34 – REGULATOR SUCCESSIVE COMPOSITION

Regulators and elements are dynamically associated and dissociated. Consequently, elements can have multiple regulators (Figure 3.35a), regulators can have multiple elements (Figure 3.35b), and mixed structures can be explored within a single configuration. Furthermore, association can be terminated at anytime.

FIGURE 3.35 – REGULATOR SHARING

Regulators support continuous as well as discrete generations. Points are regulated continuously to create shapes, and shapes are regulated discretely to create patterns and configurations. Regulators can also be regulated to create complex schema, resulting in multiple control structure.

The overall structure of a configuration can be decomposed into a set of substructures: topological, hierarchical, symmetrical, grid, and others. It is not possible to explore all these structures at the same time, because the exploration space of one structure will yield configurations that destroy relationships of another structure. Regulators offer the ability to focus on exploring a particular structure, while deactivating the others. This activation and deactivation of regulators allow users to define their sequence for exploring structures, depending on the design phase and their preferences. For example, one can explore of topological structures then geometrical structures. It also allows the exploration of configurations with and without specific relationships.

As an additional measure of exploratory flexibility, configuration elements as well as regulators can be replaced at any time, in order to explore different structures without reestablishing associations. Figure 3.36a shows the successive composition of Rotation and Translation regulators. The Translation is replaced by a Mirror in Figure 3.36b and by a Rotation in Figure 3.36c.

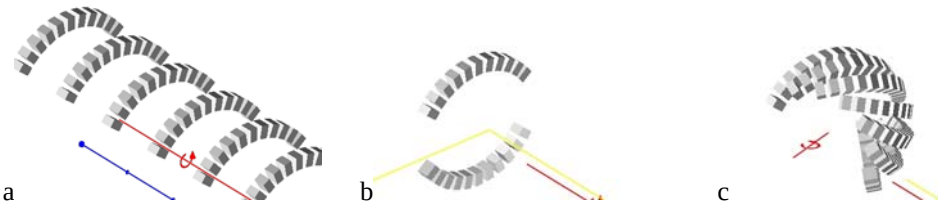


FIGURE 3.36 –REPLACING REGULATORS

Regulators can be applied at various resolutions, including local to a part of the configuration, and global to the whole configuration. The depth to which regulators can be applied is theoretically infinite. Furthermore, regulators can be defined at any stage of developing the configuration, where these can be identified with any element or extended from any element.

Tables 3.5, 3.6, and 3.7 illustrate examples of exploring building configurations with regulators. Table 3.5 shows the exploration of single floor layout, where curvatures and directions of walls are investigated. Table 3.6 shows the exploration of a building mass, where roof slopes and curvatures are investigated. Table 3.7 shows the transformation of one simple configuration, into an elaborate Chinese pagoda, with just a series of simple steps.

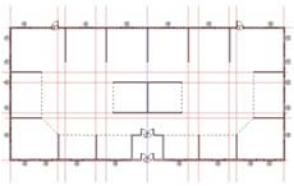
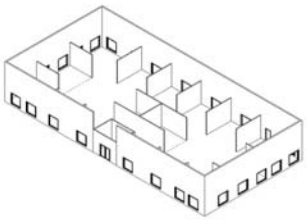
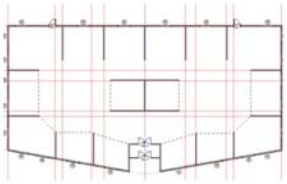
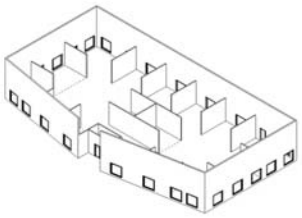
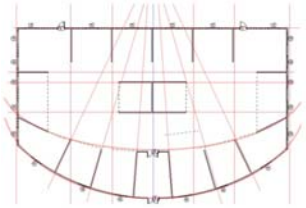
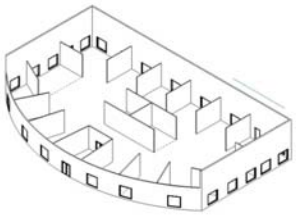
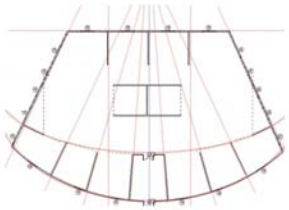
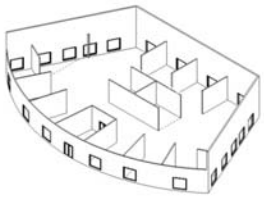
		The initial layout consist of one Mirror regulator and several Alignment regulators forming a grid. The Mirror regulator controls both the walls and the Alignment regulators.
		THE USER ROTATES ONE SIDE OF THE ALIGNMENT REGULATOR OF THE FAÇADE The Alignment regulator realigns all walls and windows according to its new orientation. The Mirror regulator ensures that the change is properly reflected on both sides.
		THE USER CHANGES THE CURVATURE OF THE ALIGNMENT REGULATOR OF THE FAÇADE The Alignment regulator re-configures the walls, doors, and windows, such that they follow the curve. The regulator of the façade also automatically reconfigures the grid lines that are perpendicular (to the curve).
		THE USER CHANGES THE ORIENTATION OF THE INTERNAL ALIGNMENT REGULATOR The central Mirror regulator ensures that the change is properly reflected on both sides.

TABLE 3.5 - EXPLORATION OF A PLAN LAYOUT AND MASSING

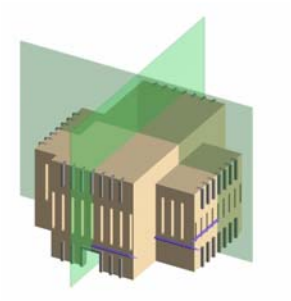
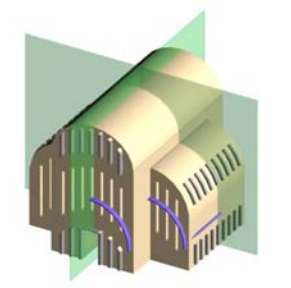
	The initial configuration consists of two Mirror regulators and three Translation regulators, which act as horizontal axes that control the windows as well as the orientation of the roof.
	THE USER ROTATES THE HORIZONTAL TRANSLATION REGULATOR The Translation regulator aligns the windows along a slanted line, and updates the roof orientation creating a gable roof.
	THE USER REDUCES THE FREQUENCY OF THE WINDOWS BY UPDATING THE NUMBER IN THE TRANSLATION REGULATOR
	THE USER TRANSFORMS THE CURVATURE OF THE TRANSLATION REGULATOR The Curve regulator realigns the windows and changes the curvature of the roof forming a vault.

TABLE 3.6 - EXPLORATION OF A MASSING CONFIGURATION AND FAÇADE

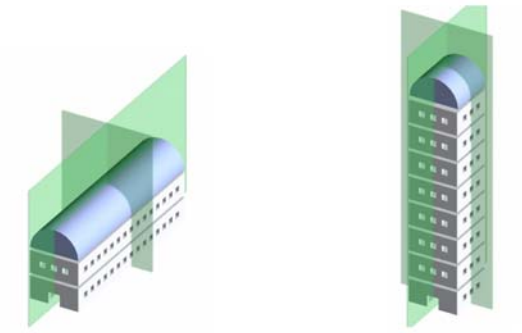
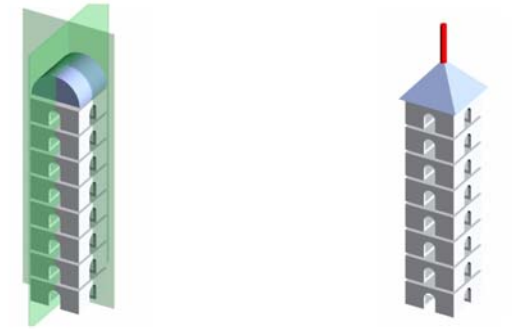
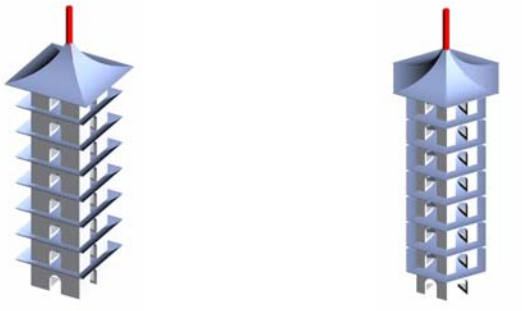
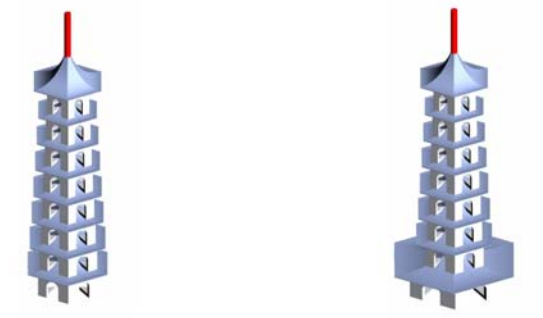
	The initial configuration consists of two Mirror regulators, two Translation regulators, an area regulator (not shown), and an adjacency regulator. THE USER DECREASES THE FOOTPRINT OF THE BUILDING. The area regulator adds floors to ensure that the area requirement be maintained after the change.
	THE USER UPDATES THE NUMBER, SIZE, AND FORM OF THE WINDOWS ON ONE FLOOR. The Translation regulator ensures that the windows of all floors are updated in the same way. THE USER REPLACES THE REFLECTION REGULATORS BY A ROTATION REGULATOR. THE USER CHANGES THE FORM OF THE ROOF.
	THE USER ADDS ONE SUB-ROOF TO ONE FLOOR The Translation regulator propagates the change to all the floors. THE USER CHANGES THE ROTATION DEGREE FROM 90° TO 60°. The plan changes from a square to a hexagon. The Translation regulator ensured that the all floors are updated.
	THE USER INTRODUCES A SCALE FACTOR TO THE TRANSLATION REGULATOR. The floors gradually decrease upon ascending. THE USER SELECTS A FLOOR AND TREATS IT AS AN EXCEPTION AND THEN APPLIES A SCALE TRANSFORMATION TO MAKE IT MORE PROMINENT. The result is a pagoda.

TABLE 3.6 - EXPLORING TRANSFORMATION ACROSS STYLES

3.3.3. NOTATION AND IMPLEMENTATION

The ICE framework has two major components: a descriptive component consisting of a formal notation, and a computational component consisting of an interactive implementation. Both use the design structure as a primary vehicle for description and transformation.

The ICE notation uses regulators for describing shapes and configurations, as a concise string, by means of its generative and relational structure. Through its symbolic representation of regulators, the ICE notation describes the generation path and transformable parameters for every configuration. For instance, in Figure 3.37, Palladio's Villa Capra is described – and can be generated discretely – by means of Mirror regulators composed successively with its top left corner as an input. The ICE notation is described in greater detail in Chapter 4.

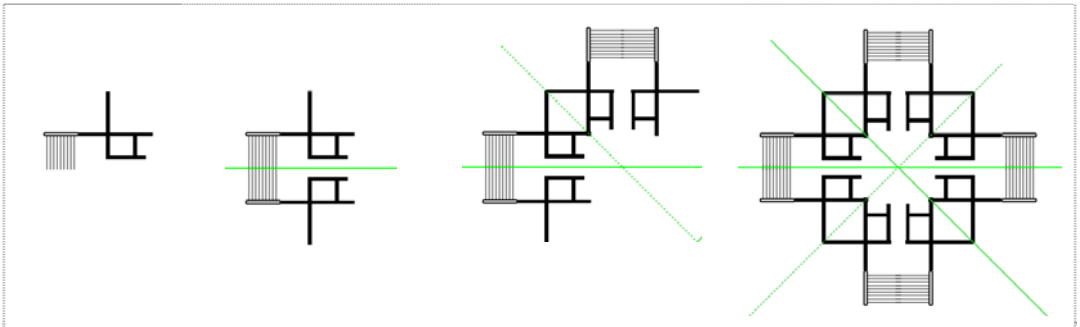
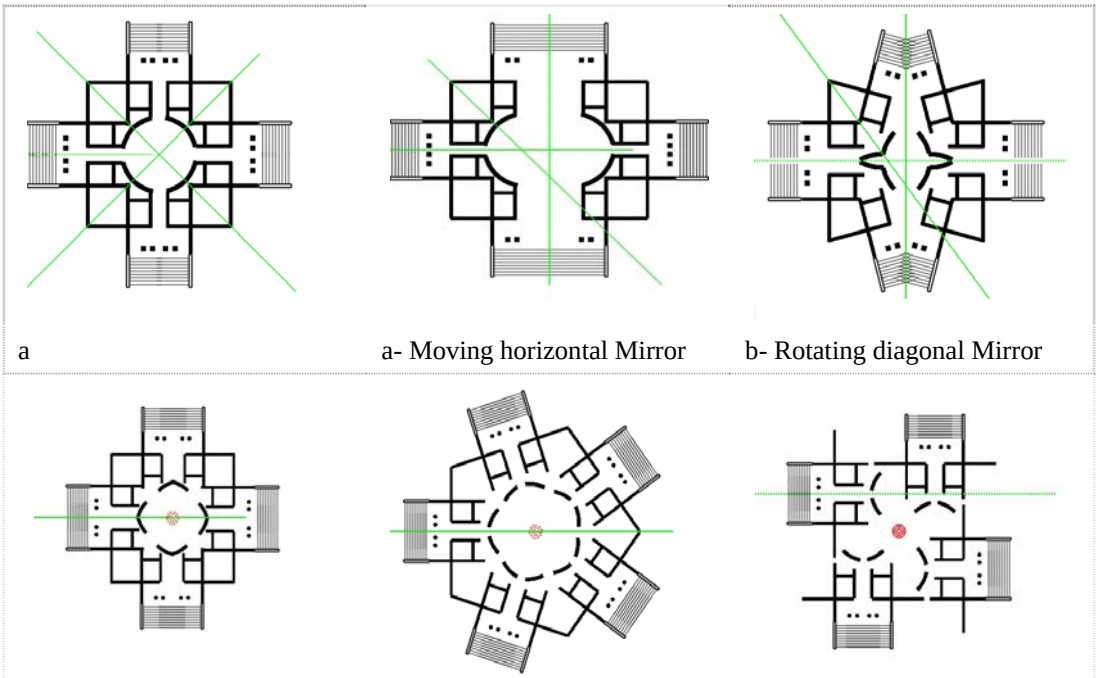


FIGURE 3.37 - PALLADIO'S VILLA CAPRA DESCRIBED USING THE ICE NOTATION

The ICE implementation uses regulators for transforming shapes and configurations through their structures. The ICE implementation provides a systematic approach to managing complex relations, while allowing users to interact with higher-level structures as opposed to lower-level details. The parameters of regulators are manipulation handles, and are used to transform configurations, slightly, as well as significantly, to accommodate various levels of exploration. Figure 3.38 shows an example of manipulating Palladio's Villa Capra by means of moving rotating, and replacing its regulators. The ICE implementation is described in greater detail in Chapter 7.



a | a- Moving horizontal Mirror | b- Rotating diagonal Mirror

c- Replace two mirrors with a rotation | d- Increasing rotation degree | e- Moving the rotation point

FIGURE 3.38 - PALLADIO'S VILLA CAPRA IS EXPLORED MANIPULATING THE GEOMETRY OF ITS GENERATIVE REGULATORS

The notational description does not correspond to the way designers would go about developing configurations with regulators. The notation provides a concise description that summarizes the relationships in the configuration by means of an ordered syntax. Designers, on the other hand, would develop elements and regulators with no particular order. They would derive initial regulators from the surrounding site and would derive additional regulators, perhaps at a later stage, from configuration elements; therefore, associating elements and regulators as the configuration is being established. Such interactions and derivations methods are illustrated in the protocols in Tables 3.1 and 3.2.

As designers are engaged in their exploration, they discover new ideas and new relations, consequently, defining new regulators and reconfiguring the structure. As structures change, novel exploration paths are formed. Exploration does not follow a predetermined path, but rather, a path that is constantly being redefined during the course of the exploration.

CHAPTER 4

THE ICE NOTATION

The ICE notation is the descriptive formalism of the ICE framework. It is a formal notation specifying complex configurations through generative and relational constructs, which are encapsulated by regulators. The ICE notation describes geometric configurations in a clear, succinct, and complete manner and supports the description of a wide range of configurations in two and three dimensions. Additionally for any given configuration, the ICE notation captures, parsimoniously, the process for its generation and as well as a set of applicable transformations that could be used for exploring the configuration.

In this chapter, I describe the syntax of the ICE notation, and focus on the various regulator types, their composition strategies, and their generation methods, from a notational perspective. I discuss the capacity of ICE to capture applicable transformations as well as the generative history for any given configuration, and I introduce the ICE transformation syntax. A gallery of shapes and patterns illustrates how the ICE notation represents shapes, patterns, and classes of compositional schemata.

4.1. THE ICE NOTATION SYNTAX

The ICE notation specifies a geometric configuration in terms of a minimal number of steps required for its generation, and the meaningful relationships for its organization.

The principal building blocks used in the ICE notation are the "point" and "regulator", which encapsulates a formula for its spatial relation. Notationally, points are indicated in lowercase, for instance, $\bar{p}$, 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 the regulator's category, for example, Δ: transformations, Φ: constraints, Ψ: hierarchies, Π: topologies, 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 represent as indices; for example $\Delta\mathbf{T}_{\mathbf{1}}$, and $\Delta\mathbf{T}_{\mathbf{2}}$ are two different instances of Translation regulators used in the same configuration.

The ICE notation can be expressed in either a short or expanded form. The former operates on a relational level, while the latter operated on a parametric level which is essential for system implementation. The short form captures the regulator and regulated objects, for instance, $\Delta\mathbf{T}(shape)$. The expanded form, additionally, includes the parameters of the regulator; these are enclosed within curly braces with vectors depicted by an overline, for example, $\Delta\mathbf{T}^{\mathbf{1}}[\ \{\bar{p}, \bar{t}, d, n\}\ (shape)\ ]$. Parameters contribute to a regulator's formula and include geometric parameters, $\bar{t}$, such as translation vectors, rotation points/lines, reflection axes, as well as generative parameters, such as translation distance, d, rotation degrees, θ, and the number of generated objects, n.

Regulators regulate points thereby creating shapes, likewise, regulate shapes to create configurations, and regulate other regulators to create complex schemata. Regulators can be generative or non-generative. Generative regulators take an input shape and create output shapes, while non-generative regulators act on the input shape. The generative

property is depicted by the presence of the "n" parameter. $\Delta\mathbf{T}^{\mathbf{1}}[\ \{\bar{p},\bar{t},d,n\}\ (shape)\ ]$ is a generative regulator , while $\Delta\mathbf{T}^{\mathbf{1}}[\ \{\bar{p},\bar{t},d\}\ (shape)\ ]$ is a non-generative one. Regulators that generate shapes from input points are applied continuously, while regulators that generate configuration from input shapes are applied discretely. The continuity factor is indicated by superscript brackets. $\Delta\mathbf{T}(\bar{s})^{<0><1><2>}$ is a discrete application generating disjoint points, while $\Delta\mathbf{T}(\bar{s})^{<0,1,2>}$ is a continuous application generating a line.

The ICE notation has a corresponding graph representation, which presents an alternative view to the ICE string and serves to visualize internal associations between elements of the ICE string namely points/shapes and regulators (Table 4.1). Graph representation is particularly significant in the description of compositions and schemata.

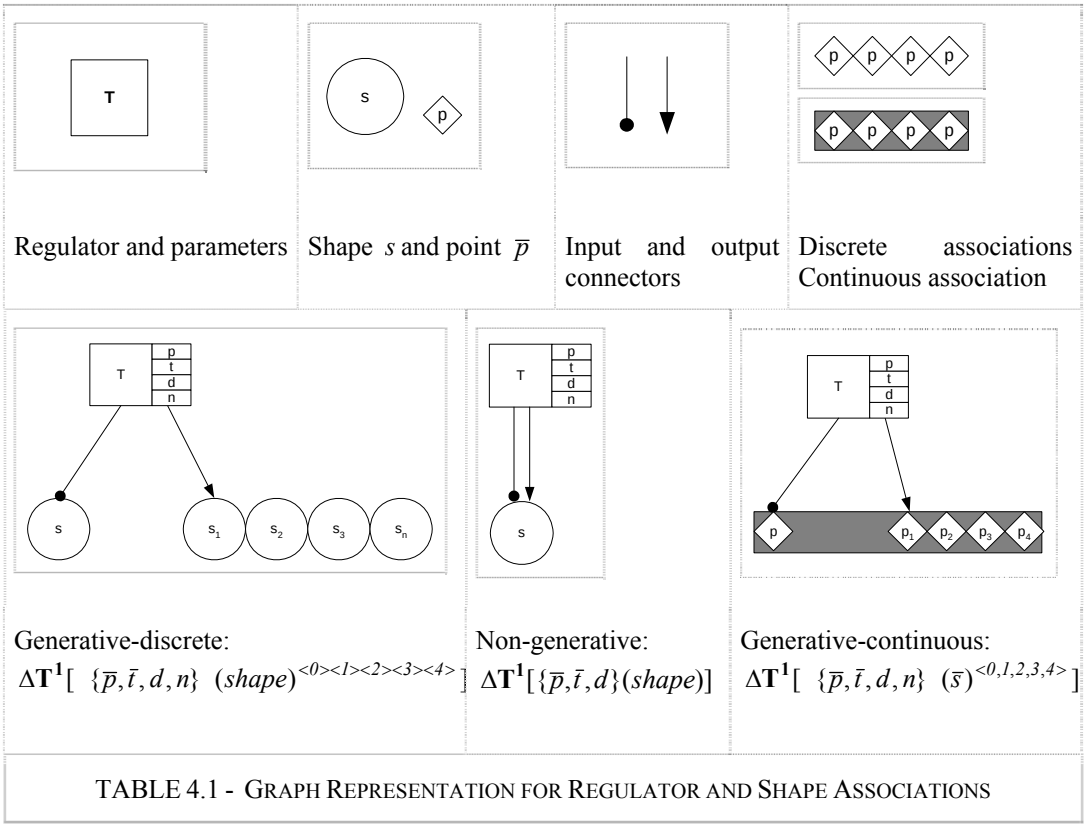


TABLE 4.1 - GRAPH REPRESENTATION FOR REGULATOR AND SHAPE ASSOCIATIONS

4.2. REGULATORS CATEGORIES AND TYPES

The notational description formalizes the concept of regulators and clarifies the various subtypes and their corresponding parameters. In this section, I describe the notation for each regulator in the ICE framework. The mathematics corresponding to the defining, transforming, and composing these regulators is presented in Appendix B.

4.2.1. TRANSFORMATION REGULATORS

Transformation regulators are the primary constructs of the ICE notation. Not only are these used as exploration tools, but as generative vehicles as well. Transformation regulators, based on isometric and affine transformations, are indicated by the Δ prefix. Transformational regulators take as input a shape or a point, and generate "n" output shapes or points. The input element is assigned index 0 and the output elements are assigned indices, $1-n$. The position of the outputs is determined by the type of transformation.

Transformation regulators operate by applying an equation to the input element to derive the output set of elements. The basic transformation regulators can be composed simultaneously, to define complex transformation effects. These regulators use the properties of their respective transformations to preserve points, lines, and planes, as a visual depiction for the regulators. Table 4.2 illustrates transformation regulators with their corresponding notations.

The ***Translation*** regulator generates n output shapes (d distance apart) along the line specified by a starting point $\bar{p}$ and a direction vector $\bar{t}$. If applied continuously, translation specifies lines and extrudes shapes. The Translation regulator is depicted by a line along the $\bar{t}$ vector.

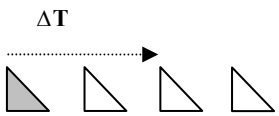
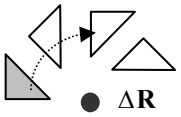
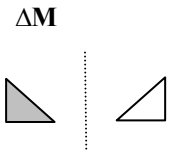
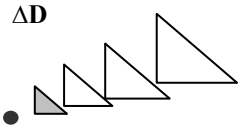
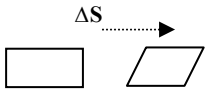
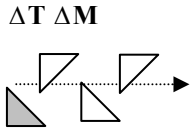
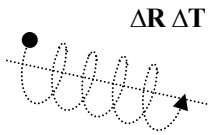
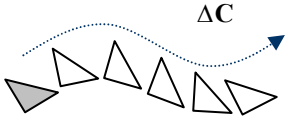
Translation	$\Delta\mathbf{T}^{\mathbf{1}}[\ \{\overline{p},\bar{t},d,n\}\ (shape)\ ]$	
Rotation	$\Delta\mathbf{R}^{\mathbf{1}}[\ \{\overline{p},\bar{t},\theta,n\}\ (shape)\ ]$ $\Delta\mathbf{R}^{\mathbf{0}}[\ \{\overline{p},\theta,n\}\ (shape)\ ]\ )\ ]$	
Mirror	$\Delta\mathbf{M}^{\mathbf{0}}[\ \{\overline{p},n\}\ (shape)\ ]$ $\Delta\mathbf{M}^{\mathbf{1}}[\ \{\overline{p},\bar{t},n\}\ (shape)\ ]$ $\Delta\mathbf{M}^{\mathbf{2}}[\ \{\overline{p},\bar{t},\bar{v},n\}\ (shape)\ ]$	
Dilation (scale)	$\Delta\mathbf{D}^{\mathbf{0}}[\ \{\overline{p},\bar{k},n\}\ (shape)\ ]$	
Shear	$\Delta\mathbf{S}\ [\ \{\bar{k},n\}\ (shape)\ ]$	
Glide	$\Delta\mathbf{T}^{\mathbf{1}}\Delta\mathbf{M}^{\mathbf{1}}[\ \{\overline{p},\bar{t},d,n\}\ (shape)\ ]$ $\Delta\mathbf{T}^{\mathbf{1}}\Delta\mathbf{M}^{\mathbf{2}}[\ \{\overline{p},\bar{t},\bar{v},d,e,n\}\ (shape)\ ]$	
Screw	$\Delta\mathbf{R}^{\mathbf{1}}\Delta\mathbf{T}^{\mathbf{1}}[\ \{\overline{p},\bar{t},d,\theta,n\}\ (shape)\ ]$	
Curve	$\Delta\mathbf{C}^{\mathbf{e}}[\ \{\overline{p},\bar{t},\alpha,n\}\ (shape)\ ]$ $\Delta\mathbf{C}^{\mathbf{h}}[\ \{\overline{p},\bar{t},\alpha,n\}\ (shape)\ ]$	

TABLE 4.2 – REGULATORS BASED ON GEOMETRIC TRANSFORMATIONS

The ***Rotation*** regulator generates n output shapes, each rotated θ degrees apart. In 3D space, rotation is about the axis specified by a starting point $\bar{p}$ and a direction vector $\bar{t}$, and in the 2D plane, the rotation is about a point $\bar{p}$. If applied continuously, the rotation specifies circles and surfaces of revolutions. The regulator is depicted by the rotation point or axis.

The ICE notation supports three subtypes of the ***Mirror*** regulator: an inversion about a point; a reflection about a line; and a reflection in the plane. These are depicted by the point, line and plane, respectively. In 3D space, reflection about a line is not orientation reversing; it is therefore, equivalent to a rotation. However, in the 2D plane, it is an actual, orientation reversing, reflection. Although reflection produces a single image, the reflection regulator allows for n output elements to accommodate the composition of mirror with other regulators.

The ***Dilation*** regulator scales successive output shapes by a factor $\bar{k}$, represented by a vector. Dilation, which is either isotropic (equal in the xyz-directions) or anisotropic, is depicted by a point $\bar{p}$ representing the origin of the scaling.

The ***Shear*** regulator shears the successive output shapes by a factor $\bar{k}$ and is depicted by an arrow showing the direction of the shear.

The ***Glide*** regulator is a composition of mirror and translation. It generates successive elements reflected about a plane or line and translated along the same line. It is achieved by means of simultaneous composition of regulators (Section 4.4). Sub-types include: a glide in 2D, along a glide line, and in 3D, about a glide plane.

The ***Screw*** rotation is a composition of a rotation and a translation. It generates successive elements rotated about the axis defined by a point $\bar{p}$ and a vector $\bar{t}$ together with a translation along this same axis. It is achieved by means of the simultaneous composition of regulators (Section 4.4).

The ***Curve*** regulator organizes the output elements along a curve in space. Ideally, curves should be described by their equations, but for ease, we identify them by their subtypes, for example, elliptical $\Delta\mathbf{C}^{\mathbf{e}}$, hyperbolic $\Delta\mathbf{C}^{\mathbf{h}}$, or trigonometric curves $\Delta\mathbf{C}^{\mathbf{s}}$. A Curve

regulator can also be achieved through simultaneous composition.

4.2.2. VARIATION REGULATORS

Variational regulators, symbolized by Ξ, are composed with generative regulators to create a variation in the output shapes. This is achieved by controlling shape attributes or regulator parameters. Variation regulators are depicted by a point indicating their presence. Table 4.3 illustrates the variation regulators and their corresponding notation.

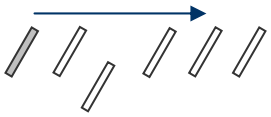
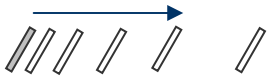
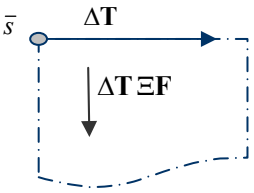
Exception	$\Xi\mathbf{E}$ [$\{a,v\}$ $(shape_0 - shape_n)$] $\Delta\mathbf{T}\Xi\mathbf{E}[\{\bar{p},\bar{t},d,n,a,v\}(\bar{s})]$	$\Delta\mathbf{T}\,\Xi\mathbf{E}$ 
Rhythm/Gradation	$\Xi\mathbf{G}$ [$\{a,\boldsymbol{f},c\}$ $(shape_0 - shape_n)$] $\Delta\mathbf{T}\Xi\mathbf{G}[\{\bar{p},\bar{t},d,n,a,f,c\}(\bar{s})]$	$\Delta\mathbf{T}\,\Xi\mathbf{G}$  $\Delta\mathbf{T}\,\Xi\mathbf{G}$ 
Differential	$\Xi\mathbf{F}$ [$\{a,\boldsymbol{f},c\}$ $(shape_0 - shape_n)$] $\Delta\mathbf{T}\Xi\mathbf{F}[\{\bar{p},\bar{t},d,n,a,f,c\}(\bar{s}_1 - \bar{s}_n)]$	$\bar{s}$ $\Delta\mathbf{T}$  $\Delta\mathbf{T}\,\Xi\mathbf{F}$

TABLE 4.3 - REGULATORS BASED ON VARIATION FORMULAE

The ***Exception*** regulator sets a shape to be an exception to the output set by overriding an attribute a, (for instance, position) with a value v.

The ***Rhythm*** regulator creates a rhythm/gradation effect within the output shapes, by applying a formula $\boldsymbol{f}$ and coefficient c to an attribute a of output elements (for instance, color), or to an attribute a of the generative regulator (for instance, the

translation distance) as it is applied to the output shapes. The formula f defines the type of rhythm, whether it is alternating or gradual, or follows the undulations of a curve.

The ***Differential*** regulator creates a variation in the output, by sweeping-copying the elements of input set differently. It applies a 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 when there are many input shapes.

4.2.3. CONSTRAINT REGULATORS

Constraint regulators, symbolized by Φ, are not generative; they restrict shapes or define relations between input shapes or points. Constraints can be defined independently or in composition with generative regulators. Constraint regulators are based on an evaluation function that determines whether or not the input element is within the constraints. Table 4.4 illustrates the constraint regulators and their corresponding notation.

The ***Equivalence*** regulator assigns and maintains a value v to an attribute a (for instance color) of a shape/s.

The ***Alignment*** regulator restricts the position or motion of elements with respect to itself. There are several subtypes of alignments: $\Phi\mathbf{A}^{\mathbf{0}}$ defined by a point $\bar{p}$, $\Phi\mathbf{A}^{\mathbf{1}}$ defined by $\bar{p}$ and a vector $\bar{t}$, and $\Phi\mathbf{A}^{\mathbf{2}}$ defined by $\bar{p}$ and vectors, $\bar{t}$ and $\bar{v}$. These regulators are depicted by a point, line, and plane respectively. The Alignment regulator can also restrict elements to a circle or curve. This regulator is depicted by $\Phi\mathbf{A}^{\mathbf{c}}$ and its parameters are defined by curve type and include the point $\bar{p}$, the vectors $\bar{t}$, and the radius r.

There are three subtypes for the ***Size*** regulator: $\Phi\mathbf{V}^{1}$ restricts length, $\Phi\mathbf{V}^{2}$ restricts area, and $\Phi\mathbf{V}^{3}$ restricts volume. The parameters are minimum/maximum value and an incremental module. These regulators are depicted by one, two, or three dimension lines, respectively.

The ***Angle*** regulator sets the angle within a shape or between two shapes. A variant $\Phi\mathbf{L}^{\mathbf{p}}$ sets shapes as being parallel. The Angle regulator is depicted by an arc joining the two

shapes or a line in the case of $\Phi\mathbf{L}^{\mathbf{p}}$.

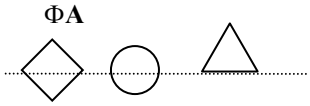
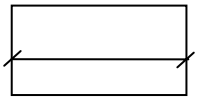
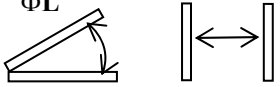
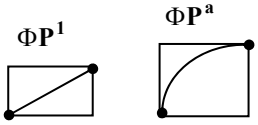
Equivalence	$\Phi\mathbf{Q}$ [$\{a,v\}$ $(shape_0 - shape_k)$]	$\Phi\mathbf{Q}$ 
Alignment	$\Phi\mathbf{A}^0$[$\{\overline{p}\}$ $(shape_0 - shape_k)$] $\Phi\mathbf{A}^1$[$\{\overline{p},\overline{t}\}$ $(shape_0 - shape_k)$] $\Phi\mathbf{A}^2$[$\{\overline{p},\overline{t},\overline{v}\}$ $(shape_0 - shape_k)$] $\Phi\mathbf{A}^C$[$\{\overline{p},\overline{t},r\}$ $(shape_0 - shape_k)$]	$\Phi\mathbf{A}$ 
Size	$\Phi\mathbf{V}$ [$\{min,max,mod\}$ $(shape)$]	$\Phi\mathbf{V}$ 
Angle	$\Phi\mathbf{L}$ [$\{min,max,mod\}$ $(shape_l - shape_k)$]	$\Phi\mathbf{L}$ $\Phi\mathbf{L}^{\mathbf{p}}$ 
Proportion	$\Phi\mathbf{P}^1$[$\{\overline{p},\overline{t},d\}$ $(shape)$]	$\Phi\mathbf{P}^{\mathbf{1}}$ $\Phi\mathbf{P}^{\mathbf{a}}$ 

TABLE 4.4 - REGULATORS BASED ON GEOMETRIC CONSTRAINTS

The ***Angle*** regulator sets the angle within a shape or between two shapes. A variant $\Phi\mathbf{L}^{\mathbf{p}}$ sets shapes as being parallel. The Angle regulator is depicted by an arc joining the two shapes or a line in the case of $\Phi\mathbf{L}^{\mathbf{p}}$.

The ***Proportion*** regulator controls the aspect ratio of a shape through a diagonal line, which also depicts the regulator. A variation of this regulator, $\Phi\mathbf{P}^{\mathbf{a}}$, controls the proportion through an arc.

4.2.4. TOPOLOGICAL REGULATORS

Topological regulators, symbolized by Π, are mostly binary and non-generative. Table 4.5 illustrates the topological regulators and their corresponding notation.

The ***Distance*** regulator, $\Pi\mathbf{J}^{+}$, defines the proximity between shapes, irrelevant of their geometry. Variations of this regulator are adjacency $\Pi\mathbf{J}^{\mathbf{0}}$, defined by a zero distance and overlap $\Pi\mathbf{J}^{-}$, defined by a negative distance. This regulator is depicted as a dimension line.

The ***Boundary*** regulator defines a legal region for a shape, with an offset o, in other words, it restricts a shape to be inside another. It inputs the boundary shape as well as the bounded shapes and is depicted by a thicker boundary shape.

The ***Connection*** regulator determines whether two shapes are connected and ensures that these remain connected upon manipulation.

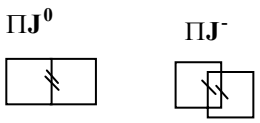
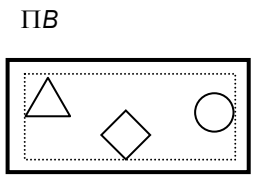
Distance	$\Pi\mathbf{J}$ **[** $\{min, max, mod\}$ $(shape_1, shape_2)$ **]**	
Boundary	$\Pi\mathbf{B}$ [$\{o\}$ $(shape_{bound}, shape_1 - shape_k)$]	
Connection	$\Pi\mathbf{C}$ **[** {} $(shape_1, shape_2)$ **]**	

TABLE 4.5 - REGULATORS BASED ON TOPOLOGICAL RELATIONS

4.2.5. HIERARCHICAL REGULATORS

Hierarchical regulators, symbolized by Ψ, define hierarchies of shapes; these can be defined independently, or in composition with other regulators. Table 4.6 illustrates the

hierarchical regulators and their corresponding notation.

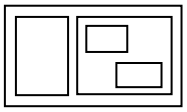
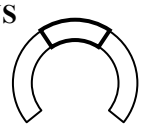
Containment	$\Psi\mathbf{H}$ [{} (*container, constituent*$_0$ – *constituent*$_n$)]	$\Psi\mathbf{H}$ 
Subshape	$\Psi\mathbf{S}$ [{} (sup*ershape, subshape*$_0$ – *subshape*$_n$)]	$\Psi\mathbf{S}$ 

TABLE 4.6 - REGULATORS BASED ON HIERARCHICAL RELATIONS

The ***Containment*** regulator creates a container-constituent relationship, irrelevant of geometry. Typically, containment inputs the container and constituents, however, it can input the container and generate the constituents, or vice versa. The containment regulator can be composed with the Subdivision and the Boundary regulators to introduce geometrical and topological dependencies in the hierarchy.

The ***Subshape*** regulator creates a geometric dependency between shapes (or more precisely between their generative regulators).

4.2.6. OPERATION REGULATORS

Operational regulators, symbolized by Ω, are generative regulators that define complex shapes from simpler ones by means of discrete transformations. Table 4.7 illustrates the operation regulators and their corresponding notation.

The **Subdivision** regulator, $\Omega\mathbf{Z}$ inputs a shape, subdivides it n times and allocates a spacing s between the subdivisions. The subdivisions produced are normal to the shape's direction (i.e., to the generative regulator's direction). A variant of subdivision, the **Cutting** regulator, $\Omega\mathbf{Z}^{\mathbf{P}}$, subdivides a shape 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.

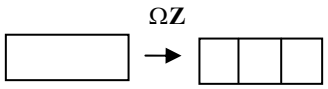
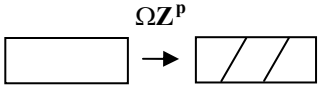
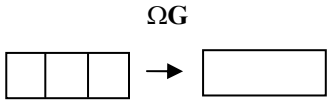
Subdivision	**ΩZ** [$\{s,n\}$ (*shape*)]	
Cutting	**ΩZ^{P}**[$\{s,n\}$ (*shape,plane*)]	
Merging	**ΩG** [{} ($shape_A$, $shape_B$)]	
Boolean Operations		
Union:	**ΩU** [{} ($shape_0 - shape_k$)]	
Intersection:	**ΩI** [{} ($shape_0 - shape_k$)]	
Difference:	**ΩD** [{} ($shape_0 - shape_k$)]	
Symmetric Difference:	**ΩM** [{} ($shape_0 - shape_k$)]	

TABLE 4.7 - REGULATORS BASED ON OPERATIONS

4.3. REGULATOR GENERATION METHODS

In order to represent the various types of shapes and patterns observed in architectural compositions, the ICE framework allows for several methods of generation. These include continuous, discrete, combination, subset, and pattern generation. This feature, which is only applicable to generative regulator, is indicated by superscripts, for instance, $\Delta\mathbf{T}(\mathrm{p})^{<0-3><6-9>}$. Brackets group continuous parts together, and the dash indicates that all shapes/points within the range are generated. Table 4.8 illustrates the generation methods supported in ICE.

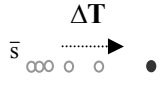
Discrete generation	$\Delta\mathbf{T}(\bar{s})^{<0>-<2>}$ $\Delta\mathbf{T}(\bar{s})^{<0><1><2>}$	
Continuous generation	$\Delta\mathbf{T}(\bar{s})^{<0-2>}$ $\Delta\mathbf{T}(\bar{s})^{<0,1,2>}$	
Combined generation	$\Delta\mathbf{T}(\bar{s})^{<0-3><4><5-6>}$ $\Delta\mathbf{T}(\bar{s})^{<0,1,2,3><4><5,6>}$	
Subset generation	$\Delta\mathbf{T}(\bar{s})^{<0,1,2><5,6>}$ $\Delta\mathbf{T}(\bar{s})^{<0-2><5-6>}$	
Pattern generation	$\Delta\mathbf{T}(\bar{s})^{\therefore m<ii>\phi}$	
Non-generative regulators	$\Delta\mathbf{T}^{\mathbf{1}}(\bar{s})^{<i>}$ $\Delta\mathbf{T}^{\mathbf{1}}[\ \{\bar{p},\bar{t},d\}\ (\bar{s})^{<i>}\ ]$	
Motion regulators	$\Delta\mathbf{T}^{\mathbf{1}}\longrightarrow(\bar{s})$ $\Delta\mathbf{T}^{\mathbf{1}}[\{\bar{p},\bar{t},0\longrightarrow d\}(\bar{s})\ ]$ $\phi\mathbf{L}^{\mathbf{1}}[\{0\longrightarrow\theta\}(\bar{s})]$	

TABLE 4.8 - GENERATION METHODS

The ***discrete generation*** method generates individual separate output elements, none connected. All previous examples were generated discretely.

The ***continuous generation*** method generates output elements that are connected and the loci of points in-between the output-elements are also generated. Continuous generation is used for creating shapes from connected vertices. The shape examples in Section 4.7 are generated continuously.

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

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

Pattern generation is intended to describe repetitive patterns, for instance a dashed line, in a concise manner. The symbol $\therefore$ indicates the start of the pattern, i denotes a generated index, m indicates the number of times the cycle is repeated, ϕ indicates an absent index, and the brackets indicate continuity.

Transformation regulators can be ***non-generative***, i.e., these transform the input shape and are characterized by the absence of the n parameter, and by the presence of only one index for the new position in the superscript bracket.

Transformation regulators can also be used to describe the ***motion*** of the input shape. This is shown by the superscript arrow, which indicates the shape moving from position 0 to position n.

4.4. REGULATOR COMPOSITION

A fundamental functionality of the ICE framework is the various ways of composing regulators in order to represent the diverse types of structures observed in architectural configurations. Composition of regulators is the primary method for connecting the "regulator building blocks" of the ICE notation. A variety of complex configurations can be specified through the composition of simple regulator units. Table 4.9 illustrates the composition methods in supported in ICE and Table 4.10 shows its corresponding graph representation. Composition strategies can be combines to define intricate schemata (see Section 4.9).

Simultaneous composition of regulators allows multiple regulators to be applied to the same set of element, i.e. multiple formulae to act simultaneously. This method allows complex regulators to be defined by composing simple ones, therefore, significantly extending the repertoire of regulators. There are no limits to the number, or type, of regulators for the composition. Glide and screw rotation, for instance, are defined by means of simultaneous composition. Notationally, the composed regulator symbols are placed in juxtaposition and the parameters for the composite are the union for the individual regulator parameters, with duplicates differentiated by subscripts.

In the ***successive form*** of composition, a regulator is applied to the output shapes of another regulator, forming a nested relationship. There are no limits to the number of regulators in the succession. Notationally, successive compositions correspond to nested parenthesized strings in which inner regulators are applied before outer regulators.

In the ***partial composition*** method, a regulator is applied to a subset of the previously generated output. This allows complex and irregular shapes/patterns to be defined. Notationally, this is indicated by a subscripted string comprising the # symbol followed by the indices of the output shape.

Sharing allows a regulator to regulate multiple input shapes, and a shape to be regulated by multiple regulators. The former allows the reuse of regulators for multiple shapes and the later allows for multiple constraints to act on a single shape. Such a situation is likely to cause conflicts. There are no limits to the number of shapes shared by a regulator or to the number of regulators shared by a shape.

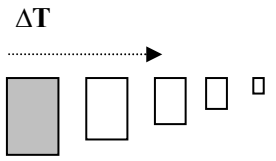
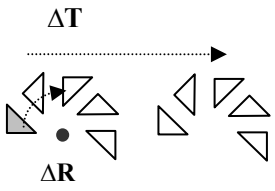
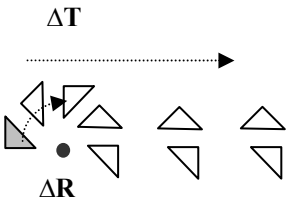
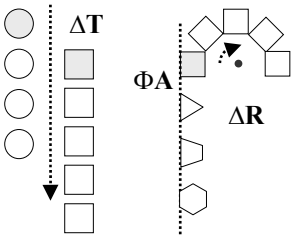
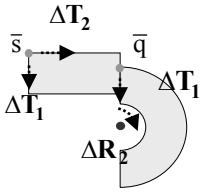
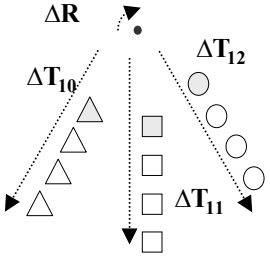
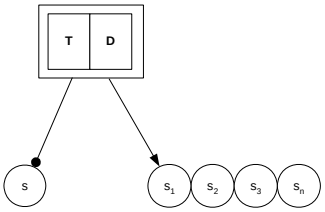
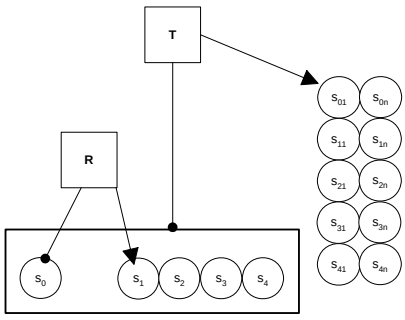
Simultaneous Composition	$\Delta\mathbf{T}^{1}\Delta\mathbf{D}^{0}[\ \{\overline{p}_T,\overline{p}_D,\overline{t},\overline{k},d,n\}\ (shape)\ ]$ $\Delta\mathbf{M}^{1}\Phi\mathbf{A}^{1}[\ \{\overline{p},\bar{t},d,n\}\ (shape)\ ]$	
Successive Composition	$\Delta\mathbf{T}^{1}[\ \{\overline{p},\bar{t},d,n\}\ (\Delta\mathbf{R}^{1}[\ \{\overline{p},\bar{t},\alpha,n\}\ (shape)]\ \ )]$	
Partial Composition	$\Delta\mathbf{T}^{1}[\ \{\overline{p},\bar{t},d,n\}\ (\Delta\mathbf{R}^{1}[\ \{\overline{p},\bar{t},\alpha,n\}\ (shape)]_{\#3,4}\ )]$	
Sharing	$\Delta\mathbf{T_2}[\{\overline{p},\bar{t},d,n\}(shape_A,shape_B)^{<0>-<4>}]$ $\Phi\mathbf{A}[\{\overline{p},\bar{t},d\}(shape_A,shape_B,shape_C,shape_D)]\wedge$ $\Delta\mathbf{R}[\{\overline{p},\bar{t},d,n\}(shape_A)^{<0>-<4>}]$	
Aggregation	$\Delta\mathbf{T_2}[\{\overline{p},\bar{t},d,n\}(\Delta\mathbf{T_1}[\{\overline{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]\wedge$ $\Delta\mathbf{R_2}[\{\overline{p},\bar{t},d,n\}(\Delta\mathbf{T_1}[\{\overline{p},\bar{t},d,n\}\ (\bar{q})^{<0-1>}])^{<0-1>}]$	
Multiple Control	$\Delta\mathbf{R}[\ \{\overline{p},\bar{t},\theta,n\}\ (\Delta\mathbf{T_1})^{<0><2>}]\ \wedge$ $\Delta\mathbf{T_{1i}}[\ \{\overline{p},\bar{t},d,n\}\ (shape_i)^{<0><3>}]$	

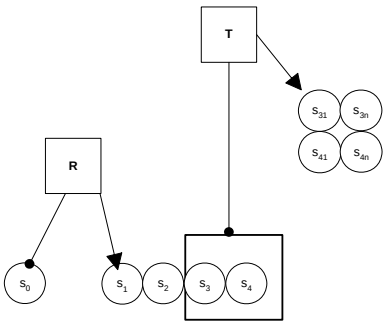
TABLE 4.9 - COMPOSITION METHODS



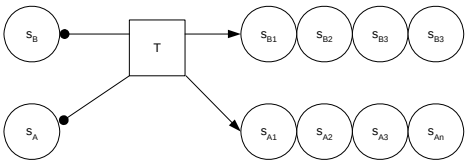
Simultaneous Composition



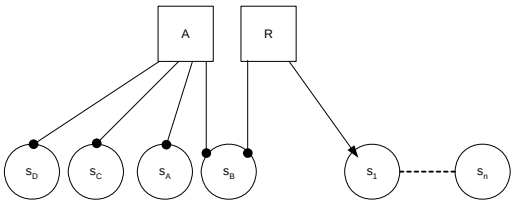
Successive Composition



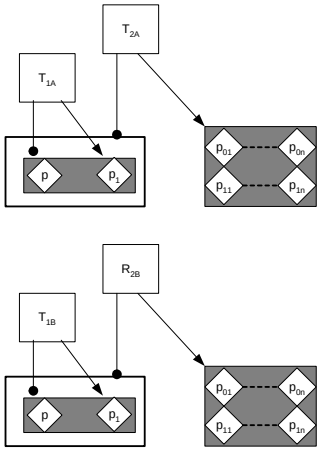
Partial Composition



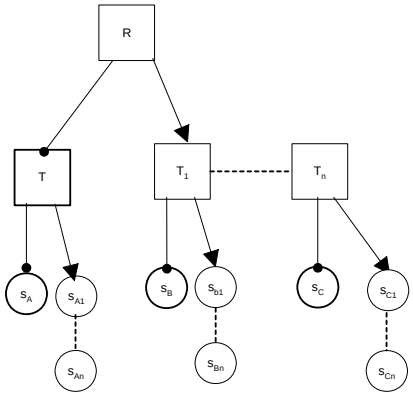
Sharing: Multiple Shapes



Sharing: Regulators Shapes



Aggregation



Multiple Control

TABLE 4.10 - GRAPH REPRESENTATION OF COMPOSITION METHODS

In the ***aggregation*** method, two independent strings are joined together to describe a complex shape. Notationally, this is indicated by the conjunction $\wedge$.

The ***multiple control*** composition method allows regulators to be regulated, therefore, regulators can be generated just like shapes and points and regulators can be constrained or related topologically or hierarchically. This method allows relations to be regulated as well as shapes, thus describing complex behaviors. When regulators control shapes, these define only one level of control for the configuration. Alternatively, when regulators control other regulators, these augment the complexity of the configuration by defining multiple levels of control. Although there are no limits on the number of levels for the regulation control hierarchy, the complexity of the configuration is directly proportional to the number of the control levels. Regulators are always regulated discretely. Notationally, generated regulators have an additional subscript to indicate its position in the generation. The multiple control mechanism allows the visionary scenario of functional regulators that control geometric regulators, which in turn regulate shapes.

4.5. ICE CONVENTIONS

For the purpose of consistency and simplification, the following conventions are used in the ICE notation throughout this document.

4.5.1. SHAPE ENCAPSULATION

In order to simplify the description of complex configurations, the ICE notation supports the notion of shape encapsulation, regulator encapsulation, parameter encapsulation and schemata (Section 4.9) encapsulation. This enables complex definitions to be captured in a simpler nomenclature, and reused within other shape definitions. For instance, in the shape encapsulation shown in Table 4.11, the *circle* object can then be used as an input for another string. Such encapsulation hierarchies can have infinite depths.

Shape Encapsulation	$circle = \Delta\mathbf{R_2}[\ \{\bar{p},\bar{t},\theta,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-n>}\ ])^{<0-n>}\ ]$ $pipe = \Delta\mathbf{T_3}[\ \{\bar{p},\bar{t},d,n\}\ (circle)^{<0-n>}\ ]$
Regulator Encapsulation	$\Delta\mathbf{G}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-n>}\ ] = \Delta\mathbf{T}\Delta\mathbf{M}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-n>}\ ]$ $\Delta\mathbf{G}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-n>}\ ] = \Delta\mathbf{T}\Delta\mathbf{M}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-n>}\ ]$
Parameter Encapsulation	$len = \{d \times n\}$ $\Delta\mathbf{T}[\ \{\bar{p},\bar{t},len\}\ (\bar{s})^{<0-n>}\ ] = \Delta\mathbf{T}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-n>}\ ]$
Schemata Encapsulation	$hi = \{d \times n\},\ \ len = \{\theta \times n\}$ $\Delta\mathbf{SPIRAL}[\ \{\bar{p},hi,len\}\ (\bar{s})^{<0-n>}\ ] = \Delta\mathbf{T}\Delta\mathbf{R}\Delta\mathbf{D}[\ \{\bar{p},\bar{t},d,\theta,\bar{k},n\}\ (\bar{s})^{<0-n>}\ ]$

TABLE 4.11 - ENCAPSULATIONS

4.5.2. INDICES AND SHAPE DIMENSION

In the context of ICE, the dimension of a shape is determined by the number of regulators defining it. Therefore, one continuous regulator defines a linear 1D shape; two continuous regulators define a planar 2D shape, and three continuous regulators define a volumetric 3D shape. As a preferred convention, regulator indices are used to indicate their position with respect to the dimension of a shape.

Special cases exist where the addition of a regulator does not imply a change in dimension. These include coplanar discrete applications such as in the case of the rhombus, or partial successions such as in the case of the polyline (Table 4.12). Therefore, in most configurations, numerical indices are used for regulators that augment dimension, while alphabetical indices are used for regulators that operate within the same dimension. However, some configurations determine special conventions for shape and regulator indices. In such cases, letters denoting vertical or horizontal may be used instead of numbers, and numbers denoting order of applications can be used instead of lowercase letters. Furthermore, complete words may be used instead of letters.

Indices for distinct shape are depicted as uppercase letters or even words, such as $shape_A$ or $shape_B$. Shapes generated by a common regulator are denoted by the indices 1 to n, $shape_0$ or $shape_n$. Similarly, regulators generated by a common regulator such indices, concatenated to their own indices: $\Delta\mathbf{T_{10}}$ to $\Delta\mathbf{T_{1n}}$. Regulator indices, denoting different shapes, are inherited from the shape index. These are depicted as uppercase letters and positioned before the dimension index. $shape_A\langle\Delta\mathbf{T_1}\rangle = \Delta\mathbf{T_{A1}}$.

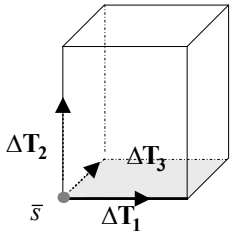
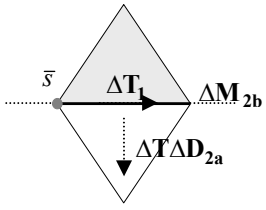
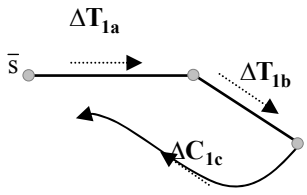
$cube = \Delta\mathbf{T_3}[\ \{\bar{p},\bar{t},d,n\}\ ($ $\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ ($ $\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}]$ $)^{<0-1>}]$ $)^{<0-1>}]$	
$rhombus = \Delta\mathbf{M_{2b}}[\ \{\bar{p},\bar{t},n\}\ ($ $\Delta\mathbf{TD_{2a}}[\ \{\bar{p},\bar{t},d,\bar{k},n\}\ ($ $\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}]$ $)^{<0-1>}]$ $)^{<0><1>}]$	
$polyline = \Delta\mathbf{T_{1c}}[\ \{\bar{p},\bar{t},d,n\}\ ($ $\Delta\mathbf{T_{1b}}[\ \{\bar{p},\bar{t},d,n\}\ ($ $\Delta\mathbf{T_{1a}}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}_{\#1}]$ $)^{<0-1>}_{\#1}]$ $)^{<0-1>}]$	

TABLE 4.12 – SHAPE DIMENSION

4.5.3. SHAPE ACCESS

The ICE notation supports the concept of accessing key elements inside shapes by using "brackets." This enables the application of regulators to certain part of a shape. For instance, an alignment can be applied to the midpoint of a shape, or a value constraint can be applied to a specific parameter of a regulator.

- To access a regulator of a shape: $shape_A\langle \Delta\mathbf{T_1} \rangle$
- To access many regulators of a shape: $shape_A\langle \Delta\mathbf{T_1}, \Delta\mathbf{T_3} \rangle$
- To access parameter n of the regulator $\Delta\mathbf{R_1}$: $\Delta\mathbf{R_1}\{n\}$ or $n_{\mathbf{R_1}}$
- To access parameter n of the regulator $\Delta\mathbf{R_1}$ within a shape: $shape_A\langle n_{\mathbf{R_1}} \rangle$
- Key-points ($\bar{e}$: endpoint, $\bar{m}$: midpoint, $\bar{s}$: start-point)
- To access key-points (such as the midpoint): $shape_A\langle m_1 \rangle$

4.5.4. SHAPE RESOLUTION

Every regulator has a resolution that is determined by the factor $(\bar{t}, \theta, \bar{k})$ multiplied by the parameter, n. This can be increased or decreased by manipulating either $(\bar{t}, \theta, \bar{k})$ or n. The same line can be generated by means of two points, which are six units of distance apart, $line = \Delta\mathbf{T_1}[\ \{\bar{p}, \bar{t}, 6, 1\}\ (\bar{s})^{<0-1>}\ ]$, or six collinear points, which are one unit distance apart, $line = \Delta\mathbf{T_1}[\ \{\bar{p}, \bar{t}, 1, 5\}\ (\bar{s})^{<0-5>}\ ]$. Typically, the former is the convention for continuous shape generation, but in the case of certain operations, the latter is used to provide access to intermediate key points. To increase the resolution, multiply the factor by $1/\lambda$ and multiply n by λ.

4.5.5. DISTRIBUTION AND IDENTITY

The ICE notation supports the following distributive property, $\Delta\mathbf{T}\ (s\mathit{hape}_1, \mathit{shape}_2) = \Delta\mathbf{T}\ (s\mathit{hape}_1) \wedge \Delta\mathbf{T}\ (\mathit{shape}_2)$, where the conjunction $\wedge$ joins the two related notation strings.

The ICE notation supports the following cancellation property. Generating only the zeroth element, is equivalent to the identity operation in ICE and can cancel the regulator in question from a complex string.

$$\Delta\mathbf{T_2}[\ \{\overline{p}, \overline{t}, d, n\}^{<0>}\ (\overline{s})\ ] = \overline{s}$$

$$\Delta\mathbf{T_2}[\ \{\overline{p}, \overline{t}, d, n\}\ (\Delta\mathbf{T_1}[\ \{\overline{p}, \overline{t}, d, n\}\ (\overline{s})^{<0>}\ ])^{<0-n>}\ ] = \Delta\mathbf{T_2}[\ \{\overline{p}, \overline{t}, d, n\}^{<0-n>}\ (\overline{s})\ ]$$

4.6. GENERATION AND TRANSFORMATION IN ICE

In addition to capturing the structure of any configuration through its regulators, the ICE notation captures two significant exploratory components: a step by step generative sequence based on structure, and a set of transformations applicable to the structure. The generation sequence is significant for analyzing a configuration and reproducing it, while the applicable transformations are significant for designating manipulation handles for exploring it.

4.6.1. CAPTURING GENERATION

The ICE notation is a vehicle for summarizing generative history, which is important for process analysis and data encoding. If a notation string is dissected and analyzed regulator by regulator, the result is a replay of the generation method. The graphic configuration in Table 4.13 is the logo designed specifically for the generative CAD systems symposium using an early version of the ICE implementation. It consists of two Rotation regulators applied successively, with the first one being composed simultaneously with Dilation and Gradation.

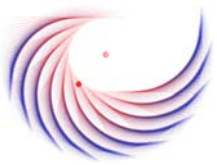
	The initial shape $shape_A$
	Apply a generative Rotation regulator. $\Delta\mathbf{R}_1^0[\{\overline{p},\theta=3,n=26,(shape_A)^{<1>-<26>}]$
	Compose the rotation with dilation. $\Delta\mathbf{R}_1^0\Delta\mathbf{D}[\{\overline{p},\theta=3,n=26,k_x=.95\}(shape_A)^{<1>-<26>}]$
	Compose the rotation with color gradation. $\Delta\mathbf{R}_1^0\Delta\mathbf{D}\Xi\mathbf{G}[\{\overline{p},\theta=3,n=26,k_x=.95,r=-.058,b=+.028\}(shape_A)^{<1>-<26>}]$
	Applying another rotation successively. $\Delta\mathbf{R}_2^0[\{\overline{p},\theta=20,n=9\}($ $\Delta\mathbf{R}_1^0\Delta\mathbf{D}\Xi\mathbf{G}[\{\overline{p},\theta=3,n=26,k_x=.95,r=-.058,b=+.028\}(\text{shape}_A)^{<1>-<26>}]$ $)^{<1>-<9>}]$

TABLE 4.13 GENERATION SEQUENCE OF THE GCAD'04 LOGO

Table 4.13 also illustrates the generation method that is based on the breakdown of the notation string considering one regulator application at a time.

4.6.2. CAPTURING TRANSFORMATION

The ICE notation is a vehicle for exploration. Its representation allows the reconfiguration of a string, therefore, the reconfiguration of a design. The transformations captured in the ice strings are categorized as follows:

- Transforming the regulated element
- Transforming the generation method
- Transforming the regulator's geometry and parameters
- Transforming the regulator composition

Transforming the regulated shape/point modifies the configuration while maintaining its geometric structure. Transformations in this category include moving the point, as well as moving-rotating-replacing the shape.

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

Transforming the parameters of the regulators 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 composition redefines the notation string and completely alters the configuration's structure. Such transformations include adding, composing, inserting, deleting, replacing or reordering regulators in a sequence.

Table 4.14 illustrates how such transformations are applied to explore the GCAD logo. The notational manipulation and the corresponding effects on the logo are illustrated.

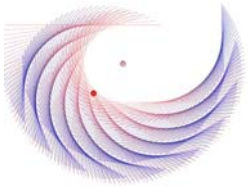
Each step in Table 4.14 is the result of applying a single transformation to the initial GCAD logo in Table 4.13 Notice how a simple notation changes represent significant changes in the geometry. Such transformation enables one to start with a configuration, then to modify it with a few steps, until a completely different configuration is achieved. When a transformation is applied to a notation string, the string is reconfigured and consequently, the set of applicable transformations, and as well as the generation sequence, are completely redefined.

The notation string captures all applicable transformations on each of these categories. The various symbols, parameters, and indices of the notation represent manipulation handles for the ICE system. The transformation syntax listed in Table 4.15 describes these transformations using two complementary ways: (i) the transformation name (in UPPPERCASE) and (ii) a left and right hand notation focusing on the parameters that are changed.

The transformation syntax is used to document exploratory actions and transitions between various configurations. It presents an alternative way of describing configurations by means of steps.

It is important to distinguish between the generative sequence, which is captured directly in ICE notation strings, and the "history" of exploration. The generative sequence is a parsimonious, one-time, generation, while the exploratory history may be an extensive, cyclic process that transforms the notation string, repeatedly, until a satisfactory configuration is achieved. The exploratory history can be captured in a sequence of configuration strings. Alternatively it can be captured by means of an initial configuration string and a sequence of transformations that will be applied to this string.

Transforming the regulated element



REPLACE_SHAPE $shape_A \Rightarrow shape_B$

$\Delta\mathbf{R}_2^0[\{\overline{p}, \theta = 20, n = 9\}($

$\Delta\mathbf{R}_1^0\Delta\mathbf{D\Xi G}[\{\overline{p}, \theta = 3, n = 26, k_x = .95, r = -.058, b = +.028\}(shape_B)^{<1>-<26>}]$

$)^{<1>-<9>}]$

Transforming the generation method

MODIFY_NUMBER $\Delta\mathbf{R}_2[\ \{n = 9\}^{<0-9>}] \Rightarrow \Delta\mathbf{R_2}[\ \{n = 4\}^{<0-4>}]$

$\Delta\mathbf{R}_2^0[\{\overline{p}, \alpha = 20, n = 4\}($

$\Delta\mathbf{R}_1^0\Delta\mathbf{D\Xi G}[\{\overline{p}, \alpha = 3, n = 26, k_x = .95, r = -.058, b = +.028\}(shape_A)^{<1>-<26>}]$

$)^{<1>-<4>}]$

Transforming the regulator parameters

MOVE_REGULATOR $\Delta\mathbf{R}[\{\overline{p} = (0,0)\}] \Rightarrow \Delta\mathbf{R}[\ \{\overline{p} = (-2,1)\}]$

$\Delta\mathbf{R}_2^0[\{\overline{p}, \alpha = 20, n = 9\}($

$\Delta\mathbf{R}_1^0\Delta\mathbf{D\Xi G}[\{\overline{p}, \alpha = 3, n = 26, k_x = .95, r = -.058, b = +.028\}(shape_A)^{<1>-<26>}]$

$)^{<1>-<9>}]$

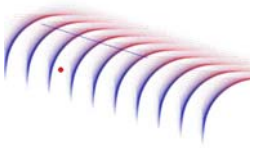
MODIFY_FACTOR $\Delta\mathbf{R}_2[\ \{\alpha = 20\}\ ] \Rightarrow \Delta\mathbf{R_1}[\ \{\alpha = 120\}\ ]$

$\Delta\mathbf{R}_2^0[\{\overline{p}, \alpha = 20, n = 9\}($

$\Delta\mathbf{R}_1^0\Delta\mathbf{D\Xi G}[\{\overline{p}, \alpha = 1, n = 26, k_x = .95, r = -.058, b = +.028\}(shape_A)^{<1>-<26>}]$

$)^{<1>-<9>}]$

Transforming the regulator composition



REPLACE_REGULATOR $\Delta\mathbf{R}_1^0 \Rightarrow \Delta\mathbf{T}_1^1$

$\Delta\mathbf{T}_2^1[\{\overline{p}, d = 25, n = 9\}($

$\Delta\mathbf{R}_1^0\Delta\mathbf{D\Xi G}[\{\overline{p}, \alpha = 3, n = 26, k_x = .95, r = -.058, b = +.028\}(shape_A)^{<1>-<26>}]$

$)]$

TABLE 4.14 TRANSFORMATIONS ON THE GCAD LOGO

REGULATED ELEMENT (applicable to point -- shape)	
INSTANTIATE_SHAPE	
MOVE_XYZ	$\bar{s} = (0,0,0) \Rightarrow \bar{s} = (0,0,1)$
MODIFY_ATTRIBUTE	$shape\langle attribute\rangle = a \Rightarrow shape\langle attribute\rangle = b$
REPLACE_SHAPE	$shape_A \Rightarrow shape_B$
GENERATION METHOD	
MODIFY_CONTINUITY	$\Delta\mathbf{T}(\bar{s})^{<0>-<4-9>} \Rightarrow \Delta\mathbf{T}(\bar{s})^{<0-9>}$
MODIFY_NUMBER	$\Delta\mathbf{T}[\{n=5\}(\bar{s})^{<0-5>}] \Rightarrow \Delta\mathbf{T}[\{n=12\}\ (\bar{s})^{<0-12>}]$
MODIFY_PATTERN	$\Delta\mathbf{T}(\bar{s})^{\because m=4<ii>\phi\phi} \Rightarrow \Delta\mathbf{T}(\bar{s})^{\because m=8<iii>\phi}$
MODIFY_GENERATED	$\Delta\mathbf{T}(\bar{s})^{<0><1><3><4>} \Rightarrow \Delta\mathbf{T}(\bar{s})^{<0><2><4>}$
REGULATOR PARAMETERS (applicable to regulator – and simultaneous composition)	
MOVE_XYZ	$\Delta\mathbf{T}[\{\bar{p}=(0,0,1)\}] \Rightarrow \Delta\mathbf{T}[\{\bar{p}=(1,1,1)\}]$
ROTATE_XYZ	$\Delta\mathbf{T}[\{\bar{t}=(0,0,1)\}] \Rightarrow \Delta\mathbf{T}[\{\bar{t}=(1,3,4)\}]$
MODIFY_FACTOR	$\Delta\mathbf{T}[\{d=5\}] \Rightarrow \Delta\mathbf{T}[\{d=8,\}]$
$d,\alpha,\bar{k},a,v,c,o,s,\min,\max,\mathrm{mod}$	$\Delta\mathbf{R}[\{\alpha=10\}] \Rightarrow \Delta\mathbf{R}[\{\alpha=30\}]$
	$\Delta\mathbf{D}[\{\bar{k}=(1,1,1.2)\}] \Rightarrow \Delta\mathbf{D}[\{\bar{k}=(1,0.8,1)\}]$
MODIFY_FORMULA	$\Xi\mathbf{G}[\{\boldsymbol{f}:(2x=1)\}] \Rightarrow \Xi\mathbf{G}[\{\boldsymbol{f}:(x^2-2)\}]$
MODIFY_DIMENSION	$\Delta\mathbf{M}^1[\{\bar{p},\bar{t},n\}] \Rightarrow \Delta\mathbf{M}^2[\{\bar{p},\bar{t},\bar{v},n\}]$
MODIFY_INDEX	$\Delta\mathbf{R}_3[\{\}] \Rightarrow \Delta\mathbf{R}_2[\{\}]$
REGULATOR COMPOSITION	
ADD_SIMULTANEOUS	$\Delta\mathbf{T}^1[\{\bar{p},\bar{t},d,n\}] \Rightarrow \Delta\mathbf{T}^1\Delta\mathbf{D}^0[\{\bar{p}_T,\bar{p}_D,\bar{t},\bar{k},d,n\}]$
REMOVE_SIMULTANEOUS	$\Delta\mathbf{T}^1\Delta\mathbf{R}^1[\ \{\bar{p}_T,\bar{p}_R,\bar{t},\alpha,d,n\}] \Rightarrow \Delta\mathbf{T}^1[\{\bar{p},\bar{t},d,n\}]$
SWAP_SIMULTANEOUS	$\Delta\mathbf{T}^1\Delta\mathbf{R}^1[(shape)] \Rightarrow \Delta\mathbf{R}^1\Delta\mathbf{T}^1[(shape)]$
ADD_SUCCESSIVE	$\Delta\mathbf{R}^1[(shape)] \Rightarrow \Delta\mathbf{T}^1[(\Delta\mathbf{R}^1[(shape)])]$
INSERT_SUCCESSIVE	$\Delta\mathbf{T}^1[(shape)] \Rightarrow \Delta\mathbf{T}^1[(\Delta\mathbf{R}^1[(shape)])]$
DELETE_SUCCESSIVE	$\Delta\mathbf{T}^1[\ (\Delta\mathbf{R}^1[\ (shape)]\ )] \Rightarrow \Delta\mathbf{T}^1[\ (shape)]$
SWAP_ SUCCESSIVE	$\Delta\mathbf{T}^1[\ (\Delta\mathbf{R}^1[\ (shape)]\ )] \Rightarrow \Delta\mathbf{R}^1[\ (\Delta\mathbf{T}^1[\ (shape)]\ )]$
REPLACE_REGULATOR	$\Delta\mathbf{R}^1\Delta\mathbf{T}^1[\{\bar{p}_T,\bar{p}_R,\bar{t},\alpha,d,n\}(shape)] \Rightarrow$
	$\Delta\mathbf{D}^0\Delta\mathbf{T}^1[\{\bar{p}_T,\bar{p}_D,\bar{t},\bar{k},d,n\}(shape)]$
	$\Delta\mathbf{T}^1[\ \{\bar{p},\bar{t},d,n\}(\Delta\mathbf{R}^1[\{\bar{p},\bar{t},\alpha,n\}(shape)])] \Rightarrow$
	$\Delta\mathbf{T}^1[\ \{\bar{p},\bar{t},d,n\}(\Delta\mathbf{D}^0[\{\bar{p},\bar{k},n\}(shape)])]$
ADD_SHARED	$\Delta\mathbf{T}^1[(shape)] \Rightarrow \Delta\mathbf{T}^1[(shape)] \wedge \Delta\mathbf{A}^1[(shape)]$

TABLE 4.15 - NOTATION FOR TRANSFORMATIONS IN ICE

4.7. SHAPE REPRESENTATION

By using transformation regulators and the continuous generation method, the ICE notation has the capacity to describe a variety of shapes. In this section, ICE's generative techniques are illustrated through a gallery of linear, planar, and volumetric shapes.

4.7.1. LINEAR SHAPES

Linear shapes are generated by the application of a single continuous regulator, or by the application of multiple regulators using partial composition, as is shown in Table 4.16.

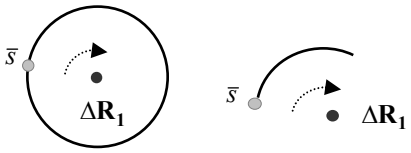
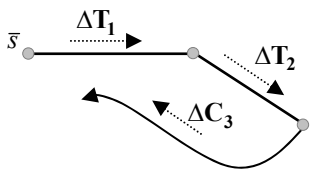
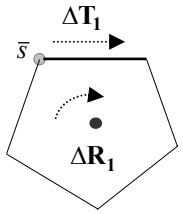
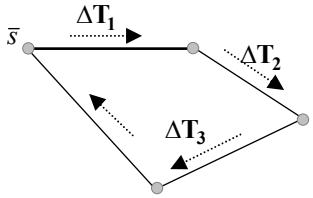
Straight line $line = \Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}]$	
Circular outline $cirlce = \Delta\mathbf{R_1}[\ \{\bar{p},\bar{t},\theta = 360,n\}\ (\bar{s})^{<0-1>}]$ $arc = \Delta\mathbf{R_1}[\ \{\bar{p},\bar{t},\theta = 98,n\}\ (\bar{s})^{<0-1>}]$	
Curved line $curve = \Delta\mathbf{C_1}[\ \{\bar{p},\theta,n\}\ (\bar{s})^{<0-1>}]$	
Complex polyline $polyline = \Delta\mathbf{C_{1c}}[\{\bar{p},\bar{t},d,n\}($ $\Delta\mathbf{T_{1b}}[\{\bar{p},\bar{t},d,n\}($ $\Delta\mathbf{T_{1a}}[\ \{\bar{p},\bar{t},d,n\}(\bar{s})_{\#1}^{<0-1>}]$ $)_{\#1}^{<0-1>}]\ )^{<0-1>}]$	
Regular polygon $pentagon = \Delta\mathbf{R_{1b}}[\{\bar{p},\bar{t},\theta = 72,n\}($ $\Delta\mathbf{T_{1a}}[\ \{\bar{p},\bar{t},d,n\}(\bar{s})_{\#1}^{<0-1>}]$ $)^{<0>-<4>}]$	
Irregular polygon $irregular = \Delta\mathbf{T_{1c}}[\{\bar{p},\bar{t},d,n\}($ $\Delta\mathbf{T_{1b}}[\{\bar{p},\bar{t},d,n\}($ $\Delta\mathbf{T_{1a}}[\ \{\bar{p},\bar{t},d,n\}(\bar{s})_{\#1}^{<0-1>}])_{\#1}^{<0-1>}]$ $)^{<0-1>}]\longrightarrow(\bar{s})$	

TABLE 4.16 - REPRESENTATION OF LINEAR SHAPES

The straight line is generated by the Translation regulator, $\Delta\mathbf{T_1}$, sweeping the starting point $\bar{s}$. The circle's outline is generated by the Rotation regulator, $\Delta\mathbf{R_1}$, sweeping the point $\bar{s}$ through 360º. If the angle is less than 360º, the result is an arc. Similarly, the curve regulator $\Delta\mathbf{C_3}$ sweeps $\bar{s}$ to create a curved line.

A polyline is generated by successive compositions; each regulator inputs only the last point of the preceding regulator. The polygon's outline is generated by translating a point to construct an edge, then by rotating it, discretely, to construct the remaining sides. The irregular polygon is also generated by successive compositions. However, the first point is the same as the last one, and this is denoted by the arrow leading to the first point.

4.7.2. PLANAR SHAPES

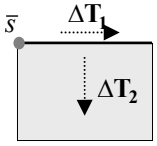
Planar shapes are generated by the application of two continuous successive regulators, or multiple regulators using partial composition and discrete generation as is illustrated in Tables 4.17 and 4.18.

A rectangle is generated by the successive composition of two Translation regulators. A solid triangle is generated by applying the composite regulator, $\Delta\mathbf{T}\Delta\mathbf{D_2}$. If the dilation factor is increased the result is a trapezoid (or quadrilateral). A solid circle is generated by sweep-rotating a line through 360º. Similarly, the semicircle is rotated through 180º, and the pie through 270º. A solid curved surface is generated by sweeping a curved line along a Translation or along another curved line.

A rhombus is generated by discretely mirroring the solid triangle, and similarly, a solid polygon is generated by discretely rotating the triangle. An irregular polygon is defined by means of quadrilaterals, using partial composition, like the polyline. The first quadrilateral is generated by sweep-scaling a line. The end line of each quadrilateral is swept to generate the subsequent quadrilateral.

Rectangle

$$rect = \Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]$$



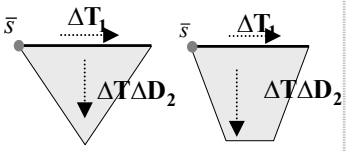
Triangle and Trapezoid

$triangle =$

$$\Delta\mathbf{T}\Delta\mathbf{D_2}[\ \{\bar{p},\bar{t},\bar{k}=(.5,1),d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]$$

$trapezoid =$

$$\Delta\mathbf{T}\Delta\mathbf{D_2}[\ \{\bar{p},\bar{t},\bar{k}=(.75,1),d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]$$

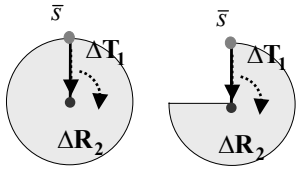


Circle and Variations

$$circle = \Delta\mathbf{R_2}[\ \{\bar{p},\bar{t},\theta=360,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]$$

$$semicircle = \Delta\mathbf{R_2}[\ \{\bar{p},\bar{t},\theta=180,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]$$

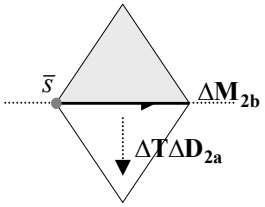
$$packman = \Delta\mathbf{R_2}[\ \{\bar{p},\bar{t},\theta=270,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]$$



Rhombus

$$triangle = \Delta\mathbf{T}\Delta\mathbf{D_{2a}}[\ \{\bar{p},\bar{t},\bar{k},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]$$

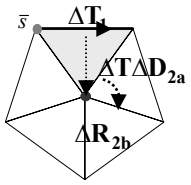
$$rombus = \Delta\mathbf{M_{2b}}[\{\bar{p},\bar{t},n\}(triangle)^{<0><1>}]$$



Solid Regular polygon

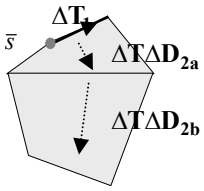
$$triangle = \Delta\mathbf{T}\Delta\mathbf{D_{2a}}[\ \{\bar{p},\bar{t},\bar{k},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]$$

$$polygon = \Delta\mathbf{R_{2b}}[\{\bar{p},\bar{t},0,n\}(triangle)^{<0><4>}]$$



Irregular Polygon

$$\begin{aligned}irregular = \Delta\mathbf{T}\Delta\mathbf{D_3}[\{\bar{p},\bar{t},d,\bar{k},n\}(&\\ &\Delta\mathbf{TD_2}[\{\bar{p},\bar{t},d,\bar{k},n\}(\\ &\quad\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}(\bar{s})^{<0-1>}])\\ &{}^{<0-1>}_{\#1}])\\ &{}^{<0-1>}]\end{aligned}$$



Solid curved surface

$$surface = \Delta\mathbf{T_2}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{C_1}[\ \{\bar{p},\theta,n\}(\bar{s})^{<0-1>}])^{<0-1>}]$$

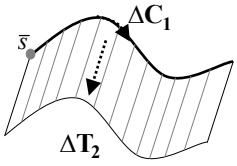


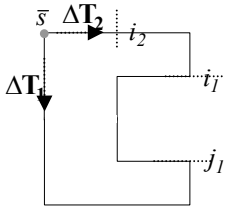
TABLE 4.17 - REPRESENTATION OF SIMPLE PLANAR SHAPES

U-shape

$part1 = \Delta\mathbf{T_2}[\{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-n_1>}])^{<0-i_2>}] \wedge$

$part2 = \Delta\mathbf{T_2}[\{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-i_1><j_1-n_1>}])^{<i_2-n_2>}]$

$ushape = part1 \wedge part2$

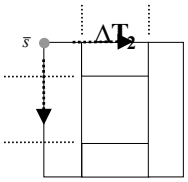


Square donut

$part1 = \Delta\mathbf{T_2}[\{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-n_1>}])^{<0-i_2><j_2-n_2>}] \wedge$

$part2 = \Delta\mathbf{T_2}[\{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-i_1><j_1-n_1>}])^{<i_2-j_2>}]$

$donut = part1 \wedge part2$



Ring, concentric, and radial

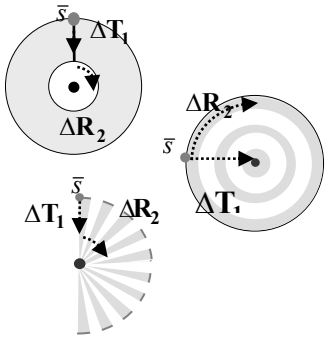
$ring = \Delta\mathbf{R_2}[\ \{\bar{p},\bar{t},\theta = 360,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]$

$concentric =$

$\Delta\mathbf{R_2}[\ \{\bar{p},\bar{t},\theta = 360,n = 20\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<ii>\varphi\varphi}])^{<0-n>}]$

$radial =$

$\Delta\mathbf{R_2}\ [\ \{\bar{p},\bar{t},\theta = 180,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-n>}])^{<ii>\varphi\varphi>}]$



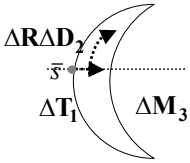
Crescent

$crescent =$

$\Delta\mathbf{M_{2b}}[\{\bar{p},\bar{t},n\}($

$\quad \Delta\mathbf{R}\Delta\mathbf{D_{2a}}[\{\bar{p},\bar{t},\theta,\bar{k},n\}(\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}(\bar{s})^{<0-1>}])^{<0-1>}]$

$)^{<0><1>}]$



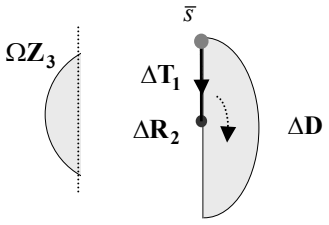
Slice

$semicircle = \Delta\mathbf{R_2}[\{\bar{p},\bar{t},\theta = 180,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]$

$\Delta\mathbf{D}\ [\ \{\bar{p},\bar{k}\}\ (semicircle)]$

$circle = \Delta\mathbf{R}\ [\ \{\bar{p},\bar{t},\theta = 360,n\}\ (\Delta\mathbf{T}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{p})^{<0-1>}])^{<0-1>}]$

$\Delta\mathbf{Z}\ [\ \{3,line\}\ (circle)^{<1>}]$



Composed shapes

$\Delta\mathbf{U}[\{\}($

$\quad \Delta\mathbf{T_{A2}}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_{A1}}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_A)^{<0-1>}])^{<0-1>}],$

$\quad \Delta\mathbf{T_{B2}}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_{B1}}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_B)^{<0-1>}])^{<0-1>}]$

$)]$

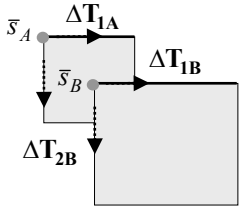


TABLE 4.18 - REPRESENTATION OF COMPLEX PLANAR SHAPES

The U-shape is generated by means of the subset generation method. The regulators are applied twice from the same starting point, each time deriving part of the shape. The same method is applicable to the L-shape. A square with a hole is described by using the subset generation method. The regulators needs to be applied twice, once to generate the vertical sides, and once to generate the horizontal sides. There are other ways of generating a square with a hole, for instance the pinwheel method, however, the two-pass method is more flexible, and can describe variations such as rectangular holes, or many holes, etc.

A ring is generated as a variant of the circle, a line (that does not intersect with the center of rotation) is sweep-rotated through 360 degrees. The concentric pattern is a subshape of a circle and defined by means of the subset generation method. The first regulator defines a pattern, while the other is continuous. Similarly, the radial pattern, which is another subshape of the circle, is a defined by subset generation, where the first regulator is continuous, and the second regulator defines the pattern. A crescent is generated by means of a scale rotating a line, then mirroring it discretely. A solid slice is defined by applying the dilation $\Delta\mathbf{D}_3$ to a semicircle, or alternatively by subdividing a circle about a line.

4.7.3. VOLUMETRIC SHAPES

Volumetric shapes are generated by applying three successive regulators as is illustrated in Table 4.19, which extends through several pages.

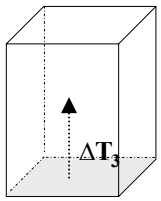
A cuboid is generated by sweeping a square along the Translation regulator $\Delta\mathbf{T}_3$. Similarly, a prism is generated by sweeping a triangular base along $\Delta\mathbf{T}_3$. A rotated prism, on the other hand, is generated by sweeping the triangular base along the screw regulator, $\Delta\mathbf{T}\Delta\mathbf{R}_3$, and a pyramid is generated by sweeping a square base along the composite regulator, $\Delta\mathbf{T}\Delta\mathbf{D}_3$. If the scale factor is decreased, the result is a frustum. The octahedron is defined by discretely mirroring a pyramid.

Cuboid

square =

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

$cuboid = \Delta\mathbf{T_3}[\ \{\overline{p},\bar{t},d,n\}\ (square)^{<0-1>}]$

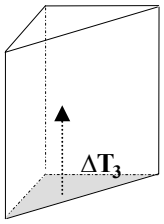


Prism

triangle =

$\Delta\mathbf{T}\Delta\mathbf{D_2}[\ \{\overline{p},\bar{t},\bar{k},d,n\}\ (\Delta\mathbf{T_1}[\ \{\overline{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}]\ )^{<0-1>}]$

$prism = \Delta\mathbf{T_3}[\ \{\overline{p},\bar{t},d,n\}\ (triangle)^{<0-1>}]$

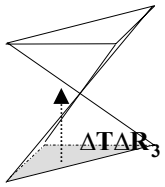


Rotated Prism

triangle =

$\Delta\mathbf{T}\Delta\mathbf{D_2}[\ \{\overline{p},\bar{t},\bar{k},d,n\}\ (\Delta\mathbf{T_1}[\ \{\overline{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}]\ )^{<0-1>}]$

$Rotated\_prism = \Delta\mathbf{T}\Delta\mathbf{R_3}[\ \{\overline{p},\bar{t},d,\theta,n\}\ (triangle)^{<0-1>}]$



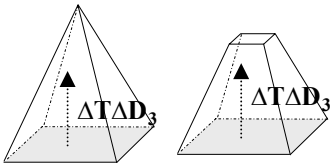
Pyramid and Frustum

square =

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

$pyramid = \Delta\mathbf{T}\Delta\mathbf{D_3}[\ \{\overline{p},\bar{t},\bar{k},\theta,n\}\ (square)^{<0-1>}]$

$frustum = \Delta\mathbf{T}\Delta\mathbf{D_3}[\ \{\overline{p},\bar{t},\bar{k},\theta,n\}\ (square)^{<0-1>}]$



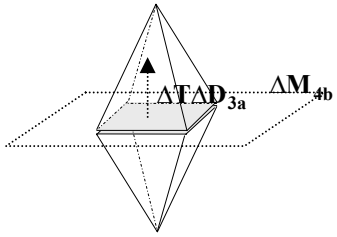
Octahedron

triangle =

$\Delta\mathbf{T}\Delta\mathbf{D_2}[\ \{\overline{p},\bar{t},\bar{k},d,n\}\ (\Delta\mathbf{T_1}[\ \{\overline{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}]\ )^{<0-1>}]$

$pyramid = \Delta\mathbf{T}\Delta\mathbf{D_{3a}}[\ \{\overline{p},\bar{t},\bar{k},\theta,n\}\ (square)^{<0-1>}]$

$octahedron = \Delta\mathbf{M_{3b}}[\ \{\overline{p},\bar{t},\bar{v},d,e,\theta,n\}\ (square)^{<0-1>}]$



Cylinder

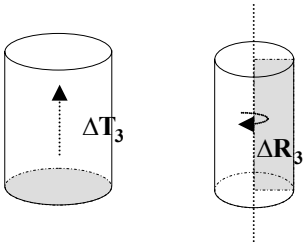
circle =

$\Delta\mathbf{R_2}[\ \{\overline{p},\bar{t},\theta=360,n\}\ (\Delta\mathbf{T_1}[\ \{\overline{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]$

$cylinder = \Delta\mathbf{T_3}[\ \{\overline{p},\bar{t},d,n\}\ (circle)^{<0-1>}]$

$rect = \Delta\mathbf{T_2}[\ \{\overline{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\overline{p},\bar{t},d,n\}\ (\overline{\mathrm{p}})^{<0-1>}])^{<0-1>}]$

$cylinder = \Delta\mathbf{R_3}[\ \{\overline{p},\bar{t},d,n\}\ (rect)^{<0-1>}]$

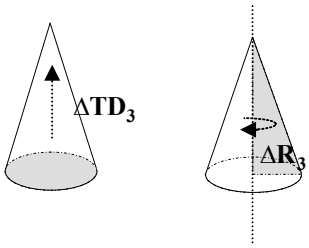


Cone

circle =

$\Delta\mathbf{R_2}[\ \{\bar{p},\bar{t},\theta=360,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]$

cone = $\Delta\mathbf{TD_3}[\ \{\bar{p},\bar{t},d,n\}\ (circle)^{<0-1>}]$



triangle =

$\Delta\mathbf{T}\Delta\mathbf{D_2}[\ \{\bar{p},\bar{t},\bar{k},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}]\ )^{<0-1>}]$

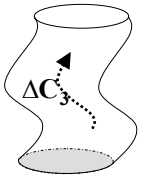
cone = $\Delta\mathbf{R_3}[\ \{\bar{p},\bar{t},d,n\}\ (triangle)^{<0-1>}]$

Slinky

circle =

$\Delta\mathbf{R_2}[\ \{\bar{p},\bar{t},\theta=360,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]$

slinky = $\Delta\mathbf{C_3}[\ \{\bar{p},\bar{t},\theta,n\}\ (circle)^{<0-1>}]$

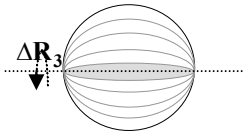


Sphere

circle =

$\Delta\mathbf{R_2}[\ \{\bar{p},\bar{t},\theta=360,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]$

sphere = $\Delta\mathbf{R_3}[\ \{\bar{p},\bar{t},\theta,n\}\ (circle)^{<0-1>}]$

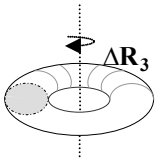


Torus

circle =

$\Delta\mathbf{R_2}[\ \{\bar{p},\bar{t},\theta=360,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]$

torus = $\Delta\mathbf{R_3}[\ \{\bar{p},\bar{t},\theta,n\}\ (circle)^{<0-1>}]$

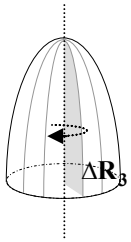


Paraboloid

slice =

$\Delta\mathbf{R_2}[\ \{\bar{p},\bar{t},\theta=360,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]$

paraboloid = $\Delta\mathbf{R_3}[\ \{\bar{p},\bar{t},\theta,n\}\ (slice)^{<0-1>}]$



Football

slice =

$\Delta\mathbf{R_2}[\ \{\bar{p},\bar{t},\theta=360,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]$

football = $\Delta\mathbf{R_3}[\ \{\bar{p},\bar{t},\theta,n\}\ (slice)^{<0-1>}]$

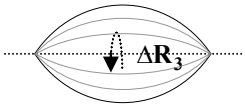


TABLE 4.19 - REPRESENTATION OF VOLUMETRIC SHAPES

A cylinder is generated either by sweeping a circular base along a Translation regulator $\Delta\mathbf{T_3}$, or by sweep-rotating a rectangle about the regulator $\Delta\mathbf{R_3}$. Likewise, a cone is generated either by sweeping a circle along the composite regulator, $\Delta\mathbf{T}\Delta\mathbf{D_3}$, or by rotating a triangle about the regulator $\Delta\mathbf{R_3}$. Similarly a slinky is generated by sweeping a circle along a curve regulator, $\Delta\mathbf{C_3}$. A sphere is generated by sweep-rotating a circle about the regulator, $\Delta\mathbf{R_3}$, positioned along the diameter of the circle; while a torus is generated by sweep-rotating a circle about the regulator $\Delta\mathbf{R_3}$, positioned outside the circle. The solid paraboloid and the solid football are generated by rotating a slice about the regulator, $\Delta\mathbf{R_3}$.

4.7.4. SHAPE TRANSFORMATIONS

The ICE notation supports the transformation of one shape to another, just by changing the definition of the regulators as is illustrated in Table 4.20.

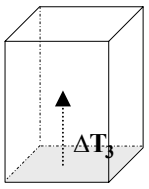
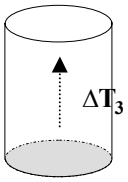
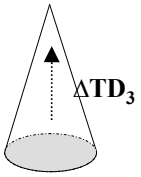
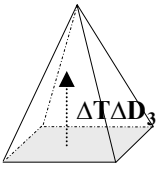
Starting shape the cuboid *cuboid* = $\Delta\mathbf{T_3}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})\ ]\ )\ ]\ )\ ]$	
REPLACE_REGULATOR $\Delta\mathbf{T_2}[\{\bar{p},\bar{t},d,n\}] \Rightarrow \Delta\mathbf{R}_2[\{\bar{p},\bar{t},\theta,n\}]$ *cylinder* = $\Delta\mathbf{T_3}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{R_2}[\ \{\bar{p},\bar{t},\theta,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})\ ]\ )\ ]\ )\ ]$	
ADD_SIMULTANEOUS $\Delta\mathbf{T_3}[\{\bar{p},\bar{t},d,n\}] \Rightarrow \Delta\mathbf{T}\ \ \Delta\mathbf{D}\ \ [\{\bar{p}_T,\bar{p}_D,\bar{t},\bar{k},d,n\}]$ *cone* = $\Delta\mathbf{TD_3}[\ \{\bar{p}_T,\bar{p}_D,\bar{t},\bar{k},d,n\}\ (\Delta\mathbf{R_2}[\ \{\bar{p},\bar{t},\theta,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})\ ]$	
REPLACE_REGULATOR $\Delta\mathbf{R}_2[\{\bar{p},\bar{t},\theta,n\}] \Rightarrow \Delta\mathbf{T_2}[\{\bar{p},\bar{t},d,n\}]$ *pyramid* = $\Delta\mathbf{TD_3}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})]\ )\ ])]$	

TABLE 4.20 TRANSFORMATION ACROSS SHAPES

4.8. PATTERN GENERATION AND TRANSFORMATION

By using transformation regulators and the discrete generation method, the ICE notation has the capacity to describe various types of patterns. In this section, I present examples of cyclic, dihedral, frieze, and wallpaper patterns, based on symmetry group classifications. The focus is on the mapping between regulators and the symmetries of the patterns, as well as the use of regulators to transform patterns. The complete set of cyclic, dihedral, frieze and wallpaper patterns, as well as the transformation among these patterns is presented in Appendix C.

4.8.1. CYCLIC AND DIHEDRAL PATTERNS

Cyclic patterns have a single center of finite rotation; dihedral patterns have, additionally, mirrors intersecting at the center of rotation. Table 4.21 shows an example of a Cyclic and a Dihedral pattern, and illustrates how to transform one to the other by means of the ICE notation.

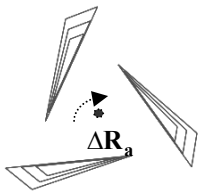
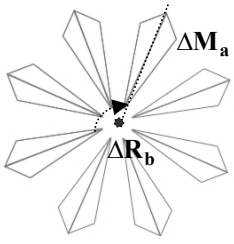
Cyclic pattern C(3) $C(3) = \Delta\mathbf{R_a}[\ \{\bar{p}, \bar{t}, \theta = 120, n\}\ (shape)^{<0>-<2>}]$	
Transforming pattern C(3) to D(8) MODIFY_FACTOR (θ) $\Delta\mathbf{R_a}[\{\theta = 120\}] \Rightarrow \Delta\mathbf{R_a}[\{\theta = 45\}]$ INSERT_SUCCESSIVE ($\Delta\mathbf{M_a}$) $\Delta\mathbf{R_a}[(shape)] \Rightarrow \Delta\mathbf{R_b}[(\Delta\mathbf{M_a}[(shape)])]$	
Dihedral pattern D(8) $D(8) = \Delta\mathbf{R_b}[\{\bar{p}, \bar{t}, \theta = 45, n\}(\Delta\mathbf{M_a}[\{\bar{p}, \bar{t}, 1\}(shape)^{<0><1>}])^{<0>-<7>}]$	
Transforming pattern D(8) to C(3) DELETE_SUCCESSIVE ($\Delta\mathbf{M_a}$) $\Delta\mathbf{R_b}[(\Delta\mathbf{M_a}[(shape)])] \Rightarrow \Delta\mathbf{R_a}[(shape)]$ MODIFY_FACTOR (θ) $\Delta\mathbf{R_a}[\{\theta = 45\}] \Rightarrow \Delta\mathbf{R_a}[\{\theta = 120\}]$	

TABLE 4.21 - CYCLIC AND DIHEDRAL PATTERNS

The cyclic pattern C(3) is generated by using the Rotation regulator $\Delta\mathbf{R_1}$. The dihedral pattern D(8) is generated by using the Mirror regulator $\Delta\mathbf{M_1}$ and the Rotation regulator $\Delta\mathbf{R_2}$. Although the pattern has four axes of mirror symmetry, ICE only uses one of those as a primary generator.

4.8.2. FRIEZE PATTERNS

Frieze patterns are periodic patterns consisting of infinite translations of a motif in a single direction. The seven frieze patterns admit half-turn rotations, horizontal and vertical mirrors and glide (Martin 1991, p78) A point of symmetry in the motif is a point of symmetry for the whole pattern and a line of symmetry in the motif is a line of symmetry for the whole pattern. Table 4.22 shows two examples of Frieze patterns, and illustrates how to transform one to the other by means of the ICE notation.

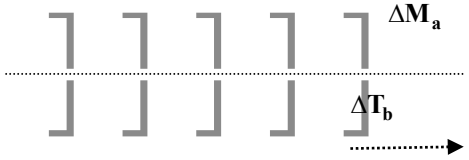
Frieze pattern p112 $p112 = \Delta\mathbf{T_b}[\ \{\bar{p},\bar{t},d,n\}\ ($ $\Delta\mathbf{R_a}[\ \{\bar{p},\bar{t},\theta = 180,l\}\ (shape)^{<0><1>}]$ $)^{<0>-<n>}]$	
Frieze pattern p1m1 $p1m1 = \Delta\mathbf{T_b}[\ \{\bar{p},\bar{t},d,n\}\ ($ $\Delta\mathbf{M_a}[\ \{\bar{p},\bar{t},l\}\ (shape)^{<0><1>}]$ $)^{<0>-<n>}]$	
Transforming pattern p1m1 to p1m1 REPLACE_REGULATOR MOVE_REGULATOR_Y	 $\Delta\mathbf{M_a} \Rightarrow \Delta\mathbf{R_a}$ $\Delta\mathbf{R_a}[\{\bar{p}\}] \Rightarrow \Delta\mathbf{R_a}[\{\bar{p}'\}]$
Transforming pattern p112 to p1m1 REPLACE_REGULATOR MOVE_REGULATOR_Y	 $\Delta\mathbf{R_a} \Rightarrow \Delta\mathbf{M_a}$ $\Delta\mathbf{M_a}[\{\bar{p}\}] \Rightarrow \Delta\mathbf{M_a}[\{\bar{p}'\}]$

TABLE 4.22 - FRIEZE PATTERNS

The Frieze pattern p1m1 is generated by using the horizontal Mirror regulator $\Delta\mathbf{M_1}$ and the Translation regulator $\Delta\mathbf{T_2}$. The pattern also has glide. The Frieze pattern p112 is generated by using a half-turn Rotation regulator $\Delta\mathbf{R_1}$ and the Translation regulator $\Delta\mathbf{T_2}$. The pattern also has another half-turn between the motifs.

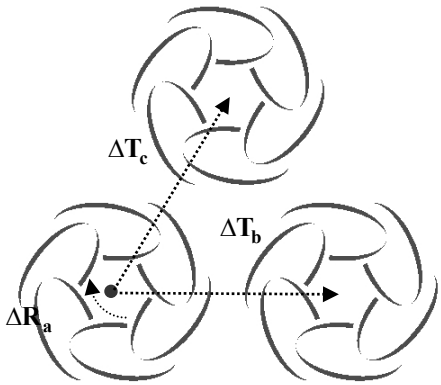
4.8.3. WALLPAPER PATTERNS

Wall paper patterns comprise infinite translations of a motif in two distinct (non-collinear) directions. These form a conceptual lattice that is either rectangular, rhombic or parallelogram. A point of symmetry in the motif is a point of symmetry for the whole pattern, and a line of symmetry in the motif is a line of symmetry for the whole pattern.

The seventeen wall paper patterns admit 2, 3, 4, or 6 centers of rotation and reflection line. (Martin 1991, p88) Table 4.23 shows two examples of wallpaper patterns, and illustrates how to transform one to the other by means of the ICE notation. The wallpaper pattern P6 is generated by using the Rotation regulator $\Delta\mathbf{R_1}$ and the Translation regulators $\Delta\mathbf{T_2}$ and $\Delta\mathbf{T_3}$. The pattern has a rhombic lattice and has 6 centers, 3 centers and 2 centers of rotation. The wallpaper pattern p4g is generated by using the Mirror regulators $\Delta\mathbf{M_1}$ and $\Delta\mathbf{M_2}$, the Rotation regulator $\Delta\mathbf{R_3}$ and the Translation regulators $\Delta\mathbf{T_4}$ and $\Delta\mathbf{T_5}$. The pattern has a rectangular lattice and has 4-centers, 2-centers of rotation as well as glide.

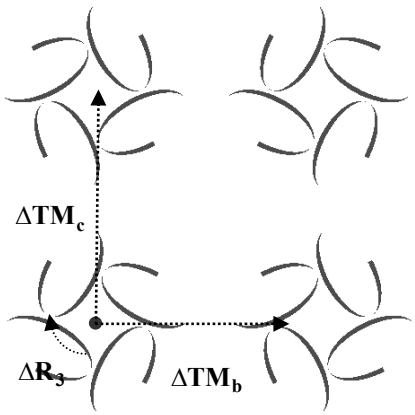
Wallpaper pattern p6

$$p6 = \Delta\mathbf{T_c}[\ \{\bar{p},\bar{t},d,n\}\ ($$
$$\Delta\mathbf{T_b}[\ \{\bar{p},\bar{t},d,n\}\ ($$
$$\Delta\mathbf{R_a}[\ \{\bar{p},\bar{t},\theta=60,n\}\ (shape)^{<0><5>}]$$
$$)^{<0>-<n>}]$$
$$)^{<0>-<n>}]$$



Wallpaper pattern p4g

$$p6 = \Delta\mathbf{TM_c}[\ \{\bar{p},\bar{t},d,n\}\ ($$
$$\Delta\mathbf{TM_b}[\ \{\bar{p},\bar{t},d,n\}\ ($$
$$\Delta\mathbf{R_a}[\ \{\bar{p},\bar{t},\theta=60,n\}\ (shape)^{<0><5>}]$$
$$)^{<0>-<n>}]$$
$$)^{<0>-<n>}]$$



Transforming pattern p6 to p4g

MODIFY_FACTOR (θ) $\quad \Delta\mathbf{R_a}[\{\theta=60\}] \Rightarrow \Delta\mathbf{R_a}[\{\theta=90\}]$

ROTATE_REGULATOR ($\Delta\mathbf{T_c}$) $\quad \Delta\mathbf{T_c}[\{\bar{t}\}] \Rightarrow \Delta\mathbf{T_c}[\{\bar{t}'\}]$

ADD_SIMULTANEOUS ($\Delta\mathbf{M}$)

$$\Delta\mathbf{T_c}[(\Delta\mathbf{T_b}[(\Delta\mathbf{R_a}[(shape)])])] \Rightarrow \Delta\mathbf{T_c}[(\Delta\mathbf{TM_b}[(\Delta\mathbf{R_a}[(shape)])])]$$

ADD_ SIMULTANEOUS ($\Delta\mathbf{M}$)

$$\Delta\mathbf{T_c}[(\Delta\mathbf{TM_b}[(\Delta\mathbf{R_a}[(shape)])])] \Rightarrow \Delta\mathbf{TM_c}[(\Delta\mathbf{TM_b}[(\Delta\mathbf{R_a}[(shape)])])]$$

Steps transforming pattern p4g to p6

MODIFY_FACTOR (θ) $\quad \Delta\mathbf{R_a}[\{\theta=90\}] \Rightarrow \Delta\mathbf{R_a}[\{\theta=60\}]$

ROTATE_REGULATOR ($\Delta\mathbf{T_c}$) $\quad \Delta\mathbf{T_c}[\{\bar{t}'\}] \Rightarrow \Delta\mathbf{T_c}[\{\bar{t}\}]$

REMOVE_SIMULTANEOUS ($\Delta\mathbf{M}$)

$$\Delta\mathbf{TM_c}[(\Delta\mathbf{TM_b}[(\Delta\mathbf{R_a}[(shape)])])] \Rightarrow \Delta\mathbf{T_c}[(\Delta\mathbf{TM_b}[(\Delta\mathbf{R_a}[(shape)])])]$$

REMOVE _ SIMULTANEOUS ($\Delta\mathbf{M}$)

$$\Delta\mathbf{T_c}[(\Delta\mathbf{TM_b}[(\Delta\mathbf{R_a}[(shape)])])] \Rightarrow \Delta\mathbf{T_c}[(\Delta\mathbf{T_b}[(\Delta\mathbf{R_a}[(shape)])])]$$

TABLE 4.23 – WALLPAPER PATTERNS

4.9. REPRESENTATIONAL SCHEMATA

The ICE notation string also encapsulates the relationships between the regulators of the configuration. These interrelationships form representational schemata, which are a higher-level classification subsuming shapes and configurations. Schemata allow us to identify distinct patterns of complex compositions where specific regulators, composition strategies and generation methods are used in combination.

Configuration schemata are discernible patterns of regulator relationships. A schema can be applied to different input shapes and get variable results with similar generation and transformation patterns. Thus, various shapes can be derived from a single higher-level schema. Consequently, schemata can be used as templates to store complex generation sequences and constraint patterns, and can later be retrieved and modified to create specific shapes. A schema can store the generation and constraints for a chair, while another schema can store the generation and constraints for a restaurant. A user can later retrieve the former schema and modify it to create variations of chairs. Transforming regulator parameters does not alter the schema; however, transforming the regulator composition or sequence transforms the schemata completely, therefore the latter form of transformations can be used to create new schemata from existing ones.

Configurational schemata include the following: simple generative schemata, complex generative schemata, hierarchical schemata, topological grid schemata, dynamic schemata. These schemata are often used in combination. In this section, I present some examples of schemata that can be generated in ICE. The graph representation illustrates how the regulators in each type of schemata are interrelated.

The notation for schemata differs from the notation for shapes in the following manner. (i) Shapes are denoted in lowercase, while schemata are denoted in uppercase. (ii) Schemata are defined by the regulator types, composition and their generation methods, while shapes are defined, additionally, by the regulator parameters. Distinct shapes, for instance the circle and the ring, can have the same schema, $CIRCULAR = \Delta\mathbf{R_2}[(\Delta\mathbf{T_1}[(\overline{p})^{<0-1>}])^{<0-1>}]$, but their actual shapes are defined by the parameters of their specific regulators.

4.9.1. SIMPLE GENERATIVE SCHEMATA

Simple generative schemata, which describe symmetric or centralized architectural layouts, utilize successive and simultaneous composition methods to generate shapes and patterns. There are no limits on the number of regulators in the sequence, or in the simultaneous composition. Patterns such as 1D frieze and 2D wallpaper, as well as shape like the cuboids, cylinders, and cones (Table 4.24) are examples of simple generative schemata. The schema for the 3D cone consists of three successive regulators: $\Delta\mathbf{T_1}$, $\Delta\mathbf{R_2}$, and $\Delta\mathbf{T}\Delta\mathbf{D_3}$, all of which are continuous, with the last regulator being a composite of Translation and Dilation.

$$ConeSCHEME = \Delta\mathbf{T}\Delta\mathbf{D_3}[\ (\Delta\mathbf{R_2}[\ (\ \Delta\mathbf{T_1}[\ (\overline{p})^{<0-1>}\ ]\ )^{<0-1>}\ ]\ )^{<0-1>}\ ]$$

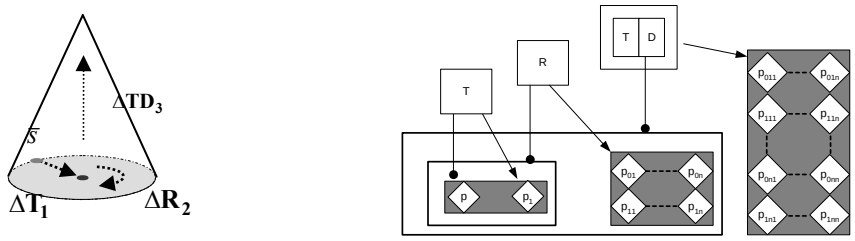


TABLE 4.24 - SIMPLE GENERATIVE SCHEMATA

4.9.2. COMPLEX GENERATIVE SCHEMATA

The complex generative schemata utilize composition methods such as sharing, aggregation, and multiple-control, in addition to simultaneous and successive compositions. These enable the description of intricate configurations and a hierarchy of control that achieves complex behaviors. Table 4.25 shows a schema that consists of a shape shared by several Translation regulators, which are, in turn, regulated by a higher-level Rotation regulator.

$\Delta \mathbf{R_b}[\{\bar{p},\bar{t}\}\ (\Delta \mathbf{T_a})^{<0-3>}] \wedge \Delta \mathbf{T_{ai}}[\{\bar{p},\bar{t},d,n\}\ (s_0)^{<0-3>}]$

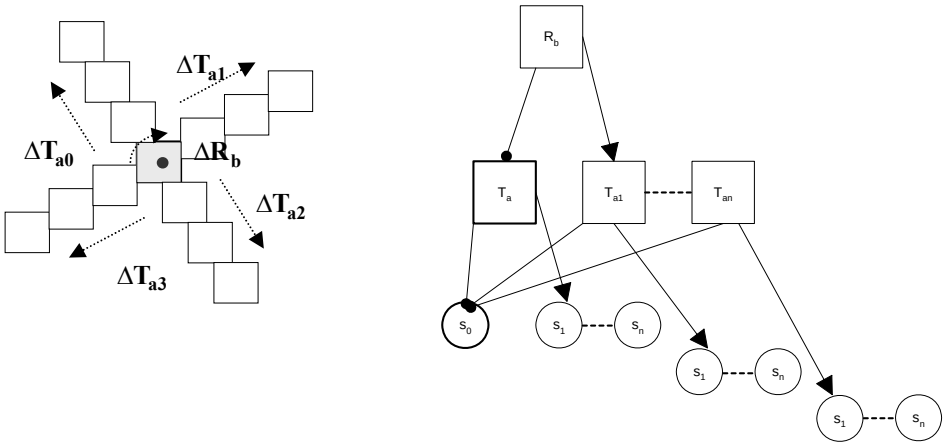


TABLE 4.25 - COMPLEX SCHEMATA

4.9.3. HIERARCHICAL SCHEMATA

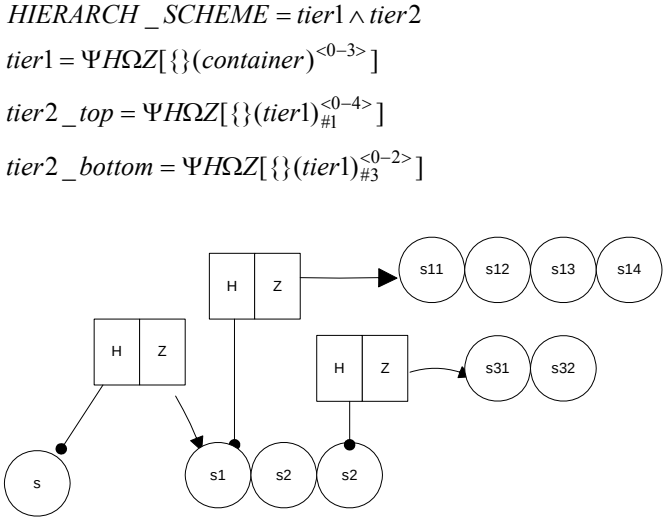
Hierarchical schemata, which describe spatial organizations in buildings, utilize hierarchical regulators to define dependencies between hierarchical structures. These are combined with other regulators (such as subdivision, size, and boundary) through simultaneous composition in order to define topological and geometric relations within hierarchies. There are no limits to the number of levels in the hierarchy. Table 4.26 shows a two-tier hierarchical configuration, which is formed by subdividing the container shape and generating constituents; these are in turn subdivided into their own constituents.

$HIERARCH\_SCHEME = tier1 \wedge tier2$

$tier1 = \Psi H \Omega Z[\{\}(container)^{<0-3>}]$

$tier2\_top = \Psi H \Omega Z[\{\}(tier1)^{<0-4>}_{\#1}]$

$tier2\_bottom = \Psi H \Omega Z[\{\}(tier1)^{<0-2>}_{\#3}]$



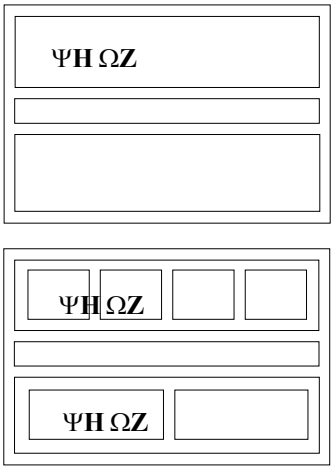
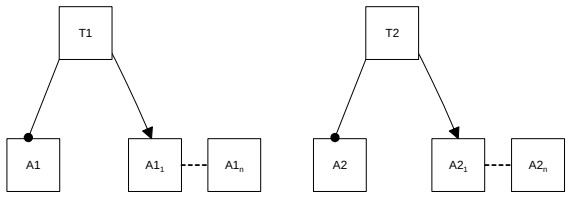


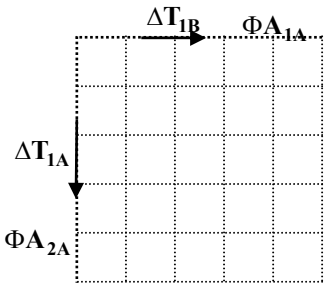
TABLE 4.26 - HIERARCHICAL SCHEMATA

4.9.4. GRID SCHEMATA

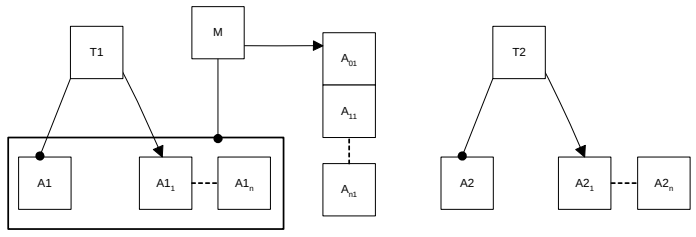
Grid schemata are of particular significance because grids are a major organization tool in Architecture. Grid schemata consist of grid lines, which are alignment or bounding regulators. Grid lines are in turn regulated by Translation or Mirror regulators, which determine the number of grid lines, the distance between them, and the axes of symmetry of the grid. Grids can be manipulated in several ways. Gross (1991) explains that grids can be selected, composed, superimposed, and used to position design elements and define relations among elements. Some of these manipulations correspond to Dürer (source: Mitchell 1990) and Darcy Thomson's strategies (Thomson 1971). In Table 4.27, *GRID1* is defined by translating the alignment regulators $\Phi\mathbf{A_{1A}}$ along $\Delta\mathbf{T_{2A}}$ and $\Phi\mathbf{A_{1B}}$ along $\Delta\mathbf{T_{2B}}$ and *GRID2* is defined the same sequence and additionally by mirroring the second set of alignment lines about $\Delta\mathbf{M_{3B}}$. Table 4.28 shows variations of these grids as transformed by the ICE transformation syntax. When parameters of the grid are changed, that does not alter its schema, however, when the regulator composition is changed, this results in a new schema, and is indicated by a change in the grid's name.

$$GRID1 = \Delta\mathbf{T_{1A}}[\ \{\overline{p},\overline{t},\theta,n\}\ (\Phi\mathbf{A_{1A}})^{<0-n>}\ ]\ \wedge$$
$$\Delta\mathbf{T_{2B}}[\ \{\overline{p},\overline{t},\theta,n\}\ (\Phi\mathbf{A_{1B}})^{<0-n>}\ ]$$





$$GRID2 = \Delta\mathbf{T_{2A}}[\ \{\overline{p},\overline{t},\theta,n\}\ (\Phi\mathbf{A_{1A}})^{<0-n>}\ ]\wedge$$
$$\Delta\mathbf{M_{3B}}[\ \{\overline{p},\overline{t},\theta,n\}(\Delta\mathbf{T_{2B}}[\{\overline{p},\overline{t},\theta,n\}(\Phi\mathbf{A_{1B}})^{<0-n>}])^{<0-n>}]$$



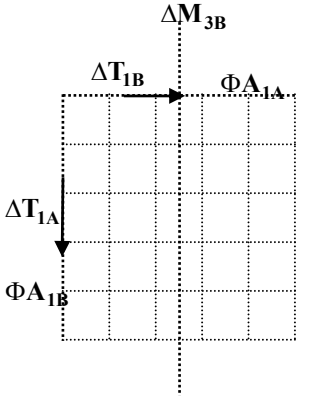


TABLE 4.27 - GRID SCHEMATA

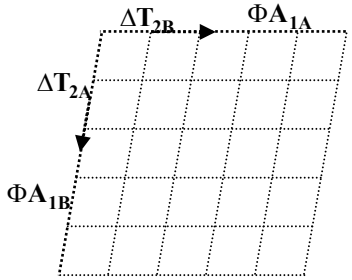
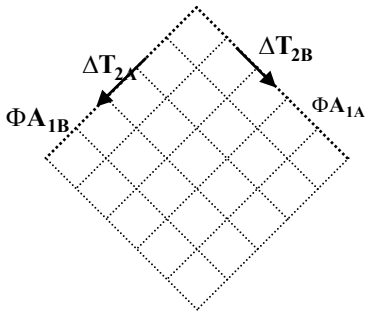
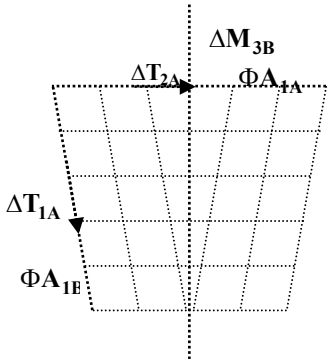
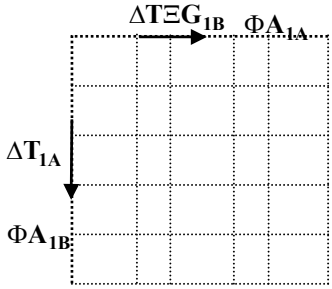
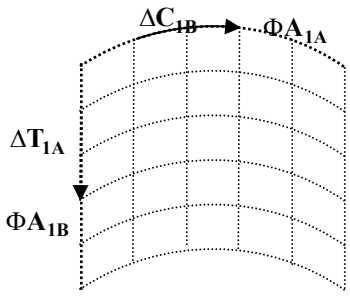
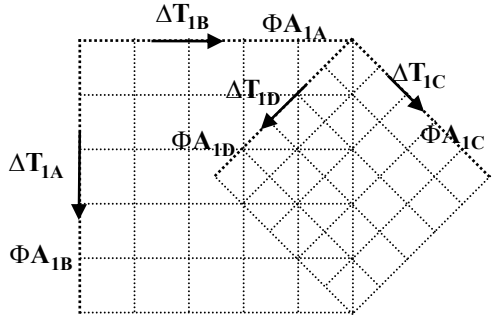
	
GRID1 = $\Delta T_{2A}[\ (\Phi A_{1A})^{<0-n>}] \wedge \Delta T_{2B}[\ (\Phi A_{1B})^{<0-n>}]$ ROTATE_REGULATOR ΔT_{2A}	*GRID1* = $\Delta T_{2A}[\ (\Phi A_{1A})^{<0-n>}] \wedge \Delta T_{2B}[\ (\Phi A_{1B})^{<0-n>}]$ ROTATE_SCHEMA *GRID1*
	
GRID2 = $\Delta T_{2A}[\ (\Phi A_{1A})^{<0-n>}] \wedge \Delta M_{3B}[\ (\Delta T_{2B}[\ (\Phi A_{1B})^{<0-n>}])^{<0-n>}]$ ROTATE_REGULATOR ΔT_{2A}	*GRID1* = $\Delta T_{2A}[\ (\Phi A_{1A})^{<0-n>}] \wedge \Delta T \Xi G_{2B}[\ (\Phi A_{1B})^{<0-n>}]$ ADD_SIMULTANEOUS ΞG
	
GRID4 = $\Delta T_{2A}[\ (\Phi A^{C}{}_{1A})^{<0-n>}] \wedge C_{2B}[\ (\Phi A_{1B})^{<0-n>}]$ REPLACE_REGULATOR (ΔT_{2B} by ΔC_{2B}) REPLACE_REGULATOR (ΦA_{1A} by $\Phi A^{C}{}_{1A}$)	*GRID5* = *GRID1* $\wedge$ *ROTATE _SCHEME*(*GRID1*)

TABLE 4.28 - VARIATIONS OF GRID SCHEMATA

4.9.5. TOPOLOGICAL SCHEMATA

Topological schemata utilize topological regulators to describe proximity relationships between spatial entities and to determine adjacency networks of spatial configurations. Table 4.29 shows set of spaces in a house that are related by means of adjacencies, distances overlaps and boundaries.

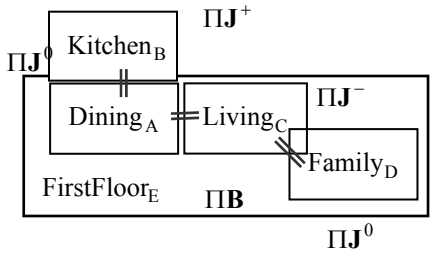
$\Phi\mathbf{J}^{0}[\ \{\}\ (Kitchen_B, Dining_A)\ ]$

$\Phi\mathbf{J}^{-}[\ \{\max = -100\}\ (Living_C, Family_D)\ ]$

$\Phi\mathbf{J}^{0}[\ \{\}\ (Dining_A, Living_C)\ ]$

$\Phi\mathbf{J}^{+}[\ \{\min = 50\}\ (Dining_A, Family_D)\ ]$

$\Pi\mathbf{B}\ [\ \{o\}\ (FirstFloor_E, Living_A - Dining_C)]$



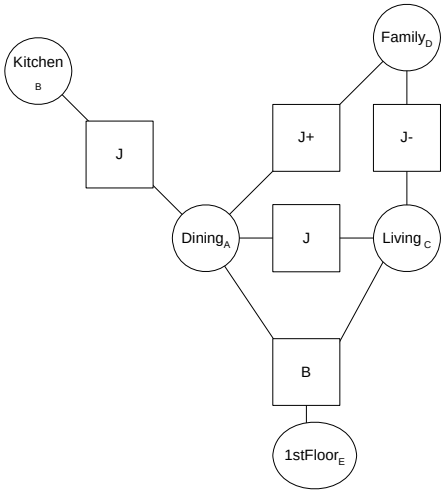


TABLE 4.29 - TOPOLOGICAL SCHEMATA

4.9.6. DYNAMIC SCHEMATA

Dynamic schemata utilize motion regulators in combination with multiple control regulators. Some types of dynamic schemata allow the description of moving components in cases where architecture and mechanical design are integrated, while other types describe complex shapes that can only be defined by means of motion. The latter type combines motion with subsequent generation of output shapes, such that the outputs are generated while the regulator is moving. Table 4.30 illustrates two examples of dynamic schemata, one is the bidirectional rotation of a moving part, and the other is an elliptical configuration generated by means of a moving regulator.

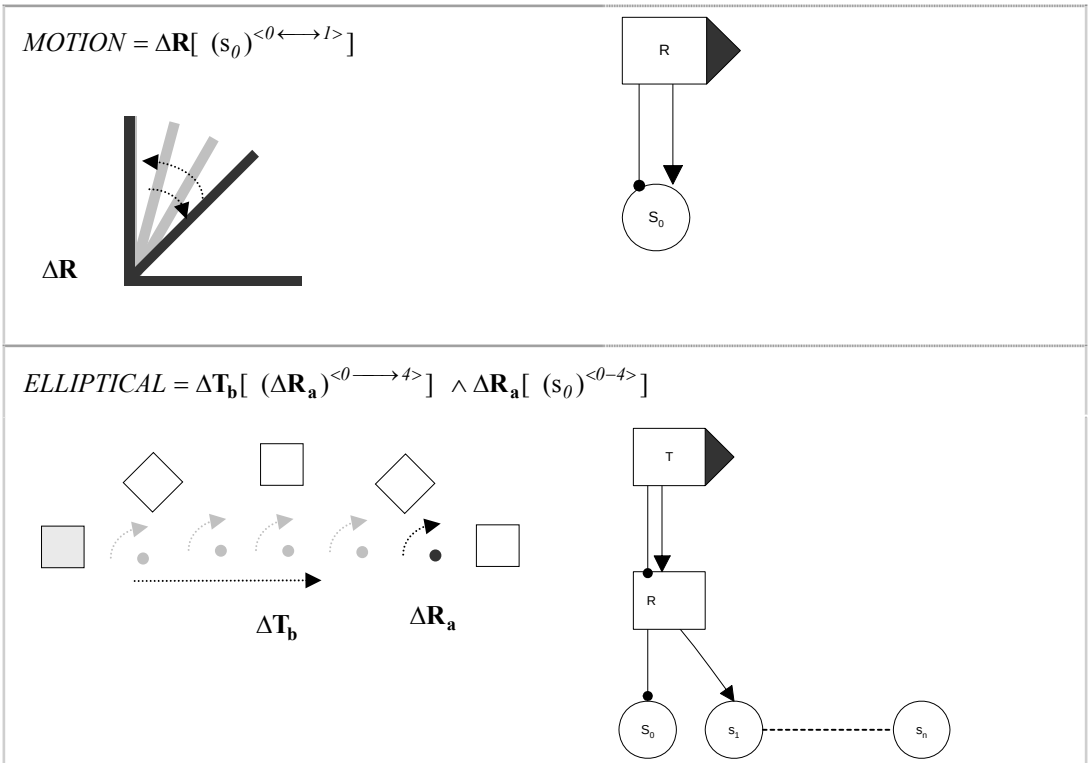


TABLE 4.30 - DYNAMIC SCHEMATA

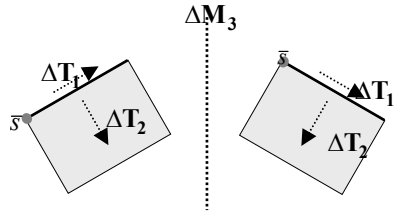
4.9.7. SCHEMATA ENCAPSULATION

Schemata encapsulation creates a multilevel schema hierarchy, where a schema is regulated like other objects. When schemata are regulated, it's starting point/shape and regulators are regulated. Schemata are always regulated discretely. Table 4.31 shows a rectangular sub-schema mirrored about $\Delta\mathbf{M_3}$ creating a super-schema.

$$RECTSCHEME = \Delta\mathbf{T_2}[\ \ (\Delta\mathbf{T_1}[\ \ (\overline{s})^{<0-1>}])^{<0-1>}]$$

$$SUPERSCHEME = \Delta\mathbf{M_a}[\ (RECTSCHEME)^{<0><1>}]$$

$$= \Delta\mathbf{M_a}[\ (\overline{s}, \Delta\mathbf{T_1}, \Delta\mathbf{T_2})^{<0><1>}]$$



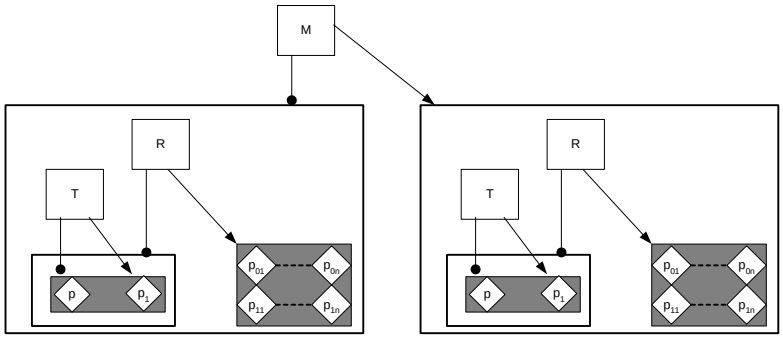


TABLE 4.31 - SCHEMATA ENCAPSULATION

CHAPTER 5

PROPERTIES OF THE ICE REPRESENTATION

The ICE notational string explicitly captures the structure of a configuration as regulators as well as a concise generation method in the order and composition of these regulators together with the applicable transformations in their parameters. Moreover, it also captures, implicitly, additional shape information that can be derived from the notational string by means of simple computations and manipulations of parameters.

In this chapter, I describe the properties and arithmetic of the ICE notation and focus on this category of implicit information and the corresponding methods and strategies by which these are be derived. Through these properties, the ICE representation is analyzed in depth with a focus on the computational significance of every parameter, and its relevance to geometry and to further algorithmic processing. These properties could be easily proven due to their correspondence with the formulae of regulators, but proving these, formally, is beyond the scope of this dissertation.

This chapter is laid out as follows. In Section 5.1, I describe geometric information that can be further derived from ICE strings by means of simple computations of the parameters or generative indices. In Section 5.2, I describe definitions and shape analogies with respect to the regulator representation. Some of these are common definitions adapted to regulators, while others are established specifically for regulators. In Section 5.3, the focus is on the interrelationships among the various regulators, and I deal with the issue of conflict in constraint-based regulators. In Section 5.4, I discuss the property of multiple representations. In Section 5.5, I discuss the transformations from one string to another. Lastly, in Section 5.6, I conclude with a discussion of the design space represented in ICE.

5.1. SHAPE INFORMATION

The ICE notation contains implicit geometrical information about shapes that can be derived through simple computation of parameters, or simple modification of the generated subset. The derivation of shape information, which includes key elements, subshapes and areas/volumes, enables the post processing of particular sub-parts or properties the a shape.

Key points (such as midpoints) or key lines (such as edges), which are not explicitly defined as symbols in the notational string, can be identified through specific treatment of the generated subsets. In this way, such key elements can be directly accessed for further manipulation. For instance, a midpoint or a specific edge can be aligned or a specific edge can be made adjacent to another edge. The identification and direct access of subshapes allows various parts of a shape to be treated differently for example different parts of a circle can be extruded at different heights. Furthermore, the identification of key elements such as vertices and edges, allow for different views of a certain shape such as wire frame or the solid view. The areas/volumes of shapes, which can be determined by multiplying parameters of the regulators, can be constrained for instance to establish a minimum area requirement, or can be used in post processing computations for example in determining amounts of materials or determining costs.

5.1.1. BOUNDARY ELEMENTS AND KEY-ELEMENTS

Key elements and subshapes can be identified by means of strategically manipulating certain parameters and subset definitions of the ICE string. Table 5.1 shows the derivation of endpoints and midpoints with respect to 1D shapes. Table 5.2 and Table 5.3 illustrate these with respect to 2D and 3D shapes, respectively.

A solid line is generated by continuous generation of a single regulator. Its endpoints, which consist of a start point $\bar{s}$ and an endpoint $\bar{e}$, are identified by generating discretely the zeroth element and the nth element of that same regulator.

The midpoint, depicted by $\bar{m}$, is identified by multiplying the factor $(\bar{r}, \theta, \bar{k})$ by $\left(1/2\right)$. Any intermediate point can be generated in this same way using another multiple. The

dotted line is generated by increasing the resolution of the regulator. This is achieved by multiplying the factor by $1/\lambda$ and by multiplying the parameter n by λ.

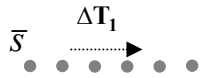
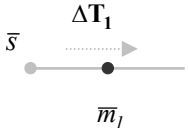
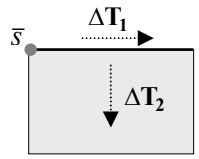
Straight line $line = \Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}\ ]$	
Both endpoints $endpoints = \Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0><1>}\ ]$ One endpoint $\bar{e}_1 = \Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<n>}\ ]$	
Dotted line $dotted = \Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},\frac{1}{\lambda}d,n\}\ (\bar{p})^{<0>-<(\lambda-1)n>}\ ]$	
Midpoint $\bar{m}_1 = \Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},\frac{1}{2}d,n\}\ (\bar{s})^{<n>}\ ]$	

TABLE 5.1 - REPRESENTATIONS OF KEY ELEMENTS IN LINEAR SHAPES

A solid rectangle is generated by continuous generation of two successive regulators. Its vertices are identified by discretely generating the zeroth element and nth element of each of these regulators. The endpoint resulting from the application of a regulator, such as $\Delta\mathbf{T_1}$, is denoted as $\bar{e}_1$; the endpoint resulting from two regulators is denoted as $\bar{e}_{1,2}$. Edges are generated by designating one regulator as continuous and one as discrete. Edges of each direction must be generated separately. The midpoint, depicted by $\bar{m}_{1,2}$, is identified by multiplying the factors $(\bar{t},0,\bar{k})$ of both regulators by $(1/2)$, and the midlines, which are midpoints of one regulator, extended continuously along the other regulator, are denoted as $\bar{m}_1$, and $\bar{m}_2$. Parallel, intermediate lines are generated by one continuous and one discrete regulator for which the resolution is increased.

Rectangle

$$rectangle = \Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}\ ])^{<0-1>}\ ]$$



Vertices

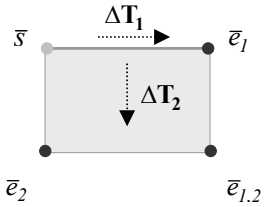
$$vertices = \Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0><1>}\ ])^{<0><1>}\ ]$$

Specific endpoints

$$\bar{e}_1 = \Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<n>}\ ]$$

$$\bar{e}_2 = \Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<n>}\ ]$$

$$\bar{e}_{1,2} = \Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<n>}\ ])^{<n>}\ ]$$

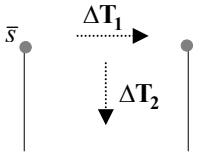


Edges

$$Outline\ edges = horizontals \wedge verticals$$

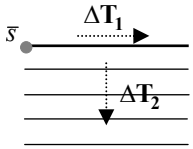
$$horizontals = \Delta\mathbf{T_2}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{T_1}[\{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0><1>}]$$

$$verticals = \Delta\mathbf{T_2}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{T_1}[\{\bar{p},\bar{t},d,n\}(\bar{s})^{<0><1>}])^{<0-1>}]$$



Parallel lines

$$lines = \Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},\frac{1}{\chi}d,n\}($$
$$\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}\ ]$$
$$)^{<0>-<(\chi-1)n>}]$$



Midlines

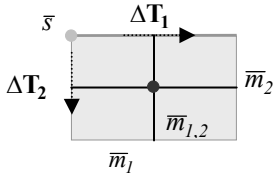
$$midline\ \ \bar{m}_1 =$$

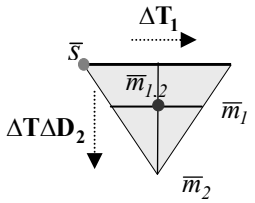
$$\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},\frac{1}{2}d,n\}\ (\bar{s})^{<n>}\ ])^{<0-n>}\ ]$$

Centroid

$$centroid\ \ \ \bar{m}_{1,2}$$

$$= \Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},\frac{1}{2}d,n\}(\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},\frac{1}{2}d,n\}\ (\bar{s})^{<n>}])^{<n>}]$$





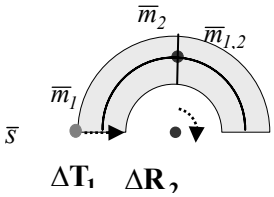
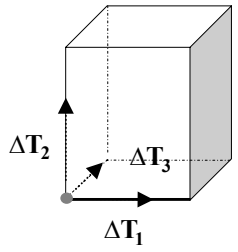


TABLE 5.2 - REPRESENTATIONS OF KEY ELEMENTS IN PLANAR SHAPES

Solid Cuboid

$$cuboid = \Delta\mathbf{T_3}[\ \{\bar{p},\bar{t},d,n\}\ ($$
$$\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ ($$
$$\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}\ ]$$
$$)^{<0-1>}\ ]\ )^{<0-1>}\ ]$$



Vertices

$$vertices = \Delta\mathbf{T_3}[\ \{\bar{p},\bar{t},d,n\}\ ($$
$$\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ ($$
$$\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0><1>}\ ]$$
$$)^{<0><1>}\ ]\ )^{<0><1>}\ ]$$

Endpoints

$$\bar{e}_1 = \Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<n>}\ ]$$

$$\bar{e}_2 = \Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<n>}\ ]$$

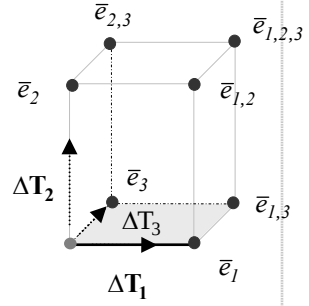
$$\bar{e}_3 = \Delta\mathbf{T_3}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<n>}\ ]$$

$$\bar{e}_{1,2} = \Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<n>}\ ])^{<n>}\ ]$$

$$\bar{e}_{1,2} = \Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<n>}\ ])^{<n>}\ ]$$

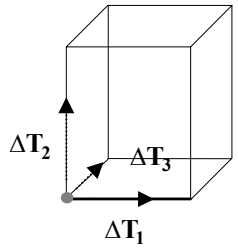
$$\bar{e}_{2,3} = \Delta\mathbf{T_3}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<n>}\ ])^{<n>}\ ]$$

$$\bar{e}_{1,2,3} =$$
$$\Delta\mathbf{T_3}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{T_2}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}(\bar{s})^{<n>}])^{<n>}])^{<n>}]$$



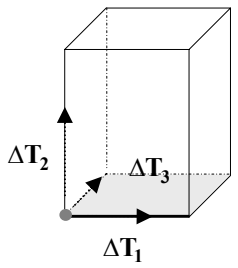
Edges

$$edges1 =$$
$$\Delta\mathbf{T_3}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{T_2}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{T_1}[\{\bar{p},\bar{t},d,n\}(\bar{s})^{<0-1>}])^{<0><1>}])^{<0><1>}]$$
$$edges2 =$$
$$\Delta\mathbf{T_3}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{T_2}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{T_1}[\{\bar{p},\bar{t},d,n\}(\bar{s})^{<0><1>}])^{<0-1>}])^{<0><1>}]$$
$$edges3 =$$
$$\Delta\mathbf{T_3}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{T_2}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{T_1}[\{\bar{p},\bar{t},d,n\}(\bar{s})^{<0><1>}])^{<0><1>}])^{<0-1>}]$$
$$cuboid\ edges = edges1 \wedge edges2 \wedge edges3$$



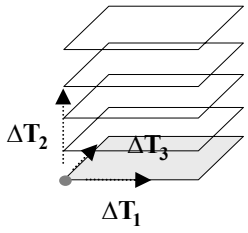
Surfaces

$$surfaces1 =$$
$$\Delta\mathbf{T_3}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{T_2}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{T_1}[\{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}])^{<0><1>}]$$
$$surfaces2 =$$
$$\Delta\mathbf{T_3}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{T_2}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{T_1}[\{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0><1>}])^{<0-1>}])^{<0-1>}]$$
$$surfaces3 =$$
$$\Delta\mathbf{T_3}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{T_2}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{T_1}[\{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0><1>}])^{<0-1>}])^{<0-1>}]$$
$$cuboid\ surfaces = surfaces1 \wedge surfaces2 \wedge surfaces3$$



Parallel planes

$$planes = \Delta\mathbf{T_3}[\ \{\bar{p},\bar{t},\frac{1}{\lambda}d,n\}\ ($$
$$\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ ($$
$$\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{p})^{<0-1>}\ ]$$
$$)^{<0-1>}\ ]$$
$$)^{<0><(\lambda-1)n>}\ ]$$



Mid-surfaces

$midsurface_1\ \ m_1 =$

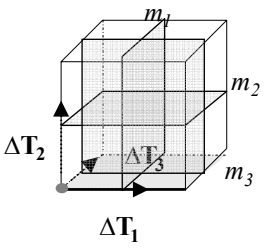
$$\Delta\mathbf{T_3}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},\frac{1}{2}d,n\}\ (\bar{s})^{<n>}\ ]\ )^{<0-n>}\ ]\ )^{<0-n>}$$

$midsurface_2\ \ m_2 =$

$$\Delta\mathbf{T_3}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},\frac{1}{2}d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-n>}\ ]\ )^{<n>}\ ]\ )^{<0}$$

$midsurface_3\ \ m_3 =$

$$\Delta\mathbf{T_3}[\ \{\bar{p},\bar{t},\frac{1}{2}d,n\}\ (\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-n>}\ ]\ )^{<0-n>}\ ]\ )$$



Mid-lines

$midline\ \ \bar{m}_{1,2} =$

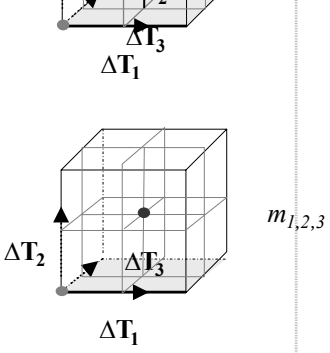
$$\Delta\mathbf{T_3}[\{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},\frac{1}{2}d,n\}(\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},\frac{1}{2}d,n\}(\bar{s})^{<n>}\ ]\ )^{<n>}\ ]\ )^{<0-n>}]$$

$midline\ \ \bar{m}_{1,3} =$

$$\Delta\mathbf{T_3}[\{\bar{p},\bar{t},\frac{1}{2}d,n\}\ (\Delta\mathbf{T_2}[\{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},\frac{1}{2}d,n\}\ (\bar{s})^{<n>}\ ]\ )^{<0-n>}])^{<n>}]$$

$midline\ \ \bar{m}_{2,3} =$

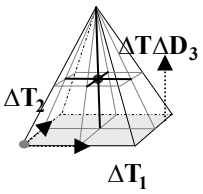
$$\Delta\mathbf{T_3}[\{\bar{p},\bar{t},\frac{1}{2}d,n\}(\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},\frac{1}{2}d,n\}(\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-n>}\ ]\ )^{<n>}])^{<n>}]$$



Centroid

$Centroid_{1,2,3} =$

$$\Delta\mathbf{T_3}[\ \{\bar{p},\bar{t},\frac{1}{2}d,n\}\ (\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},\frac{1}{2}d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},\frac{1}{2}d,n\}\ (\bar{p})^{<n>}$$



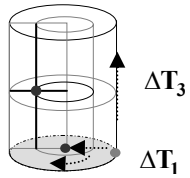


TABLE 5.3 – REPRESENTATION OF KEY ELEMENTS IN VOLUMETRIC SHAPES

A solid cuboid is generated by continuous generation of three successive regulators. Its vertices are identified by discretely generating the zeroth element and nth element of each of these regulators. The endpoints resulting from the application of one regulator are denoted as $\bar{e}_1$, the endpoints of two regulators are denoted as $\bar{e}_{1,2}$, and the endpoint of the three regulators combined is denoted as $\bar{e}_{1,2,3}$.

Edges are generated by designating one regulator as continuous and two regulators as discrete. Edges of each direction are generated separately. Similarly, surfaces (for each direction) are generated by designating two regulators as continuous and one regulator as discrete.

For 3D shapes, there is one centroid, three midlines and three mid-surfaces. The midpoint, depicted by $\bar{m}_{1,2,3}$, is identified by multiplying the factors $(\bar{\imath}, \theta, \bar{k})$ of all three regulators by $\left(1/2\right)$. The midlines, which represent the midpoint of two regulators extended continuously along the third regulator, are denoted as $\bar{m}_{2,3}$. The mid surfaces represent the midpoints of one regulator extended continuously across the two other regulators and denoted as $\bar{m}_1$. Parallel, intermediate, planes are generated by two continuous and one discrete regulator for which the resolution is increased.

In the ICE framework, shapes are not defined by the Cartesian coordinate system. Each shape has its internal coordinate system defined by its generative regulators. These can be polar coordinates or cylindrical, or any other curvilinear configuration determined by the regulator's form. Therefore, elements are identified through the parameters defining these coordinate systems.

5.1.2. SUB-SHAPES

In the ICE framework, subshapes depend on the definition of the shape's regulators. Subshape definition requires an increase in the resolution of the regulators, which is achieved by multiplying the factor by $1/\lambda$ and by multiplying the parameter n by λ. Subshapes, such as the dashed line, the sub-rectangles and the sub-cubes are all defined by means subset generation, where the range i-j (that define the subshape) must be between 0 and n. Table 5.4 shows sample subshapes and their corresponding notation.

Subshape 1D

$sub-line = \Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<i-j>}\ ]$

$(i > 0 \wedge j < n)$

Dashed line

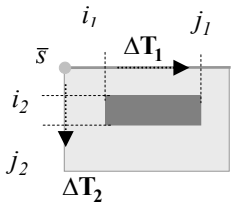
$dotted = \Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-i><j-n>}\ ]$

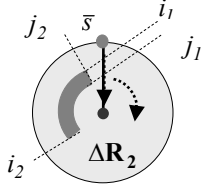
Subshape 2D

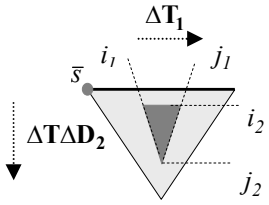
$sub-rectangle =$

$\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<i-j>}\ ])^{<i-j>}\ ]$

$(i_1 > 0 \wedge j_1 < n_1) - (i_2 > 0\ \wedge j_2 < n_2)$





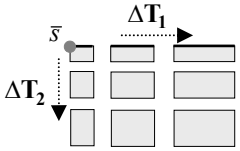


Sub-planes

$subplanes = \Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ ($

$\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-i><j-k><l-n>}\ ]$

$)^{<0-i><j-k><l-n>}\ ]$



Subshape 3D

$sub-volume =$

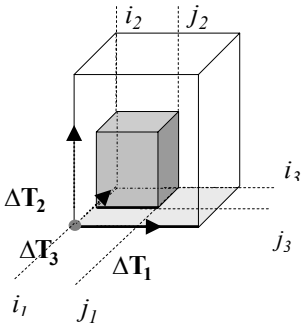
$\Delta\mathbf{T_3}[\ \{\bar{p},\bar{t},d,n\}\ ($

$\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ ($

$\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<i-j>}]$

$)^{<i-j>}])^{<i-j>}]$

$(i_1 > 0 \wedge j_1 < n_1),(i_2 > 0 \wedge j_2 < n_2),(i_3 > 0 \wedge j_3 < n_3)$



Sub-cubes

$subcubes = \Delta\mathbf{T_3}[\ \{\bar{p},\bar{t},d,n\}\ ($

$\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ ($

$\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-i><j-n>}]$

$)^{<0-i><j-n>}]$

$)^{<0-l>}]$

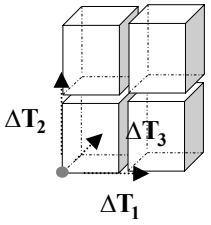


TABLE 5.4 – DERIVING SUBSHAPES FROM THE ICE SHAPES

5.1.3. LENGTHS, AREA, AND VOLUMES

Length and area computations are important in architecture, at various levels, from accommodating space requirements, to determining budgets. The ICE notation enables the computation of lengths, areas, and volumes of shapes by means of multiplying the generative parameters of its regulators. Tables 5.5, 5.6, and 5.7, show the derivation of lengths, areas, and volumes, respectively. Each row focuses on one of the major transformation regulators.

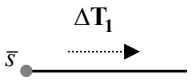
The basic strategy for computing the length of a shape defined by one regulator is to multiply the factor $(\bar{t}, 0, \bar{k})$ with the parameter n, with specific considerations for certain regulators, such as Rotation and Dilation. The length of the curve is determined by integration. The curve is subdivided into small units (by increasing the curve's resolution) and the sum of these units gives the total length of the curve. For composite regulators, the factors of each of the composites are taken into consideration for computing the length. For 2D shapes, each continuous regulator determines the length of one side of the shape.

For determining the area of a shape, the length of the first regulator, $length_1$ is multiplied with that of the second regulator. In case of shapes defined by additional discrete regulators, the area is computed by taking the resulting area of the first two regulators and multiplying it with the number n of the discrete regulator.

For determining the volume, the area of the base shape produced by the first two regulators, $area_{1,2}$, is multiplied by the length produced by the third regulator.

Translation : $\Delta\mathbf{T_1}$ [$\{\bar{p}, \bar{t}, d, n\}$ $(\bar{s})$],

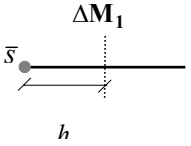
$length = (d_1 \times n_1)$



Mirror : $\Delta\mathbf{M_1}$[$\{\bar{p}, \bar{t}, \bar{v}, n\}$ $(\bar{s})$]

$length = (2 \times h_1)$

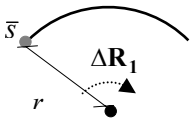
h is the perpendicular distance between $\bar{s}$ and $\Delta\mathbf{M}$



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

$$length = \left(\frac{\theta_1 \times n_1}{360} \times 2 \times \Pi \times r\right)$$

r is the perpendicular distance between $\bar{s}$ and $\Delta\mathbf{R_1}$

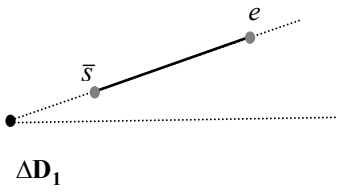


Scale: $\Delta\mathbf{D}^{\mathbf{0}}$[$\{\bar{p}, \bar{k}, n\}$ $(\bar{s})$]

$length = magnitude(\bar{e} - \bar{s})$

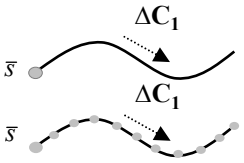
$\bar{s} = (s_x \quad s_y \quad s_z)$

$\bar{e} = (s_x k_x^n - k_x p_x + p_x \quad s_y k_y^n - k_y p_y + p_y \quad s_z k_z^n - k_z p_z + p_z)$



Curve: $\Delta\mathbf{C}^{\mathbf{e}}$[$\{\bar{p}, \bar{t}, \alpha, n\}$ $(\bar{s})$]

$$length = \sum_{i=0}^{n-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2 + (z_i - z_{i+1})^2}$$



COMPOSITION OF REGULATORS

Translation and scale: $\Delta\mathbf{TD}$ [$\{\bar{p}, \bar{t}, d, \bar{k}, n\}$ $(\bar{s})$]

$length = magnitude(\bar{e} - \bar{s})$

$\bar{s} = (s_x \quad s_y \quad s_z)$

$$\bar{e} = \begin{pmatrix} s_x k_x + d_x - k_x p_x + p_x \\ s_y k_y + d_y - k_y p_y + p_y \\ s_z k_z + d_z - k_z p_z + p_z \end{pmatrix}$$

Screw Rotation: $\Delta\mathbf{T}\Delta\mathbf{R}$ [$\{\bar{p}, \bar{t}, d, \theta, n\}$ $(\bar{s})$]

$$length = \sqrt{\left(\frac{\theta_1}{360} \times 2 \times \Pi \times r\right)^2 + (d)^2} \times n$$

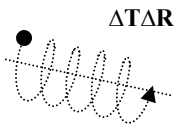


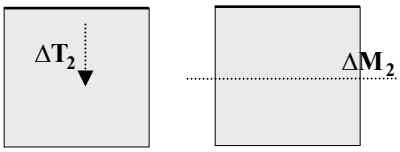
TABLE 5.5 – LENGTHS COMPUTATIONS

Translation

$$area_{1,2} = length_1 \times (d_2 \times n_2)$$

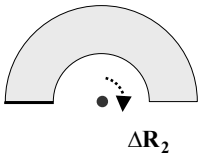
Mirror

$$area_{1,2} = length_1 \times (2 \times h_2)$$



Rotation

$$area_{1,2} = \frac{\theta_2 \times n_2}{360} \times \Pi \times \left(length_1{}^2 + 2 \times (length_1) \times r_2\right)$$

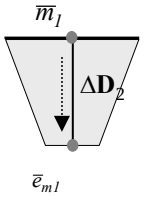


Scale

$$area_{1,2} = length_1 \times magnitude(\bar{e}_{m1} - \bar{m}_1) \times \left(k_{x2}{}^n \times k_{y2}{}^n \times k_{z2}{}^n\right)$$

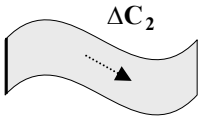
$$\bar{m}_1 = \begin{pmatrix} m_x & m_y & m_z \end{pmatrix}$$

$$\bar{e}_{m1} = \begin{pmatrix} m_x k_x^n - k_x p_x + p_x & s_y k_y^n - k_y p_y + p_y & s_z k_z^n - k_z p_z + p_z \end{pmatrix}$$



Curve

$$area_{1,2} = length_1 \times \sum_{i=0}^{n-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2 + (z_i - z_{i+1})^2}$$



Shape defined by discrete generation

$$total\_area = unit\_area \times n$$

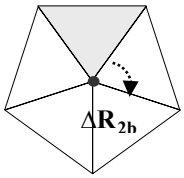
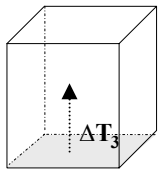


TABLE 5.6 – AREA COMPUTATIONS

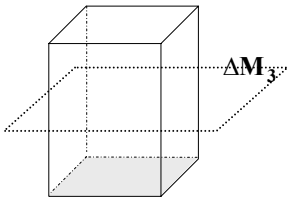
Translation

$$volume_{1,2,3} = area_{1,2} \times (d_3 \times n_3)$$



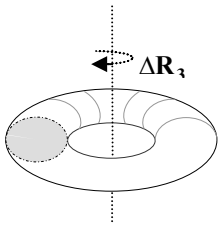
Mirror

$$volume_{1,2,3} = area_{1,2} \times (2 \times h_3)$$



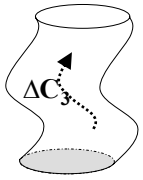
Rotation

$$volume_{1,2,3} = \frac{\theta_3 \times n_3}{360} \times \Pi \times \left(area_{1,2}{}^2 + 2 \times area_{1,2} \times r_3\right)$$



Curve

$$volume_{1,2,3} = area_{1,2} \times \sum_{i=0}^{n-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2 + (z_i - z_{i+1})^2}$$



Scale

$$volume_{1,2,3} = area_{1,2} \times magnitude(\bar{e}_{m1} - \bar{m}_1) \times \left(k_{x3}{}^n \times k_{y3}{}^n \times k_{z3}{}^n\right)$$

$$\bar{m}_1 = (m_x \quad m_y \quad m_z)$$

$$\bar{e}_{m1} = \left(m_x k_x^n - k_x p_x + p_x \quad s_y k_y^n - k_y p_y + p_y \quad s_z k_z^n - k_z p_z + p_z\right)$$

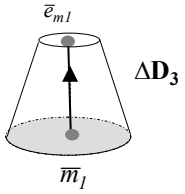


TABLE 5.7 – VOLUME COMPUTATIONS

5.2. DEFINITIONS AND ANALOGIES

In this section, I present known geometric definitions and shape analogies with respect to the regulator representation with an emphasis on the relevant parameters. What does it mean for two ICE notational strings to be equivalent for instance? Are they any conditions subsumed in the notation? And, are additional conditions necessary? Are their subclasses of equivalence based on regulators? Such definitions are particularly important for understanding, in depth, the regulator representation, for defining the structure of regulator interrelationships, and not least, for defining the fundamentals of test, or for further algorithmic computations.

5.2.1. EQUALITY AND EQUIVALENCES

Besides equality (Figure 5.1), the ICE notation supports several types of equivalences: directional-equivalence, factor-equivalence, and continuity-equivalence for regulators (Figure 5.2); as well as distance-equivalence, angular-equivalence, and proportional-equivalence for shapes (Figure 5.3).

Regulator equality: Two regulators, $\Delta\mathbf{T_a}$ and $\Delta\mathbf{T_b}$, are equal whenever (i) they are of the same type, (ii) their parameters $\bar{t}_a$, d_a, n_a and $\bar{t}_b$, d_b, n_b are respectively equal, and (iii) the continuity pattern of $\Delta\mathbf{T_a}$ is identical that of $\Delta\mathbf{T_b}$. The parameter $\bar{p}$ denoting the starting point of the regulator needs not to be equal. Additionally, in order to achieve equality, curvilinear regulators need to have same formula.

Shape equality: $shape_A$ and $shape_B$ are equal, whenever (i) their defining regulators $\Delta\mathbf{T_{A1}} - \Delta\mathbf{T_{An}}$ and $\Delta\mathbf{T_{B1}} - \Delta\mathbf{T_{Bn}}$ are respectively equal and (ii) the distances between regulators and the starting point $\bar{s}$ of the shapes are respectively equal. Note that this definition of shape equality does not address multiple representations.

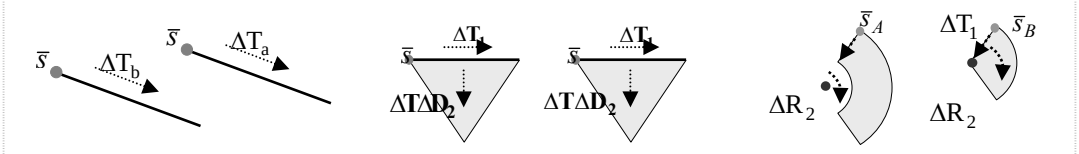


FIGURE 5.1 - EQUAL REGULATORS, EQUAL SHAPES, AND UNEQUAL SHAPES

Directional-equivalence: Two regulators, $\Delta\mathbf{T_a}$ and $\Delta\mathbf{T_b}$ are directionally-equivalent if (i) they are of the same type, and (ii) their directional parameter $\bar{t}_a$ and $\bar{t}_b$ are equal. Since the formula defines the direction for curvilinear regulators, their formula must be identical.

Factor-equivalence: The regulators $\Delta\mathbf{T_a}$ and $\Delta\mathbf{T_b}$ are factor-equivalent whenever (i) they are of the same type, and (ii) their distance factors d are equal. Factors include the angle θ for Rotation and scaling $\bar{k}$ for Dilation regulators.

Continuity-equivalence: The regulators $\Delta\mathbf{T_a}$ and $\Delta\mathbf{T_b}$ are continuity-equivalent whenever (i) they are of the same type and (ii) the continuity pattern of $\Delta\mathbf{T_a}$ is identical to the pattern in $\Delta\mathbf{T_b}$.

Inverted-equivalence: Two regulators, $\Delta\mathbf{T_a}$ and $\Delta\mathbf{T_b}$, are inverse-equivalent to one another, whenever they are of the same type, and the parameters $\bar{t}_a$ and $\bar{t}_b$ are equal, but the factors d_a and d_b are opposites (one is the negative of the other).

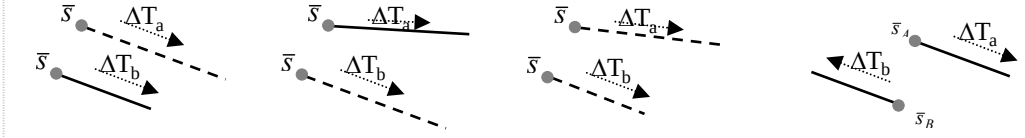


FIGURE 5.2 – DIRECTIONAL, FACTOR, CONTINUITY, AND INVERTED EQUIVALENCE

Shape equivalence: $shape_A$ and $shape_B$ are equivalent, whenever (i) their defining regulators $\Delta\mathbf{T_{A1}} - \Delta\mathbf{T_{An}}$ and $\Delta\mathbf{T_{B1}} - \Delta\mathbf{T_{Bn}}$ are respectively equivalent. Shapes maintain the same geometric integrity and preserve angles if their regulators are directionally-equivalent; they maintain the same dimensions if their regulators are factor-equivalent; and they maintain the same continuity patterns if their regulators are continuity-equivalent. Shape schema describes shapes that are continuity equivalent.

Distance-equivalence: $shape_A$ and $shape_B$ are distance-equivalent whenever the distances between the starting points $\bar{s}$ of the shapes and their regulators are respectively equal.

Angular-equivalence: $shape_A$ and $shape_B$ are angular-equivalent whenever the angles between the constituent regulators for each shape are equal.

Proportional-equivalence: $shape_A$ and $shape_B$ are proportionally-equivalent whenever the regulator factors and the distance between the starting point $\bar{s}$ and the regulators are proportionally related.

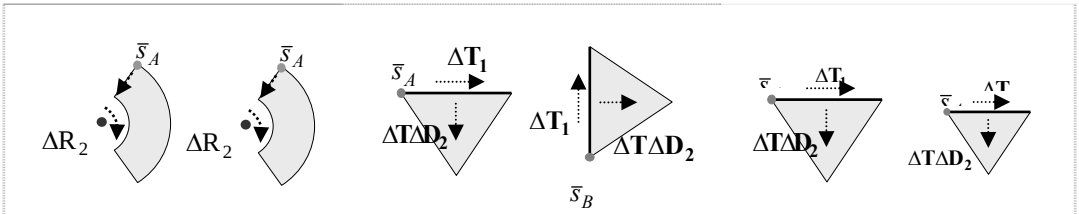


FIGURE 5.3 – DISTANCE, ANGULAR , AND PROPORTIONAL EQUIVALENCE

These definitions of shape and regulator equivalences do not address multiple representations.

5.2.2. COINCIDENCE AND EXTENSION

Coincidence in ICE depends on regulator operations. A point $\bar{q}$ is coincident on $shape_A$, whenever (i) $\bar{q}$ can be generated by $shape_A$'s defining regulators using its starting point $\bar{s}_A$, and (ii) $\bar{q}$ is within the range defined by $<0-n>$ of $shape_A$. This is achieved by means of a multiple applied to the main factor of the regulator. If $\bar{s}_A$ can be obtained by applying the inverse of the regulators' matrices to $\bar{q}$ with the factor multiple λ between 0 and 1, then $\bar{q}$ is coincident on $shape_A$. If $shape_A$ is discontinuous, the factor must be verified with respect to the continuity patterns of its regulators.

Table 5.8 shows coincidence as it applies to linear, planar and volumetric shapes. For linear shapes, coincidence describes points anywhere on the line. If the factor multiple λ_1 equals 0, then the point $\bar{q}$ is coincident to the start point and if the multiple equals 1, it's

coincident to the endpoint. For planar shapes, coincidence describes points that are either inside the plane or on its boundary. If the both multiples λ_1 and λ_2 equal 0, then the point $\bar{q}$ is coincident to the start point, if one multiple equals 0 or equals 1 then the point $\bar{q}$ is on the boundary. For volumetric shapes, coincidence describes points inside the shape, on its surfaces, edges or vertices. If the all three multiples λ_1, λ_2 and λ_3 equal 0, then the point $\bar{q}$ is coincident to the start point, if one or two multiple equal zero or equal one then the point $\bar{q}$ is on an edge of the volume, if one multiple equals 0 or equals 1, then the point $\bar{q}$ is on the surface of the volume.

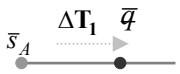
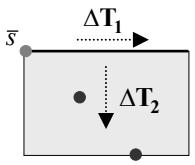
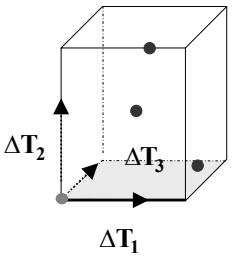
Coincidence on linear shapes $line = \Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},\lambda d,n\}\ (\bar{s})^{<0-1>}]$	
Coincidence on planar shapes $rectangle = \Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},\lambda d,n\}\ ($ $\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},\lambda d,n\}\ (\bar{s})^{<0-1>}]$ $)^{<0-1>}]$	
Coincidence on volumetric shapes $cuboid = \Delta\mathbf{T_3}[\ \{\bar{p},\bar{t},\lambda d,n\}\ ($ $\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},\lambda d,n\}\ ($ $\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},\lambda d,n\}\ (\bar{s})^{<0-1>}]$ $)^{<0-1>}]$ $)^{<0-1>}]$	

TABLE 5.8 – LINEAR, PLANAR , AND VOLUMETRIC COINCIDENCE

Internal-coincidence: A point $\bar{q}$ is internal-coincident on $shape_A$ (linear, planar, or volumetric) whenever its entire factor multiples λ are greater than 0 and less than 1.

Boundary-coincidence: A point $\bar{q}$ is boundary-coincident on $shape_A$ (linear, planar, or volumetric) whenever its one of its factor multiples λ is equal to 0 or 1.

Primary-coincidence: A point $\bar{q}$ is primary-coincident on $shape_A$ whenever it can be generated by only one of its regulator. For a linear shape, coincidence is always primary. For planar shapes, primary-coincidence is contingent on one of λ multiples to be equal

to 0. For volumetric shapes, primary-coincidence is contingent on two of λ multiples to be equal to 0.

Secondary-coincidence: A point $\bar{q}$ is secondary-coincident on $shape_A$ whenever it can be generated by only two regulators. For planar shapes, coincidence is secondary if none of the λ multiples are equal to 0. For volumetric shapes, secondary-coincidence is contingent on one λ multiples to be equal to 0.

Directional-coincidence: A regulator $\Delta\mathbf{T_a}$ is directionally coincident on $shape_A$ whenever (i) $\Delta\mathbf{T_a}$'s starting point $\bar{p}$ is on the boundary of $shape_A$ and (ii) $\Delta\mathbf{T_a}$ generates points that are inside $shape_a$.

Extension: A point $\bar{q}$ is an extension of a $shape_A$, whenever $\bar{q}$ can be generated by $shape_A$'s defining regulators using its starting point $\bar{s}_A$, and $\bar{q}$ extends beyond the range defined by $<0-n>$ of $shape_A$. The factor multiple λ_1 can be negative or can be greater than 1. Extension can be primary or secondary as illustrated in Figure 5.4.

Primary-extension: Primary-extensions are defined by a single regulator. Linear shapes can only describe primary extensions. For planar shapes, a primary extension is defined if one of the λ multiples is equal to 0. For a volumetric shape to define a primary extension, two of the λ multiples must equal 0.

Secondary-extension: Secondary extensions are defined by two regulators. Planar shapes define secondary extension if none of the λ multiples are zero, and volumetric shapes define secondary extensions if only one of the λ multiples equals zero.

FIGURE 5.4 – PRIMARY AND SECONDARY EXTENSIONS

5.2.3. COINCIDENCE-BASED RELATIONS AND OPERATIONS

Complex relations and operations can be simply defined by means of coincidence with respect to regulators. These include collinearity, coplanarity, connectedness. Computing intersections and are illustrated in Figures 5.5 and 5.6.

Collinear: Two linear shapes, $shape_A$ and $shape_B$, are collinear whenever (i) their defining regulators are directionally-equivalent and (ii) the starting point of $shape_B$ is primary-coincident or a primary-extension with respect to $shape_A$. This definition is applicable to curvilinear lines as well.

Coplanar: Two planar shapes, $shape_A$ and $shape_B$, are coplanar whenever (i) their defining regulators are respectively directionally-equivalent, and (ii) the starting point of $shape_B$ is primary (or secondary) coincident, or a primary (or secondary) extension with respect to $shape_A$. This definition is also applicable to curvilinear surfaces.

Connectedness: Two shapes, $shape_A$ and $shape_B$, are connected whenever the starting point of $shape_B$ (or is coincident to) $shape_A$. The shapes are ***primary-connected*** if the starting point of $shape_B$ is primary-coincident to $shape_A$

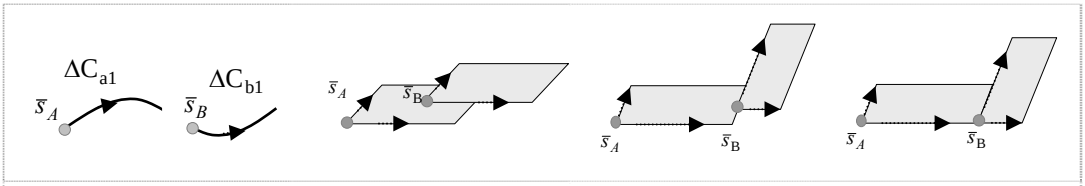


FIGURE 5.5 – COLLINEAR, COPLANAR, CONNECTED, AND PRIMARY CONNECTED

Point of intersection: A point of intersection of two linear shapes $shape_A$ and $shape_B$, is a point $\bar{q}$ which is coincident on both $shape_A$ and $shape_B$. This means that it can be generated by regulators of $shape_A$ using the start-point $\bar{s}_A$ and by regulators of $shape_B$ using the starting point $\bar{s}_B$.

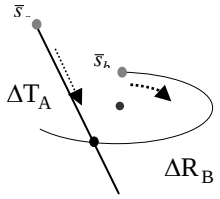
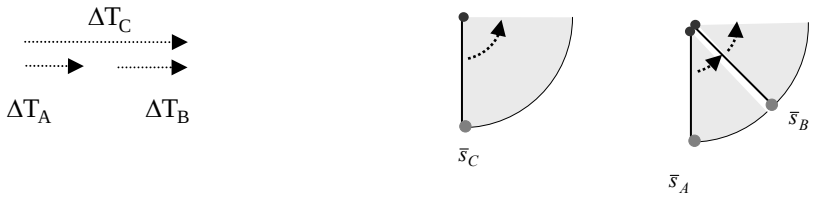


FIGURE 5.6 – POINTS OF INTERSECTION

5.2.4. MAXIMAL AND SUBSHAPE

Maximal regulators: A regulator $\Delta\mathbf{T_C}$ is the maximal of the regulators $\Delta\mathbf{T_A}$ and $\Delta\mathbf{T_B}$ whenever (i) all three regulators are directionally and continuity-equivalent and (ii) the factor $(d,\theta,\bar{k})$ multiplied by the parameter n of $\Delta\mathbf{T_C}$ equals the sum of the factor $(d,\theta,\bar{k})$ multiplied by the parameter n of $\Delta\mathbf{T_A}$ and those of $\Delta\mathbf{T_B}$. A maximal regulator can replace several regulators and produce a maximal shape. Figure 5.7 shows maximal regulators and shapes.

Maximal shape: $shape_C$ is the maximal of the shapes $shape_A$ and $shape_B$, whenever (i) at least one set of corresponding regulators are equivalent, (ii) the regulator of $shape_C$ is the maximal of the regulator of $shape_A$ and $shape_B$, (iii) $shape_A$ is primary-connected to $shape_B$ (along the equivalent regulator) (iv) $shape_A$ and $shape_B$ are subshapes of $shape_C$ and (v) the start-point of $shape_C$ is coincident with the start-point of $shape_A$, and (vi) the endpoint of $shape_C$ is coincident with the endpoint of $shape_B$.



a b

FIGURE 5.7 – MAXIMAL REGULATOR AND MAXIMAL SHAPE

Sub-regulator: $\Delta\mathbf{T_B}$ is a sub-regulator of $\Delta\mathbf{T_A}$ whenever (i) both are directionally and continuity equivalent, (ii) the factor $(d,\theta,\bar{k})$ multiplied by the parameter n of $\Delta\mathbf{T_B}$ is less than the factor $(d,\theta,\bar{k})$ multiplied by the parameter n of $\Delta\mathbf{T_A}$. Figure 5.8 shows sub-regulators and subshapes.

Subshape: $shape_B$ is a sub-shape of $shape_A$ whenever (i) their corresponding defining regulators are sub-regulators and (ii) the starting point and the endpoints $shape_B$ are coincident or internally-coincident on $shape_A$.

FIGURE 5.8 – SUB-REGULATOR AND SUBSHAPE

5.3. REGULATOR INTERRELATIONSHIPS

The categories of regulators (transformational, variational, operational, constraints, hierarchical, topological) are significantly different in their technique for regulating shapes. These regulators can be further categorized into three sets: the primary generative set, the secondary constructive set, and the ternary relational set. The primary generative regulators, consisting of transformation regulators, define the geometry. The secondary set, consisting of variational and operational regulators affect the geometry and are used in conjunction with the primary set to create complex forms. The ternary set, consisting of constraint, hierarchical, and topological regulators, define relationships and establish order among shapes. These are applied to shapes defined by the primary and secondary sets. Figure 5.9 shows these sets of regulators and their interrelationships.

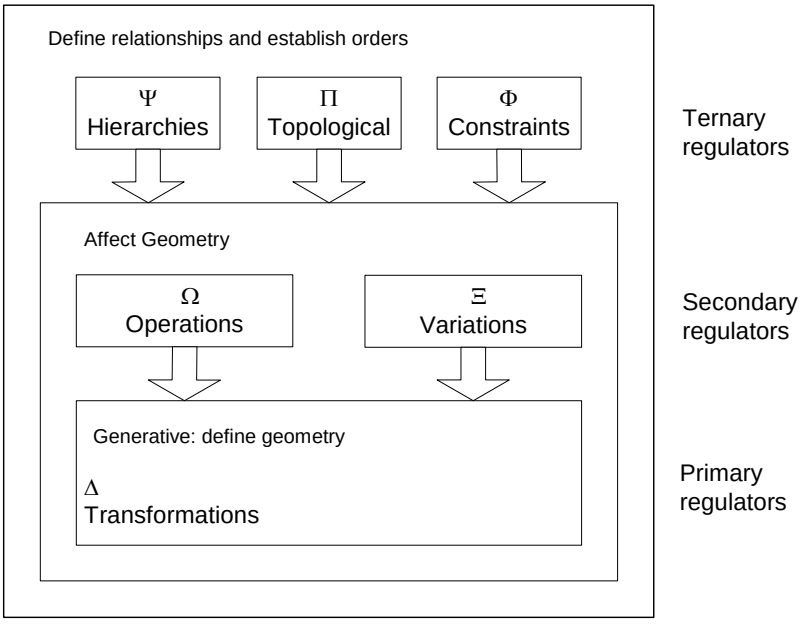


FIGURE 5.9 - REGULATOR INTERRELATIONSHIPS

Since the secondary and ternary sets of regulators are applied to shapes defined by primary regulators, they are in fact controlling parameters of generative regulators. Understanding regulator interrelationship in detail is significant for understanding the consequences of applying regulators; it is critical for the identifying conflicts within configurations and for system implementation. In this section, I describe how secondary and ternary regulators control primary regulators.

5.3.1. VARIATIONAL REGULATORS

Variational regulators are composed simultaneously with transformation regulators and thus control their transformational factors.

The ***Exception*** regulator, $\Xi\mathbf{E}$, affects an output shape by making it non-responsive to the transformation regulator's influence, therefore, creating an exception to the set of outputs.

The ***Rhythm/Gradation*** regulator, $\Xi\mathbf{G}$, creates a variation by multiplying a coefficient, c, to an attribute of the output shape/points, or to the factor $(d,\theta,\bar{k})$ of its composed generative regulator as is applied to the output shapes/points.

The ***Differential*** regulator, $\Xi\mathbf{F}$, creates a variation by multiplying a coefficient c to the factor $(d,\theta,\bar{k})$ of its composed generative regulator as it is applied to various input shapes.

5.3.2. CONSTRAINT REGULATORS

When constraint regulators are applied to a shape, they are actually applied to the parameters of the generative regulators of this shape. Constraint regulators restrict parameters to a maximum or minimum value, or to a value defined by an incremental module. Constraint regulators are applicable to one or many shapes at a time.

The attribute ***Equivalence*** regulator, $\Phi\mathbf{Q}$, controls shape attributes. These include the factor $(d,\theta,\bar{k})$, the parameter n and the direction vector $\bar{t}$ of any of the generative regulators. This regulator is also applicable to non-geometric attributes of shapes.

The ***Alignment*** regulator has several variations: $\Phi\mathbf{A}^0$, $\Phi\mathbf{A}^1$, and $\Phi\mathbf{A}^2$. $\Phi\mathbf{A}^0$ restricts the starting point $\bar{s}$ (or any key-point $\bar{k}$) of the shape to be coincident to an alignment point $\bar{p}$. $\Phi\mathbf{A}^1$ restricts the point $\bar{k}$ and the directional vector $\bar{t}$ (of one the defining regulators of the shape). $\bar{k}$ becomes coincident to the alignment line (defined by $\bar{p}$ and $\bar{t}$) and $\bar{t}$ becomes directionally-equivalent to the vector $\bar{t}$ of the alignment regulator. $\Phi\mathbf{A}^2$ restricts the point $\bar{k}$ and the vector $\bar{t}$ (of two the defining regulators). $\bar{k}$ must be coincident to the alignment plane (defined by $\bar{p}$, $\bar{t}$, and $\bar{v}$) and $\bar{t}$ must be coplanar to

the plane defined by the vectors $\bar{t}$ and $\bar{v}$ of the alignment regulator. $\Phi\mathbf{A}^{c}$ restricts the point $\bar{k}$ and the vector $\bar{t}$ of two the defining regulators to align with the tangent and radius of the circle. The desired key-point and regulators needs to be specified, otherwise, the alignment will apply to the starting point, $\bar{s}$, and the first regulator $\Phi\mathbf{T}_{1}$.

- $\Phi\mathbf{A}^{0}$[$\{\bar{p}\}$ (shape$_{\mathrm{A}}\langle m_{1}\rangle$)]
- $\Phi\mathbf{A}^{1}$[$\{\bar{p},\bar{t}\}$ (shape.$\langle\bar{s},\Delta\mathbf{T}_{3}\rangle$)]
- $\Phi\mathbf{A}^{2}$[$\{\bar{p},\bar{t},\bar{v}\}$ (shape$\langle\bar{s},\Delta\mathbf{T}_{1},\Delta\mathbf{T}_{2}\rangle$)]
- $\Phi\mathbf{A}^{c}$[$\{\bar{p},\bar{t},r\}$ (shape$\langle\bar{s},\Delta\mathbf{T}_{1},\Delta\mathbf{T}_{2}\rangle$)]

The ***Size*** regulator, $\Phi\mathbf{V}$, restricts the lengths of the shape by means of restricting the factor $(d,\theta,\bar{k})$ and the parameter n of the constituent regulator. $\Phi\mathbf{V}^{\mathbf{1}}$ applies to one regulator, thus restricting the length, $\Phi\mathbf{V}^{\mathbf{2}}$ applies to two regulators, thus restricting the area, and $\Phi\mathbf{V}^{\mathbf{3}}$ applies to three regulators, thus restricting the volume.

- $\Phi\mathbf{V}^{1}$[$\{min,max,mod\}$ (shape$\langle\Delta\mathbf{T}_{2}\rangle$)]
- $\Phi\mathbf{V}^{2}$[$\{min,max,mod\}$ (shape$\langle\Delta\mathbf{T}_{1},\Delta\mathbf{T}_{3}\rangle$)]
- $\Phi\mathbf{V}^{3}$[$\{min,max,mod\}$ (shape)]

The ***Angle*** regulator restricts the angle between two directional vectors $\bar{t}$ of constituent regulators. These two regulators can be pertaining to a single shape, or these can be pertaining to two distinct shapes.

- $\Phi\mathbf{L}$ [$\{min,max,mod\}$ (shape$\langle\Delta\mathbf{T}_{1},\Delta\mathbf{T}_{3}\rangle$)]
- $\Phi\mathbf{L}$ [$\{min,max,mod\}$ (shape$\langle\Delta\mathbf{T}_{1}\rangle$, shape$\langle\Delta\mathbf{T}_{3}\rangle$)]

The ***Proportion*** regulator restricts the aspect ratio of a shape determined by the factor $(d,\theta,\bar{k})$ and the parameter n of two (or three) of its defining regulators. $\Phi\mathbf{P}^{1}$[$\{\bar{p},\bar{t},d\}$ (shape$\langle\Delta\mathbf{T}_{1},\Delta\mathbf{T}_{3}\rangle$)]

5.3.3. TOPOLOGICAL REGULATORS

Topological regulators are mostly binary regulators that establish a relationship between two shapes, $shape_A$ and $shape_B$, by controlling their key points.

The ***Distance*** regulator, $\Pi\mathbf{J}^+$, restricts a key-point from $shape_A$ and a key-point from $shape_B$ to be within a specific distance. The Adjacency regulator, $\Pi\mathbf{J}$, restricts a key-point or key element of $shape_A$ and a key-point of $shape_B$ to be coincident. The Overlap regulator, $\Pi\mathbf{J}^-$, restricts at least one key-point of $shape_B$ to be internally-coincident $shape_A$.

- $\Pi\mathbf{J}^+[\ \{\}\ (\text{shape}_\text{A}\langle\bar{e}_{1,2}\rangle, \text{shape}_\text{B}\langle\bar{s}\rangle)\ ]$
- $\Pi\mathbf{J}^0\ [\ \{\}\ (\text{shape}_\text{A}\langle\bar{e}_{1,2}\rangle, \text{shape}_\text{B}\langle\bar{s}\rangle)\ ]$
- $\Pi\mathbf{J}^-\ [\ \{\}\ (\text{shape}_\text{A}, \text{shape}_\text{B}\langle\bar{s}\rangle)\ ]$

The ***Boundary*** regulator restricts the start-point, endpoint and all key-points of bounded shape to be internally-coincident the boundary shape.

- $\Phi\text{B}^2[\ \{o\}\ (\text{shape}_{\text{boundary}}, \text{shape}_1)]$

The ***Connected*** regulator, $\Pi\mathbf{C}$, restricts a key-point of $shape_A$ to be coincident with a key-point of $shape_B$.

- $\Pi\mathbf{C}\ [\ \{\}\ (\text{shape}_\text{A}\langle\bar{e}_{1,2}\rangle, \text{shape}_\text{B}\langle\bar{s}\rangle)\ ]$

5.3.4. HIERARCHICAL REGULATORS

Hierarchical regulators establish order between shapes.

The ***Containment*** regulator, $\Psi\mathbf{H}$, is independent of geometry therefore does not affect the parameters of the generative regulators.

The ***Subshape*** regulator, $\Psi\mathbf{S}$, ensures that the following conditions are always satisfied (i) the defining regulators of the shapes are directionally-equivalent by controlling their directional parameter $\bar{t}$ (or formula), (ii) the start point of the Subshape is internally

coincident or coincident with respect to the Super-shape, and (iii) the factor $(d,\theta,\bar{k})$ multiplied by the parameter n of the each regulator in the Subshape is always less than the corresponding ones of the Super-shape:

- superShape $= \Delta\mathbf{R_2}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}\ )^{<0-n>}\ ])^{<0-n>}\ ]$
- subShape $= \Delta\mathbf{R_2}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}\ )^{<0-n>}\ ])^{<0-n>}\ ]$

5.3.5. OPERATION REGULATORS

The operation regulators apply a discrete operation to the input set, and create a resultant set of output shapes.

The ***Subdivision regulator***, $\Omega\mathbf{Z}$, divides a shape into sequentially connected subshapes (Table 5.9a). In order to subdivide a shape, it is necessary to subdivide at least one of its defining regulators. The original regulator is a maximal with respect to its sub-regulators; similarly the original shape is the maximal with respect to its subshapes.

The regulator $\Delta\mathbf{R_A}$ of the original $shape_A$ will be replaced by n_{sub} directionally equivalent sub-regulators $\Delta\mathbf{R_B}$, $\Delta\mathbf{R_C}$, etc. The n parameter of each sub-regulator remains the same as the n parameter of $\Delta\mathbf{R_A}$. The factor $(d,\theta,\bar{k})$ of $\Delta\mathbf{R_A}$ is scaled according to the number of subdivisions. For each subshape, a new start point will be created at the position determined by the scale factor, and the remaining regulators (which are not subdivided) will be duplicated.

Another form of subdivision by means of the ***Cutting*** regulator, $\Omega\mathbf{C}$, is achieved by a cutting line or cutting surface, which is not directionally-equivalent to the regulators of the shape (Table 5.9b). The points of intersection between the shape and the plane are identified and a new shape is created. Its starting point is determined by the points of intersection with the plane, and its regulators are directionally-equivalent to original regulator. A Differential variation regulator is applied to each of the subdivided regulators to create the slanted effects in the subdivided shapes.

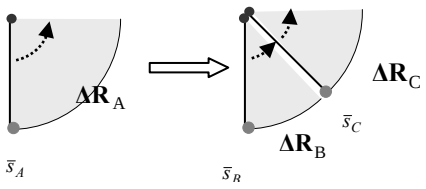
The ***Merging*** regulator is the inverse of the subdivision regulator (Table 5.9c). It replaces a primary-connected shape with their maximal representation. To merge a set of

connected shapes, it is necessary to merge at least one corresponding set of their defining regulators. These must be directionally-equivalent.

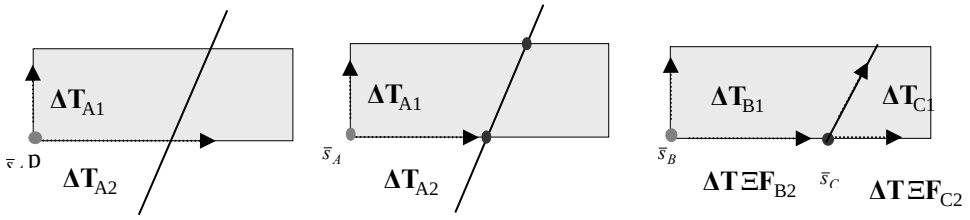
The original regulators $\Delta\mathbf{T}_A$ and $\Delta\mathbf{T}_B$ are replaced by their directionally-equivalent maximal regulator $\Delta\mathbf{T}_C$. The n parameter of $\Delta\mathbf{T}_C$ is the sum of the parameter n of the original regulators. The factor $(d,\theta,\bar{k})$ of the maximal $\Delta\mathbf{T}_C$ is the average of the factor of the original regulators, $\Delta\mathbf{T}_A$ and $\Delta\mathbf{T}_B$. Start-points and regulators of the original shapes will be deleted. The starting point of the first shape is the starting point of the maximal shape, and the endpoint of the last shape is the ending point of the maximal shape.

a - Subdivision

$$\Delta\mathbf{R_A}\{n\} = \Delta\mathbf{R_B}\{n\} = \Delta\mathbf{R_C}\{n\}$$
$$\Delta\mathbf{R_A}\{\theta\} = \Delta\mathbf{R_B}\{\theta\} + \Delta\mathbf{R_C}\{\theta\}$$



b- Subdivision by plane



c - Merging

$$\Delta\mathbf{T_C}\{n\} = \Delta\mathbf{T_A}\{n\} + \Delta\mathbf{T_B}\{n\}$$
$$\Delta\mathbf{T_C}\{d\} = \frac{\Delta\mathbf{T_A}\{d\} + \Delta\mathbf{T_B}\{d\}}{2}$$

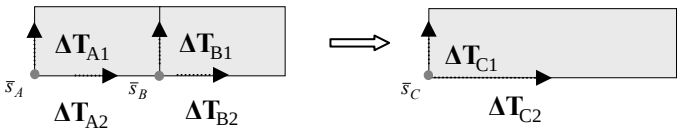


TABLE 5.9 – SUBDIVISION AND MERGING REGULATORS

The ***Boolean operation*** regulators control the generated subsets of input shapes to create a new output shape. The Boolean regulator determines the resultant shapes by (i) identifying boundary vertices of intersected shape, (ii) increasing the resolution of the inputs, and (iii) generating the output based on subsets of the original input shape.

The vertices bounding the intersected shape are identified by computing the points of

intersection between the input shapes and computing the key-points of $shape_A$ that are coincident in $shape_B$ and vice versa. The output shape is generated by increasing the resolution of the input shapes, and as the output shape is generated, each intermediate point is tested to determine its coincidence with $shape_A$ and $shape_B$.

- Union: All points internally-coincident to shape A, or internally-coincident to shape B, are generated.
- Intersection: All points internally-coincident to shape A, and internally-coincident to shape B, are generated.
- Difference: All points internally-coincident to shape A, and not coincident to shape B, are generated.

The linear boundary of the intersected shape is constructed by taking each boundary vertex as a starting point of the intersected shape and applying a regulator from the input shape. The applicable regulators are determined by testing the boundary vertex with all the regulators. If the regulator test generates line with is directionally-coincident to both shapes, the regulator is applied to create a valid boundary line. If there are directionally-equivalent (or inverted-coincident) regulators, only one will be considered per bounding point. The process is illustrated in Figure 5.10.

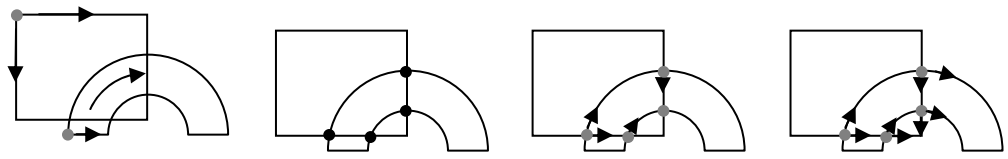


FIGURE 5.10 - GENERATION OF THE LINEAR BOUNDARY OF THE INTERSECTION

5.3.6. CONFLICT IDENTIFICATION

Since the ICE notation supports constraints among its many regulator types, there is potential for conflicting situations. This is particularly the case in an implementation environment intended for design exploration where conflicts can cause annoying interruptions. From the perspective of the ICE framework, conflicts can occur when

multiple regulators control a common shape (or point), each restricting its attributes in a different and opposing way.

ICE notational strings contain explicit information about constraints and constrained shapes, and therefore, these strings contain implicit information about possible conflicts for every constrained shape. In this section, I present a strategy for identifying conflicts, and the parameters involved, from an ICE notational string.

Constraints on shapes can be categorized as positional, directional or dimensional or a combination. Positional constraints control the starting point $\bar{s}$ of a shape or the starting point $\bar{p}$ of a regulator. Directional constraints control the vector $\bar{t}$ of one (or many) of the constituent regulators. Dimensional constraints affect the factor $(d, \theta, \bar{k})$ of one the constituent regulators.

The following steps constitute the strategy for identifying conflicts as well as the shapes and parameters involved, by analyzing the notational string:

- Identify the constraining regulators (These include topological and hierarchical regulators).
- Identify the constraint type, and the parameters that it typically controls.
- Identify constrained shapes, their constrained key-points and parameters.
- Identify other regulators controlling the constrained shape.
- Identify possible transformations that may change the constrained key-points and parameters (This includes regulator transformation or user manipulations)

Table 5.10 shows examples of simple conflicts that can be identified using this strategy.

The first example shows a Rotation regulator controlling a set of bounded shapes. Potential conflicts can occur if the shapes are moved beyond their boundary. The conflict causing parameters are identified as follows.

- Positional constraint regulator: **ΦB**
- Constrained shapes: $shape_0 - shape$
- Constrained key-points: $\text{shape}_0 \langle \bar{s}_0, \bar{e}_1, \bar{e}_2, \bar{e}_{1,2} \rangle$

- Other control regulators: $\Delta\mathbf{R}$, $\Delta\mathbf{T_1}$ and $\Delta\mathbf{T_2}$
- Possible transformation parameters that can cause conflicts: $\Delta\mathbf{R}\{\bar{p}\}$, $\Delta\mathbf{R}\{\theta\}$, $\Delta\mathbf{T}_1\{d,n\}$, $\Delta\mathbf{T}_2\{d,n\}$

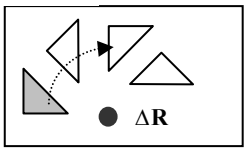
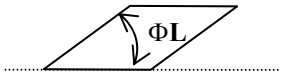
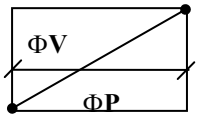
Rotation and boundary $\text{shape}_0 =$ $\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_0)^{<0-n>}\ ])^{<0-n>}\ ]$ $\text{shape}_{\text{Boundary}} =$ $\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_{\text{Boundary}})^{<0-n>}\ ])^{<0-n>}\ ]$ $\Delta\mathbf{R}\ [\ \{\bar{p},\bar{t},\theta,n\}\ (shape_0)^{<0>-<n>}\ ]$ $\Phi\mathbf{B}\ [\ \{o\}\ (shape_{boundary}, shape_0 - shape_n)\ ]$	$\Phi\mathbf{B}$ 
Angle and alignment $\text{shape}_0 =$ $\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_0)^{<0-n>}\ ])^{<0-n>}\ ]$ $\Phi\mathbf{L}\ [\ \{\theta\}\ (shape_0\langle\Delta\mathbf{T}_1,\Delta\mathbf{T}_2\rangle\ )^{<0>-<n>}\ ]$ $\Phi\mathbf{A}\ [\ \{\bar{p},\bar{t}\}\ (shape_0\langle\Delta\mathbf{T}_1\rangle\ )\ ]$	 $\Phi\mathbf{A}$
Proportion and Dimension $\text{shape}_0 = \Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_0)^{<0-n>}\ ])^{<0-n>}\ ]$ $\Phi\mathbf{P}\ [\ \{\bar{t}\}\ (shape_0)^{<0>-<n>}\ ]$ $\Phi\mathbf{V}^1\ [\ \{\min = d\}\ (shape_0\langle\Delta\mathbf{T}_1\rangle\ )\ ]$	

TABLE 5.10 – EXAMPLES OF CONFLICTING SITUATIONS

The second example shows a shape with a fixed angle controlled by an Alignment regulator. Potential conflicts can occur if the angle is changed beyond the alignment, or if the alignment is rotated beyond the allowable angle range. The conflict causing parameters are identified as follows.

- Directional constraint: $\Phi\mathbf{L}$
- Positional constraint: $\Phi\mathbf{A}$

- Constrained objects: $shape_0$
- Constrained parameters:

  By angle: $\text{shape}_0\left\langle \bar{t}_{\Delta \mathbf{T}_1} \right\rangle$, $\text{shape}_0\left\langle \bar{t}_{\Delta \mathbf{T}_2} \right\rangle$

  By alignment: $\text{shape}_0\left\langle \bar{t}_{\Delta \mathbf{T}_1} \right\rangle$, $\text{shape}_0\left\langle \bar{p}_{\Delta \mathbf{T}_1} \right\rangle$
- Possible transformation causing conflicts: $\Phi\mathbf{A}\{\bar{p}\}$, $\Phi\mathbf{A}\{\bar{t}\}$, $\Phi\mathbf{L}\{\theta\}$

The third example shows a shape with a minimum dimension on one side, controlled by a Proportion regulator. Potential conflicts can occur if the proportion line is rotated, thus causing a change in the length beyond the allowable range. The conflict causing parameters are identified as follows.

- Dimensional constraint: $\Phi\mathbf{P}\{\theta\}$ and $\Phi\mathbf{V}^1\{\theta\}$
- Constrained objects: $shape_0$
- Constrained parameters:

  By proportion: $\text{shape}_0\left\langle d_{\Delta \mathbf{T}_1}, n_{\Delta \mathbf{T}_1} \right\rangle$, $\text{shape}_0\left\langle d_{\Delta \mathbf{T}_2}, n_{\Delta \mathbf{T}_2} \right\rangle$

  By length: $\text{shape}_0\left\langle d_{\Delta \mathbf{T}_1}, n_{\Delta \mathbf{T}_1} \right\rangle$
- Possible transformation causing conflicts: $\Phi\mathbf{P}\{\bar{t}\}$, $\Phi\mathbf{V}\{\min\}$

This strategy can be developed further into an algorithm that not only identifies potential conflicts, but deactivates the responsible regulators when the conflicts occur.

5.4. MULTIPLE REPRESENTATION

Although the ICE notation is not ambiguous, it is also not unique: the same configuration can be represented by different notation strings. For a given configuration, these strings would capture different processes of generation, and a different set of applicable transformations. This property of multiple representations allows for different options for generating a shape, as well, different options for manipulating it. Therefore, the ICE framework supports different exploration paths leading to the same configuration, thus allowing users to select variable strategies for their exploration.

The simplest example is the solid square, which can be defined using ICE in several ways. Table 5.11 illustrates the possible generations of the square using continuous generation only. The square can be defined by two continuous Translations; it can be defined by two continuous Mirrors; or alternatively, it can be defined by one Translation and one Mirror, both continuous, in any order.

The number of possible generations is greatly augmented when the continuous and discrete generation methods are combined, as shown is Table 5.12. Notice that this latter form of representation is recursive. A square can be defined by two continuous Translations then a discrete Mirror; it can be defined by two continuous Translations then two discrete Mirrors; it can be defined by two continuous Translations then four discrete Mirrors. Notice how recursive this representation can be. Alternatively, the square can be defined by two continuous Translations then a discrete Rotation; it can be defined by Translation then a Translation composed with a Dilation (both continuous) then a discrete Mirror or the last regulator can be a discrete Rotation.

Multiple representations result in multiple schema for the same configuration, one scheme for every exploration path. Therefore, each representation for the square is a separate schema.

Equivalent configurations having a different set of defining regulators, allow for different applicable transformations. Each configuration will have shared as well as distinct regulators. The distinct regulators will provide a distinct set of manipulations. Furthermore, the shared regulators provide manipulations that produce different results. Table 5.13 and 5.14 illustrate the difference in manipulation effects for distinct

representations of the square. Table 5.13 shows continuous representations, while Table 5.14 shows continuous-discrete examples. In Table 5.13, manipulating $\Delta\mathbf{T_1}$ results in a parallelogram for one representation, and it results in a trapezoid for the other. In Table 5.14, manipulating the shared Regulator $\Delta\mathbf{T_2}$, results in a quadrilateral symmetric configuration for one representation, and a pinwheel for the other. In both tables, the third row shows manipulations of distinct (not shared) regulators.

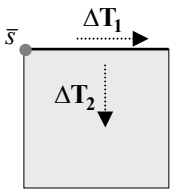
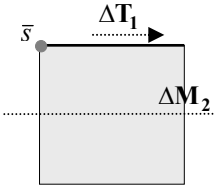
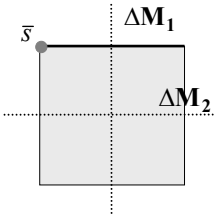
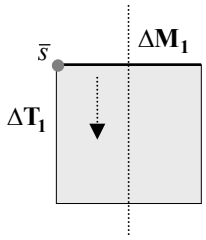
$square = \Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ ($ $\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}\ ]$ $)^{<0-1>}\ ]$	
$square = \Delta\mathbf{M_2}[\ \{\bar{p},\bar{t},n\}\ ($ $\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}\ ]$ $)^{<0-1>}\ ]$	
$square = \Delta\mathbf{M_2}[\ \{\bar{p},\bar{t},n\}\ ($ $\Delta\mathbf{M_1}[\ \{\bar{p},\bar{t},n\}\ (\bar{s})^{<0-1>}\ ]$ $)^{<0-1>}\ ]$	
$square = \Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ ($ $\Delta\mathbf{M_1}[\ \{\bar{p},\bar{t},n\}\ (\bar{s})^{<0-1>}\ ]$ $)^{<0-1>}\ ]$	

TABLE 5.11 - REPRESENTATIONS OF THE SQUARE (CONTINUOUS GENERATION)

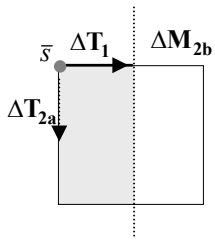
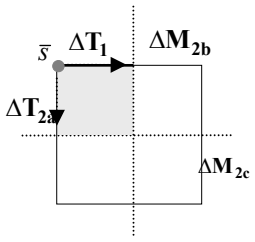
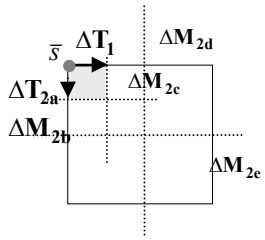
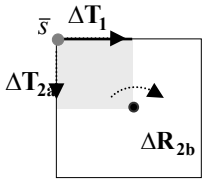
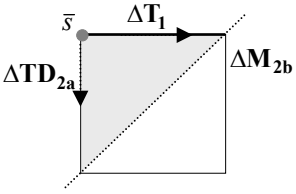
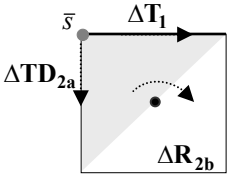
$\Delta\mathbf{M_{2b}}[\ \{\bar{p},\bar{t},n\}$ $(\Delta\mathbf{T_{2a}}[\ \{\bar{p},\bar{t},d,n\}$ $(\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])$ $^{<0-1>}])^{<0><1>}]$	
$\Delta\mathbf{M_{2c}}[\ \{\bar{p},\bar{t},n\}$ $(\Delta\mathbf{M_{2b}}[\ \{\bar{p},\bar{t},n\}$ $(\Delta\mathbf{T_{2s}}[\ \{\bar{p},\bar{t},d,n\}$ $(\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])$ $^{<0-1>}])^{<0><1>}])^{<0><1>}]$	
$\Delta\mathbf{M_{2e}}[\ \{\bar{p},\bar{t},n\}$ $(\Delta\mathbf{M_{2d}}[\ \{\bar{p},\bar{t},n\}$ $(\Delta\mathbf{M_{2c}}[\ \{\bar{p},\bar{t},n\}$ $(\Delta\mathbf{M_{2b}}[\ \{\bar{p},\bar{t},n\}$ $(\Delta\mathbf{T_{2a}}[\ \{\bar{p},\bar{t},d,n\}$ $(\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])$ $^{<0-1>}])^{<0><1>}])^{<0><1>}])^{<0><1>}])^{<0><1>}]$	
$\Delta\mathbf{R_{2b}}[\ \{\bar{p},\bar{t},90^\circ,n\}$ $(\Delta\mathbf{T_{2a}}[\ \{\bar{p},\bar{t},d,n\}$ $(\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])$ $^{<0-1>}])^{<0><1>}]$	
$\Delta\mathbf{M_{2b}}[\ \{\bar{p},\bar{t},n\}$ $(\Delta\mathbf{T}\Delta\mathbf{D_{2a}}[\ \{\bar{p},\bar{t},d,\bar{k},n\}$ $(\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])$ $^{<0-1>}])^{<0><1>}]$	
$\Delta\mathbf{R_{2b}}[\ \{\bar{p},\bar{t},180^\circ,n\}$ $(\Delta\mathbf{T}\Delta\mathbf{D_{2a}}[\ \{\bar{p},\bar{t},d,\bar{k},n\}$ $(\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])$ $^{<0-1>}])^{<0><1>}]$	

TABLE 5.12 - REPRESENTATIONS OF THE SQUARE (CONTINUOUS AND DISCRETE GENERATION)

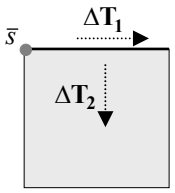
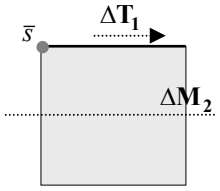
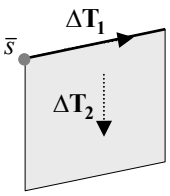
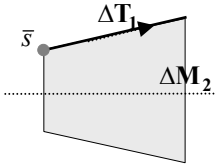
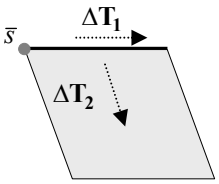
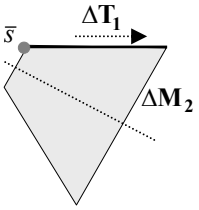
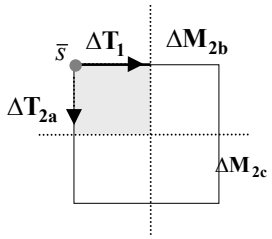
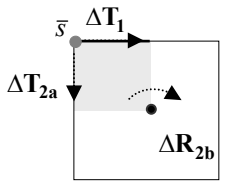
	
$square = \Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ ($ $\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}\ ]$ $)^{<0-1>}\ ]$	$square = \Delta\mathbf{M_2}[\ \{\bar{p},\bar{t},n\}\ ($ $\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}\ ]$ $)^{<0-1>}\ ]$
	
ROTATE_REGULATOR $\Delta\mathbf{T_1}$ $\Delta\mathbf{T_1}[\{\bar{t}=(1,0)\}] \Rightarrow \Delta\mathbf{T_1}[\{\bar{t}=(1,0.25)\}]$	ROTATE_REGULATOR $\Delta\mathbf{T_1}$ $\Delta\mathbf{T_1}[\{\bar{t}=(1,0)\}] \Rightarrow \Delta\mathbf{T_1}[\{\bar{t}=(1,0.25)\}]$
	
ROTATE_REGULATOR $\Delta\mathbf{T_2}$ $\Delta\mathbf{T_2}[\{\bar{t}=(0,-1)\}] \Rightarrow \Delta\mathbf{T_2}[\{\bar{t}=(0.25,-1)\}]$	ROTATE_REGULATOR $\Delta\mathbf{M_2}$ $\Delta\mathbf{M_2}[\{\bar{t}=(1,0)\}] \Rightarrow \Delta\mathbf{M_2}[\{\bar{t}=(1,-1)\}]$

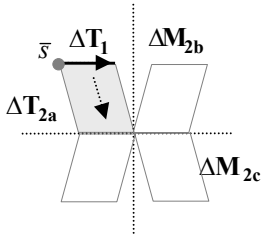
TABLE 5.13 - TRANSFORMATIONS FOR DISTINCT CONTINUOUS REPRESENTATIONS OF THE SQUARE



$\Delta\mathbf{M_{2c}}[\ \{\bar{p},\bar{t},n\}$
$(\Delta\mathbf{M_{2b}}[\ \{\bar{p},\bar{t},n\}$
$(\Delta\mathbf{T_{2s}}[\ \{\bar{p},\bar{t},d,n\}$
$(\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])$
$^{<0-1>}])^{<0><1>}])^{<0><1>}]$

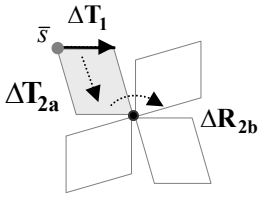


$\Delta\mathbf{R_{2b}}[\ \{\bar{p},\bar{t},90^\circ,n\}$
$(\Delta\mathbf{T_{2a}}[\ \{\bar{p},\bar{t},d,n\}$
$(\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])$
$^{<0-1>}])^{<0><1>}]$



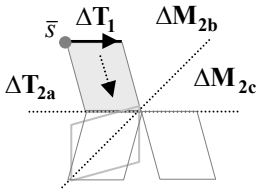
ROTATE_REGULATOR $\Delta\mathbf{T_2}$

$\Delta\mathbf{T_2}[\{\bar{t}=(0,-1)\}] \Rightarrow \Delta\mathbf{T_1}[\{\bar{t}=(0.3,-1)\}]$



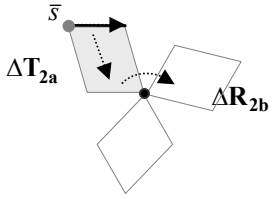
ROTATE_REGULATOR $\Delta\mathbf{T_2}$

$\Delta\mathbf{T_2}[\{\bar{t}=(0,-1)\}] \Rightarrow \Delta\mathbf{T_1}[\{\bar{t}=(0.3,-1)\}]$



ROTATE_REGULATOR

$\Delta\mathbf{M_{2b}}[\{\bar{t}=(0,1)\}] \Rightarrow \Delta\mathbf{M_{2b}}[\{\bar{t}=(1,1)\}]$



MODIFY_FACTOR

$\Delta\mathbf{R_{2b}}[\{\theta=90\}] \Rightarrow \Delta\mathbf{R_{2b}}[\{\theta=120\}]$

TABLE 5.14 - TRANSFORMATION FOR DISTINCT DISCRETE REPRESENTATIONS OF THE SQUARE

Multiple representations are caused by equivalent relationships between the generative regulators as illustrated in Tables 5.15 and 5.16. The symbol ≈ is used for representational equivalence.

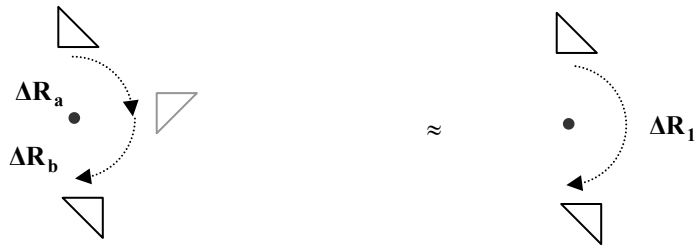
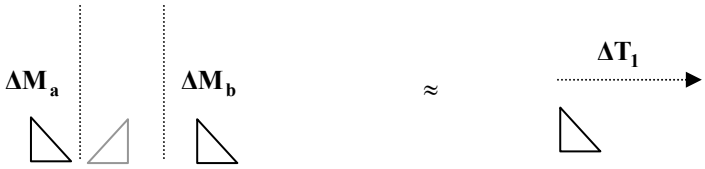
Two Translations ≈ $\Delta\mathbf{T_b}[\ \{\bar{p},\bar{t}_b,d_b\}\ (\Delta\mathbf{T_a}[\ \{\bar{p},\bar{t}_a,d_a\}\ (\bar{s})_{\#1}^{<0><1>})^{<1>}\ ]$	One Translation $\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t}_{a+b},d_{a+b}\}\ (\bar{s})^{<0><1>}\ ]$
	
Two Rotations (same center) ≈ $\Delta\mathbf{R_b}[\ \{\bar{p},\bar{t},\theta_b\}\ (\Delta\mathbf{R_a}[\ \{\bar{p},\bar{t},\theta_a\}\ (\bar{s})_{\#1}^{<0><1>})^{<1>}\ ]$	One Rotation $\Delta\mathbf{R_1}[\ \{\bar{p},\bar{t},\theta_{a+b}\}\ (\bar{s})^{<0-1>}\ ]$
	
Two parallel Mirrors $\Delta\mathbf{M_b}[\ \{\bar{p}_b,\bar{t}\}\ (\Delta\mathbf{M_a}[\ \{\bar{p}_a,\bar{t}\}\ (\bar{s})_{\#1}^{<0><1>})^{<1>}\ ]$	One Translation $\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d\}\ (\bar{s})^{<0-1>}\ ]$
	
Two intersecting Mirrors $\Delta\mathbf{M_b}[\ \{\bar{p}_b,\bar{t}_b\}\ (\Delta\mathbf{M_a}[\ \{\bar{p}_a,\bar{t}_a\}\ (\bar{s})_{\#1}^{<0><1>})^{<1>}\ ]$	One Rotation $\Delta\mathbf{R_1}[\ \{\bar{p},\theta\}\ (\bar{s})^{<0-1>}\ ]$
	

TABLE 5.15 - EXAMPLES OF EQUIVALENCES IN TRANSFORMATION REGULATORS IN 2D

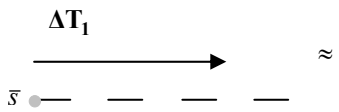
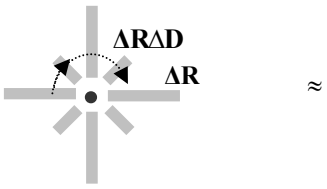
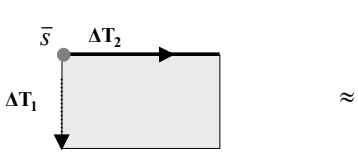
Combined generation $\Delta\mathbf{T_1}[\{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1><2-3><4-5>}]$	Two successive regulators (of the same type) $\Delta\mathbf{T_{1b}}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_{1a}}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}\ ])^{<0-1>}\ ]$
	
Two successive regulators $\Delta\mathbf{R}[\ \{\bar{p},\bar{t},\theta,n\}($ $\Delta\mathbf{R}\Delta\mathbf{D}[\ \{\bar{p},\bar{t},d,n\}\ (shape)^{<0-1>}]$ $)^{<0><3>}]$	Rhythmic variation $\Delta\mathbf{R}\Xi\mathbf{G_1}[\ \{\bar{p},\bar{t},\theta,n,a,\boldsymbol{f},c\}\ (shape)^{<0><7>}\ ]$
	
Maximal $\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ ($ $\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}\ ]$ $)^{<0-1>}\ ]$	Two subdivisions $\Delta\mathbf{T_{A2}}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{T_{A1}}[\{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]\wedge$ $\Delta\mathbf{T_{B2}}[\{\bar{p},\bar{t},d,n\}(\Delta\mathbf{T_{B1}}[\{\bar{p},\bar{t},d,n\}\ (\bar{s})^{<0-1>}])^{<0-1>}]$
	
Boolean union $\mathbf{\Omega U}[\ \{\}\ (shape_A, shape_B)\ \ ]$	Boolean difference $\mathbf{\Omega D}[\ \{\}\ (shape_C, shape_D)\ \ ]$
	

TABLE 5.16- EXAMPLES OF EQUIVALENCES IN VARIATION, AND OPERATION REGULATORS

Two consecutive Translations, $\mathbf{\Delta T_a}$ and $\mathbf{\Delta T_b}$ are equivalent to a single Translation $\mathbf{\Delta T_1}$, where the factors and the direction vectors are related as follows: $d_a + d_b = d_1$ and $\bar{t}_a + \bar{t}_b = \bar{t}_1$. Two consecutive Rotations about the same center, $\mathbf{\Delta R_a}$ and $\mathbf{\Delta R_b}$, are equivalent to a single Rotation $\mathbf{\Delta R_1}$, where the factors are related as follows: $\theta_a + \theta_b = \theta_1$. Two parallel Mirrors, $\mathbf{\Delta M_a}$ and $\mathbf{\Delta M_b}$ are equivalent to a single translation $\mathbf{\Delta T_1}$, where the direction vector $\Delta\mathbf{T_1}\{\bar{t}\}$ is perpendicular to $\Delta\mathbf{M_1}\{\bar{t}\}$ and the factor is equal to twice the distance between the two mirror lines: $\Delta\mathbf{T_1}\{d\} = 2\times(p_a - p_b)$. Two intersecting Mirrors, $\mathbf{\Delta M_a}$ and $\mathbf{\Delta M_b}$ are equivalent to a single Rotation $\mathbf{\Delta R_1}$, where the rotation point $\Delta\mathbf{R_1}\{\bar{p}\}$ is the intersection of the mirror lines $\Delta\mathbf{M_{1a}}\{\bar{t}\}$ and $\Delta\mathbf{M_{1b}}\{\bar{t}\}$, and the rotation degree $\Delta\mathbf{R_1}\{\theta\}$ equals twice the angle between $\Delta\mathbf{M_{1a}}\{\bar{t}\}$ and $\Delta\mathbf{M_{1b}}\{\bar{t}\}$.

The capacity of ICE to determine equivalent representations (same shape different generation method) depends on the regulators chosen as well as on the generation method chosen. It also varies if the representation is recursive.

If maximal representations are used to eliminate the recursive factor, and if isometric regulators are considered exclusively, it would be possible to algorithmically determine equivalent representations for an ICE string. Two consecutive isometric transformation regulators can be replaced by a single equivalent regulator or vice versa. However, once other regulators (such as affine transformations, variations or operations) are used, and once generation methods are combined, additional representational options are introduced. Consequently, determining equivalences becomes increasingly complex, and it would be extremely difficult to determine, algorithmically, possible equivalences for a given configuration.

5.5. DETERMINING TRANSFORMATION STEPS

The ICE notation provides a flexible representation that supports transformation from one configuration into another. Determining the precise transformation sequence is not always a simple task, especially when the configurations are very different. However, it is critical to identify a strategy for achieving a goal shape or configuration from an initial one. In his section, I introduce a computational algorithm that automates the derivation "sequential steps" for such transformations, provided both configurations are represented using the ICE notation.

Given an initial and a goal string, a precise sequence of individual transformations can be determined by using simple string operations such as juxtaposition, insertion, deletion, and replacement. The objective of the Transformation Steps (TS) algorithm is to make the initial string identical to the goal string, with the minimum number of steps. These steps are a list of transformations based on the ICE transformational syntax (Table 4.15).

The algorithm cycles through the strings several times, each time addressing a different element, identifying a particular transformation step and updating the working string (which is the intermediate string going from the initial to the goal). Earlier iterations or phases consider the whole string, while focusing on regulator types, and later phases consider individual regulators, and focus on particular attributes. If the strings are broken down into levels of encapsulations, each of these levels is compared, using the same algorithm, recursively.

The algorithm's major iterations are as follows

- Match the strings
- Adjust the regulator sequence (# of regulators changes)
- Adjust the regulator sequence (same # of regulators)
- Adjust the regulator composition
- Update indices
- Adjust the regulator parameters
- Adjust the generation method
- Adjust the regulated element (points)

Table 5.18 describes each step of the algorithm in detail with an example of transforming a 2D pattern into a 3D shape, using the initial and goal configuration strings in Table 5.17.

Initial String

$p6mm = \Delta\mathbf{T_c}(\Delta\mathbf{R_b}(\Delta\mathbf{M_a}(\Delta\mathbf{C_1}(s_p)\ )\ )\ )$

$p6mm =$

$\Delta\mathbf{T^1_d}[\{\bar{p}, \bar{t} = (1, \sqrt{3}, 0), d, n\}\ ($

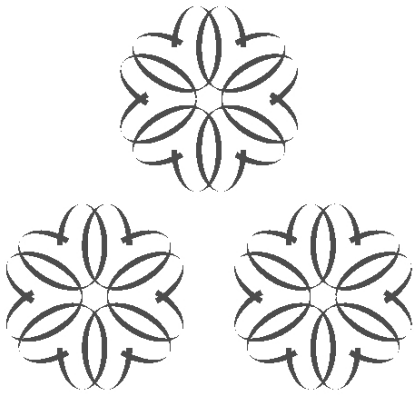
$\quad \Delta\mathbf{T^1_c}[\{\bar{p}, \bar{t} = (1,0,0), d, n\}\ ($

$\quad\quad \Delta\mathbf{R^1_b}[\{\bar{p}, \bar{t}, \theta = 60^o, n = 5\}\ ($

$\quad\quad\quad \Delta\mathbf{M^1_a}[\{\bar{p}, \bar{t} = (0,1,0), n = 1\}\ ($

$\quad\quad\quad\quad \Delta\mathbf{C_1}[\{\bar{p}, \theta, n\}\ (s_p)^{<0-1>}]$

$)^{<0><1>}]\quad)^{<0>-<5>}]\quad)^{<0>-<n>}]\ )^{<0>-<n>}]$



Goal String

$cone = \Delta\mathbf{TD_3}(\Delta\mathbf{R_2}(\Delta\mathbf{T_1}(s_c)\ )\ )$

$cone =$

$\Delta\mathbf{TD^1_3}[\{\bar{p}, \bar{t} = (0,0,1), d, n\}\ ($

$\quad \Delta\mathbf{R^1_2}[\{\bar{p}, \bar{t}, \theta = 360^o, n = 1\}\ ($

$\quad\quad \Delta\mathbf{T^1_1}[\{\bar{p}, \bar{t} = (1,0,0), n\}\ (s_c)^{<0-1>}]$

$)^{<0-1>}]$

$)^{<0-1>}]$

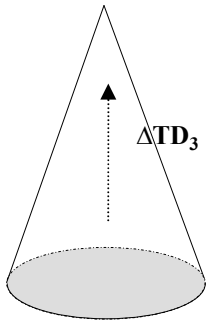


TABLE 5.17 - INITIAL AND GOAL CONFIGURATIONS

Match the strings

Initial = ΔT_d ΔT_c ΔR_b ΔM_a ΔC_1 s_p

Goal = ΔTD_3 ΔR_2 ΔT_1 s_c

Working = ΔT_d ΔT_c ΔR_b ΔM_a ΔC_1 s_p

Adjust the regulator sequence (# of regulator changes)

Working = ΔT_d ΔT_c ΔR_b ΔM_a ΔC_1 s_p

Goal = ΔTD_3 ΔR_2 ΔT_1 s_c

New Working = ΔT_c ΔR_b ΔC_1 s_p

DELETE_SUCCESSIVE ΔM_a

DELETE_SUCCESSIVE ΔT_d

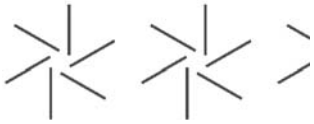
Adjust the regulator sequence (# of regulator is the same)

Working = ΔT_c ΔR_b ΔC_1 s_p

Goal = ΔTD_3 ΔR_2 ΔT_1 s_c

New Working = ΔT_c ΔR_b ΔT_1 s_p

REPLACE_REGULATOR $\Delta C_1 \Rightarrow \Delta T_1$



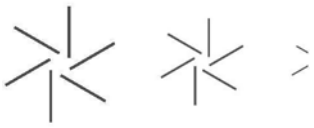
Adjust the regulator composition

Working = ΔT_c ΔR_b ΔT_1 s_p

Goal = ΔTD_3 ΔR_2 ΔT_1 s_c

New Working = ΔTD_c ΔR_b ΔT_1 s_p

ADD_SIMULTANEOUS ΔT_c => ΔTD_c



Update indices and dimensions

Working = $\Delta T^1 D^0_c$ ΔR^1_b ΔT^1_1 $\bar{s}_p$

Goal = $\Delta T^1 D^0_3$ ΔR^1_2 ΔT^1_1 s_c

New Working = ΔTD_3 ΔR_2 ΔT_1 $\bar{s}_p$

MODIFY_INDEX ΔTD_c => ΔTD_3

MODIFY _INDEX ΔR_b => ΔR_2

Adjust the regulator parameters

Working = $\boxed{\Delta\mathbf{TD_3}\{\bar{p},\bar{t}=(1,0,0),d,n\}}$ $\boxed{\Delta\mathbf{R_2}\{\bar{p},\bar{t},\theta=60^o,n=5\}}$ $\boxed{\Delta\mathbf{T_1}\{\bar{p},\theta,n\}}$ $\boxed{s_p}$

Goal = $\boxed{\Delta\mathbf{TD_3}\{\bar{p},\bar{t}=(0,0,1),d,n\}}$ $\boxed{\Delta\mathbf{R_2}\{\bar{p},\bar{t},\theta=360^o,n=1\}}$ $\boxed{\Delta\mathbf{T_1}\{\bar{p},\bar{t}=(1,0,0),n\}}$ $\boxed{s_c}$

New Working = $\boxed{\Delta\mathbf{TD_3}\{\bar{p},\bar{t}=(0,0,1),d,n\}}$ $\boxed{\Delta\mathbf{R_2}\{\bar{p},\bar{t},\theta=360^o,n=1\}}$ $\boxed{\Delta\mathbf{T_1}\{\bar{p},\theta,n\}}$ $\boxed{s_p}$

ROTATE_XYZ $\Delta\mathbf{TD_3}.\bar{t}=(1,0,0)\Rightarrow\Delta\mathbf{TD_3}.\bar{t}=(0,0,1)$

MODIFY_FACTOR $\Delta\mathbf{R_2}.\theta=60^o\Rightarrow\Delta\mathbf{R_2}.\theta=360^o$

INSERT_PARAMETER $\Delta\mathbf{T_1}.\bar{t}=(1,0,0)$

MODIFY_NUMBER, $\Delta\mathbf{R_2}.n=5\Rightarrow\Delta\mathbf{R_2}.n=1$

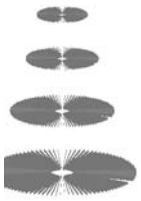
Adjust the generation method

Working = $\boxed{\Delta\mathbf{TD_3}^{<0>-<n>}}$ $\boxed{\Delta\mathbf{R_2}^{<0>-<n>}}$ $\boxed{\Delta\mathbf{T_1}^{<0-1>}}$ $\boxed{s_p}$

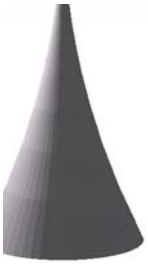
Goal = $\boxed{\Delta\mathbf{TD_3}^{<0-1>}}$ $\boxed{\Delta\mathbf{R_2}^{<0-1>}}$ $\boxed{\Delta\mathbf{T_1}^{<0-1>}}$ $\boxed{s_c}$

New Working = $\boxed{\Delta\mathbf{TD_3}^{<0-1>}}$ $\boxed{\Delta\mathbf{R_2}^{<0-1>}}$ $\boxed{\Delta\mathbf{T_1}^{<0-1>}}$ $\boxed{s_p}$

MODIFY_CONTINUITY $\boxed{\Delta\mathbf{R_2}^{<0>-<n>}\Rightarrow\Delta\mathbf{R_2}^{<0-1>}}$



MODIFY_CONTINUITY $\boxed{\Delta\mathbf{TD_3}^{<0>-<n>}\Rightarrow\Delta\mathbf{TD_3}^{<0-1>}}$



Adjust the regulated element (points and their attributes)

Working = $\boxed{\Delta\mathbf{T_c}}$ $\boxed{\Delta\mathbf{R_b}}$ $\boxed{\Delta\mathbf{T_1}}$ $\boxed{s_p}$

Goal = $\boxed{\Delta\mathbf{TD_3}}$ $\boxed{\Delta\mathbf{R_2}}$ $\boxed{\Delta\mathbf{T_1}}$ $\boxed{s_c}$

Final = $\boxed{\Delta\mathbf{TD_c}}$ $\boxed{\Delta\mathbf{R_b}}$ $\boxed{\Delta\mathbf{T_1}}$ $\boxed{s_c}$

REPLACE_SHAPE $s_p\Rightarrow s_c$

TABLE 5.18 - THE TRANSFORMATION STEPS ALGORITHM

The "Match the strings" phase of the TS algorithm considers the whole regulator sequence and focuses on regulator types. It is a process of alignment where the working string is moved with respect to the goal string until the greatest number of regulators is

matched. Two regulators such as $\Delta\mathbf{T_c}$ and $\Delta\mathbf{TD_3}$ are considered as a semi-match.

The first "Adjust the regulator sequence" phase consists of an alignment operation that recursively moves the non-matching regulators to derive the greatest match between the working and goal strings. It focuses on the whole string at the regulator level of abstraction, and identifies regulators that need to be deleted and creates empty spaces for insertions. ADD_SUCCESSIVE, INSERT_SUCCESSIVE, DELETE_SUCCESSIVE are the possible transformations that could be identified in this phase.

The second "Adjust the regulator sequence" phase compares both strings, focusing on one regulator at a time, and replacing or swapping non-matching regulators. The possible transformations that could be identified in this phase are SWAP_SUCCESSIVE and REPLACE_REGULATOR.

The "Adjust the regulator composition" phase focuses on individual semi-matching regulators, adding or removing types from their composition. ADD_SIMULTANEOUS, REMOVE_SIMULTANEOUS, and SWAP_SIMULTANEOUS are the possible transformations of this phase.

The "Update indices and dimensions" phase focuses on the subscripts and superscripts of individual regulators. At the end of this phase, the sequence of regulators in the working string and goal string are identical. The possible transformations for this phase are ADJUST_INDEX and ADJUST_DIMENSION.

The "Adjust the regulator parameters" phase focuses on the parameters inside the curly brackets, of individual regulators. Transformations for this phase are MOVE_XYZ, ROTATE_XYZ, MODIFY_FACTOR, MODIFY_NUMBER, and MODIFY_FORMULA.

This "Adjust the generation method" phase focuses on the generation methods (inside the subscript brackets) of individual regulators. Transformations for this phase are MODIFY_CONTINUITY, MODIFY_PATTERN, and MODIFY_GENERATED.

The "Adjust the regulated element" phase focuses on regulated elements, which are the innermost elements in a regulator sequence. Transformations identified are MOVE_XYZ,

MODIFY_ATTRIBUTE, REPLACE_SHAPE.

With this algorithm, not only can the ICE framework be used for interactive exploration towards an unknown goal, it can also be used to identify precise strategies to reach a desired goal.

5.6. DESIGN SPACE IN THE ICE REPRESENTATION

In the context of Simon's cognitive model of design (Simon, 1969), the ICE notational string, which capture a generation method and the applicable transformations, presents a snapshot within the exploration process. A string represents one state in an immense space of possibilities. The applicable transformations represent the possible transitions from this state to other states. The generation method is one non-cyclic exploration path that leads to this state from an initial start state (the point).

In the ICE representation there is no distinction between terminal and non-terminal symbols. Consequently, there is no classification of intermediary and final states. All symbols can be transformed at anytime. All states can be transformed at any time, although these same states are considered final at any time.

There are implicit rules for transitions in ICE, establishing the way element in the string are transformed. Only regulators can replace regulators; only generation methods can be reconfigured within the brackets, and only parameters can change in value. These rules are encoded in the well defined structure of the ICE string and in the transformation syntax of the ICE notation.

For each state, there are numerous possible transformations, each leading to another other design state. However, in certain states not all transformations are applicable: for instance, DELETE_REGULATOR is not applicable to the initial state consisting of a single point. Design transformations, which are manipulations on the ICE string, can have variable effects, depending on the parameter values. REPLACE_REGULATOR will produce different states if the new regulator is a Rotation or a Dilation; similarly, MODIFY_FACTOR will produce different states for different values.

Because of continuous parameter values, it is not possible to enumerate all design states that can be generated from a given state, thus making the design space an infinite space. Furthermore, since every object can be transformed into every other object using the TS algorithm, the infinite design space encompasses all objects that could be generated in ICE.

The universe described by the ICE notation is two fold: the universe of shapes and configurations, and the universe of manipulations. These are determined by the regulators in use, as well as on the generation method.

Transformation regulators applied continuously define a universe of 1D, 2D and 3D shapes in 3D space. Translation regulators, used exclusively, define a universe of rectilinear shapes, while isometric regulators (Translation, Rotation and Mirror) define a universe consisting of Euclidean shapes. Affine regulators such as Dilation, and Shear, significantly increase the universe of possible shapes and so do Curve and Deformation regulators.

Generation methods also influence the resulting universe. Discrete generation, used exclusively, the result would be a universe of points. If the generation were restricted to one continuous regulator and unlimited discrete regulators, the resulting universe would consist of linear objects in a 3D Space. If continuous and discrete generation were used in combination, irregular shapes and shapes with holes could be produced.

Variation and operation regulators significantly increase the universe of possible shapes as well as the complexity of the shapes generated; while constraint, topological and hierarchical regulators influence ICE's universe of manipulations.

Since the set of regulators described in this document is not an exhaustive list. The universe of the configurations and the universe of manipulations described in ICE are not clearly determined. As more regulators are added according to the necessity, the universe is redefined.

CHAPTER 6

ARCHITECTURAL EXAMPLES

In this Chapter, I demonstrate the capacity of the ICE notation to describe architectural configurations and transformations across these, through architectural works. These include the representation of architectural components (Section 6.1), a hypothetical generation and transformation of Hejduk's half house (Sections 6.2), the roof structure of Calatrava's Art Museum in Milwaukee, and an ethnographic observation in a design studio (Section 6.4). Each of these examples represents a different aspect of design. Some are finished products, while others are snapshots of actual evolving designs. Yet others represent hypothetical generations.

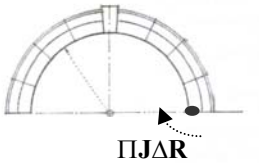
The strategy for describing effectively using ICE depends on the properties of the configuration represented. Most architectural configurations are constructed by means of repetitive elements, and often such configurations have repetitive relations. It is important to capture these repetitions in ICE in order to define the most effective and parsimonious description and the shortest generation path. Most configurations can be divided into smaller units, and represented in a top-down or bottom-up manner. Shape encapsulation in ICE allows the representation of such configurations at multiple levels of abstraction, including detailed descriptions, as well as higher-level ones.

6.1. ARCHITECTURAL ELEMENTS

In this section, I illustrate architectural components, such as arches, domes, vaults, stairs, roofs, and trusses, as they are described with the ICE notation. This section serves as a preamble to the more elaborate and exploratory examples of the following sections.

Table 6.1 shows round and pointed arches. The round arch is described by rotating the unit bricks while the pointed arch is described by rotating then mirroring these bricks. In addition, these bricks are constrained by an Adjacency regulator composed with the generative rotations and translations. The curved bricks are described by continuous Translation and Rotation regulators, while the keystone of the pointed arch is described by a Mirror, composed with Cutting and Adjacency regulators.

$$RoundArch =$$
$$\Pi\mathbf{J}\Delta\mathbf{R}_{arch}[\ \{\bar{p},\bar{t},\theta,n\}\ (curvedBrick)]^{<0>-<n>} \wedge keystone$$
$$curvedBrick =$$
$$\Delta\mathbf{R}_{\mathbf{3}}[\ \{\bar{p},\bar{t},\theta,n\}\ ($$
$$\Delta\mathbf{R}_{\mathbf{2}}[\ \{\bar{p},\bar{t},\theta,n\}\ ($$
$$\Delta\mathbf{T}_{\mathbf{1}}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_{brick})^{<0-n>}]$$
$$)^{<0-n>}]\ )^{<0-n>}]$$



$$pointedArch =$$
$$\Delta\mathbf{M}[\ \{\bar{p},\bar{t},\theta,n\}\ ($$
$$\Delta\mathbf{R}\Pi\mathbf{J}[\ \{\bar{p},\bar{t},\theta,n\}\ (curvedBrick)]^{<0>-<n>}])^{<0>-<n>}]$$
$$\wedge keystone$$
$$Keystone =$$
$$\Pi\mathbf{J}\Delta\mathbf{M}\Omega\mathbf{C}[\ \{\bar{p},\bar{t},n\}\ ($$
$$\Delta\mathbf{R}_{\mathbf{3}}[\ \{\bar{p},\bar{t},\theta,n\}\ ($$
$$\Delta\mathbf{R}_{\mathbf{2}}[\ \{\bar{p},\bar{t},\theta,n\}\ ($$
$$\Delta\mathbf{T}_{\mathbf{1}}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_{keystone})^{<0-n>}]$$
$$)^{<0-n>}]\ ^{<0-n>}]\ )^{<0><1>}]$$

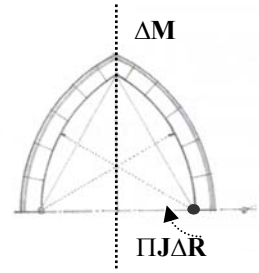


TABLE 6.1 – ARCHES (SOURCE CHING 1997, P14)

The Roofs in Table 6.2 are described by means of Translation and Mirror regulators composed with adjacencies.

gableRoof =

$\Pi\mathbf{J}\Delta\mathbf{M}[\ \{\bar{p},\bar{t},n\}\ ($

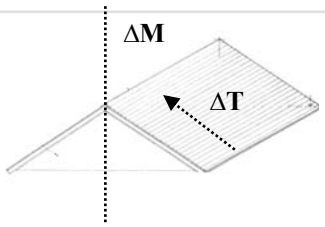
$\Pi\mathbf{J}\Delta\mathbf{T}[\ \{\bar{p},\bar{t},\theta,n\}\ (beam)]^{<0>-<n>}])^{<0>-<n>}]$

beam =

$\Delta\mathbf{T_3}[\ \{\bar{p},\bar{t},d,n\}\ ($

$\Delta\mathbf{T_2}[\ \{\bar{p},\bar{t},d,n\}\ ($

$\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_{beam})^{<0-n>}]\ )^{<0-n>}]\ )^{<0-n>}]$



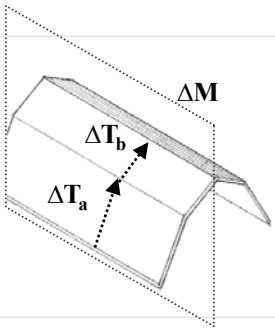
mansardRoof =

$\Pi\mathbf{J}\Delta\mathbf{M}[\ \{\bar{p},\bar{t},n\}\ ($

$\Delta\mathbf{T_b}[\ \{\bar{p},\bar{t},\theta,n\}\ ($

$\Delta\mathbf{T_a}[\ \{\bar{p},\bar{t},\theta,n\}\ (beam)^{<0>-<n>}_{\#n}]$

$)^{<0>-<n>}]\ )^{<0>-<n>}]$



hipRoof =

$\Delta\mathbf{M_h}[\ \{\bar{p},\bar{t},n\}\ ($

$\Delta\mathbf{T}\Delta\mathbf{D_a}[\ \{\bar{p},\bar{t},\bar{k},\theta,n\}\ (beam_1)^{<0>-<n>}]\ )^{<0>-<n>}]$

$\Pi\mathbf{J}\Delta\mathbf{M_v}[\ \{\bar{p},\bar{t},n\}\ ($

$\Delta\mathbf{T}\Delta\mathbf{D_b}[\ \{\bar{p},\bar{t},\bar{k},\theta,n\}\ (beam_2)^{<0>-<n>}]\ )^{<0>-<n>}]$

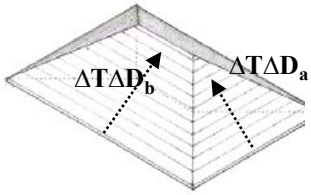


TABLE 6.2 – ROOFS (SOURCE CHING 1997, P208)

The truss in Table 6.3 is described by means of Alignment regulators. The start-point of each inclined member is aligned to the horizontal post, and its endpoint is aligned to the diagonal post. The horizontal Alignment regulator is composed with a Translation regulator in order to generate the inclined members.

belgianTruss =

$\Delta\mathbf{M}[\ \{\bar{p},\bar{t},n\}\ ($

$\Delta\mathbf{T}\Phi\mathbf{A_{horizontal}}[\{\bar{p},\bar{t},n\}\ (members\langle\bar{s}\rangle)^{<0><n>}]$

$\Phi\mathbf{A_{slanted}}[\{\bar{p},\bar{t},n\}\ (member\langle\bar{e}_1\rangle)^{<0><n>}]$

$)^{<0><1>}]$

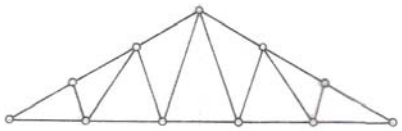


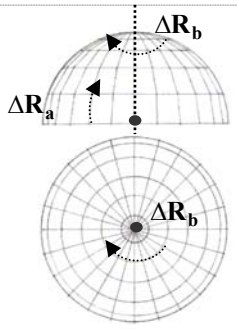
TABLE 6.3 – TRUSSES (SOURCE CHING 1997, P261)

The domes in Table 6.4 are described by means of compound rotations of their unit structural members. Additionally, horizontal structural members are kept adjacent, and diagonal structural members are aligned to both horizontal and vertical members.

radialDome =

$\Delta\mathbf{R_b}[\ \{\bar{p},\bar{t},n\}\ (\Delta\mathbf{R_a}[\ \{\bar{p},\bar{t},\theta,n\}\ (line_{vertical})^{<0-n>}])^{<0>-<n>}]$

$\Pi\mathbf{J}\Delta\mathbf{R_b}[\ \{\bar{p},\bar{t},n\}\ (\Delta\mathbf{R}\Delta\mathbf{D_a}[\ \{\bar{p},\bar{t},\theta,n\}\ (line_{horizontal})^{<0-n>}])^{<0>-<n>}]$



schwedlerDome =

$\Delta\mathbf{R_b}[\ \{\bar{p},\bar{t},\theta,n\}\ (\Delta\mathbf{R}\Delta\mathbf{D_a}[\ \{\bar{p},\bar{t},\theta,n\}\ (line_v)^{<0-n>}])^{<0>-<n>}]$

$\Pi\mathbf{J}\Delta\mathbf{R_b}[\ \{\bar{p},\bar{t},\theta,n\}\ (\Delta\mathbf{R}\Delta\mathbf{D_a}[\ \{\bar{p},\bar{t},\theta,n\}\ (line_h)^{<0-n>}])^{<0>-<n>}]$

$\Delta\mathbf{R_b}[\ \{\bar{p},\bar{t},\theta,n\}\ (\Delta\mathbf{R}\Delta\mathbf{D_a}[\ \{\bar{p},\bar{t},\theta,n\}\ (line_d)^{<0-n>}])^{<0>-<n>}]$

$\Phi\mathbf{A_h}[\ \{\bar{p},\bar{t},n\}\ (line_d)] \wedge$

$\Phi\mathbf{A_v}[\ \{\bar{p},\bar{t},n\}\ (line_d)]$

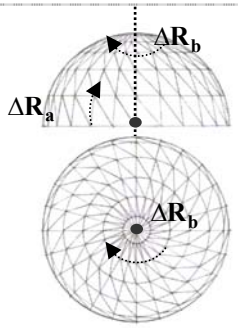


TABLE 6.4 – DOMES (SOURCE CHING 1997, P60)

Table 6.5 shows two vaults. The quadripartite vault is described by creating the curved surface, then mirroring it, rotating it, and cutting all surfaces beyond the intersection. The annular vault is defined by rotating the arch-like cross section.

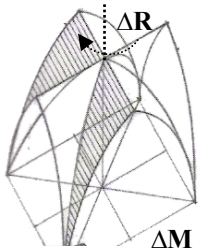
quadriPartiteVault =

$\Pi\mathbf{J}\Delta\mathbf{R}\Omega\mathbf{C}[\ \{\bar{p},\bar{t},\theta,n\}\ ($

$\Delta\mathbf{M}\ [\ \{\bar{p},\bar{t},n\}\ (curvedSurface))^{<0-n>}]\ )^{<0-n>}]$

curvedSurface =

$\Delta\mathbf{R_2}[\ \{\bar{p},\bar{t},\theta,n\}\ (\Delta\mathbf{T_1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_{vault})^{<0-n>}])^{<0-n>}]$



annularVault =

$\Delta\mathbf{R}$ [$\{\bar{p},\bar{t},\theta,n\}$ $(arcade, outerwall, vaultCrossSection)^{<0><n>}$]

VaultCrossSection =

$\Delta\mathbf{R_2}$[$\{\bar{p},\bar{t},n\}$ ($\Delta\mathbf{T_1}$[$\{\bar{p},\bar{t},n\}$ $(\bar{s}_{vault})^{<0-n>}$)$^{<0-n>}$]

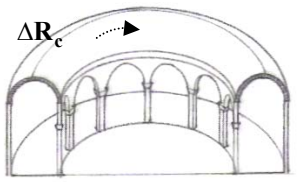


TABLE 6.5 – VAULTS (SOURCE CHING 1997, P263)

In Table 6.6, half turn stairs are described by means of translations and one 180º screw rotation, while spiral stairs are described by means of only screw rotations.

halfTurnStair =

$\Pi\mathbf{J}\Delta\mathbf{T}\Delta\mathbf{R}$[$\{\bar{p},\bar{t},d,n\}$ (

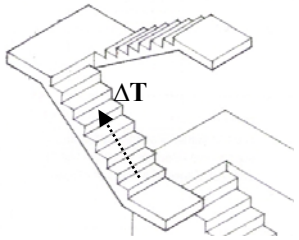
$\Pi\mathbf{J}\Delta\mathbf{TJ}$[$\{\bar{p},\bar{t},d,n\}$ $(tread)^{<0-n>}$])$^{<0-n>}$]

tread =

$\Delta\mathbf{TD_3}$[$\{\bar{p},\bar{t},d,n\}$ (

$\Delta\mathbf{T_2}$[$\{\bar{p},\bar{t},d,n\}$ (

$\Delta\mathbf{T_1}$[$\{\bar{p},\bar{t},d,n\}$ $(\bar{s}_{tread})^{<0-n>}$])$^{<0-n>}$])$^{<0-n>}$]



spiralStair =

$\Delta\mathbf{T}\Delta\mathbf{R}$[$\{\bar{p},\bar{t},d,n\}$ $(tread, support)^{<0-n>}$] $\wedge$ *centralPost*

tread =

$\Delta\mathbf{R_3}$[$\{\bar{p},\bar{t},d,n\}$ (

$\Delta\mathbf{T_2}$[$\{\bar{p},\bar{t},d,n\}$ (

$\Delta\mathbf{T_1}$[$\{\bar{p},\bar{t},d,n\}$ $(\bar{s}_{tread})^{<0-n>}$])$^{<0-n>}$])$^{<0-n>}$]

centralPost =

$\Delta\mathbf{T_3}$[$\{\bar{p},\bar{t},d,n\}$ (

$\Delta\mathbf{R_2}$[$\{\bar{p},\bar{t},\theta,n\}$ (

$\Delta\mathbf{T_1}$[$\{\bar{p},\bar{t},d,n\}$ $(\bar{s}_{post})^{<0-n>}$])$^{<0-n>}$])$^{<0-n>}$]

Support =

$\Delta\mathbf{TD_3}$[$\{\bar{p},\bar{t},\bar{k},d,n\}$ (

$\Delta\mathbf{T_2}$[$\{\bar{p},\bar{t},d,n\}$ (

$\Delta\mathbf{T_1}$[$\{\bar{p},\bar{t},d,n\}$ $(\bar{s}_{post})^{<0-n>}$])$^{<0-n>}$])$^{<0-n>}$]

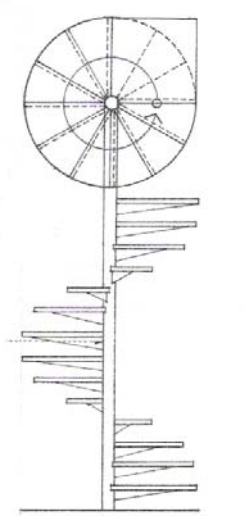


TABLE 6.6 – STAIRS (SOURCE CHING 1997, P234)

6.2. HALF HOUSE TO HOUSE 10

In this section, the ICE notation is used to describe two of John Hejduk's architectural endeavors: Half House and House 10 (Figure 6.1, sources: Ching p.187 and p.12). Both houses have simple, non-repetitive parts, yet include repetitive relationships. Both consist of semi-primary shaped rooms linked by common spaces. These houses were chosen for their simplicity and elegance. The global asymmetry of these houses illustrates ICE's descriptive capacity in encapsulating local symmetries and geometrical constraints. A hypothetical exploratory-generation of Half House is presented in Section 6.2.1, and a hypothetical transformation of Half House to House 10 is presented in Section 6.2.2.

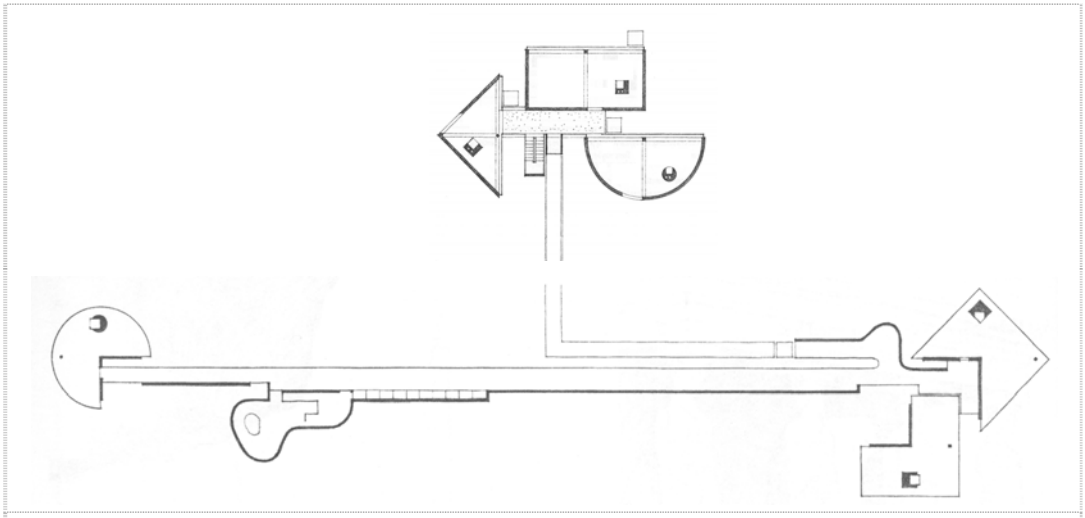


FIGURE 6.1 – HALF HOUSE AND HOUSE 10

6.2.1. THE GENERATION OF HALF HOUSE

Table 6.7 shows the exploratory step by step generation of Half House from an initial rectangle. An exploratory generation allows users to explore while modeling. Therefore, it includes intermediate exploration steps that are not captured in the final generation sequence. Each step in the table shows the current state of generation, its corresponding notation and the operation that resulting in this state, from the previous one.

In the 1st step of Table 6.7, the outline of the rectangular unit, $unit_A$, is generated from

two Translation regulators, $\Delta\mathbf{T_v}$ and $\Delta\mathbf{T_h}$, and one Mirror regulator, $\Delta\mathbf{M}$. This sequence is strategically chosen, among the multiple representations of the rectangle to describe the local bilateral symmetry of each space.

In the 2nd step, $unit_B$ and $unit_C$ are generated by applying the Translation regulators, $\Delta\mathbf{T_B}$ and $\Delta\mathbf{T_C}$, respectively to $unit_A$.

In the 3rd step, alignments between the three rectangles are identified. $unit_C$ is aligned vertically to $unit_A$, along $\Phi\mathbf{A_v}$, and is aligned horizontally to $unit_B$, along $\Phi\mathbf{A_h}$.

In the 4th step, the rectangular units are defined as being composed of an enclosure and a line, and the thickness for the enclosure is incorporated by means of the Dilation regulator, $\Delta\mathbf{D}$. The enclosure and the line are defined to be adjacent.

In the 5th step, the structural constraint between the midpoints of the three shapes ($\bar{p}_A$, $\bar{p}_B$ and $\bar{p}_C$) is identified as a right triangle, which is represented by the Angle regulator $\Phi\mathbf{L}$. Each midpoint of the shape is actually the endpoint of the line defined by the regulator $\Delta\mathbf{T_h}$ and is constrained to the Mirror regulator, $\Delta\mathbf{M}$. The directional parameters, $\bar{t}$, of both Translation regulators, $\Delta\mathbf{T_B}$ and $\Delta\mathbf{T_C}$, are adjusted to be consistent with this right angle.

In the 6th step, $unit_B$ is rotated 90 degrees from its midpoint $\bar{p}_C$. This manipulation is equivalent to replacing the Translation regulator $\Delta\mathbf{T_B}$, which relates $unit_A$ to $unit_B$, with the Rotation regulator $\Delta\mathbf{R_B}$.

In the 7th step, the window, column, beams, terrace, and the corner square are integrated into the representation to create more detailed units. The window and its opening are defined as subshapes of the enclosure.

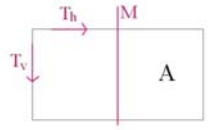
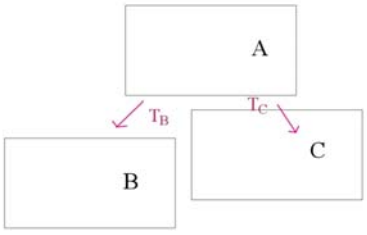
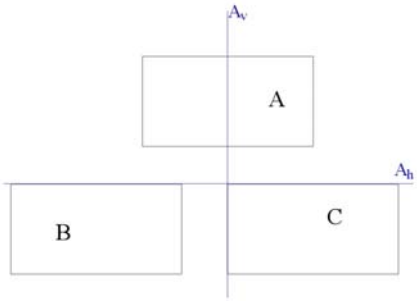
In the 8th step, the elements joining the three basic units, namely the corridor, the staircase and the walkway, are added to the representation.

In the 9th step, the configuration is explored by changing the rectangular units into

triangular ones. This involves removing the regulator $\Delta\mathbf{T_h}$, and rotating the regulator $\Delta\mathbf{T_v}$ by 45º. Since the window is a subshape of the enclosure, it is redefined according to the new regulators of the enclosure. Furthermore, the window is moved from one side to another by moving its starting point, $\bar{s}_5$.

In the 10th step the configuration is explored, once more, by changing the triangular units into semicircular ones. This involves replacing the regulator $\Delta\mathbf{T_v}$ by the regulator $\Delta\mathbf{R}$.

The last step of Table 6.7 shows the completed configuration, emphasizing the differences between the individual units and their relational constraints. The regulators $\Delta\mathbf{T_B}$ and $\Delta\mathbf{T_C}$ relate only the starting points and specific regulators, not the whole units as in previous configurations.

1	INSTANTIATE_SHAPE $unit_A$ $unit_A =$ $\Delta\mathbf{M^1}[\{\bar{p},\bar{t},n\}($ $\Delta\mathbf{T_h^1}[\ \{\bar{p},\bar{t},d,n\}\ ($ $\Delta\mathbf{T_v^1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_0)^{<0-n>}]$ $)^{<0-n>}]$ $)^{<0><1>}]$	
2	ADD_SUCCESSIVE($\Delta\mathbf{T_B}$ and $\Delta\mathbf{T_C}$) $unit_B = \Delta\mathbf{T_B^1}[\ \{\bar{p},\bar{t},d,n\}\ (unit_A)^{<0><1>}]$ $unit_C = \Delta\mathbf{T_C^1}[\ \{\bar{p},\bar{t},d,n\}\ (unit_A)^{<0><1>}]$	
3	ADD_SHARED ($\Phi\mathbf{A_v}$ and $\Phi\mathbf{A_h}$) $\Phi\mathbf{A_v^1}[\ \{\bar{p},\bar{t}\}\ (unit_A\langle\Delta\mathbf{M^1}\rangle, unit_B)]$ $\Phi\mathbf{A_h^1}[\ \{\bar{p},\bar{t}\}\ (unit_B, unit_C)]$	

4 DISTRIBUTE

ADD_SUCCESSIVE ($\Delta\mathbf{D}$)

$unit_A = \Delta\mathbf{M^1}[\{\bar{p},\bar{t},n\}\ (enclosure_A \wedge line_A)^{<0><1>}]$

$enclosure_A =$

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

$\Delta\mathbf{T_h^1}[\ \{\bar{p},\bar{t},d,n\}\ ($

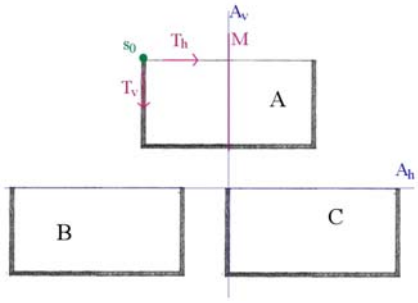
$\Delta\mathbf{T_v^1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_0)_{\#n}^{<0-n>}]$

$)^{<0-n>}]$

$)^{<0><1>}]$

$line_A = \Delta\mathbf{T_h^1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_0)^{<0-n>}]$

$\Phi\mathbf{J}\ [\ (enclosure_A, line_A)]$



5 ADD_SHARED ($\Phi\mathbf{L}, \Phi\mathbf{A}_A, \Phi\mathbf{A}_B, \Phi\mathbf{A}_C$)

$p_A = line_A\langle\bar{e}_h\rangle$

$p_B = line_B\langle\bar{e}_h\rangle$

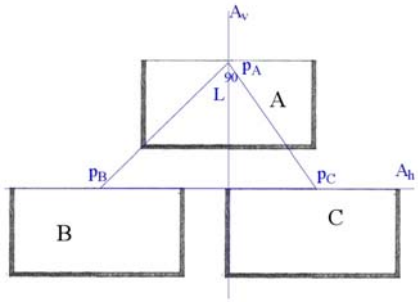
$p_C = line_C\langle\bar{e}_h\rangle$

$\Phi\mathbf{A_A^1}[\ \{\bar{p},\bar{t},n\}\ (p_A, \Delta\mathbf{M_A^1})]$

$\Phi\mathbf{A_B^1}[\ \{\bar{p},\bar{t},n\}\ (p_B, \Delta\mathbf{M_B^1})]$

$\Phi\mathbf{A_C^1}[\ \{\bar{p},\bar{t},n\}\ (p_C, \Delta\mathbf{M_C^1})]$

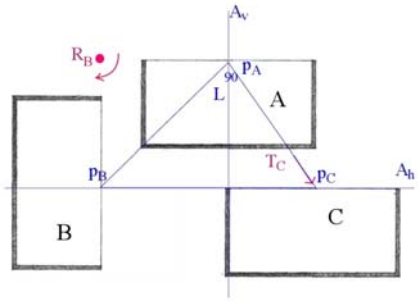
$\Phi\mathbf{L^0}[\ \{\bar{p},\bar{t},n\}\ (p_A, p_B, p_C)]$



6 REPLACE_REGULATOR ($\Delta\mathbf{T_B}$, $\Delta\mathbf{R_B}$)

$unit_B = \Delta\mathbf{R_B^0}[\ \{\bar{p},90°,n\}\ (unit_A)^{<0><1>}]$

$unit_C = \Delta\mathbf{T_C^1}[\ \{\bar{p},\bar{t},d,n\}\ unit_A)^{<0><1>}]$



7 INSTANTIATE_SHAPES

$unit_A =$

$\Delta\mathbf{M^1}[\{\bar{p},\bar{t},n\}\ (enclosure_A, articulations_A)^{<0><1>}] \wedge$

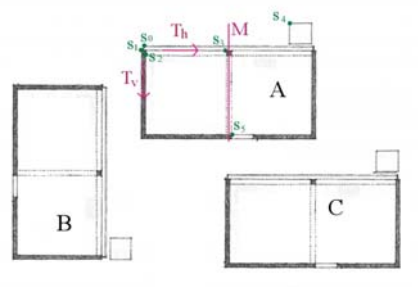
$window_A, square_A, column_A$

$articulations_A =$

$\Delta\mathbf{T_h^1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_0,\bar{s}_1,\bar{s}_2,)^{<0-n>}]$

$\Delta\mathbf{T_v^1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_3)^{<0-n>}]$

$\Phi\mathbf{J}\ [\ (enclosure_A, articualtions_A)]$



$column_A =$

$\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{h}}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{v}}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_3)^{<0-1>}])^{<0-1>}]$

$square_A =$

$\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{h}}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{v}}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_4)^{<0-1>}])^{<0-1>}]$

$window_A =$

$\Omega\mathbf{U}[\ (\Delta\mathbf{S}[\ (enclosure_A, opening_A)])\ frame]$

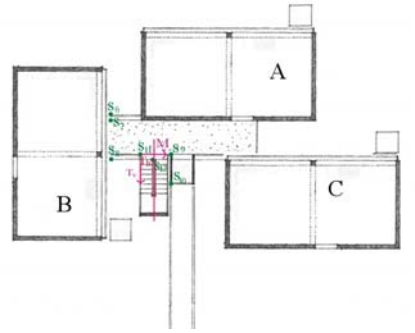
$opening = \Delta\mathbf{D}^{\mathbf{0}}[\ \{\bar{p},\bar{k},n\}\ (\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{h}}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_5)^{<0-n>}])^{<0-n>}$

$frame = \Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{v}}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{h}}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_5)^{<0-n>}])^{<0><1>}$

$\Pi\mathbf{S}\ [\ (enclosure_A, opening_A)]$

$\Pi\mathbf{S}\ [\ (enclosure_A, frame_A)]$

8 INSTANTIATE_SHAPES



$corridor =$

$\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{h}}[\ \{\bar{p},\bar{t},d,n\}\ ($

$\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{v}}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_6)^{<0-1>}], \bar{s}_7, s_8$

$)^{<0-1>}]$

$walkway =$

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

$\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{h}}[\ \{\bar{p},\bar{t},d,n\}\ ($

$\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{v}}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_9)^{<0-n>}_{\#n}]$

$)^{<0-n>}]$

$\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{v}}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_{10})^{<0-n>}]$

$)^{<0-n>}]$

$staircase = \Delta\mathbf{M}^{\mathbf{1}}[\ \{\bar{p},\bar{t},1\}\ (treads, outline)^{<0><1>}]$

$treads =$

$\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{v}}[\ \{\bar{p},\bar{t},d,n\}\ ($

$\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{h}}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_{11})^{<0-n>}]$

$)^{<0><n>}]$

$outline =$

$\Delta\mathbf{D}^{\mathbf{0}}[\ \{\bar{p},\bar{t},d,n\}\ ($

$\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{h}}[\ \{\bar{p},\bar{t},d,n\}\ ($

$\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{v}}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_{11}, \bar{s}_{12})^{<0-n>}_{\#n}]$

$)^{<0-n>}]$

$)^{<0-n>}]$

9 DELETE_REGULATOR ($enclosure\left\langle \Delta \mathbf{T_h} \right\rangle$)

ROTATE_REGULATOR ($\Delta \mathbf{T_v}$)

MOVE_SHAPE (*window*)

$enclosure_A =$

$\Delta \mathbf{D^0}[\ \{\bar{p},\bar{k},n\}$

$\Delta \mathbf{T_v^1}[\ \{\bar{p},\bar{t}_{45},d,n\}\ (\bar{s}_0)^{<0-n>}\ ]$

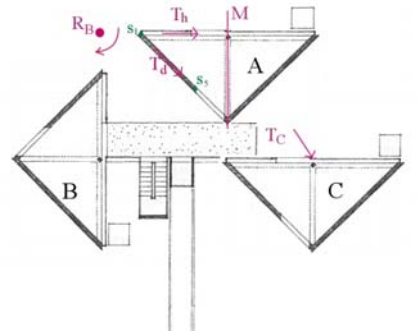
$)^{<0-n>}\ ]$

$opening =$

$\Delta \mathbf{D^0}[\ \{\bar{p},\bar{k},n\}\ (\Delta \mathbf{T_v^1}[\ \{\bar{p},\bar{t}_{45},d,n\}\ (\bar{s}_5)^{<0-n>}\ ])^{<0-n>}\ ]$

$frame =$

$\Delta \mathbf{T_e^1}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta \mathbf{T_v^1}[\ \{\bar{p},\bar{t}_{45},d,n\}\ (\bar{s}_5)^{<0-n>}\ ])^{<0><1>}\ ]$



10 REPLACE_REGULATOR ($\Delta \mathbf{T_v}$)

$enclosure_A =$

$\Delta \mathbf{D^0}[\ \{\bar{p},\bar{k},n\}$

$\Delta \mathbf{R_v^0}[\ \{\bar{p},\bar{t},90,n\}\ (\bar{s}_A)^{<0-n>}\ ]$

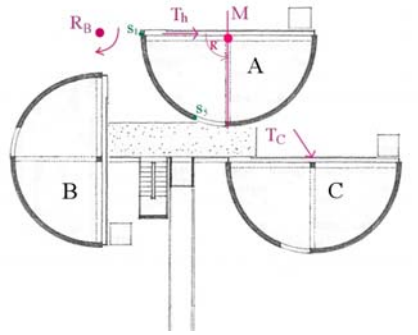
$)^{<0-n>}\ ]$

$opening =$

$\Delta \mathbf{D^0}[\ \{\bar{p},\bar{k},n\}\ (\Delta \mathbf{R^0}[\ \{\bar{p},\bar{t},18,n\}\ (\bar{s}_5)^{<0-n>}\ ])^{<0-n>}\ ]$

$frame =$

$\Delta \mathbf{T_v^1}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta \mathbf{R^0}[\ \{\bar{p},\bar{t},18,n\}\ (\bar{s}_5)^{<0-n>}\ ])^{<0><1>}\ ]$



11 COMPLETE CONFIGURATION (summary)

$HalfHouse = unit_A \wedge unit_B \wedge unit_C \wedge commonlinks$

$commonlinks = corridor \wedge walkway \wedge staircase$

$unit_A =$

$\Delta \mathbf{M^1}[\{\bar{p},\bar{t},n\}\ (enclosure_A, articulations_A)^{<0><1>}\ ] \wedge$

$window_A, square_A, column_A$

$enclosure_A =$

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

$\Delta \mathbf{T_h^1}[\ \{\bar{p},\bar{t},d,n\}\ ($

$\Delta \mathbf{T_v^1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_A)_{\#n}^{<0-n>}\ ]$

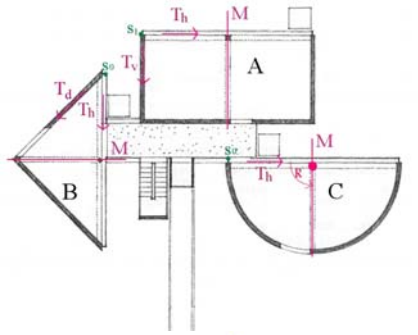
$)^{<0-n>}\ ]$

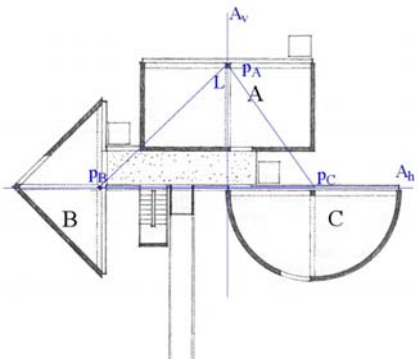
$)^{<0-1>}\ ]$

$unit_B =$

$\Delta \mathbf{M^1}[\{\bar{p},\bar{t},n\}\ (enclosure_B, articulations_B)^{<0><1>}\ ] \wedge$

$window_B, square_B, column_B$





$$
\begin{aligned}
&enclosure_B = \\
&\quad \Delta\mathbf{D^0}[\ \{\bar{p},\bar{k},n\} \\
&\qquad \Delta\mathbf{T_v^1}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_B)^{<0-n>}\ ] \\
&\quad)^{<0-n>}\ ] \\
&\\
&unit_C = \\
&\quad \Delta\mathbf{M^1}[\{\bar{p},\bar{t},n\}\ (enclosure_C, articulations_C)^{<0><1>}\ ] \wedge \\
&\quad window_C, square_C, column_C \\
&enclosure_C = \\
&\quad \Delta\mathbf{D^0}[\ \{\bar{p},\bar{k},n\} \\
&\qquad \Delta\mathbf{R^0}[\ \{\bar{p},\bar{t},90,n\}\ (\bar{s}_C)^{<0-n>}\ ] \\
&\quad)^{<0-n>}\ ] \\
&\\
&s_C = \Delta\mathbf{R_C^1}[\ \{\bar{p},90°,n\}\ (s_A)^{<0><1>}\ ] \\
&\Delta\mathbf{M_C^1} = \Delta\mathbf{R_C^1}[\ \{\bar{p},90°,n\}\ (\Delta\mathbf{M_A^1})^{<0><1>}\ ] \\
&\Delta\mathbf{T_{Ch}^1} = \Delta\mathbf{R_C^1}[\ \{\bar{p},90°,n\}\ (\Delta\mathbf{T_{Ah}^1})^{<0><1>}\ ] \\
&s_B = \Delta\mathbf{T_B^0}[\ \{\bar{p},\bar{t},d,n\}\ (s_A)^{<0><1>}\ ] \\
&\Delta\mathbf{M_B^1} = \Delta\mathbf{T_B^0}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{M_A^1})^{<0><1>}\ ] \\
&\Delta\mathbf{T_{Bh}^1} = \Delta\mathbf{T_B^0}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_{Ah}^1})^{<0><1>}\ ] \\
&\\
&p_A = line_A\langle\bar{e}_h\rangle \\
&p_B = line_B\langle\bar{e}_h\rangle \\
&p_C = line_C\langle\bar{e}_h\rangle \\
&\Phi\mathbf{A^1}[\ \{\bar{p},\bar{t},n\}\ (p_A, \Delta\mathbf{M_A^1})] \\
&\Phi\mathbf{A^1}[\ \{\bar{p},\bar{t},n\}\ (p_B, \Delta\mathbf{M_B^1})] \\
&\Phi\mathbf{A^1}[\ \{\bar{p},\bar{t},n\}\ (p_C, \Delta\mathbf{M_C^1})] \\
&\Phi\mathbf{L^0}[\ \{\bar{p},\bar{t},n\}\ (p_A, p_B, p_C)] \\
&\\
&\Phi\mathbf{A_v^1}[\ \{\bar{p},\bar{t}\}\ (unit_A\langle\Delta\mathbf{M^1}\rangle, unit_B)] \\
&\Phi\mathbf{A_h^1}[\ \{\bar{p},\bar{t}\}\ (unit_B, unit_C)]
\end{aligned}
$$

FIGURE 6.7 – HYPOTHETICAL GENERATION OF HALF HOUSE

6.2.2. THE TRANSFORMATION OF HALF HOUSE TO HOUSE 10

Table 6.7, which extends through several pages, shows a set of exploratory manipulations used while generating Half House. These include instantiation of elements, replacing elements, increasing in the levels of details, identification of constraints, and adjusting to

fit constraints. Table 6.8 uses similar manipulations to transform Half House to House 10.

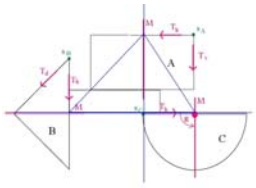
The 1st step of Table 6.8 represents the abstraction of Half House into its basic primary shapes and their relations. In the 2nd step, half house is rotated 180 degrees. This involves rotating the direction vectors of the shape's defining regulators.

In the 3rd step, the dimension of the corridor is changed and the rectangular and triangular units, $unit_A$ and $unit_B$, are moved to new positions. Resizing the corridors involves changing the distance factor of its horizontal regulator, and moving the units involves moving their starting points. The vertical alignment is broken, and the triangle connecting all three units is no longer a right angle.

In the 4th step, the shapes of the three units are transformed by rotating their mirror lines 45 degrees. The rotation degree of $unit_C$ is updated to 135º, the regulator $\Delta\mathbf{T_h}$ is inserted in $unit_B$ to achieve the trapezoidal shape, and the right angle is re-established when the midpoint of $unit_A$ is repositioned.

In the 5th step, the walls thickness is defined by means of a Dilation regulator. In the 6th step, the corridor is redefined to accommodate the curvilinear spaces, using the technique of partial succession of regulators (used for polylines). All the regulators defining the corridor are either Translation or Curve regulators. These are referred to as $\Delta\mathbf{T_1} - \Delta\mathbf{T_n}$ or $\Delta\mathbf{C_1} - \Delta\mathbf{C_n}$. The last step of Table 6.8 shows House 10 in its final form, emphasizing the units and their relationships.

1



$$HalfHouse = unit_A \wedge unit_B \wedge unit_C \wedge corridor$$

$$unit_A = \Delta\mathbf{M^0}[\ \{\bar{p},\bar{t},n\}\ (\Delta\mathbf{T_h^1}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_v^1}[\ \{\bar{p},\bar{t}_{90},d,n\}\ (\bar{s}_A)^{<0-n>}\ ])^{<0-n>}\ ])^{<0><1>}\ ]$$

$$unit_B = \Delta\mathbf{M^0}[\ \{\bar{p},\bar{t},n\}\ (\Delta\mathbf{T_h^1}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T_d^1}[\ \{\bar{p},\bar{t}_{45},d,n\}\ (\bar{s}_0)^{<0-n>}\ ])^{<0-n>}\ ])^{<0><1>}\ ]$$

$$unit_C = \Delta\mathbf{M^0}[\ \{\bar{p},\bar{t},n\}\ (\Delta\mathbf{T_h^1}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{R^1}[\ \{\bar{p},\bar{t},90,n\}\ (\bar{s}_0)^{<0-n>}\ ])^{<0-n>}\ ])^{<0><1>}\ ]$$

$corridor = \Delta\mathbf{T}_{\mathbf{h}}^{\mathbf{1}}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T}_{\mathbf{v}}^{\mathbf{1}}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_6)^{<0-1>}],\bar{s}_7,s_8)^{<0-1>}]$

$p_A = line_A\langle\bar{e}_h\rangle$

$p_B = line_B\langle\bar{e}_h\rangle$

$p_C = line_C\langle\bar{e}_h\rangle$

$\Phi\mathbf{A}^{\mathbf{1}}[\ \{\bar{p},\bar{t},n\}\ (p_A,\Delta\mathbf{M}_{\mathbf{A}}^{\mathbf{1}})]$

$\Phi\mathbf{A}^{\mathbf{1}}[\ \{\bar{p},\bar{t},n\}\ (p_B,\Delta\mathbf{M}_{\mathbf{B}}^{\mathbf{1}})]$

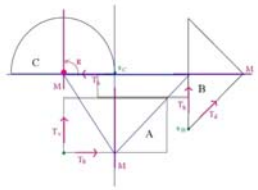
$\Phi\mathbf{A}^{\mathbf{1}}[\ \{\bar{p},\bar{t},n\}\ (p_C,\Delta\mathbf{M}_{\mathbf{C}}^{\mathbf{1}})]$

$\Phi\mathbf{L}^{\mathbf{0}}[\ \{\bar{p},\bar{t},n\}\ (p_A,p_B,p_C)]$

$\Phi\mathbf{A}_{\mathbf{v}}^{\mathbf{1}}[\ \{\bar{p},\bar{t}\}\ (unit_A\langle\Delta\mathbf{M}^{\mathbf{1}}\rangle,unit_B)]$

$\Phi\mathbf{A}_{\mathbf{h}}^{\mathbf{1}}[\ \{\bar{p},\bar{t}\}\ (unit_B,unit_C)]$

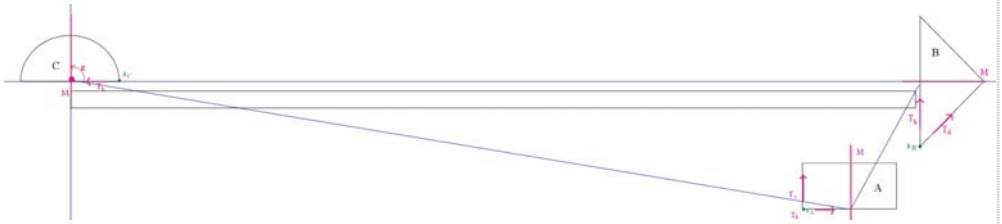
2



ROTATE_SHAPE (*HalfHouse*)

$\Delta\mathbf{R}^{\mathbf{1}}[\{\bar{p},\bar{t},180°,n\}(HalfHouse)^{<1>}] = \Delta\mathbf{R}^{\mathbf{1}}[\{\bar{p},\bar{t},180°,n\}(unit_A \wedge unit_B \wedge unit_C \wedge corridor)^{<1>}]$

3



ADJUST_FACTOR ($\Delta\mathbf{T}_{\mathbf{h}}\{d\}$)

$corridor = \Delta\mathbf{T}_{\mathbf{h}}^{\mathbf{1}}[\ \{\bar{p},\bar{t},5d,n\}\ (\Delta\mathbf{T}_{\mathbf{v}}^{\mathbf{1}}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_6)^{<0-1>}],\bar{s}_7,s_8)^{<0-1>}]$

DELETE_REGULATOR ($\Delta\mathbf{A}_{\mathbf{v}}$)

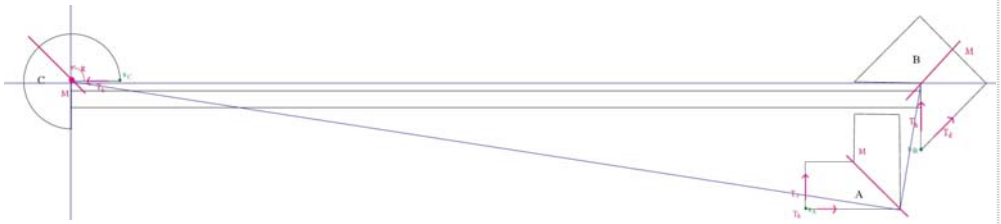
$\Phi\mathbf{A}_{\mathbf{h}}^{\mathbf{1}}[\ \{\bar{p},\bar{t}\}\ (unit_B,unit_C)]$

ADJUST_FACTOR ($\Delta\mathbf{L}\{v\}$)

$\Phi\mathbf{L}^{\mathbf{0}}[\ \{100°\}\ (p_A,p_B,p_C)]$

MOVE_SHAPE ($unit_A$, $unit_B$)

4



ROTATE_REGULATOR ($\Delta\mathbf{M_A}, \Delta\mathbf{M_B}, \Delta\mathbf{M_C}$)

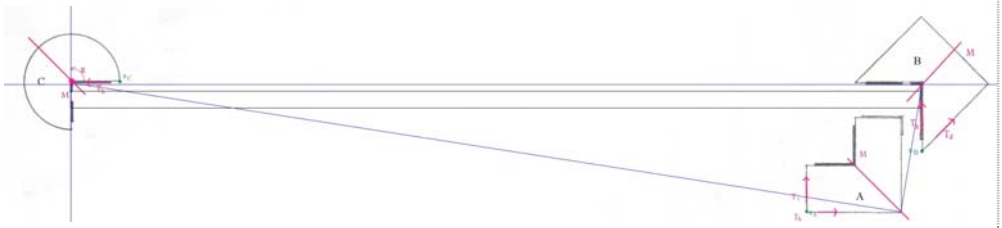
$unit_A = \Delta\mathbf{M^0}[\ \{\bar{p}, \bar{t}_{45}, n\}\ (\Delta\mathbf{T_h^1}[\ \{\bar{p}, \bar{t}, d, n\}\ (\Delta\mathbf{T_v^1}[\ \{\bar{p}, \bar{t}, d, n\}\ (\bar{s}_A)^{<0-n>}\ ])^{<0-n>}\ ])^{<0><1>}\ ]$

$unit_B = \Delta\mathbf{M^0}[\ \{\bar{p}, \bar{t}_{45}, n\}\ (\Delta\mathbf{T_h^1}\Delta\mathbf{D_h^0}[\ \{\bar{p}, \bar{t}, d, n\}\ (\Delta\mathbf{T_d^1}[\ \{\bar{p}, \bar{t}_{45}, d, n\}\ (\bar{s}_0)^{<0-n>}\ ])^{<0-n>}\ ])^{<0><1>}\ ]$

$unit_C = \Delta\mathbf{M^0}[\ \{\bar{p}, \bar{t}_{45}, n\}\ (\Delta\mathbf{T_h^1}[\ \{\bar{p}, \bar{t}, d, n\}\ (\Delta\mathbf{R^1}[\ \{\bar{p}, \bar{t}, 135, n\}\ (\bar{s}_0)^{<0-n>}\ ])^{<0-n>}\ ])^{<0><1>}\ ]$

$\Phi\mathbf{L^0}[\ \{90°\}\ (p_A, p_B, p_C)]$

5



ADD_SUCCESSIVE ($\Delta\mathbf{D}$)

$unit_A = \Delta\mathbf{M^0}[\ \{\bar{p}, \bar{t}_{45}, n\}\ ($

$\Delta\mathbf{T_h^1}[\ \{\bar{p}, \bar{t}, d, n\}\ (\Delta\mathbf{T_v^1}[\ \{\bar{p}, \bar{t}_{90}, d, n\}\ (\bar{s}_A)^{<0-n>}\ ])^{<0-n>}\ ]$

$\Delta\mathbf{D^0}[\ \{\bar{p}, \bar{k}, n\}\ (\Delta\mathbf{T_h^1}[\ \{\bar{p}, \bar{t}_{90}, d, n\}\ (\bar{s}_A)^{<0-n>}\ ])^{<0-n>}\ ]$

$)^{<0><1>}\ ]$

$unit_B = \Delta\mathbf{M^0}[\ \{\bar{p}, \bar{t}_{45}, n\}\ ($

$\Delta\mathbf{T_h^1}\Delta\mathbf{D^0}[\ \{\bar{p}, \bar{t}, d, n\}\ (\Delta\mathbf{T_d^1}[\ \{\bar{p}, \bar{t}_{45}, d, n\}\ (\bar{s}_0)^{<0-n>}\ ])^{<0-n>}\ ]$

$\Delta\mathbf{D^0}[\ \{\bar{p}, \bar{k}, n\}\ (\Delta\mathbf{T_h^1}[\ \{\bar{p}, \bar{t}_{45}, d, n\}\ (\bar{s}_0)^{<0-n>}\ ])^{<0-n>}\ ]$

$)^{<0><1>}\ ]$

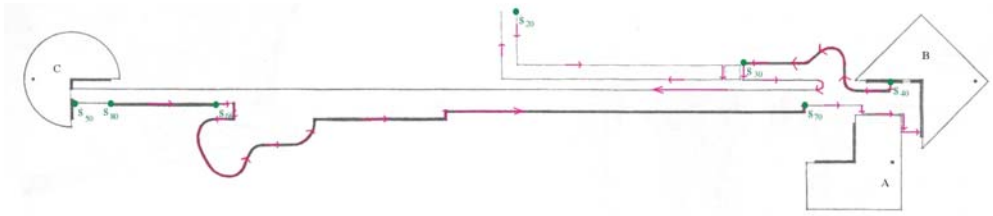
$unit_C = \Delta\mathbf{M^0}[\ \{\bar{p}, \bar{t}_{45}, n\}\ ($

$\Delta\mathbf{T_h^1}[\ \{\bar{p}, \bar{t}, d, n\}\ (\Delta\mathbf{R^1}[\ \{\bar{p}, \bar{t}, 135, n\}\ (\bar{s}_0)^{<0-n>}\ ])^{<0-n>}\ ]$

$\Delta\mathbf{D^0}[\ \{\bar{p}, \bar{k}, n\}\ (\Delta\mathbf{T_h^1}[\ \{\bar{p}, \bar{t}, d, n\}\ (\bar{s}_0)^{<0-n>}\ ])^{<0-n>}\ ]$

$)^{<0><1>}\ ]$

6



INSTANTIATE_SHAPES

$upper\_walkway =$

$\Delta\mathbf{T}_{d5}(\Delta\mathbf{T}_{d4}(\Delta\mathbf{T}_{d3}(\Delta\mathbf{T}_{d2}(\Delta\mathbf{T}_{d1}(\bar{s}_{20})_{\#n})_{\#n})_{\#n})_{\#n})$

$upper\_corridor =$

$\Delta\mathbf{T}_{d15}(\Delta\mathbf{C}_{d14}(\Delta\mathbf{T}_{d13}(\Delta\mathbf{T}_{d12}(\bar{s}_{30})_{\#n})_{\#n})_{\#n})$

$uppper\_curvilinear\_space =$

$$\Delta\mathbf{D}\ (\Delta\mathbf{T}_{d11}\Delta\mathbf{C}_{d10}(\Delta\mathbf{C}_{d9}(\Delta\mathbf{C}_{d8}(\Delta\mathbf{T}_{d7}(\Delta\mathbf{T}_{d6}(\bar{s}_{40})_{\#n})_{\#n})_{\#n})_{\#n})_{\#n})_{\#n}))$$

$lower\_corridor =$

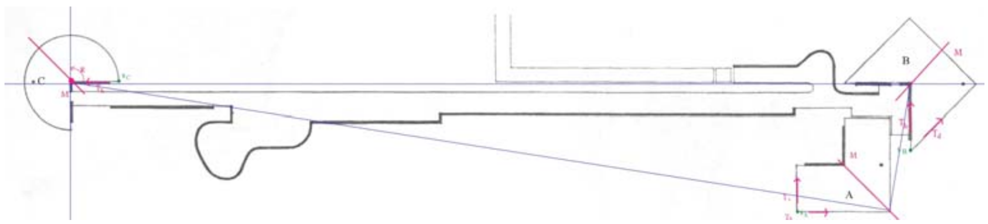
$$\Delta\mathbf{T}_{d16}(\bar{s}_{50}) \wedge \Delta\mathbf{T}_{d17}(\bar{s}_{60}) \wedge \Delta\mathbf{T}_{d22}(\Delta\mathbf{T}_{d21}(\Delta\mathbf{T}_{d20}(\Delta\mathbf{T}_{d19}(\Delta\mathbf{T}_{d18}(\bar{s}_{70})_{\#n})_{\#n})_{\#n})_{\#n})$$

$lower\_curvilinear\_space =$

$$\Delta\mathbf{D}\ (\Delta\mathbf{T}_{d34}(\Delta\mathbf{T}_{d33}(\Delta\mathbf{C}_{d32}(\Delta\mathbf{C}_{d31}(\Delta\mathbf{C}_{d30}(\Delta\mathbf{C}_{d29}(\Delta\mathbf{C}_{d28}(\Delta\mathbf{C}_{d27}(\Delta\mathbf{T}_{d26}(\Delta\mathbf{T}_{d25}(\Delta\mathbf{T}_{d24}(\Delta\mathbf{T}_{d23}(\bar{s}_{80})$$

$$_{\#n})_{\#n})_{\#n})_{\#n})_{\#n})_{\#n})_{\#n})_{\#n})_{\#n})_{\#n})_{\#n})_{\#n})_{\#n}))$$

7



COMPLETE CONFIGURATION (SUMMARY)

$House10 = unit_A \wedge unit_B \wedge unit_C \wedge corridor$

$corridor =$

$upper\_walkway \wedge upper\_corridor \wedge upper\_curvilinear\_space \wedge$
$lower\_corridor \wedge lower\_curvilinear\_space$

$$unit_A = \Delta\mathbf{M}^{\mathbf{0}}[\ \{\bar{p},\bar{t}_{45},n\}\ ($$
$$\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{h}}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{v}}[\ \{\bar{p},\bar{t}_{90},d,n\}\ (\bar{s}_A)^{<0-n>}])^{<0-n>}]$$
$$\Delta\mathbf{D}^{\mathbf{0}}[\ \{\bar{p},\bar{k},n\}\ (\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{h}}[\ \{\bar{p},\bar{t}_{90},d,n\}\ (\bar{s}_A)^{<0-n>}])^{<0-n>}]$$
$$)^{<0><1>}]$$

$$unit_B = \Delta\mathbf{M}^{\mathbf{0}}[\ \{\bar{p},\bar{t}_{45},n\}\ ($$
$$\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{h}}\Delta\mathbf{D}^{\mathbf{0}}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{d}}[\ \{\bar{p},\bar{t}_{45},d,n\}\ (\bar{s}_0)^{<0-n>}])^{<0-n>}]$$
$$\Delta\mathbf{D}^{\mathbf{0}}[\ \{\bar{p},\bar{k},n\}\ (\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{h}}[\ \{\bar{p},\bar{t}_{45},d,n\}\ (\bar{s}_0)^{<0-n>}])^{<0-n>}]$$
$$)^{<0><1>}]$$

$$unit_C = \Delta\mathbf{M}^{\mathbf{0}}[\ \{\bar{p},\bar{t}_{45},n\}\ ($$
$$\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{h}}[\ \{\bar{p},\bar{t},d,n\}\ (\Delta\mathbf{R}^{\mathbf{1}}[\ \{\bar{p},\bar{t},135,n\}\ (\bar{s}_0)^{<0-n>}])^{<0-n>}]$$
$$\Delta\mathbf{D}^{\mathbf{0}}[\ \{\bar{p},\bar{k},n\}\ (\Delta\mathbf{T}^{\mathbf{1}}_{\mathbf{h}}[\ \{\bar{p},\bar{t},d,n\}\ (\bar{s}_0)^{<0-n>}])^{<0-n>}]$$
$$)^{<0><1>}]$$

$$\Phi\mathbf{A}^{\mathbf{1}}_{\mathbf{h}}[\ \{\bar{p},\bar{t}\}\ (unit_B, unit_C)]$$

$$\Phi\mathbf{L}^{\mathbf{0}}[\ \{90°\}\ (p_A, p_B, p_C)]$$

TABLE 6.8 - HYPOTHETICAL TRANSFORMATION OF HALF HOUSE TO HOUSE 10

6.3. CALATRAVA'S ART MUSEUM AT MILWAUKEE

In this section, I use the ICE notation to describe the winged roof structure of Calatrava's Art Museum at Milwaukee (Figure 6.2, source: Tzonis 2004, p.291, p.298, p.299). This building was chosen because of its visual appeal, and because, in its concepts and technology, it symbolizes Calatrava's poetics of structure and movement (Tzonis 2004, p.290).

Overlooking Lake Michigan, Calatrava's Art museum at Milwaukee is exceptional for its visual connection with its environment. The sculptural elements of the roof create the impression of a great seagull landing on the shore, while the remaining parts of the building (with the bridge supported by a single inclined mass) are reminiscent of another marine image, a ship. The towering glass roof over the main hall and the system of movable wings allow one to control the light and temperature of the interior, while completely transforming its character. When closed, it is a covered protected space, and when open, it is a vast open air installation in which the distinction between interior and exterior dissolves. (Molinari 1999, p.142)

The folding roof structure, consisting of a brise-soleil, is constructed out of steel plates which are welded and stiffened inside. The two winged elements, each formed by 36 fins whose length ranges from 32 to 8 meters, are cantilevered by a rotating spine joined by five rows of tying tubular sections. The angle at which each spine meets the rotating spine is different so that when closed the brise-soleil forms a ruled surface with a conical shape. The spine and the parallel mast of the bridge are both 47º incline. The brise-soleil is controlled either manually, to accommodate the requirements of the museum, or automatically through a computer system, which responds to bad weather and excessive wind speeds by closing the brise-soleil. (Tzonis 2004, p.290).

a- Closed wings

b- Open wings, side view

c- Back view against Lake Michigan

d- Brise-soleil in operation

FIGURE 6.2 - THE ART MUSEUM AT MILWAUKEE

6.3.1. DESCRIBING THE ROOF STRUCTURE USING ICE

Table 6.9 describes the step by step description, using the ICE notation, of the roof structure and mast of Calatrava's Art Museum at Milwaukee. The figures in the table are all generated using the ICE-3D implementation.

The 1st step of Table 6.9 describes the beam means of its generative regulators. The 2nd and 3rd steps describe the generation of the conic roof structure, by using circular and diagonal Alignments to define the half-conic, then a Mirror regulator to generate the whole conic structure. Within these Alignment regulators, the diagonal line, which is composed with a generative translation, aligns the start-point of the beams, while the circle aligns the endpoints.

The 4th and 5th steps describe the wings, by using Translation composed with Alignments and Gradation regulators to define one wing, then a Mirror regulator to generate both

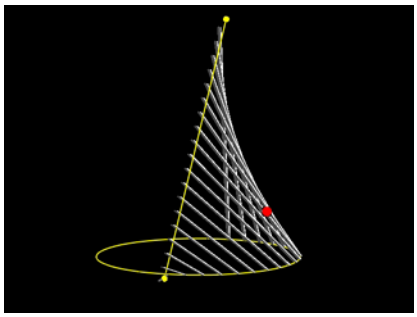
wings. The Gradation includes both a gradual rotation factor and gradual scale factor.

The 6th step describes the connection between the conic beams and the wings. The 7th step describes the constraints tying each beam of the conic with its corresponding beam of the wing thought its connection object. Notice that the mirror of the wings, the mirror of the connection axis, and the mirror of the conic is the same, $\Delta\mathbf{M}_{\mathbf{axis}}$ regulator.

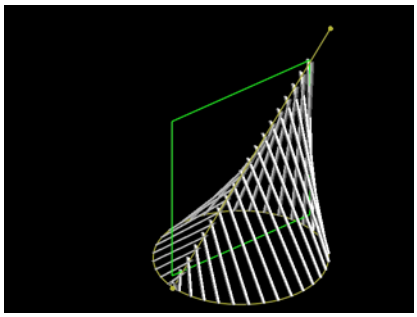
The 8th step describes the mast of the bridge by means of its generative regulators and the 9th step describes the linear supports of the bridge aligned to both the mast and the horizontal thought their endpoints.

The 10th step describes the constraint defining the inclination of both the mast and the wings-connection-conic diagonal axis to be at 47º.

The last step of Table 6.9 describes the motion of the brise-soleil by means of motion regulators, which sets the angle of the wing beams with respect to the connection axis. When the brise-soleil is closed the angle is at its minimum; when it is open the angle is at its maximum. The gradual rotation factor determines angle of each beam differently.

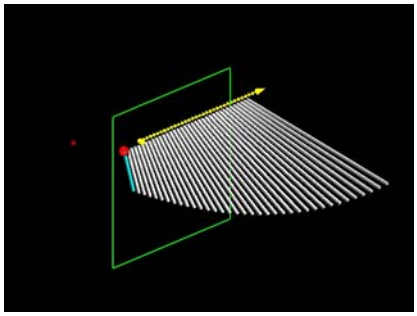
1	$beam =$ $\Delta\mathbf{T}_{\mathbf{3}}[\ \{\bar{p},\bar{t},d,n\}\ ($ $\Delta\mathbf{T}_{\mathbf{2}}[\ \{\bar{p},\bar{t},d,n\}\ ($ $\Delta\mathbf{T}_{\mathbf{1}}[\{\bar{p},\bar{t},d,n\}(\bar{s}_{beam})^{<0-1>}])^{<0-1>}])^{<0-1>}]$	
2	$half\_conic =$ $\Phi\mathbf{A}_{\mathbf{diag}}\Delta\mathbf{T}[\ \{\bar{p},\bar{t}_{47^o},n=18\}\ (beam_{conic}\langle\bar{s}\rangle)^{<0>-<18>}$ $\Phi\mathbf{A}_{\mathbf{circ}}[\ \{\bar{p},\bar{t}_{0^o},n=18\}\ (beam_{conic}\langle\bar{e}\rangle)^{<0>-<18>}]$	

3 $conic = \Delta\mathbf{M}_{\mathbf{axis}}[\ \{\bar{p},\bar{t},n\}\ (half\_conic)^{<0><1>}]$



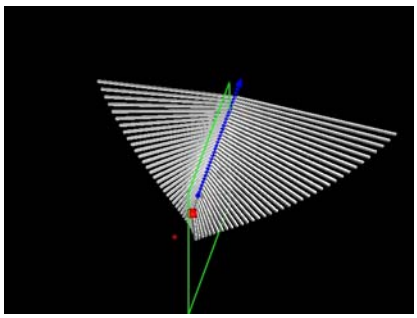
4 $wing =$

$\Delta\mathbf{T}\,\Xi\mathbf{G}\Phi\mathbf{A}[\{\bar{p},\bar{t},\bar{k},\theta,36\}\ (beam_{wing})^{<0>-<n>}]$



5 $both\_wings =$

$\Delta\mathbf{M}_{\mathbf{axis}}[\ \{\bar{p},\bar{t},n\}\ (wing)^{<0><1>}]$



6 $connection\_axis =$

$\Delta\mathbf{M}_{\mathbf{axis}}[\ \{\bar{p},\bar{t},n\}\ ($

$\Delta\mathbf{T}\ [\ \{\bar{p},\bar{t},d,n\}\ (connection)^{<0>-<n>}]$

$)^{<0><1>}] \wedge pointedEnd$

7 $connection\_constraint =$

$\Phi\mathbf{A}^{\mathbf{0}}_{\mathbf{i}}[\ \{\bar{p},\bar{t},n\}$

$(connection_i, beam_{conic-i}, beam_{wing-i})\ ]$

8 $mast\_axis =$

$\Delta\mathbf{TD}\ [\ \{\bar{p},\bar{t},n\}\ ($

$\Delta\mathbf{R}\ [\ \{\bar{p},\bar{t},\theta,n\}\ (s_{mast})^{<0>-<n>}]$

$)^{<0><1>}]$

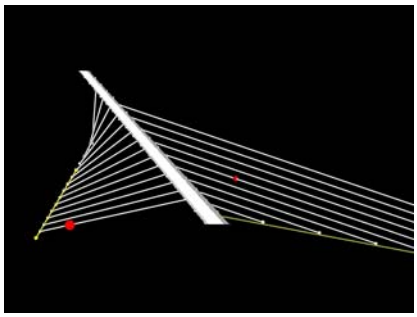
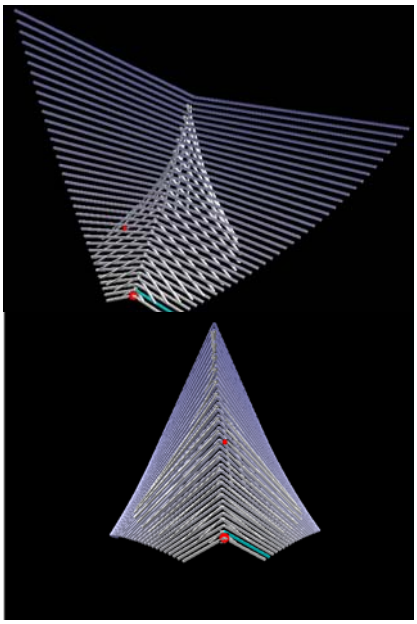
9	*linear_supports* = $\Phi\mathbf{A}_{\mathbf{diag}}[\ \{\bar{p},\bar{t}_{47^\circ},n=9\}\ (support\langle\bar{s}\rangle)^{<0>-<9>}]\wedge$ $\Phi\mathbf{A}_{\mathbf{horizontal}}[\ \{\bar{p},\bar{t}_{0^\circ},n=9\}\ (support\langle\bar{e}\rangle)^{<0>-<9>}]$	
10	*inclination_constraint* = $\Delta\mathbf{L}[\{\theta = 47^\circ\}\ (mast\_axis)(connection\_axis)]$	
11	*both_wings* = *IN_MOTION* $\Delta\mathbf{M}_{\mathbf{axis}}[\ \{\bar{p},\bar{t},n\}\ ($ $\Delta\mathbf{T}\,\Xi\mathbf{G}\Phi\mathbf{A}\ [\ \{\bar{p},\bar{t},\bar{k},0\longrightarrow\theta^\circ,n=36\}(beam_{wing})^{<0>-<36>}]$ $)^{<0><1>}]\wedge$ $\Delta\mathbf{L}\ [\ \{0\longrightarrow\theta^\circ\}\ (beam_{wing})(beam_{connection})\ ]$ 	

TABLE 6.9 - THE ART MUSEUM AT MILWAUKEE

6.4. ETHNOGRAPHIC EXAMPLE

In this example, the ICE notation is used for describing snapshots from an annotated design studio, in which the entire graphic output of a student and the annotations of her faculty have been ethnographically recorded. This is a realistic design situation, in which the configuration is evolving and several ideas are explored. The ICE notation enables the formal and unambiguous codification the design process in stages, which are defined by each drawing in the sequence of the design development, as well as the codification of transitions between those drawings.

6.4.1. THE ANNOTATED STUDIO

A vertical design studio in the School of Architecture, at Carnegie Mellon University was offered during the summer of 2002, by Professor Omer Akin, where students ranged from 2nd year, to 5th year of their college education. The entire studio work was recorded through digital photographs of student work brought to each class session and the midterm and final reviews (Akin, 2004). These graphic records were accompanied by daily diary annotations kept by the instructor for each student's progress as well as the overall progress of the studio. Three different problems emerged: international housing prototype, dormitory housing, and a toy manufacturer's headquarters building.

6.4.2. SNAPSHOTS FROM THE DESIGN STUDIO

In this section, I present a sequence of sketches and models created by Subject-W for her[1] dormitory housing project. This sequence starts about a quarter of the way into the studio and runs through to the end, highlighting all major formal solutions produced. The ICE description is a hypothetical analysis of each drawing after the completion of the project; these designs have been generated by Subject-W independent of ICE. Subject-W's project was selected mainly because she constantly changes directions in her development. It gives the opportunity to illustrate how the ICE notation represents changes in actual design exploration paths.

[1] We used "she" or "her" to refer to all subjects--students and critics--of the annotated studio for the purpose of anonymity. No gender implications are intended.

In Subject-W's project, the main design constructs repeated throughout the drawing sequence are rooms, dorm units, entrances and common spaces. Subject-W's repeatedly reorganizes these elements and further details are not completely resolved in most of her drawings. Therefore, the ICE notation is used to represent these elements and the evolving relations between them.

Table 6.10 shows the sequence of Subject-W's drawings, their corresponding ICE representation, and steps to transform one drawing to the next. Notice how Subject-W alternates between the sketchy mode and the refined drawings/models, with the sketchier ones expressing re-organization of ideas.

In the 1st step of Table 6.10, "Subject-W begins by drawing her housing hierarchy of rooms, units, unit-clusters, wings, buildings, and building-clusters by means of a seemingly unstructured swirling shape" (Akin 2004). With the assumption that the whole drawing represents a building and its individual flower-like objects are abstractions of dorm units, it is possible to identify the underlying structure, which can be described in ICE by means of Rotation, Curve, and Dilation regulators. In the 2nd step of Table 6.10, "Subject-W presents a rectilinear scheme in which the modular bays of the dormitory are clustered to create a large and integrated form on the site, creating a 'beads-on-a-string' type scheme" (Akin 2004). The building's structure can be generated in ICE through the following steps: (1) reflecting the dorm unit to create the dorm cluster, (2) translating and rotating the dorm cluster (4) and reflecting these to generate the whole building. Similarly, the entrance is defined by sweeping points along a curve, then reflecting the curved lines.

The configuration in steps 3 and 4 go back to the curvilinear theme. In the 3rd step, "the beads-on-a-string type arrangement has yielded to a "serpentine" form that curves with the contours, creating a concave edge for the public and a convex one for the private side of the site lot" (Akin 2004). This serpentine form is described by sweeping a point along two consecutive Curve regulators, $\Delta\mathbf{C_1}$ then $\Delta\mathbf{C_2}$. Both common spaces are described in the same way. In the 4th step, "the serpentine form is refined. Curves turn into rotations and the central axis of symmetry from the previous configuration is reinstituted" (Akin 2004). The building is generated by means of Reflection and Rotation regulators. To achieve the curved axis (from the rectilinear configuration in step 2), the Translation

regulator is deleted and Rotation regulator is adjusted. The dorm units are refined and re-oriented, while the dorm cluster's reflection axis, $\Delta\mathbf{M_1}$, is rotated thought 45º.

In the 5th step of Table 6.10, "the next formal overhaul involves one end of the 'serpentine' form bifurcating into two wings, allowing the development of a 'commons' area and lobby from one of the major access edges of the site" (Akin 2004). In this configuration, the curve is broken into segments, and the rotation is replaced by a reflection, which becomes the dominant relationship in all parts of the drawing. This building is defined by reflecting the dorm cluster twice, then reflecting the individual dorm units to achieve the bifurcations. The central axis is slightly rotated and several local axes emerge.

"During midterm review, Subject-W's work shows little development over the previous critique" (Akin 2004). The most significant development is the layout of the dorm units illustrated in the 6th step. The Containment regulator, $\Psi\,\mathbf{H}$, indicates that the dorm unit consists of two successive reflections for the rooms and one for the bathroom, as well as a kitchen and a balcony.

The configurations in steps 7 and 8 are variations suggested during the midterm review. The 7th step is achieved by moving the common space, and the 8th step is achieved by rotating the dorm cluster 180 degrees and converting the Mirrors, $\Delta\mathbf{M_4}$ and $\Delta\mathbf{M_5}$, into Rotations.

In step 9, the configuration shows a return to the curvilinear axis through rotation. "A new aspect of the scheme emerges. Drawings lack architectonic qualities, such as material, construction and structural specificity" (Akin 2004). Although, this is speculation, it appears that Subject-W has created this configuration not by developing the midterm solution, but by working from the drawings in step 2, while pair-wise integrating the common spaces from the midterm's configuration. To achieve this configuration from the midterm configuration in step 5, the bifurcation Mirrors, $\Delta\mathbf{M_4}$ and $\Delta\mathbf{M_5}$, are removed, and the secondary reflection, $\Delta\mathbf{M_3}$, is replaced by a Rotation $\Delta\mathbf{R}$.

"The configuration in the 10th step is a mixed bag. While the dorm units gain

architectonic clarity, the main entrance, circulation and commons areas continue to resemble spaghetti" (Akin 2004). This configuration has the same underlying structure as the previous configuration (step 9), but with the incorporation spaghetti region of the public spaces.

In the 11th step, "the development is along the same lines as before. The "spaghetti" scheme dominates the formal development. Circulation paths are configured as tubes without incorporating circulation and social hubs" (Akin 2004). The general structure of this configuration is similar to the one in step 10, with the secondary Rotation, $\Delta\mathbf{R}$, being replaced by the reflection, $\Delta\mathbf{M}_3$.

"The 12th step marks a significant return to architecture and architectonics. The "spaghetti" is gone, dissolved in the interstitial space between two parallel dorm wings" (Akin 2004). Its structure is the same as the configuration in step 11.

The 13th step represents the submission two days prior to the final review, where "there are still basic issues of development and resolution, including the incorporation of the other building systems" (Akin 2004). It has a slight development in the dorm cluster, where a glide relationship, depicted by $\Delta\mathbf{G}$, is explored.

The final review, illustrated in the 14th step of Table 6.10, "does not bring any surprises or further development of the scheme" (Akin 2004). However, it still shows some signs of exploration in the dorm clusters. This time the units are slightly sheared. The Reflection and Glide sequence $\Delta\mathbf{G}(\Delta\mathbf{M}_1())$ is replaced by a composition of Translation and Shear $\Delta\mathbf{TS}$ regulators.

1

$$building = \Delta\mathbf{C}_3[\{\bar{p},\bar{t},\alpha,n\}\ (dormCluster)^{<0>-<7>}] \wedge commonSpace$$

$$dormcluster = \Delta\mathbf{RD}[\{\bar{p},\bar{t},\theta,\bar{k},n\}\ (flowerShape)^{<0>-<3>}]$$

$$flowershape = \Delta\mathbf{RD}[\{\bar{p},\bar{t},\theta,\bar{k},n\}\ (room)^{<0>-<4>}]$$

$$room = \Delta\mathbf{C}_1[\{\bar{p},\bar{t},\alpha,n\}\ (s_1)^{<0-n>}]$$

Thursday, May 23, 2002

Extract from *Design The Art and Science of the Synthetic* unpublished manuscript by Ömer Akin ©

$commonSpace = \Delta\mathbf{C_2}[\{\bar{p},\bar{t},\alpha,n\}\ (s_2)^{<0-n>}]$

1→2

REPLACE_SHAPE (*dormCluster*)

REPLACE_REGULATOR ($\Delta\mathbf{C_3}$)

$\Delta\mathbf{C}_3[(dormCluster)] \Rightarrow \Delta\mathbf{M}[(\Delta\mathbf{R}[(\Delta\mathbf{T}[(dormCluster)])_{\#1}])]$

2

building =

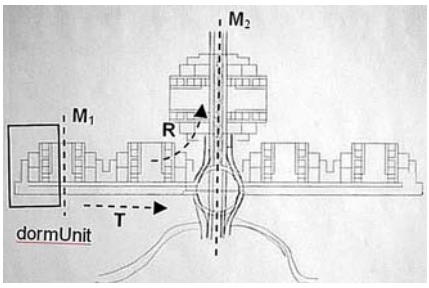
$\Delta\mathbf{M}_2[\{\bar{p},\bar{t},n\}($

$\Delta\mathbf{R}\ [\{\bar{p},\bar{t},\theta,n\}($

$\Delta\mathbf{T}[\{\bar{p},\bar{t},d,n\}(dormCluster)^{<0><1>}]$

$)_{\#1}^{<0><1>}]$

$)^{<0><1>}]$



Tuesday, June 4, 2002

dormcluster = $\Delta\mathbf{M}_1[\{\bar{p},\bar{t},n\}\ (dormUnit)^{<0><1>}]$

Entrance =

$\Delta\mathbf{M}[\{\bar{p},\bar{t},n\}\ ($

$\Delta\mathbf{C}_2[\{\bar{p},\bar{t},\alpha,n\}\ (s_2)^{<0-n>}]$

$)^{<0-n>}]$

2→4

DELETE_REGULATOR ($\Delta\mathbf{T}$)

$\Delta\mathbf{M}_2[(\Delta\mathbf{R}[(\Delta\mathbf{T}[(dormCluster)])_{\#1}])] \Rightarrow \Delta\mathbf{M}_2[(\Delta\mathbf{R}[(dormCluster)])]$

MODIFY_FACTOR ($\Delta\mathbf{R}\{\theta\}$)

$\Delta\mathbf{R}\{\theta = 90\} \Rightarrow \Delta\mathbf{R}\{\theta = 30\}$

INSERT_SHARED (*commonSpace*)

REPLACE_SHAPE (*dormUnit*)

ROTATE_REGULATOR($\Delta\mathbf{M}_1$)

$\Delta\mathbf{M}_1\{\bar{t} = (0,1)\} \Rightarrow \Delta\mathbf{M}_1\{\bar{t} = (1,1)\}$

3

building = *serpentine* ∧

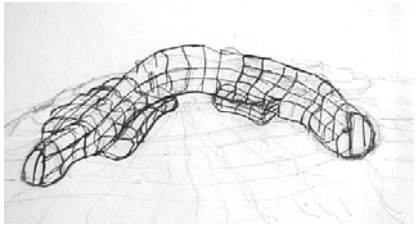
$commonSpace_1 \wedge commonSpace_2$

serpentine =

$\Delta\mathbf{C}_2[\{\bar{p},\bar{t},\alpha,n\}\ ($

$\Delta\mathbf{C}_1[\{\bar{p},\bar{t},\alpha,n\}\ (s_1)^{<0-n>}]$

$)^{<0-n>}]$



Thursday, June 6, 2002

commonSpace =

$\Delta\mathbf{C}_2[\{\bar{p},\bar{t},\alpha,n\}\ ($

$\Delta\mathbf{C}_1[\{\bar{p},\bar{t},\alpha,n\}\ (s_1)^{<0-n>}]$

$)^{<0-n>}]$

4 *building* =

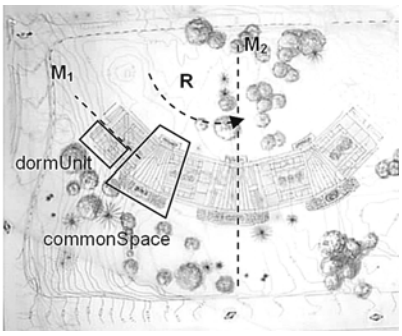
$\Delta\mathbf{M_2}[\{\bar{p},\bar{t},n\}($

$\Delta\mathbf{R}\ [\{\bar{p},\bar{t},\theta,n\}((dormCluster)^{<0><1>}]$

$)^{<0><1>}]\wedge$

$\Delta\mathbf{R}\ [\{\bar{p},\bar{t},\theta,n\}((commonSpace)^{<0><2>}]$

$dormcluster = \Delta\mathbf{M}_1[\{\bar{p},\bar{t},n\}\ \ (dormUnit)^{<0><1>}]$



Thursday, June 6, 2002

4→5

REPLACE_REGULATOR ($\Delta\mathbf{R}$)

$\Delta\mathbf{M_2}[(\Delta\mathbf{R}[(dormCluster)])] \Rightarrow \Delta\mathbf{M_2}[(\Delta\mathbf{M}_3[(dormCluster)])]$

5 *building* =

$\Delta\mathbf{M_2}[\{\bar{p},\bar{t},n\}($

$\Delta\mathbf{M_3}[\{\bar{p},\bar{t},n\}((dormCluster_1)^{<0><1>}_{\#1}]$

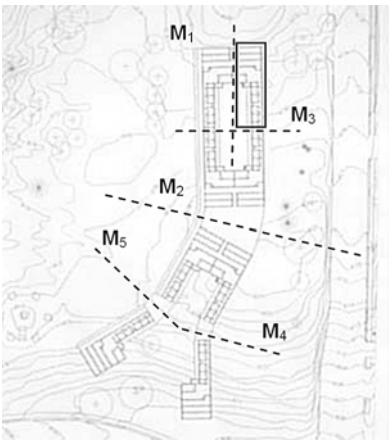
$)^{<0><1>}]\wedge$

$\Delta\mathbf{M_4}[\{\bar{p},\bar{t},n\}((dormUnit_4)^{<0><1>}]\wedge$

$\Delta\mathbf{M_5}[\{\bar{p},\bar{t},n\}((dormUnit_5)^{<0><1>}]\wedge$

$commonSpace$

$dormcluster = \Delta\mathbf{M}_1[\{\bar{p},\bar{t},n\}\ \ (dormUnit)^{<0><1>}]$



Wednesday, June 12

5→7

MOVE_SHAPE (*commonSpace*)

5→8

ROTATE_SHAPE (*dormCluster*)

MOVE_SHAPE (*commonSpace*)

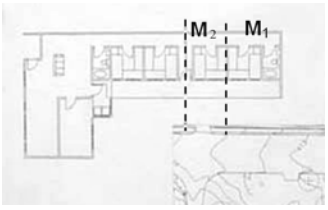
REPLACE_REGULATOR ($\Delta\mathbf{M_4}$, $\Delta\mathbf{M_5}$)

$\Delta\mathbf{M_4}[\{\bar{p},\bar{t},n\}((dormUnit_4)\ ] \Rightarrow \Delta\mathbf{R_4}[\{\bar{p},\bar{t},\theta,n\}((dormUnit_4)]$

$\Delta\mathbf{M_5}[\{\bar{p},\bar{t},n\}((dormUnit_5)\ ] \Rightarrow \Delta\mathbf{R_5}[\{\bar{p},\bar{t},\theta,n\}((dormUnit_5)]$

6

$$dormUnit_1 = \Pi\, \mathbf{H}\,[\{\}(\ \Delta\mathbf{M}_2\,[\{\bar{p},\bar{t},n\}(\ \Delta\mathbf{M}_1\,[\{\bar{p},\bar{t},n\}(room)^{<0><1>}]\,,\ bathroom\ )^{<0><1>}]\wedge\ kitchen, balcony\ )]$$

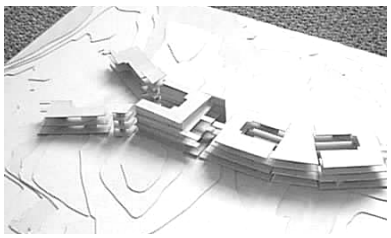


Monday, June 17, 2002: MIDTERM

7

$$building = \Delta\mathbf{M}_2[\{\bar{p},\bar{t},n\}(\ \Delta\mathbf{M}_3[\{\bar{p},\bar{t},n\}((dormCluster_1)^{<0><1>}_{\#1}]\ )^{<0><1>}]\wedge\ \Delta\mathbf{M}_4[\{\bar{p},\bar{t},n\}((dormUnit_4)^{<0><1>}]\wedge\ \Delta\mathbf{M}_5[\{\bar{p},\bar{t},n\}((dormUnit_5)^{<0><1>}]\wedge\ commonSpace$$

$$dormcluster = \Delta\mathbf{M}_1[\{\bar{p},\bar{t},n\}\ \ (dormUnit)^{<0><1>}]$$

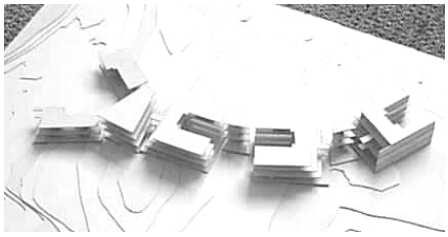


Monday, June 17, 2002: MIDTERM

8

$$building = \Delta\mathbf{M}_2[\{\bar{p},\bar{t},n\}(\ \Delta\mathbf{M}_3[\{\bar{p},\bar{t},n\}((dormCluster_1)^{<0><1>}_{\#1}]\ )^{<0><1>}]\wedge\ \Delta\mathbf{R}_4[\{\bar{p},\bar{t},n\}((dormUnit_4)^{<0><1>}]\wedge\ \Delta\mathbf{R}_5[\{\bar{p},\bar{t},n\}((dormUnit_5)^{<0><1>}]\wedge\ commonSpace$$

$$dormcluster = \Delta\mathbf{M}_1[\{\bar{p},\bar{t},n\}\ \ (dormUnit)^{<0><1>}]$$



Monday, June 17, 2002: MIDTERM

5→9

DELETE_REGULATOR($\Delta\mathbf{M}_4$, $\Delta\mathbf{M}_5$)

REPLACE_REGULATOR($\Delta\mathbf{M}_3$)

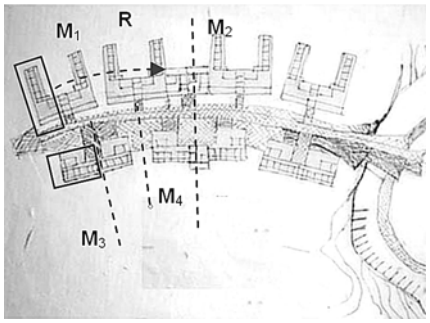
$$\Delta\mathbf{M}_2[(\Delta\mathbf{M}_3[(dormCluster)])] \Rightarrow \Delta\mathbf{M}_2[(\Delta\mathbf{R}[(dormCluster)])]$$

9

$$building = \Delta\mathbf{M_2}[\{\bar{p},\bar{t},n\}(\Delta\mathbf{R}\ [\{\bar{p},\bar{t},\theta,n\}(dormCluster_1)^{<0><1>}]\ )^{<0><1>}]\wedge \Delta\mathbf{M_2}[\{\bar{p},\bar{t},n\}(\Delta\mathbf{M}_4[\{\bar{p},\bar{t},\theta,n\}(dormCluster_2)^{<0><1>}_{\#1}]\ )^{<0><1>}]\wedge \Delta\mathbf{R}\ [\{\bar{p},\bar{t},\theta,n\}(commonSpace)^{<0><n>}]$$

$$dormCluste\quad r_1 = \Delta\mathbf{M}_1[\{\bar{p},\bar{t},n\}\ (dormUnit$$

$$dormCluster_2 = \Delta\mathbf{M_3}[\{\bar{p},\bar{t},n\}(dormUnit_2)^{<0><1>}]$$



Friday, June 21, 2002

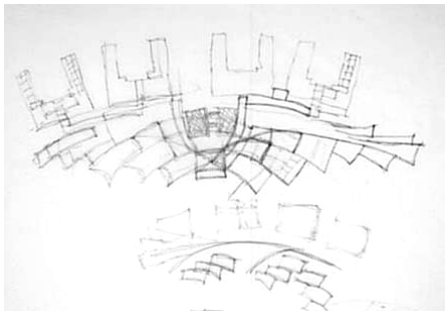
9→10

INSTANTIATE_SHAPE (*spagettiSpace*)

10

$$building = \Delta\mathbf{M_2}[\{\bar{p},\bar{t},n\}(\Delta\mathbf{R}\ [\{\bar{p},\bar{t},\theta,n\}(dormCluster_1)^{<0><1>}]\ )^{<0><1>}]\wedge spagettiSpace$$

$$dormCluster = \Delta\mathbf{M}_1[\{\bar{p},\bar{t},n\}\ (dormUnit_1)^{<0><1>}]$$



Monday, June 24, 2002

10→11

REPLACE_REGULATOR($\Delta\mathbf{R}$)

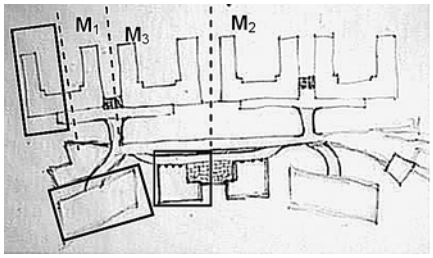
$$\Delta\mathbf{M_2}[(\Delta\mathbf{R}[(dormCluster)])] \Rightarrow \Delta\mathbf{M_2}[(\Delta\mathbf{M_3}[(dormCluster)])]$$

INSTANTIATE_SHAPE (*circulation*)

11

$$building = \Delta\mathbf{M_2}[\{\bar{p},\bar{t},n\}(\Delta\mathbf{M_3}[\{\bar{p},\bar{t},\theta,n\}(dormCluster_1)^{<0><1>}]\ dormCluster_2, commonSpace, circulation\ )^{<0><1>}]$$

$$dormCluster = \Delta\mathbf{M_1}[\{\bar{p},\bar{t},n\}\ (dormUnit_1)^{<0><1>}]$$



Wednesday June 26, 2002

11→12

SAME STRUCTURE

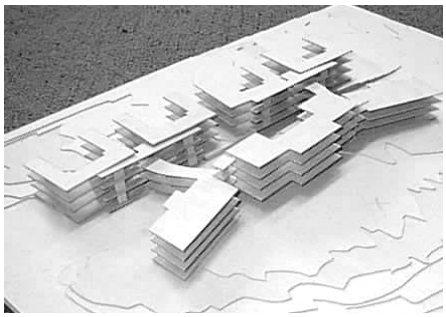
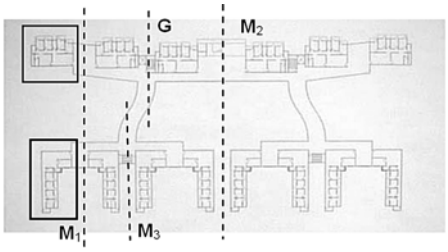
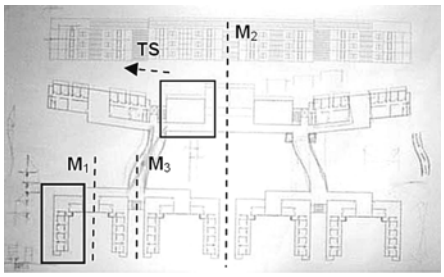
12	*building* = $\Delta\mathbf{M_2}[\{\bar{p},\bar{t},n\}($ $\Delta\mathbf{M_3}[\{\bar{p},\bar{t},\theta,n\}(dormCluster_1)^{<0><1>}]$ $dormCluster_2, circulation, commonSpace$ $)^{<0><1>}]$ $dormCluster = \Delta\mathbf{M_1}[\{\bar{p},\bar{t},n\}\ (dormUnit_1)^{<0><1>}]$	 Monday July 1, 2002
	12→13 INSERT_SUCCESSIVE ($\Delta\mathbf{G}$, $\Delta\mathbf{M}_1$)	
13	*building* = $\Delta\mathbf{M_2}[\{\bar{p},\bar{t},n\}($ $\Delta\mathbf{M_3}[\{\bar{p},\bar{t},n\}(dormCluster_1)^{<0><1>}]$ $\Delta\mathbf{G}[\{\bar{p},\bar{t},n\}($ $\Delta\mathbf{M_1}[\{\bar{p},\bar{t},n\}(dormCluster_2)^{<0><1>}_{\#1}]$ $)^{<0><1>}]\wedge$ $commonSpace, circulation$ $)^{<0><1>}]$ $dormCluster = \Delta\mathbf{M_1}[\{\bar{p},\bar{t},n\}\ (dormUnit_1)^{<0><1>}]$	 Wednesday, July 3, 2002
	13→14 REPLACE_REGULATOR ($\Delta\mathbf{G}$, $\Delta\mathbf{M_1}$) $\Delta\mathbf{G}[\{\bar{p},\bar{t},n\}(\Delta\mathbf{M_1}[\{\bar{p},\bar{t},n\}(dormCluster_2)_{\#1}])] \Rightarrow \Delta\mathbf{T}\Delta\mathbf{S}\ [\{\bar{p},\bar{t},n\}(dormCluster_2)]$	
14	*building* = $\Delta\mathbf{M_2}[\{\bar{p},\bar{t},n\}($ $\Delta\mathbf{M_3}[\{\bar{p},\bar{t},n\}(dormCluster_1)^{<0><1>}]$ $\Delta\mathbf{T}\Delta\mathbf{S}\ [\{\bar{p},\bar{t},n\}(dormCluster_2)^{<0><2>}]$ $commonSpace, circulation$ $)^{<0><1>}]$ $dormCluster = \Delta\mathbf{M_1}[\{\bar{p},\bar{t},n\}\ (dormUnit_1)^{<0><1>}]$	 Tuesday, July 10, 2002

TABLE 6.10 – SUBJECT W'S DESIGN SEQUENCE ILLUSTRATED IN ICE

6.4.3. MULTIPLE REPRESENTATIONS OF A SNAPSHOT

Each step in Table 6.10 represents a stage of Subject-W's development. Although each of these steps is represented using ICE in a single manner, it is possible to represent it in multiple ways using ICE's property of multiple representations. Table 6.11 shows the

abstraction of Subject-W's midterm submission (see Table 6.10 step 5) as it is represented by distinct generation paths, and consequently yielding different ICE notation strings and distinct applicable transformations. Each of these steps is generated using the ICE implementation. In steps 1 and 2, a dorm unit is created then reflected about $\Delta\mathbf{M_1}$. In step 3, the same arrangement is obtained (step 3A) by a reflection about $\Delta\mathbf{M_2}$, and in (step 3B) a rotation about $\Delta\mathbf{R_1}$. The generation sequence continues in distinct paths though steps 4 and 5, yielding different arrangements. In step 6, however, two different actions, reflecting about $\Delta\mathbf{M_5}$ and reflecting about $\Delta\mathbf{M_6}$, bring the arrangement back to equivalence. At this point the two shapes are identical, but not the notation, since it also captures the way in which each shape was generated.

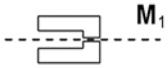
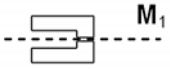
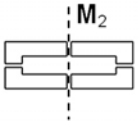
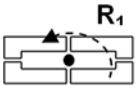
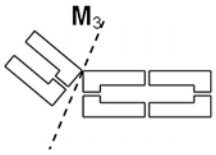
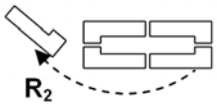
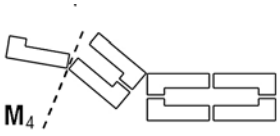
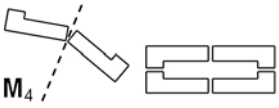
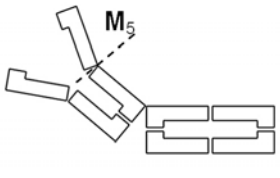
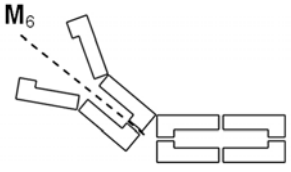
	A	B
1		
2		
3		
4		
5		
6		
	$\Delta\mathbf{M}_3(\Delta\mathbf{M}_2(\Delta\mathbf{M}_1(\text{dormUnit}_1)))$ $\Delta\mathbf{M}_4(\text{dormUnit}_4)$ $\Delta\mathbf{M}_5(\text{dormUnit}_5)$	$\Delta\mathbf{R}_1(\Delta\mathbf{M}_1(\text{dormUnit}_1))$ $\Delta\mathbf{M}_6(\Delta\mathbf{M}_4(\Delta\mathbf{R}_2(\text{dormUnit}_1)))$

TABLE 6.11 - TWO GENERATIVE REPRESENATIONS OF SUBJECTW'S DESIGN

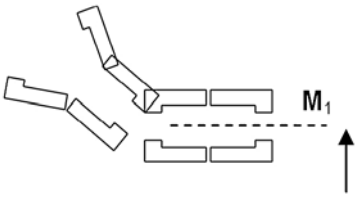
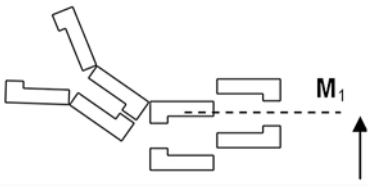
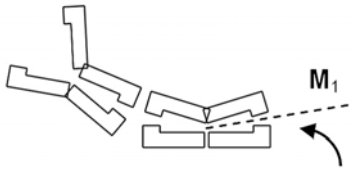
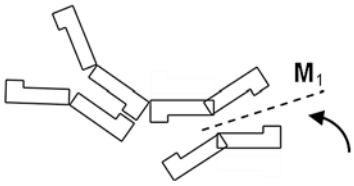
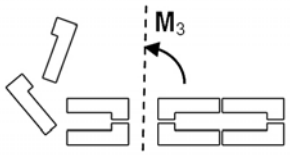
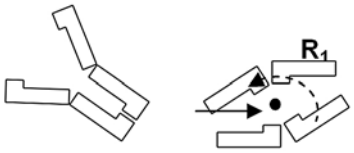
	A	B
1		
	Move mirror $\Delta\mathbf{M_1}$ upward	Move mirror $\Delta\mathbf{M_1}$ upward
2		
	Rotate mirror $\Delta\mathbf{M_1}$ counterclockwise	RotateMirror $\Delta\mathbf{M_1}$ counterclockwise
3		
	Rotate mirror $\Delta\mathbf{M_3}$ counterclockwise	Move rotation point $\Delta\mathbf{R}$ to the right

TABLE 6.12 - TWO SETS OF APPLICABLE TRANSFORMATION OF SUBJECTW'S DESIGN

The notation clearly expresses the difference(s) between the two generative sequences. The resulting configuration (step 6) in Sequence B of Table 6.11 has different handles than the same one in Sequence A. These results are illustrated in steps1-3 of Table 6.12. Identical manipulation-actions (for instance moving shared regulators) would result in totally different graphic configurations. The different handles (non-shared regulators) allow for a different set of manipulations per graphic configuration, such as moving the rotation point $\Delta\mathbf{R_1}$ or rotating the mirror line $\Delta\mathbf{M_3}$. There are numerous possible manipulations for each sequence; those shown were just a few. Additionally, redefining the notation string by insertion, deletion, or replacement would expand the manipulation possibilities even further and redirect the exploration paths.

CHAPTER 7

THE ICE IMPLEMENTATION

In this chapter, I describe the implementation of the ICE framework, which is a software environment that supports real-time exploration of 3D shapes and configurations by means of regulators.

The ICE implementation supports creation of structures, integration of structure, preservation of structure, transformation of structure, as well as breaking of structure. Creation of structure is achieved by designing regulators (the axes, centers and alignments) and associating them to the elements of composition. Integration of structure is the superimposition of several local and global sub-structures, within a single composition, by integrating distinct sets of regulators and elements. Preservation of structure is achieved by propagating changes though regulators. Transformation of structures is achieved by manipulating regulators. Breaking of structures is achieved by dissociating and deactivating regulators, in order to explore the configurations without their corresponding relationships.

7.1. OVERVIEW

In the ICE implementation, the design configuration is represented as two levels of abstraction: one for the regulated elements and the other for the structure encapsulated by regulators. Although the point is only element in ICE, and all shapes are defined by points and regulators, the implementation includes additional pre-programmed primitive elements. These are introduced for the purpose of processing speed. The regulators control the behavior of elements and propagate changes across the configuration. Elements and regulators can be dynamically associated with each other. This allows superimposed structures, and enables multiple elements per regulator as well as multiple regulators per element. Furthermore, regulators can control other regulators, defining structure hierarchies with multiple levels of control.

Flexibility is the primary goal for the ICE implementation in both its engineering and usability components. Flexibility is necessary to support an exploration that begins with one configuration, and proceeds by means of gradual transformations in order to arrive at a different configuration. The absence of flexibility would confine this exploratory process and limit exploratory directions.

In the ICE implementation, flexibility is expressed in the interaction of users with structures. To create structures, users must be able to associate regulators with elements at any time during the exploration. Conversely, to break structures, users have the ability to dissociate regulators from elements. To momentarily explore without specific structures, users must be able to deactivate and reactivate regulators. To transform structures, user must be able to manipulate all parameters of regulators and regulated elements at any given time. Furthermore, users must be able to interchange regulators (and interchange regulated elements) anytime during exploration without reworking the configuration, such that they may never be "stuck" to a specific situation.

Additional goals for the ICE implementation include efficiency, ease of use, and user control of all elements as well as attributes of the configuration.

7.1.1. ENGINEERING CONCEPTS

The ICE implementation was designed using an object oriented software engineering

approach; use-cases, interaction diagrams, and an object model were developed. Appendix D presents selected use-cases and interaction diagrams for creating and transforming elements and regulators.

Figure 7.1 shows the object model for the ICE implementation (see Appendix D for an enlarged version). The main objects include Shape, Regulator, Association, Schema, which represents the groups of associated elements and regulators, and the IceModel, which represents the whole configuration. Polymorphism has a primary role in all the constructs of the ICE implementation, and particularly, in the way lists are implemented and objects are interchanged. IceObject is the primary abstraction that subsumes all other objects of the implementation, and IceList is a list that organizes all these objects.

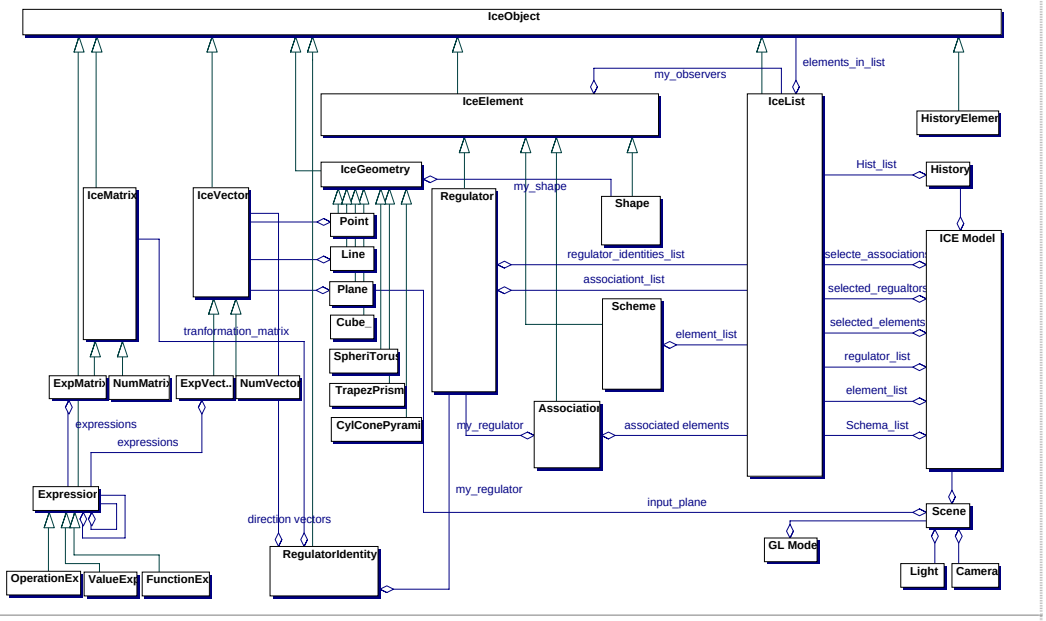


FIGURE 7.1 - THE ICE OBJECT MODEL

There were several significant issues in the design of the object model. The bidirectional associativity between regulators and associated elements resulted in an adaptation of the Observer mechanism (Gamma 1997, p293). The need to interchange elements and regulators during exploration resulted in an adaptation of the Bridge pattern (Gamma 1997, p151). The numerous possible combinations of regulators resulted in a strategy for

combining simple mathematical modules to define complex formulae for regulators. This modular approach extends the notion of describing complex shapes though simple regulator modules of the ICE framework.

Regulators control elements though the observer mechanism. A regulator 'observes' the elements (inputs and outputs) that are associated with it. When an element is changed, it notifies its regulator; some regulators constrain the change, while others propagate the change to other elements. When a regulator's parameters are changed, the elements that it regulates are updated accordingly. The observer pattern, which encapsulates the dependencies between elements and regulators, is adapted to accommodate the control functionality of the regulator in addition to the observation functionality. This allows bidirectional associative networks to be constructed at run-time. Figure 7.2 shows the observer in the context of the ICE implementation. IceElement subsumes the objects of the observation mechanism, which are the shape, the regulator and the association. This allows the association object to include shapes and regulators, such that regulators can be regulated, and enables regulators and associations to act as observers/controllers. When the regulators and shapes are associated in succession, these form a tree; when shapes share regulators or regulators share shapes, these form an acyclic graph.

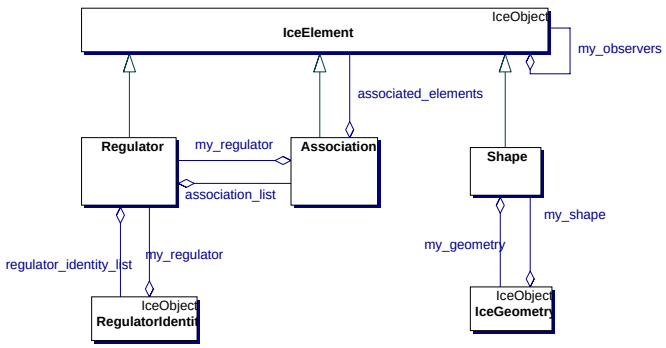


FIGURE 7.2 - OBSERVER MECHANISM

Replacing regulators (or shapes), while these are associated to numerous elements, can lead to reconfigurations numerous reassignment of associations. This is an error prone process, which affects the robustness of the implementation. Such internal reconfigurations are significantly simplified by using the Bridge pattern, which, in the

context of the ICE implementation, uses polymorphism to interchange a specific formula (or geometry) at run-time, without changing additional parameters or reassigning any of the numerous associations.

7.1.2. USABILITY AND INTERACTION CONCEPTS

Every object in the ICE implementation can be accessed through the interface. Shapes, regulators associations, schemata, and models, have specialized windows housing their controls. Interaction with structures is achieved by manipulating the regulator's parameters. Direct manipulation is implemented for selection and moving, but other operations, such as rotation, scaling and changing attributes, are achieved through value sliders. Furthermore, every element in the ICE implementation – shapes, regulators, associations, schemata and models—can be either viewed or hidden in order to provide focused views as well as integrated views of the configuration.

The implementation starts up with the model. A user instantiates shapes as well as regulators by choose and click. Associations and schemata are generated by the system, when users associate regulators and elements. Every element in the ICE implementation, whether it is instantiated by users generated by the system, can be selected and manipulated separately.

Modes of interaction include drawing modes, and selection modes (Figure 7.3). Drawing modes are for instantiating, copying, and moving elements and regulators. Selection modes include single selection, multiple selection, and specialized selections. The latter form selection includes selecting all shapes of an association, all shapes of a regulator, and all associations of a regulator, efficiently, with just a single click. Additionally, when an object is selected, its relevant interface widgets appear on its window.

FIGURE 7.3 - MODES OF INTERACTION

7.2. REGULATED ELEMENTS

The vocabulary of regulated elements consists of the point and a finite set of 3D shapes. These can be manipulated in various ways, including common manipulations such as translation, scaling, rotation, and shear, and manipulation specific to each shape such as adjusting holes sizes, smoothness of surface, and dimensions of upper and lower faces. Additionally, all the shape's physical properties such as color, fill, line width, and transparency are adjustable. Shapes are regulated thought their key points, which include their centroids, or their upper or lower midpoints.

In the ICE implementation, shapes can be interchanged at any time. The shape abstraction is separated from its specific geometry by means of the bridge pattern (Figure 7.4), allowing the geometry to be changed without disrupting any of the intricate associations.

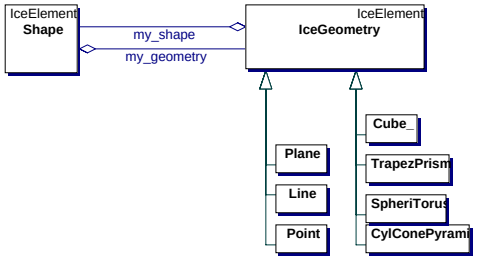


FIGURE 7.4 - SHAPE BRIDGE PATTERN

Figure 7.5 shows the shape window, the shape controls, and illustrates examples of ICE primitive shapes, for which the proportion can be manipulated in many ways.

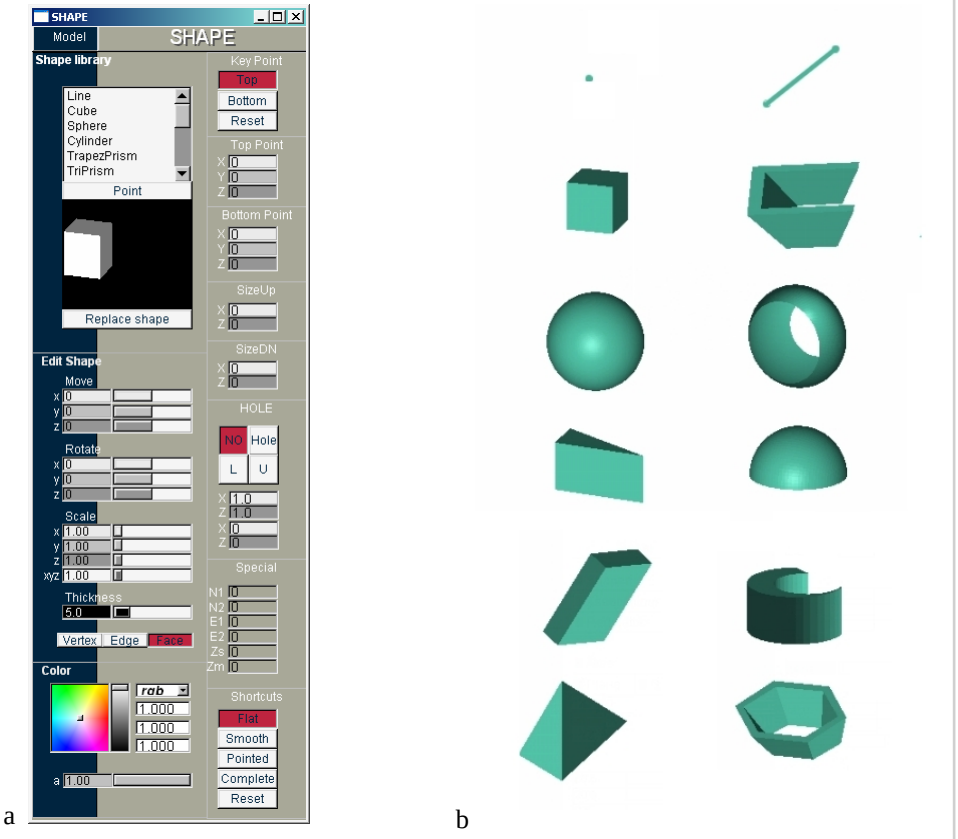


FIGURE 7.5 - SHAPE WINDOW AND SHAPE EXAMPLES

7.3. REGULATORS

A regulator has generative as well as manipulative capabilities: it generates one or more outputs from a (user-defined) input; it maintains a persistent relation between outputs and corresponding inputs upon manipulation. Changes to regulators transform outputs, and may change the classification of the configuration, for instance its symmetry group.

Transformation-based regulators (for example, translation, rotation, mirror, and dilation) control the relation between an input and its outputs. Each regulator computes the position/orientation of outputs according to its specific transformation matrix. It is used to generate outputs from an input, and to update outputs when the input is manipulated. The inverse transformation is used to update the input when the outputs are manipulated. Variation regulators control elements by means of a formula, and constraint regulators control elements by means of an evaluation function.

Only a subset of the regulators of the ICE framework is realized in the ICE implementation. In the transformation category, Translation, Rotation, Mirror, Dilation, and Shear are implemented. In the variation category, Rhythm/Gradation and Exception are implemented. In the constraint category, Alignment is implemented. All regulators function in 3D space. The regulator parameters described in the ICE notation are the manipulation handles in the ICE system.

Transformation-based regulators work by multiplying the regulated elements by the regulator's main transformation matrix. When their geometry is updated the transformation matrix is pre- and post- multiplied by the rotation and translation matrices, therefore modifying the regulator's main transformation matrix. Transformation regulators (Figure 7.6) include translation along a line, rotation about a line, mirror about a point, line, plane, dilation about a point in the xyz direction and shear about a point along one direction.

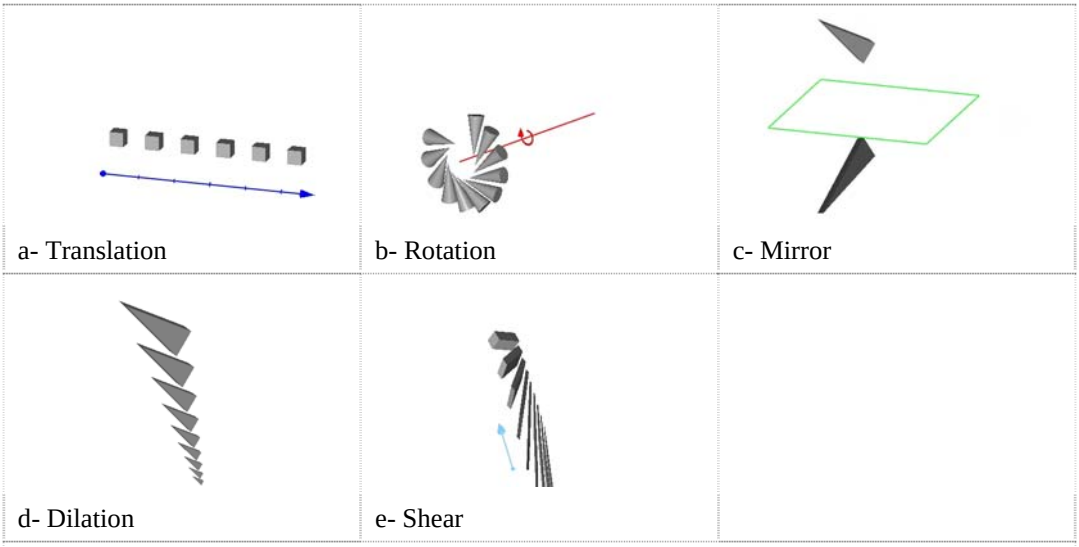


FIGURE 7.6 - TRANSFORMATION-BASED REGULATORS

The Rhythm/gradation regulator can only be used in composition with other transformation regulators. These work by applying a formula to a specific attribute of the shape or point. For instance, position, relative rotation, color and relative scale. There are several version of this regulator, including a gradual rhythm, a cyclic rhythm, a sine curve rhythm, and an upwards and downward rhythm. The gradual and cyclic Rhythm types are illustrated in Figure 7.7. The exception regulator works by overriding the value of a specific attribute of the regulated shape or point.

FIGURE 7.7 - RHYTHM

The Alignment regulator can be generative or can be used in combination with generative transformation regulators. Alignment works by constraining the position of the key-point in the regulated shape about a point or along a line, plane, or circle. Key-points include the shape's centroid, or its upper or lower midpoints. Alignment regulators can be combined to regulate different points in a set of shapes. Figure 7.8 illustrates a linear

element regulated by two simultaneous alignment regulators, with its upper midpoint aligned by a line and its lower midpoint aligned by a line (7.8a), point (7.8b), or circle (7.8c).

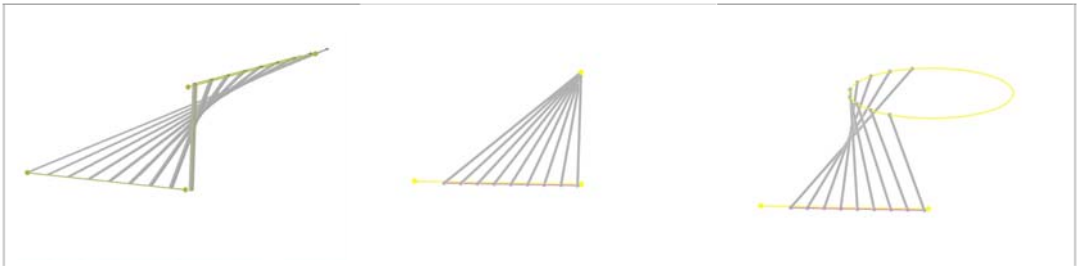


FIGURE 7.8 - ALIGNMENT

7.3.1. DYNAMIC ASSOCIATIONS

Elements can be associated with regulators at any stage of the exploration. An element supporting multiple regulators maintains a list of its regulators. Likewise, a regulator supporting multiple elements maintains a list of associations, each consisting of an input and *n* outputs.

Within associations, images are indexed, and the index is treated as an active variable in its regulator formula. Each association has a positive and negative output list, one on each side of the input (Figure 7.9). An association traverses though the list of images, updating these according to the transformation matrix or constraint of its regulator. The index (positive or negative) is combined with the active parameter of the regulators transformation matrix and therefore, determines the precise transformation for each output.

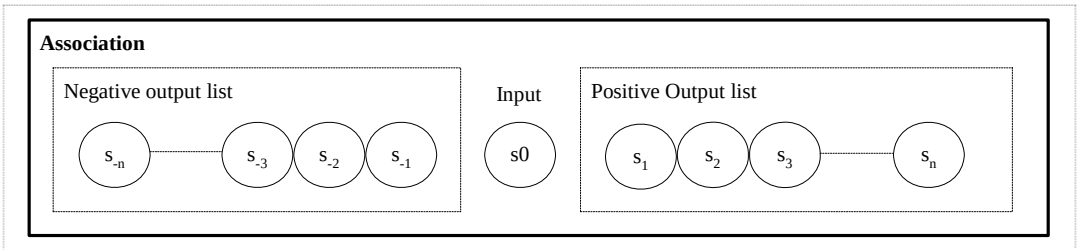


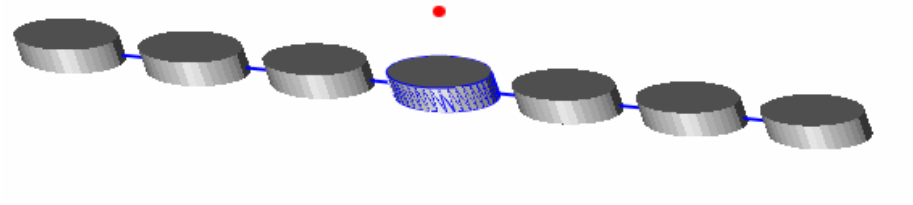
FIGURE 7.9 - THE ASSOCIATION OBJECT

The number of outputs generated is a function of the association, in this way, distinct associations belonging to the shape regulator can have a different number of outputs. The user can adjust the number of elements within an association anytime. Furthermore, the user can choose whether to update all properties of the associated elements or just to update the geometry.

Associations of non-generative regulators transform the input, and do not have any outputs. Associations can also accommodate several inputs, thereby supporting regulators that input several objects to output one resultant object, such as the union or intersection regulators.

On the model, associations are displayed by means of a line linking associate elements together and emphasizing the input elements (Figure 7.10a). Therefore, in a multiple regulator schema, associations can be traced (Figure 7.10b)

a



b

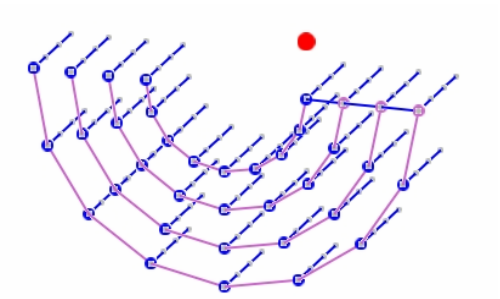


FIGURE 7.10 - ASSOCIATION DISPLAY

There are several types of associations: a user can associate one element (or regulator) to a regulator, or associate all elements (of a schema) to a regulator (Figure 7.11). An association can also associate a schema, thus creating multiple output schemata from an

input schema. Additionally a user can relate separate elements to a regulator, thus establishing a relation between them and creating an association. This functionality serves when structures are discovered within configuration elements.

FIGURE 7.11 - ASSOCIATION TYPES

7.3.2. SIMULTANEOUS COMPOSITION OF REGULATORS

Simultaneous composition takes advantage of the regulator's bridge pattern, where the regulator object is separated from its formula or matrix, which is encapsulated in a RegulatorIdentity object (Figure 7.12). This strategy enables various combinations of regulators to be defined at runtime. Regulators can be composed, added, removed, or replaced, and therefore, changing the regulation formulae and the behavior of the regulated elements, without disrupting any associations.

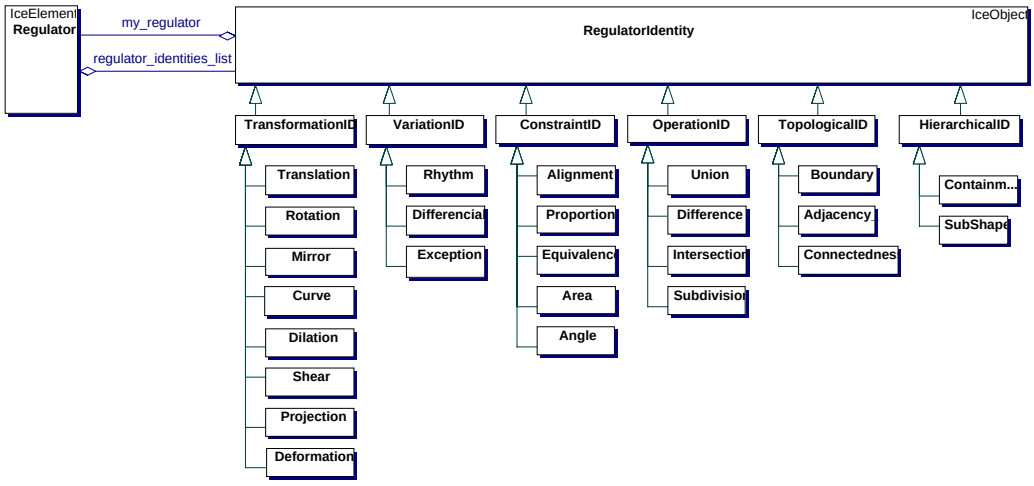


FIGURE 7.12 - BRIDGE PATTERN FOR REGULATORS

The regulator object contains all the regulator's parameters and associations. The RegulatorIdentity object encapsulates the formula as well as the geometry of the regulator, therefore, enabling distinct geometries per identity. For example, in a

composition of mirror and translation, the mirror is a plane while the translation is a line, and each of these can be manipulated separately.

The regulator has a list of RegulatorIdentity objects, which accommodates multiple matrices or formulae. Upon regulation, the regulator goes thought this list and combines all the matrices of formulae of these RegulatorIdentity objects. Transformation-based regulators produce a single composite transformation matrix (and its inverse) by means of matrix multiplication of all the unit matrices in the list as illustrated in Figure 7.13. The composite matrix controls the positive list in the association, while its inverse controls the negative list. Variation and constraint regulators produce a compound formula by combining the formulae of their regulator identities.

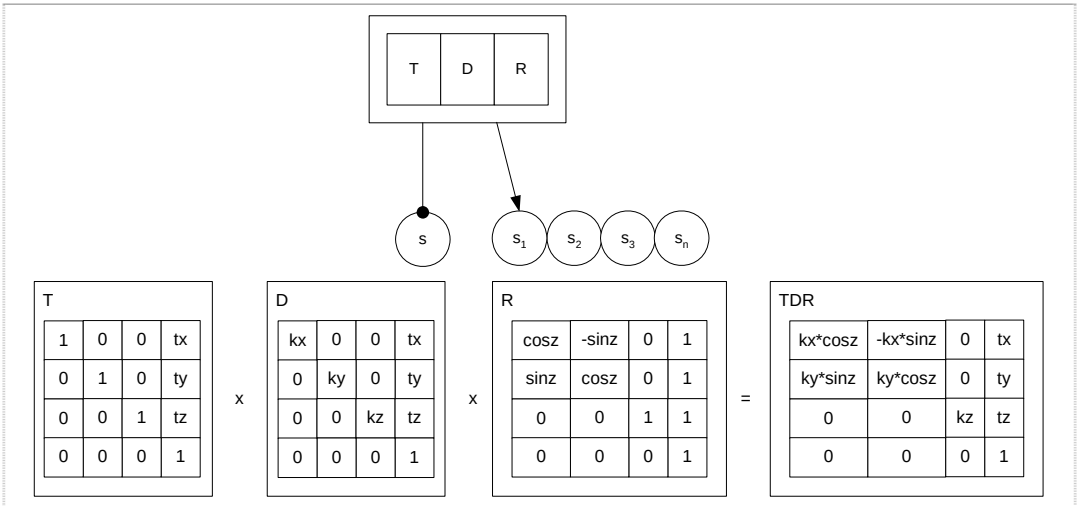


FIGURE 7.13 - SIMULTANEOUS COMPOSITION

Direction vectors and transformation matrices are implemented as Expression objects that store compound numerical expressions and trigonometric functions as well as numbers (Figure 7.14)

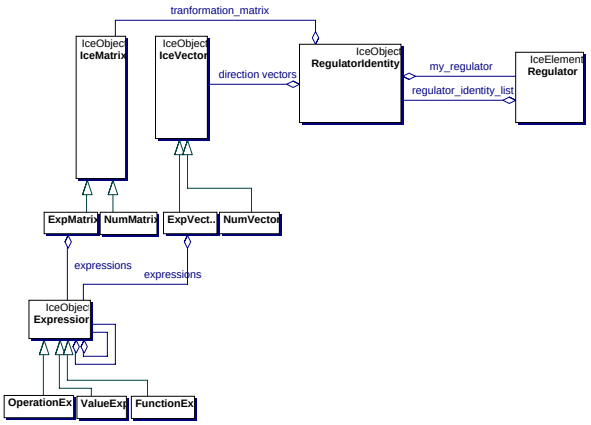


FIGURE 7.14 - THE MATRIX-VECTOR OBJECT MODEL

7.3.3. REGULATOR CONTROLS

The ICE implementation supports both the geometric manipulation of the composition of regulators and the geometric manipulation of the single regulator.

Regulators are controlled thought the regulator window (Figure 7.15). The user selects a regulator (or multiple regulators) from a list, and then positions these on the model to be instantiated. The user can add, remove, or replace regulators from a simultaneous composition at any time. Each category of regulators, transformations, constraints, and variations are listed in a separate box. The user can select single regulators from this box in order to manipulate them separately. The geometric parameters of the regulators as well as regulator-specific parameters and factors are controlled by means of sliders. The regulator identity objects are hidden form the user. The interface gives the impression of composing whole regulators objects. When a regulator is selected, only the applicable widgets appear in the regulator window.

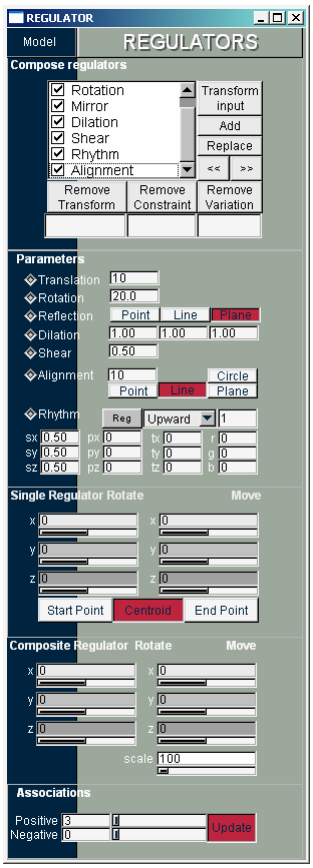


FIGURE 7.15 - REGULATOR CONTROLS

Associations are also manipulated thought the regulator window, where the numbers of positive and negative output elements, as well as the property to update all associated elements, are adjusted.

7.4. SCHEMATA

Schemata encapsulate configurations comprising of elements associated to regulators. In addition to allowing multiple shapes to share a regulator, and multiple regulators to shape a shape, schemata facilitate the interaction by allowing the manipulation of the structure as a whole and the superimposition of distinct structures.

Schemata encapsulate a sequence of regulators. Regulators can be inserted in a sequence, deleted from a sequence, or two regulators can be swapped within a sequence, thereby redefining configurations, especially those with non-commutative regulators.

Such discrete manipulations of schemata (as well as the continuous geometric manipulations of moving, rotating, and scaling) are achieved through the schema window (Figure 7.16).

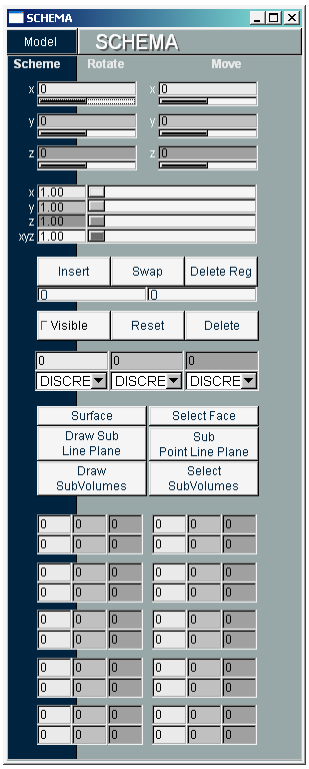


FIGURE 7.16 - SCHEMA WINDOW

Regulators use polymorphism to regulate shapes, points, and other regulators in a

seamless manner. Points are the simplest elements to regulate, with only position as the regulated property. Shapes need additional properties to be regulated, such as their orientation, scale, and physical and formal properties. Regulators, on the other hand are the most complex to regulate, because their matrices need to be regulated as well as their geometric and physical properties.

Once an element is regulated, it can be momentarily deleted, but it will not be completely discarded. If a schema is reset, all the deleted points, shapes and regulators are re-activated.

7.4.1. SUCCESSIVE COMPOSITION OF REGULATORS

In a multiple regulator schema, elements can take on the role of both inputs and outputs simultaneously (Figure 7.17). The root of the tree is the first input; intermediate nodes are outputs of the regulator above and inputs of the level below; and leaf nodes are outputs of the last regulator. For a multiple element and multiple regulator schemata, this tree becomes an acyclic graph.

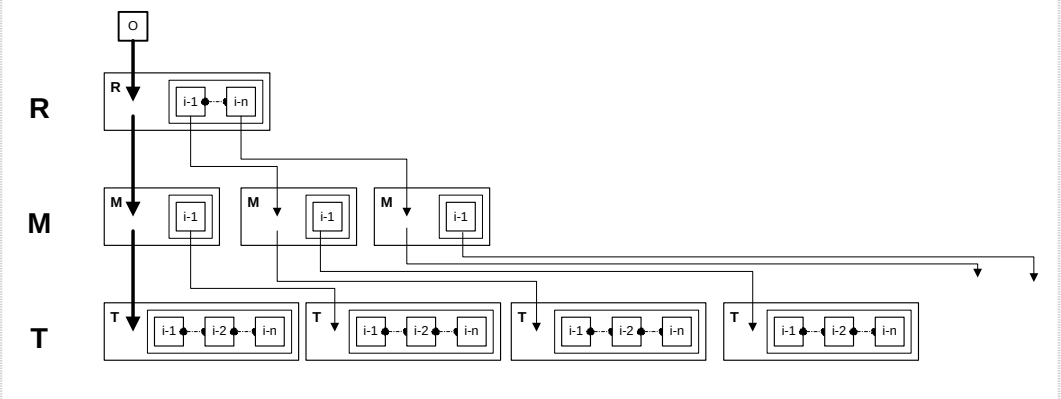


FIGURE 7.17 - THE REGULATOR TREE

When an element is associated with multiple regulators, its outputs are subject to a composition of transformations. In principle, there is no limit to the number of regulators that can be composed successively.

For multiple regulator schemata, a change in one element will initiate a chain reaction of changes that propagate across all the regulators. Changes in regulators are recursively

propagated “forward” so that they affect outputs of subsequent regulators. Changes in elements are recursively propagated "forwards and backwards" and affect outputs of subsequent, as well as antecedent, regulators. The user can select whether the change propagation is to proceed forwards, backwards, or in both directions.

7.4.2. REGULATING CONTINUOUS SHAPES

Complex shapes are regulated through continuous generation of points. When a regulator is updated, the shape is redefined in a virtually plastic manner. The resolution of the shapes can be adjusted by increasing or decreasing the number of points.

In OpenGL, 3D shapes are rendered as surfaces. So it is necessary to map every ICE shape representation to a representation of its outer surfaces in order to display it properly. Such surfaces can be displayed or hidden at will.

The ICE implementation supports the selection of various subsets of a shape, whether it’s a sub-point, sub-line, sub-surface, or sub-volume.

When defining a 3D shapes (with 3 regulators), it is possible to specify any combination of discrete or continuous regulators to define a shape with its constituent linear or planar components (Figure 7.18). Furthermore, it is possible to define subshapes within a shape by indicating the indices to be generated (Figure 7.19).

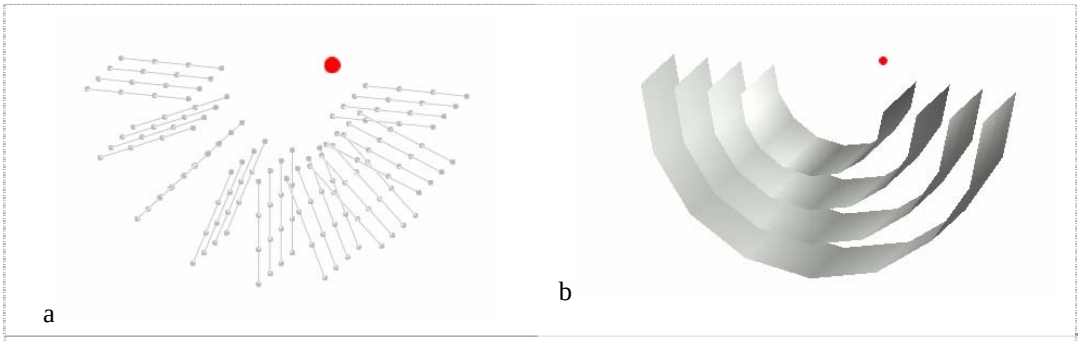


FIGURE 7.18 - COMBINATION OF DISCRETE AND CONTINUOUS REGULATORS

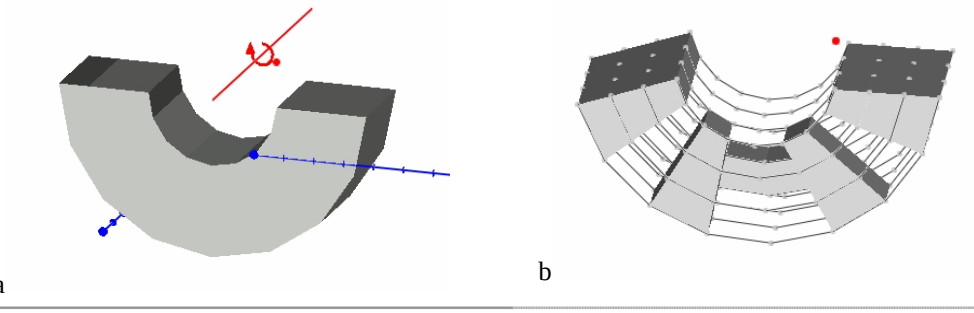


FIGURE 7.19 – COMPLETE SHAPE GENERATION AND SUBSET GENERATION

7.4.3. REGULATING REGULATORS

Regulators can be regulated creating multiple levels of control. When regulators are regulated, the transformation matrix of the super-regulator influences the matrix of the sub-regulators. This matrix is pre- and post- multiplied by the conjugates (the matrix and its inverse) of the super-regulator in order to produce the correct regulation matrix for the sub regulator.

Furthermore, the ICE implementation supports the regulation of schemata, therefore enabling a regulator to input a shape (described by regulators) and create a novel configuration by means of discrete generation. Regulating schemata involves regulating the constituent regulators.

7.5. THE MODEL

In the ICE implementation, the model has two major components: the configuration component consisting of the elements and regulators of the ICE framework, and the graphic component, consisting of viewing, camera, and lighting. The controls for the model display can be considered as view regulators.

The user can select between one, two, and four simultaneous view-ports, (Figure 7.20) allowing the display of several views of the model simultaneously. These can be axonometric, perspective, top, front, back, or side views. Each view-port can be selected and manipulated independently. The user can move, rotate or scale, the model as a whole and can control the camera to focus on particular aspects of the configuration model. Additionally, wire frame views as well as shaded views of the model are available. The views and camera controls are located on the right side of the main model window, which is designated for common manipulations (Figure 7.20), or alternatively, on the view window (Figure 7.21a) where less frequent manipulations are placed. The main model window also serves for setting interaction modes and accessing other windows.

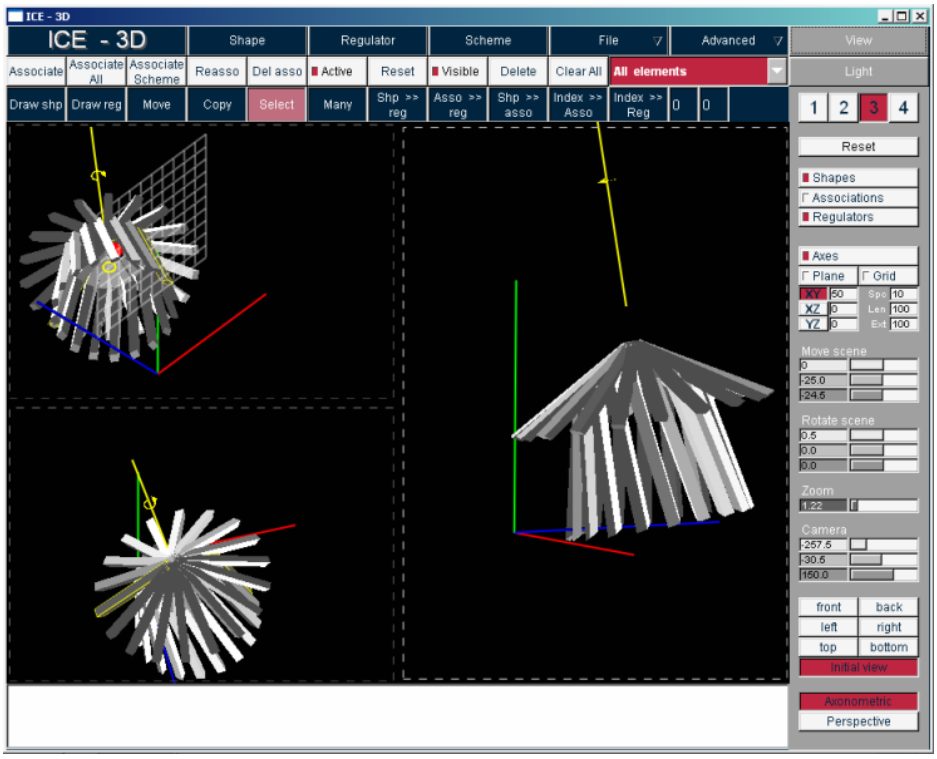


FIGURE 7.20 - THE MAIN MODEL WINDOW

Since the ICE implementation is in 3D, it is important to have adequate lighting to shade the model elements properly. Lights can be turned on and off, and the position, orientation, intensity, and color of the lights, are adjustable, as well as their ambient, diffuse, and secular components. All lighting controls are located in the light window (Figure 7.20b).

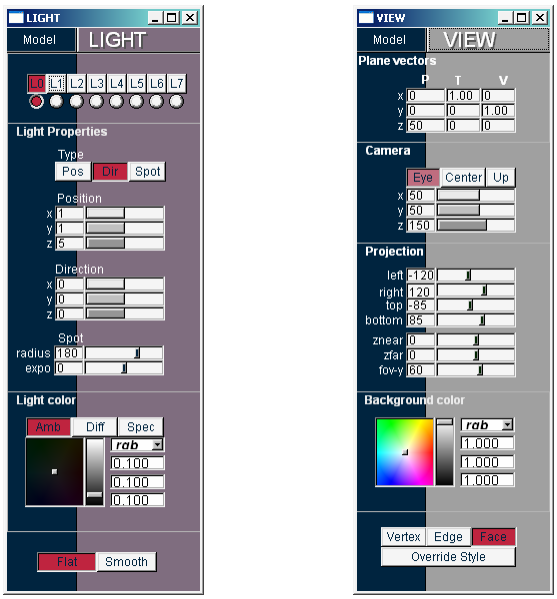
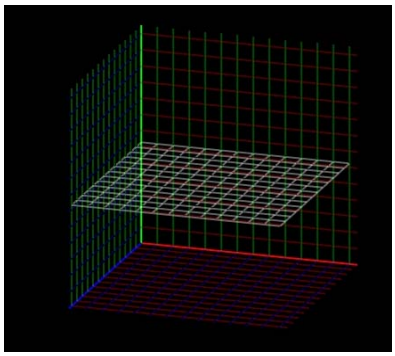
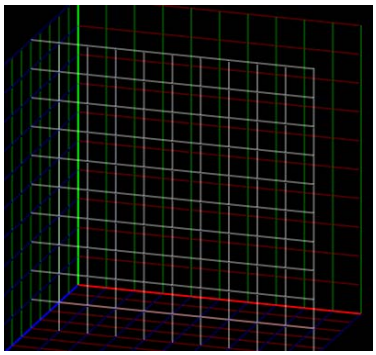


FIGURE 7.21 - VIEW AND LIGHTING CONTROLS

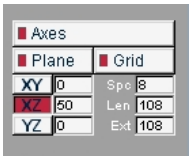
Additional model elements include the coordinate axes, and the positioning plane (Figure 7.22). The coordinate axes intersect at the origin, forming the three coordinate planes; each has a corresponding grid with adjustable resolution. The plane serves as a positioning device for direct input. A point on the 2D screen corresponds to infinitely many positions (forming a line) in its 3D model projection. The intersection of this line with the input plane serves to determine the precise position of the input on the 3D model. The plane can be set parallel to any of the 3D coordinate planes or can be set to any orientation by changing its direction vectors.



a



b



c

FIGURE 7.22 - THE AXES, THE POSITIONING PLANE, AND THEIR CONTROLS

7.6. HISTORY AND PROCESS CAPTURE

In the ICE implementation, every exploratory action, whether it is generative or transformative is recorded, in order to track the process. The current history controls, located in the history window (Figure 7.23) include playback, playback speed, loading, saving history and displaying the history list. In this way the whole process of exploration can be recorded and analyzed for further studies.

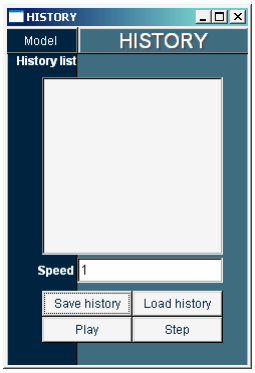


FIGURE 7.23 - HISTORY WINDOW

CHAPTER 8

DISCUSSION

This chapter concludes the dissertation by a discussion of the ICE framework in comparison to other representations, as well as a review the contributions of this work, and a presentation on the future directions for the ICE framework.

8.1. COMPARATIVE ANALYSIS

This section presents a comparative analysis of the ICE framework with constraint-based, grammar-based, associative, and mathematical representations.

8.1.1. ICE AND CONSTRAINT-BASED REPRESENTATIONS

The ICE representation differs from constraint-based representations in its capacity to encapsulate lower-level constraints into higher-level regulating constructs. While constraint-based representations operate with basic unary and binary constraints, ICE operates with more complex, yet more, intuitive entities. From a usability perspective, regulators overcome some of the problems found in typical constraint-based systems. Constraints are either system-defined (Briar, Sketcher) or user-defined (SketchPad, CoDraw). System defined constraints can lead to misinterpretation of user intent. On the other hand, users specifying and updating every constraint can become increasingly cumbersome with complex configurations, and can distract from the major design task. Regulators simplify user interaction with multiple constraints by grouping related constraints, and enabling users to generate and manipulate them simultaneously. Users interact with higher-level regulators, while regulators manage lower-level constraints. In this manner, users are relieved from the burden of specifying and updating numerous lower-level constraints. Still, users maintain control over the constraint definitions of their configuration though regulators.

Through this higher-level regulation, composing, transforming and redefining structures, both in a continuous and discrete manner, provide a more stimulating exploratory experience.

8.1.2. ICE AND ASSOCIATIVE REPRESENTATIONS

The ICE representation is an associative representation. However, unlike GenerativeComponents, ICE supports bi-directional change propagation; and unlike ReDraw with its three-tier hierarchical structure, ICE support multiple levels of control structures. Although ReDraw's pencil lines can be considered a primitive form of an alignment regulator, the ICE regulators are more diverse, and compositions strategies are more elaborate. Furthermore, ReDraw imposes an order on the drawing sequence (pencil lines first, then ink lines), while ICE requires no order for generation or manipulation.

8.1.3. ICE AND DESIGN GRAMMARS

ICE differs from design grammars in its representation for configurations, as well as in its strategy for transformations.

In grammars, the configuration is represented by shapes, which are in turn represented by lines and points. The shape itself does not recall its generation path nor does it capture relations between its parts. The only exception is Carlson's structured grammars that capture a transformation, which maps a shape from the origin to its intended position and orientation. ICE, on the other hand, represents the shape by means of relations among its parts. These relations determine the way the shape is generated, and their corresponding parameters guide further transformations. This representation makes the shapes themselves much richer in information.

In grammars, configurations are transformed through the application of production rules in a sequential manner; transformations (as well as relations and constraints) are implicitly encapsulated in those rules. In ICE, however, transformations are explicitly encapsulated in regulator parameters. The transformation syntax in ICE is comparable to grammar rules, but the recognition process (for which the left hand of the rule is matched to the configuration) is much simpler in ICE, where the recognition process is explicitly built in the string definition. In grammars, the recognition process is based on spatial

algorithms mapping the points and lines of the left hand side of the rule to the configuration. In ICE, however, there is no process for recognizing spatially equivalent, yet, notationally different configuration.

8.1.4. ICE AND MATHEMATICAL REPRESENTATIONS

Although both ICE and Cha's representation of shape patterns use predicate and arguments, the purpose of Cha's representation is shape analysis, while the purpose of the ICE representation is design exploration. ICE differs from Cha's shape patterns in numerous ways. ICE is a 3-Dimensional representation that includes a description for individual shapes as well as patterns. ICE is broader in scope (more relationships are supported) and more diverse in generation methods (continuous discrete and subpart generation are supported). From a syntactic perspective, the ICE's representation for output sequences is iterative, but it allows variations within the iterations, while Cha's representation for each element in the sequences is recursive. ICE uses mnemonic symbols to denote its relations, and it uses brackets and indices consistently, thus promoting readability. ICE also has a shape encapsulation strategy that simplifies the description. In addition to the configuration syntax, ICE has a complementary syntax for transformations.

The ICE representation and Leyton's representation are based on the same mathematical principles, however, their respective goals and approaches are different. Leyton's purpose is to represent existing shapes and configurations with a rigorous theoretic foundation. His objective is to maximize transfer and maximize recoverability. ICE's objective is to maximize the exploratory potential of design configurations, and to minimize the complexity of their corresponding descriptions. In Leyton's representation there are no objects, just actions. ICE has the opposite approach. In ICE, actions are encapsulated by objects (regulators), therefore, allowing these to be grabbed and manipulated after the action initially takes place. This approach converts actions, which are typically one time events, to persistent controllable events.

Leyton represents all shapes and configurations by means of symmetry groups and transfer structures. He claims that all design processes are effectively asymmetry building. In his effort to maximize transfer and recoverability, Leyton imposes an order

of symmetry, then asymmetry, on the action sequences for shape generation. This often complicates the generation of the shape. ICE, on the other hand, has a significant amount of recoverability, but does not impose a specific sequence on the generation process. ICE allows for multiple representations in order to maximize the options for shape generation.

Leyton always uses transfer structures as necessary constructs for describing all parts of a configuration; ICE, however, use transfer structures for describing shapes and promoting exploration, but transfer is not a necessary condition for all parts of the configuration. For instance, two distinct cubes can be described in ICE without transfer.

ICE's regulator structure can also be described by groups; however, not all shapes can be described simply by means of groups. For instance, a regular hexagon has a dihedral group, but not the semi-hexagon because it does not satisfy the group property of closure under composition. ICE supports the generation of such a shape (by three steps) in the same way the hexagon is generated, but with different parameters. Leyton needs to create the symmetrical hexagon first, then, define other groups to remove parts of it, therefore lengthen the process of generation.

From a syntactic perspective, Leyton's notation does not denote all the necessary parameters for shape generation. Additionally, the syntax for unfolding groups is quite complex.

The ICE representation simplifies the process of shape generation as much as possible and the syntax uses a minimum number of steps (motions) to create shapes. These steps are comparable to a sequence of pen motions in 3-Dimensional space.

8.1.5. ICE AND SOLID MODELING

The ICE framework differs from solid modeling and constructive solid geometry representations in that ICE relies on compositions of relations to define forms rather than a fixed set of components and operations.

8.1.6. ICE AND COMPUTING LANGUAGES

ICE resembles a computing language in its functional approach as well as object oriented

quality. It has input and output parameters, functional nesting as well as object encapsulation. However, it is not quite a language, because it does not have explicit conditional and procedural definitions nor does it explicitly support inheritance. Although some of these are captured within the regulators — for example, constraints encapsulate conditionals, while operations encapsulate procedural algorithms — the ICE notation itself cannot be used to define conditionals or procedures, explicitly.

8.2. CONTRIBUTIONS

This research describes a novel way of representing and exploring design configurations by means of generative and relational structures. It advances the state of the art in computational design representations through the following contributions:

A Modular Approach to Organizing Lower-Level Relations into Higher-Level Structures. Regulators of the ICE framework capture design relations as simple building blocks that are composed (in parallel and in sequence) to define the structure of complex configurations.

The Support for Iterative Explorations. Regulators, in their capacity to be decomposed, modified (in various ways), and replaced, offers the ability to completely transform and redefine configurations by changing a few structural parameters, thus reducing the labor involved in exploring with structures.

The Encapsulation Design Descriptions. The ICE notation introduces syntax for representing geometric configurations, completely, accurately and succinctly by means of a string. Its short form and shape encapsulation mechanism augment the notation with flexibility and enhance readability. Furthermore, describing complex geometrical configurations by means of a concise string has the additional computational advantages of minimizing the size of storage, maximizing the speed of file transfer, and facilitating the analysis of configurations.

Deleted: a syntax

The Encapsulation Design History. The ICE notation string, in its capacity to capture parsimoniously the generative process of a configuration, encapsulates its generative history. In its capacity for recording transformations, the ICE notation captures the exploratory history for a configuration.

The Derivation of Additional Information from Notational Descriptions. The notational description allows for the derivation of additional shape information, such as volumes and subshapes, by simple computation of regulator parameters. Furthermore, with some algorithmic interpretation, the notational description enables the deduction of the precise steps for transforming an initial configuration to the goal configuration. This derivational mechanism allows an unprecedented level of detail to be stored in such a concise string.

A Novel Method of Interaction with Design Configurations. The ICE implementation supports direct-manipulation of structures, while providing instantaneous visualization of the effects for these manipulations. It enables designers to transform structures, discretely and continuously, thus allowing the complete redefinition of configurations with minimum steps. Such iterative manipulation, coupled with visualization, facilitates the modification of earlier decisions, and provides a new exploratory experience in design.

Strategic Exploration. The ICE implementation allows users to define their manipulation handles for any object, based on the relationships they choose. Therefore, they are able to influence the direction of the subsequent explorations that will occur during design. In this way users can strategically compose their regulators with the intention of specific future explorations.

8.3. FUTURE WORK

The ICE framework, both in its notation and implementation forms, has potential for future investigation. Several venues for extension as well as potential applications are discussed below.

8.3.1. EXTENDING THE ICE FRAMEWORK

Regulators Representing Non-Geometric Information. Although regulators were described as geometric in nature, the vocabulary of the ICE framework can be extended to include non-geometric design information. These include physical/material properties (such as light reflectance, thermal transmission, and acoustic absorption), budgets constraints and design requirements (such as privacy or climatic considerations). With such semantic additions, the ICE framework would evolve into a complete design language relating semantics to geometry, and therefore, enabling the control of a design through its requirements and through its semantic property.

Regulators in Other Design Domains. Although regulators were primarily conceived for architectural design, this concept can be utilized in other domains, such as mechanical, industrial, and graphic design. Geometric regulators are easily applicable, while other domain specific regulators can be further developed, in particular, motion regulators can be of great potential in exploring mechanical and industrial design. Furthermore, by adapting non-geometric regulators to semantic properties, the regulator approach can be applied to domains that do not rely on geometry.

Recognition of Implied and Emergent Structures. Gero (1998) and others investigated recognition of emergent structures in design. Although recognizing structures in ICE is a complex task, a module for recognizing design structures would complement the ICE implementation, and would uncover implied and hidden structures in any configuration. Therefore, it would enable the identification of the geometrically equivalent, yet notationally different, representations, in cases where multiple representations exist.

Two-way Integration between the Notation and the Implementation. The interaction between the notation and the 3D model is not sufficiently integrated. A parser that converts the notational string into a 3D model and converts a model into a string would

provide the two-way integration, where a change in the notation would appear on the model, and a change in the model would be updated on the notation. Furthermore, an editor for the notation, where the syntax (brackets, commas, superscripts, etc) is managed automatically would greatly enhance the usability of the notation.

Usability. Although ICE represents complex geometric relationships in a simple way, interaction with the ICE models in 3-dimension is not ideal. Complexities of converting 2D interaction in 3D space are still prevalent. A significant research venue would be to investigate novel interaction hardware applicable to the design exploration activities of the ICE system. The ICE generation sequences can be mapped to gestures of drawing with the pen in a 3D sketch environment. Manipulation would also be mapped to gestures, without intermediate windows and widgets.

Cognitive Implications and Predictability. Predictability of the behavior of the ICE models upon manipulation is proportional to the complexity of the configuration and depends on whether the manipulations are discrete or continuous. Discrete manipulations are far less predictable and more surprising than continuous ones. Predictability is also affected by the type of transformation that is applied. A user study documenting factors in ICE that influence predictability, as well as the cognitive implication of predictably, is worthy of further investigation.

8.3.2. POTENTIAL APPLICATIONS FOR THE ICE FRAMEWORK

Process Analysis. ICE captures history on two levels: (1) the generative sequence captured in the shape definition; and (2) a record of transformations that occurred in the process of creating the design. Keeping track of the history is a valuable tool in analyzing the course of design processes precisely, and completely. Furthermore, history can be used effectively as a multidimensional element of the exploration. Users can step through their history, forwards and backwards, and change the course of the exploration while replaying their design actions. This would result in a history tree of branching exploration paths, instead of a linear history list.

Case-base Adaptation. The ICE representation can be integrated to case base systems, where cases are represented by means of the ICE notation, and the adaptation of a case to a new problem can be achieved readily through regulator transformations. As novel

shapes and configurations are defined by regulators, these can be stored in the configuration library, then later retrieved, re-used, and manipulated, as part of other configurations.

Application of the ICE Notation to Genetic Algorithms. The ICE representation can be used as the basis for genetic algorithms. Configurations would be represented in ICE and the evolution patterns would be based on patterns of random ICE transformations. These would result in more intricate evolution patterns than those produced by typical binary mutations used in genetic algorithms.

Integration with Evaluation Systems. ICE can be integrated with a design evaluation system: as a user explores alternate solutions, his/her design can be evaluated in real time, thereby enabling him/her to continuously compare the results of the exploration. In this scenario, regulators and evaluators work together to guide users in transforming design configurations in ways that improves the quality of the design.

Integration with Generative Systems. Regulators can be augmented to generative systems, in order to enable users to further manipulate the generated results. Shape configurations can be represented as ICE strings, while generative rules would be represented as ICE transformations. In the present context, users generate and control regulators. In a generative context, the system can generate regulators as part of configurations, therefore making generated configurations very flexible. Furthermore, generative systems can focus on the use of specific regulators, in order to promote exploration within certain styles.

APPENDIX A

BIBLIOGRAPHY

Aish, 2005 Aish, R. – Introduction to GenerativeComponents**. –** *http://www.bentley.com/en-US/Markets/Building/White+Papers/White+Papers.htm*

Akin, 1978 Akin, O. – How Architects Design. – *Artificial Intelligence and Pattern Recognition in Design, North Holland Publishing Company, Amsterdam, 1978.*

Akin, 1978b Akin, O. – Quantification of Three-Dimensional Structure. – *Journal of Experimental Psychology, American Psychology Association, 1978*.

Akin, 1980 Akin, O. – Structural Properties of Three Dimensional Block Arrangements. –*Technical Report, Carnegie Mellon University, Pittsburgh. 1980*

Akin, 1986 Akin, O. – Psychology of Architectural Design. – *Pion Ltd., London, 1986.*

Akin, 1987 Akin, O. – Problem Structuring in Architectural Design. – *Technical Report, Carnegie Mellon University, Pittsburgh, 1987.*

Akin, 1994 Akin, O. – Psychology of Early Design. – *Technical Report, Carnegie Mellon University, Pittsburgh, 1994.*

Akin, 1996a Akin, O., Akin C. – Expertise and Creativity in Architectural Design. – *Descriptive Models of Design, Istanbul, 1996.*

Akin, 1996b Akin, O., Akin, C. – Frames of Reference in Architectural Design: Analyzing the Hyper Exclamation! AHA!. – *Design Studies, vol.17, Elsevier Science, New York, 1996.*

Akin, 2003 Akin, Ö., Moustapha H. – Strategic Use of Representation in Architectural Massing. – *Design Studies, vol.25 #1, Elsevier Ltd., London, 2003.*

Akin, 2004 Akin, Ö., Moustapha, H. – Formalizing Generation and Transformation in Design: a studio case study. – *First International Conference on Design Computing and Cognition (DCC'04), Kluwer Academic publisher, the Netherlands, 2004.*

AutoDesk, 2005 AutoDesk. – An Introduction to AutoDesk Revit. http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=3782406

AutoDesk, 2005 AutoDesk. – AutoDesk Revit User Guide. http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=5107070

Archea, 1987 Archea, J. – Puzzle Making: What Architects Do When No One is Looking. – *Computability of Design, John Wiley and Sons Inc., New York, 1987.*

Argawal, 1997 Argawal, M. – Shape Grammars and their Languages: a Methodology for Product Design and Product Representation. – *ASME, Sacramento, 1997.*

Badros, 1998 Badros, G., et, al. – Cassowary Linear Arithmetic Constraint Solving Algorithm. – *Technical Report, University of Washington, Seattle, 1998.*

Baglivo, 1983 Baglivo, J., Graver, J. – Incidence and Symmetry in Design and Architecture. – *Cambridge University Press, London, 1983.*

Baird, 1973 Baird I.C. – Designing with Volumes. – *Cantlab Press, Cambridge England, 1973.*

Baker, 1996 Baker, G. – Design Strategies in Architecture. – *E & FN Spon, 1996.*

Baudish, 1996 Baudish, P. – The Cage: Efficient Construction in 3D Using a Cubic Adaptive Grid. – *UIST 96, ACM Press, Seattle, 1996.*

Baykan, 1997 Bakan, C., Fox, M. – Spatial Synthesis by Disjunctive Constraint Satisfaction. – *AI EDAM, Cambridge University Press, 1997.*

Blackwell, 1984 Blackwell, W. – Geometry in Architecture. – *John Wiley and Sons Inc., 1984.*

Booth, 1983 Booth, N. – Basic Elements of Landscape Architectural Design. – *Elsevier, New York, 1983.*

Brier, 1986 Brier, E. – Stone, M. Snap-Dragging. – *ACM Siggraph, vol.20, #4, ACM Press, Dallas, 1986.*

Brier, 1988 Bier, E. – Snap-Dragging: Interactive Geometric Design in Two and Three Dimensions. – *Technical Report, University of California at Berkley, 1988.*

Bruegge, 2003 Bruegge, B., Dutoit, A. – Object Oriented Software Engineering: Using UML, Patterns and Java. – *2nd edition. Prentice Hall, NJ, 2003.*

Cagan, 2001 Cagan J. – Engineering with Grammars. *Formal Engineering Design Synthesis. – Cambridge University Press, New York, 2001.*

Carlson, 1989 Carlson, C. – Structured Grammars and Their Applications to Design. – *Technical Report, Carnegie Mellon University, Pittsburgh, 1989.*

Carlson, 1990 Woodbury R., Carlson C. – Hands on exploration of Recursive forms. – *Technical Report, Carnegie Mellon University, Pittsburgh, 1990.*

Carlson, 1993 Carlson, C. – Grammatical Programming. – *Ph.D. Disseration, Carnegie Mellon University, Pittsburgh, 1993.*

Cha, 1998 Cha, M.Y., Gero, J. – Shape pattern recognition using a computable shape pattern representation. – *Artificial Intelligence in Design '98, Kluwer, Dordrech, 1998.*

Cha, 1999 Cha, M.Y., Gero, J. – Style Learning: Inductive Generalization On Architectural Shape Patterns. – *Architectural Computing from Turing to 2000, eCAADe, University of Liverpool, Liverpool.*

Cha, 200? Cha, M.Y., Gero, J. – Shape Pattern Representation for Design Computation. – *http://www.arch.usyd.edu.au/%7Ejohn/publications/ChaGero.pdf*

Chase, 1988 Chase, S. – Shapes and Shape Grammars: from Matematical Models to Computer Implementations. – *Environment and Planning B, vol.16 Pion Ltd., London, 1988.*

Chase, 1996 Chase, S. – Representing Designs with Logic Formulations of Spatial Relations. –*Visual Representation, Reasoning and Interaction in Design, 4th Conference on Artificial Intelligence in Design, Stanford University, 1996.*

Ching, 1996 Ching, F. – Form, Space and Order. – *John Wiley & Sons Inc., New York, 1996.*

Ching, 1997 Ching, F. – A Visual Dictionary of Architecture. – *John Wiley & Sons Inc., New York, 1997.*

Clark, 1982 Clark, R. – Analysis of Precedents. – *North Carolina State University, 1982.*

Clark, 1985 Clark, R. – Precedence in Architecture. – *Van Nostrand Reinhold, New York, 1985.*

Coxeter, 1987 Coxeter, H. – Geometry Revisited. – *The Mathematical Association of America, Washington D.C. 1987.*

Cromwell, 1997 Cromwell, P. – Polyhedra. – *Cambridge University Press, Cambridge, 1997.*

Curtis, 1935 Curtis, N. – Architectural Composition. – *J. H. Jansen, C, 1935.*

Dix, 1993 Dix et. al. – Human Computer Interaction. – *Prentice Hall, New York, 1993.*

Dohmen, 1995 Dohmen. M. – A Survey of Constraint Satisfaction Techniques. – *Computer and Graphics, vol.19 #6, Elsevier Science Ltd., New York 1995.*

Eastman, 1987 Eastman, C. – The Design of Assemblies. – *Technical Report, Carnegie Mellon University, Pittsburgh, 1987.*

Economou, 1999- Economou, A. – The Symmetry Lessons of Froebel Gifts. – *Environment and Planning B, vol.26, Pion Ltd., London 1999.*

Eggink, 2001 Eggink, et. al. – Smart Objects: Constraints and Behavior in a 3D Design Environment. – *Modeling and Planning, Virtual Environments, New York, 2001*.

Emmer, 1993 Emmer, M. – The Visual Mind. – *MIT Press, Cambridge, 1993.*

Fernando 1993 Fernando et. al. – Interactive Constraint Based Solid Modelers Using Allowable Motion. – *Second Symposium on Solid Modeling and Applications, ACM press, New York, 1993.*

Ferrante, 1991 Ferrante A. et al. – Computer Graphics for Architects and Engineers. – *Elsevier, New York, 1991.*

Finke, 1989-- Finke, R. – Principles of Mental Imagery. – *MIT Press, Cambridge, Mass, 1989.*

Flemming, 1990 Flemming U. – Syntactic Structure in Architecture. – *The Electronic Design Studio, MIT Press, Cambridge,1990.*

Flemming, 1995 Flemming, U., Chien. S. Schematic Layout Design in SEED Environment. – *Journal of Architectural Engineering American Society of Civil Engineers, 1995.*

Foley, 1996 Foley, J., VanDam, A. – Computer Graphics: Principles and Practices. – *Addison Wesley Publishing Co., Holland, 1996.*

Foz, 1973 Foz, A. – Observation on Designer Behavior in the Parti. – *In The Design Activity International Conference, vol.1, Printing Unit, University of Strathclyde, Glasgow, 1973.*

Frohlich, 1996 Frohlich, D. – Direct Manipulation and Other Lessons. – *Hewlett Packard Laboratories, Bristol 1996.*

Gabriel, 1997 Gabriel J., F. – Beyond The Cube: The Architecture of Space Frames and Polyhedra. – *John Wiley & Sons Inc., 1997.*

Gamma, 1996 Gamma, E., Helm, R. – Design Patterns: Elements of Reusable Object-Oriented Software. – *Addison Wesley Publishing Co., Holland, 1994.*

Gero, 1994 Gero, J., Yan, M. – Shape Emergence by Symbolic Reasoning. – *Environmental and Planning B, vol.21, Pion Ltd., London 1994.*

Gero, 1995 -- Gero J., Jun, H. Getting Computers to Read the Architectural Semantic of Drawings. – *ACADIA'95, Seattle, 1995.*

Gero, 1997 Gero, J., Damsky, J.C. – A Symbolic Model for Graphical Emergence. – *Environmental and Planning B, vol.24, Pion Ltd., London 1997.*

Gero, 1998 -- Gero, J., Jun, H. – Emergence of Shape Semantics of Architectural Shapes. – *Environmental and Planning B, vol.25, Pion Ltd., London 1998.*

Gleicher, 1990 Gleicher, M. – Snap Together Mathematics. – *Technical Report, Carnegie Mellon University, Pittsburgh, 1990.*

Gleicher, 1991 Gleicher, M., Witkin, A. – Creating and Manipulating Constrained Models. – *Technical Report, Carnegie Mellon University, Pittsburgh, 1991.*

Gleicher, 1993a Gleicher, M. – Drawing with Constraints. – *In the Visual Computer, Springer, Berlin, 1993.*

Gleicher-1993b Gleicher, M., Witkin, A. – Supporting Numerical Computations in Interactive Contexts. – *Graphics Interfaces '93, Morgan Kauffmann, San Francisco, CA, 1990.*

Gleicher, 1994 Gleicher, M. – A Differential Approach to Graphical Interaction. – *Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, 1994.*

Goldshmidt, 1990 Goldshmidt, G. – On Visual Design Thinking: The Vis Kids of Architecture. – *Design Studies, vol.15 #2, Butterworth and Co. Ltd., 1990.*

Goldsmidt 1996 Goldshmidt, G. – Capturing Indeterminism: Representation of the Design Problem Space. – *Descriptive Models of Design, Takisla, Istanbul, 1996.*

Grim., 1995 Grim, et. al. – Visual Interfaces for Solid Modeling. – *UIST 95, Pittsburgh, ACM Press, 1995.*

Gross, 1987 Gross, M. et al. – Designing with Constraints. – *In Computability of Design, John Wiley and Sons Inc., New York, 1987.*

Gross, 1990 Gross, M. – Relational Modeling A Basis From Computer Aided Design. – *The Electronic Design Studio, MIT Press, Cambridge, 1990.*

Gross, 1991 Gross, M. – Grids in Design and CAD. – *Proceedings ACADIA 91-Reality and Virtual Reality, Los Angeles, 1991.*

Gross, 1992 Gross, M. – Graphical Constraints in CoDraw. – *IEEE Workshop on Visual Languages, Seattle WA, 1992.*

Gross, 1996a Gross, M. – Why can't CAD be more like Lego. – *Automation in construction, Elsevier, New York, 1996*

Gross, 1996b Gross, M. – Form Writer. – *CAAD Futures 2001, Kluwer Academic Publishers, The Netherlands, 1996.*

Guan, 1996-- Guan, X., et al. – A Prototype System for Early Geometric Configuration Design. – *Computers in Industry, vol.30, 1996.*

Harada, 1995 Harada, M., et. al. – Interactive Physically Based Manipulation of Discrete Continuous Models. – *SIGGRAPH'95, Computer Graphics Proceeding Annual Conference Series, ACM Press, Montreal 1995.*

Harada, 1997 Harada, M. – Discrete/Continuous Design Exploration by Direct Manipulation. – *Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, 1997.*

Hargittai, 1998 Hargittai, Istvan. – Symmetry II: Unifying Human Understanding. – *Pergamon Press, Oxford, 1998.*

Heiserman, 1993 Heiserman, J. Woodbury, R. – Generating Languages of Solid Models. – *Second Symposium on Solid Modeling and Applications, ACM Press, New York, 1993.*

Herndon, 1992 Herndon, K., et, al. – Interactive Shadows. – *UIST '92, ACM Press, New York, 1992.*

Hill, 2001 Hill, F. – Computer Graphics Using Open OpenGL. – *Prentice Hall, New Jersey, 2001.*

Hoffmann, 1989 Hoffmann, C. – Geometric and Solid Modeling: An Introduction. – *Morgan Maufmann Publishers Inc., San Mateo, CA, 1989.*

Hoggar, 1992 Hoggar, S. – Mathematics for Computer Graphics. – *Cambridge University Press 1992.*

Honda, 1999 Honda et.al. – Integrated Manipulation: Context Aware Manipulation of 2D Diagrams. – *UIST 99, Asheville, 1999.*

Hudson, 1996 Hudson S., Smith, I. – Ultra Lightweight Constraints. – *UIST 96, ACM Press, Seattle, 1996.*

Huybers, 1993 Huybers, P. – Computer Aided Design of Polyhedral Building Structures. – *Design Studies, vol.14, #1, Butterworth and Co. Ltd., 1993.*

Josuttis, 1999 Josuttis, N. – The C++ Standard Library: A Tutorial and Reference. – *Addison Wesley Longman Inc, Reading Mass., 1997.*

Knight, 1983 Knight, T. – Transformation of Languages of of Design (part 1 to 3). – *Environmental and Planning B, vol.10, Pion Ltd., London, 1983.*

Knight, 1994 Knight, T. – Shape Grammars and Color Grammars in Design. – *Environmental and Planning B, vol.21, Pion Ltd., London, 1994.*

Knight, 1995a Knight, T. – Constructive Symmetry. – *Environment and Planning B, vol.21, Pion Ltd., New York, 1995.*

Knight, 1995b Knight, T. – Transformations in Design: A Formal Approach to Stylistic Change and Innovation in Visual Art. – *Environment and Planning B, vol. 21, Pion Ltd., New York,* 1995.

Knight, 1999 Knight, T. – Shape Grammars: Six types. – *Environmental and Planning B, vol 26, Pion Ltd., London, 1999.*

Kolarevic, 1993 Kolarevic, B. – Geometric Relations as a Framework Design Conceptualization. – *Ph.D. Dissertation, Harvard University, Graduate School of Design, 1993.*

Kolarevic, 1997 Kolarevic, B. – Lines And Geometric Relations as a Framework for Exploring Shape, Dimension and Geometric Organization in Design. – *CAAD Futures 97, Kluwer Academic Publishers, Munich, 1997.*

Krawczyk, 1997 Krawczyk, R. – Programs as Pencils: Investigating Form Generation. – *ACADIA 9, Ohio, 1997.*

Krawczyk, 2001 Krawczyk, R. – Curving Spirolaterals. – Mathematics and Design 2001, Third International Conference, *Deakin University, Geelong, 2001.*

Krier, 1988a Krier, R. – Architectural Composition. – *Rizolli International Publication Inc., New York, 1988.*

Krier, 1988b Krier, R. – Urban Space. – *Rizolli International Publication Inc., New York, 1988.*

Krishnamurti, 1979 Krishnamurti, R. – On the Generation and Enumeration of Tessellation Designs. – *Environment and Planning B, vol.6, Pion Ltd., London 1979.*

Krishnamurti, 1992 Krishnamurti, R. – The Maximal Representation of a Shape. – *Environment and Planning B, vol.19, Pion Ltd., London 1992.*

Laffra, 1995 Laffra. C. – Object Oriented Programming For Graphics. – *Springer, New York, 1995.*

Laseau, 1992 Laseau, P., Tice J. – Frank Lloyd Wright: Between Principle and Form. – *Van Nostrand Reinhold, 1992.*

Le Corbusier, 1960 Le Corbusier. – Towards a new Architecture. – *Preager Publishers Inc., 1960*.

Leeuven, 1997 Leeuven, J. – Architectural Design by Features. – *CAAD Future' 97 Proceedings, Kluwer Academic Publishers, Munich, 1997.*

Leeuven, 1997 Leeuven, J. – Architectural Design by Features. – *CAAD Futures'99 Proceedings, Kluwer Academic Publishers, Munich 1997.*

Leyton, 1992 Leyton, M. – Symmetry, Causality, Mind. – The MIT Press, Cambridge, Mass., 1992.

Leyton, 1999 Leyton, M. – Group Theory and Architecture I and II. – Visual Mathematics, vol.1 #3, and vol.1 #4, 1999.
http://members.tripod.com/vismath/pap.htm

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

Liu, 2000 Liu, Y. – Computational Symmetry. – Technical Report, Carnegie Mellon University, Pittsburgh, 2001**.**

Mahdavi, 1997 Mahdavi, A. Suter, G. – On Implementing a Computational Facade Design Tool. – *Environment and Planning B, vol.24, Pion Ltd., London, 1997.*

Madrazo, 1994 Madrazo, L. Durant and The Science of Architecture. – *Journal of Architectural Education, vol.48 #1, Association of Collegiate Schools of Architecture, Washington, May 1994.*

Madrazo, 1995 Madrazo, L. – Concept of Type in Architecture. – *Federal Institute of Technology Zurich, 1995*.

Maher, 1990 Maher M. – Processes Models for Design Synthesis. – *AI Magazine, vol.11 #1, American Associations for Artificial Intelligence, 1990.*

Mantyla, 1988 Mantyla, M. – An Introduction To Solid Modeling. – *The Computer Science Press Inc., Maryland, 1988.*

March, 1972 March, L., Martin, L. – Urban Space and Structures. – *Cambridge University Press, London, 1972.*

March, 1974 March, L., Steadman P. – The Geometry of the Environment. – *The MIT Press, Cambridge, Massachusetts, 1974.*

March, 1976 March L. – The Architecture of Form. – *Cambridge University Press, London, 1976.*

Martin, 1991 Martin G. – Transformation Geometry: An Introduction to Symmetry. – *Springer, New York, 1991*.

Medjdoub, 1999 Medjdoub, B. – Interactive 2D Constraint-Based Geometric Construction. – *CAAD Futures 99, Kluwer Academic Publishers, Munich, 1999.*

Mitchell, 1977 Mitchell, W. – Computer Aided Architectural design. – *Charter, New York 1977.*

Mitchell, 1990 Mitchell, W. – The logic of Architecture. – *The MIT Press, Cambridge, 1990.*

Mitchell, 1993 Mitchell, W. – A Computational View of Design Creativity. – *Modeling Creativity and Knowledge Based Creative Design. Lawrence Erlbaum Associates Publishers, Hillsdale, New Jersey, 1993.*

Molinari, 1999 Molinari, L. – Santiago Calatrava. – *Skira, Milan, 1999.*

Mortenson, 1995 Mortenson, M. – Geometric Transformations. – Industrial Press, New York, 1995.

Mortenson, 1997 Mortenson, M. – Geometric Modeling. – Second Edition, *John Wiley and Sons, New York, 1997.*

Mortenson, 1999 Mortenson, M. – Mathematics for Computer Graphics Applications. – Industrial Press, New York, 1999.

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

Moustapha, 2004 Moustapha, H. – A Formal Representation for Generation and Transformation in Design. – *Generative CAD Systems Symposium (GCAD'04), Carnegie Mellon University, Pittsburgh, 2004.*

Museum, 1972 Museum of Modern Art. – Five Architects: Eisenman, Graves, Gwathmey, Hejduk, Meier. – *Wittenborn, New York, 1972.*

Neilsen, 1993 Neilsen, J. – Usability Engineering. *AP Professional, Boston, 1993.*

Onat, 1991 Onat, E. – Architecture, Form, and Geometry. – *Yem Yayin Istambul, 1991.*

Oxman, 2002 Oxamn, R. – The Thinking Eye: Visual Recognition in Design Emergence. – *Design Studies vol.19 #5, New York, 2002.*

Papazian, 1993 Papazian, P. – Incommensurability of Criteria and Focus in Design Generation. – *CAAD FUTURES 93, Elsevier, New York. 1993.*

Pohl, 1994 Pohl, I. – C++ for C Programmers. – *Second Edition, The Benjamin Cummings Publishing Company Inc., Redwood City, CA., 1994.*

Pomerantz, 1991 Pomerantz, J. – The Structure of Visual Configurations. – *In The Perception of Structure, G. Lockhead ed. The American Psychological Association, 1991.*

Purcell, 1998 Purcell, A., Gero, J. – Drawings and the Design Process. – *Design Studies vol. 19 #4, New York, 1998.*

Raisamo, 1996 Raisamo et. al. – Techniques for Aligning Objects in Drawing Programs. – *Technical Report, University of Tampere Finland, 1996.*

Rappoport, 1997 Rappoport, A., Spitz, S. – Interactive Boolean Operations for Conceptual Design of 3-D Solids. – *Siggraph 97, LA, 1997.*

Robinson, 1914 Robinson, J. – Architectural Composition. – *Van Nostrand Company. London, 1914.*

Rosen, 1995a Rosen, J. – Symmetry in Science. – *Springer-Verlag, New York, 1995.*

Rosen, 1995b Rosen, K. – Discrete Mathematics and its Applications. – *McGraw Hill, New York, 1995.*

Rumbaugh, 1996 Rumbaugh, J., Blaha, M. – Object-Oriented Modeling and Design. – *Prentice-Hall, Englewood Cliffs NJ, 1991.*

Sapossnek, 1991 Sapossnek M. – Research on Constraint-Based Design Systems. – *Technical Report, Carnegie Mellon University, Pittsburgh 1991.*

Schild, 1991 *Schild H.* – C, The Pocket Reference. *Second Edition.* – *McGraw Hill, Berkeley, CA., 1990*

Shepard, 1978 Shepard, R. – The Mental Image. – *American Psychologist, vol.23 #2, 1978.*

Shepard, 1982 Shepard, R. – Mental Images and Their Transformation. – *MIT Press, Cambridge, Mass, 1982.*

Shubnikov, 1974 Shubnikov, A., Koptsik V. – Symmetry in Art and Science. – *Plenum Press, New York, 1974.*

Simon, 1969 Simon H. – The Science of the Artificial. – *MIT Press, Cambridge, Mass, 1969.*

Simon, 1984 Simon H. – The Structure of Ill-structured Problems. – *Developments in Design Methodology, John Wiley and sons, New York, 1984.*

Speray, 1990 Speray, D. – Volume Probes: Interactive Data Exploration on Arbitrary Grids. – *Computer Graphics, vol.24, #5, 1990.*

Stamps, 1998 Stamps, A. – Measures of Architectural Mass: From vague impressions to definite design features. – *Environmental and Planning B, vol.25, Pion Ltd., London 1998.*

Steadman, 1983 Steadman P. – Architectural Morphology. – *Pion Ltd., London, 1983.*

Steadman, 1994 Steadman, P. – Built Forms and Building Types: Some Speculations. – *Environmental and Planning B, vol.21, Pion Ltd., London 1994.*

Steadman, 1998 Steadman, P. – Sketch for an Archetypal Building. – *Environmental and Planning B, Anniversary Issue, Pion Ltd., London 1998.*

Stiny, 1976 Stiny, G. – Two Exercises in Formal Compositions. – *Environment and Planning B, vol.3, Pion Ltd., New York, 1976.*

Stiny, 1977 Stiny, G. – Ice Ray: a Note on the Generation of Chinese Lattice Designs. – *Environmental and Planning B, vol.4, Pion Ltd., London, 1977.*

Stiny, 1980a Stiny G. – Introduction to Shape and Shape Grammars. – *Environment and Planning B, vol.7, Pion Ltd., London 1980.*

Stiny, 1980b Stiny G. – What Designers Do that Computers Should. – *The Electronic Design Studio, MIT Press, Cambridge,1990.*

Stiny, 1980c Stiny, G. – Kindergarten Grammars: Designing with Froebels Building Gifts. – *Environment and Planning B, vol.7, Pion Ltd., London, 1980.*

Stouffs, 1994 Stouffs, R. – The Algebra of Shapes. – *PhD Dissertation, Carnegie Mellon University, Pittsburgh, 1994.*

Stouffs, 2001 Stouffs, R., Krishnamurti, R. – Sortal Grammars as a Framework for Exploring Grammar Formalisms. – *Mathematics and Design 2001, Third International Conference, Deakin University, Geelong, 2001.*

Stroustrup, 1997 Stroustrup, B. – The C++ Programming Language. – *Third Edition, Addison Wesley Longman Inc, Reading, Mass., 1997.*

Suter, 1999 Suter, G. – A Representation for Design Manipulations and Performance Simulation. – *Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, 1999.*

Sutherland, 1963 Sutherland, I.E. – Sketchpad: A Man Made Graphical Communication System. – *Massachusetts institute of technology, 1963.*

Tan, 1990 Tan, M. – Saying what is by what is like. – *The Electronic Design Studio, MIT Press, Cambridge, 1990.*

Thadani, 1992 Thadani, D. – Five Architects: Twenty Years Later. – *University of Maryland, College Park, 1992.*

Thompson, 1971 Thompson, D. – On Growth and Form. – *Cambridge University Press, London, 1971.*

Tobin, 1991 Tobin M. – Constraint-Based Three Dimensional Modeling as a Design Tool. – *Reality and Virtual Reality (ACADIA 1991 proceedings) Association for Computer Aided Design in Architecture, 1991.*

Turabian, 1987 Turabian, K. – A Manual for Writers. – *Fifth Edition, The University of Chicago Press, Chicago, 1987.*

Tzoniz, 2004 Tzonis, A. – Santiago Calatrava: The Complete Works. – *Rizzoli, New York, 2004.*

VanLeeuven, 1997 Van Leeuven, J. – Architectural Design by Features. – *CAAD Futures 97, Kluwer Academic Publishers, The Netherlands, 1997.*

Veltkamp, 1996 Veltkamp, R., Arbab, F. – Interactive Geometric Constraint Satisfaction. – *Stichting Mathematish Centrum, Amsterdam 1996.*

Verstijen, 1998 Verstijen, I., et. al. – Sketching and Creative Discovery. – *Design Studies, vol. 23 #2, New York 1998.*

Viega, 1996 Viega, et. al. – 3D Magic Lenses. – *UIST 96, ACM Press, Seattle, 1996.*

Weichsel, 1999 Weichsel, J. – Pattern Formation under Various Tiling Rules. – *Computer and Graphics, vol.23, Elsevier Science, New York, 1999.*

Weterguard, 1992 Weterguard, C. – Visualizing negative space. – *Computer Supported Design in Architecture, Mission, Method and Madness, ACADIA, 92 Charleston, 1992.*

White, 1980 White, E. – Concept Sourcebook: A Vocabulary of Architectural Forms. – *Architectural Media Ltd., 1980.*

Williams, 1999 Williams, K. – Symmetry in Architecture. – *Visual Mathematics, vol.1 #1, 1999.* http://members.tripod.com/vismath/pap.htm

Wisskirchen, 1990 Wisskirchen, P. – Object Oriented and mixed Programming paradigms. – *Springer New York, 1990.*

Witkin, 1990 Witkin, A., et al. – Linking Perception and Graphics: Modeling with Dynamic Constraint. – *Images and understanding, Cambridge University Press, 1990.*

Wong, 1993 Wong, W. – Principles of Form and Design. – *John Wiley and Sons Inc., New York, 1993.*

Woo, 1998 Woo, M. et. al. – OpenGL Programming Guide. – Second Edition, Addison Wesley Longman Inc, Reading Mass., 1998.

Woodbury, 1987 Woodbury, R. – Strategies for Interactive Design Systems. – *In Computability of Design, John Wiley and Sons Inc., New York, 1987.*

Woodbury, 1987 Woodbury, R. – Knowledge Based Representation and Manipulation of Geometry. – *Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, 1987.*

Woodbury, 1995 Woodbury, R., Cheng, T. – Massing and Enclosure Design with SEED Config. – *Journal of Architectural Engineering American Society of Civil Engineers, 1995.*

Yale, 1968 Yale, P. – Geometry and Symmetry. – *Dover, New York, 1968.*

Yessios, 1987 Yessios, C. – Computability of Void Architectural Modeling. – *In Computability of Design, John Wiley and Sons Inc., New York, 1987.*

APPENDIX B

THE MATHEMATICS OF REGULATORS

Regulators encapsulate a formula, a polynomial equation, for controlling its associated elements. In most cases, this formula is determined by the geometry of the regulator. In this Appendix, I discuss the geometric representation of regulators in general, and the specific mathematical properties for each regulator type, as well as for transforming and compositing regulators.

B.1. THE GEOMETRY OF REGULATORS

The geometry of regulators is based on vector mathematics. Point regulator, line regulator and plane regulators are represented as vectors in 3D space.

B.1.1. REPRESENTATION OF POINTS

A point $\overline{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$

Distance between two points

$$\overline{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \text{ and } \overline{q} = \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} = \sqrt{(q_x - p_x)^2 + (q_y - p_y)^2 (q_z - p_z)^2}$$

B.1.2. REPRESENTATION OF LINES

A line l with starting point $\bar{p}$ and end point $\bar{q}$ and direction along $\bar{t}$

$l = \bar{q} = \bar{p} + d\bar{t} \;=\; \bar{q} = \bar{p} + d(\bar{q} - \bar{p})$ where $\bar{t} = (\bar{q} - \bar{p})$

$$q_x = p_x + dt_x \qquad \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} + d \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$
$$q_y = p_y + dt_y$$
$$q_z = p_z + dt_z$$

Magnitude:

Line: $|l| = \sqrt{(q_x - p_x)^2 + (q_y - p_y)^2 (q_z - p_z)^2}$ Vector: $|\bar{t}| = \sqrt{(t_x)^2 + (t_y)^2 (t_z)^2}$

Unit vector: $\bar{u} = \bar{t} / |t|$

Direction cosines:

$$t_x = |t| \cos\psi \qquad u_x = \cos\psi$$
$$t_y = |t| \cos\theta \qquad u_y = \cos\theta$$
$$t_z = |t| \cos\phi \qquad u_z = \cos\phi$$

Where ψ, θ, ϕ are the angles of vector $\bar{t}$ with the x, y, z axes respectively.

Midpoint of a line:

$$Mid_x = \frac{\bar{p}_x + \bar{q}_x}{2}, \; Mid_y = \frac{\bar{p}_y + \bar{q}_y}{2}, \; Mid_z = \frac{\bar{p}_z + \bar{q}_z}{2}$$

Point on the line

New point
if s < d then the point is on the line
if s > d then the point extends beyond the line

B.1.3. REPRESENTATION OF PLANES

Given three points in a plane $\overline{P}_0$, $\overline{P}_1$ and $\overline{P}_2$.

$\overline{P} = \overline{P}_0 + d\bar{t} + e\bar{v} = \overline{P}_0 + d(\overline{P}_1 - \overline{P}_0) + e(\overline{P}_2 - \overline{P}_1)$

The plane's normal $\overline{N} = \lfloor N_x \quad N_y \quad N_z \rfloor$

$\overline{N} = \bar{t} \otimes \bar{v} = (\overline{P}_1 - \overline{P}_0) \otimes (\overline{P}_2 - \overline{P}_1)$

$\overline{N} \bullet \bar{t} = 0$

$\overline{N} \bullet (\overline{P}_0 - \overline{P}_1) = 0$

$\overline{N}_x \bar{t}_x + \overline{N}_y \bar{t}_y + \overline{N}_z \bar{t}_z = 0$

If $\overline{N} = \lfloor A \quad B \quad C \rfloor$ and $\bar{t} = \lfloor x \quad y \quad z \rfloor$ the Plane equation = $Ax + By + Cz + D = 0$

To determine the angle between a plane and the major coordinate planes (e.g. x-y plane), it is necessary to compute the angle between the normal of both planes.

B.2. TRANSFORMATION REGULATORS

Transformation regulators are based on isometry and affine transformations. An isometry transformation is a collineation i.e. it preserves linearity. It also preserves, distances, angles, areas, parallels, perpendiculars, between-ness and midpoints. The determinant of the transformation matrix for an isometry transformation is ± 1. Even isometries preserve orientations (determinant = $+1$), while odd isometries reverse orientations (determinant = -1). A similarity transformation is any combination of an isometry with a uniform scaling. Together, these form the group of similarities, (every similarity has an inverse, which is also a similarity, and the product of two similarities is a similarity). The group of similarity transformations subsumes the group of isometries. Affine transformations subsume similarity transformations. These preserve collinearity and parallelism. The inverse of an affine transformation is affine, and the product of two affine transformations is also affine. Distances and angles, however, are not preserved. The determinant of its coefficient matrix is $\neq 0$.

Transformation regulators operate by applying an equation (or transformation matrix) to the input element to derive the output set of elements. These regulators use the properties of their respective transformations to preserve points, lines, and planes, as a visual depiction for the regulators. Transformations in space have a polynomial equation. Their arguments are expressed by the

coefficients of the (4x4) transformation matrix (using homogenous coordinates). In order for a transformation to apply to a shape, it must be multiplied to every vertex. A vector $\bar{v}$ is transformed by T resulting in $\bar{v}'$

Function notation: $T(\bar{v}) = \bar{v}'$ Vector/matrix notation: $\bar{v}' = T\bar{v}$

$$x' = a_{11}x + a_{12}y + a_{13}z + a_{14}$$
$$y' = a_{21}x + a_{22}y + a_{23}z + a_{24}$$
$$z' = a_{31}x + a_{32}y + a_{33}z + a_{34}$$

$$\begin{bmatrix} v'_x \\ v'_y \\ v'_z \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ 1 \end{bmatrix}$$

B.2.1. GROUP PROPERTIES OF TRANSFORMATIONS (MORTENSON 1995)

In general, a group consists of a set of elements and an operator acting on these elements. In this case, the transformations are the elements, and the operator is the composition of transformation. In order for a set of transformations to form a group it needs to have the following properties.

- Closure: $A \in S \wedge B \in S \Rightarrow A \circ B \in S$ if two transformations are in a group then their composition is also in the group.
- Identity: $\exists I \therefore A \circ I = A \wedge I \circ A = A$. There exists an identity transformation I, such that A composed with I leaves A unchanged.
- Inverse: $\forall A \in S$.....$\exists A^{-1} \ \therefore A \circ A^{-1} = I$. For every transformation A in the group, there exists an inverse A-1 such that A composed with A-1 results in the identity transformation.
- Associativity: $A \circ (B \circ C) = (A \circ B) \circ C$
- Commutativity: (only for Abelian groups) $A \circ B = B \circ A$.

The following is a description for each transformation that corresponds to a regulator. It is applied about the origin and uses the xyz-coordinate axes. The common strategies for transforming about arbitrary points, lines and planes, are described in section B.3.

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

Translation is an even isometry. The determinant of its coefficient matrix is +1. Translations form an Abelian group because of the following:

Translation Matrix

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

The product of two or more translations is a translation

$$T_{x,y,z} \circ T_{a,b,c} = T_{x+a,y+b,z+c}$$

The inverse of a translation is a translation.

$$T_{x,y,z}{}^{-1} = T_{-x,-y,-z}$$

Composition of translation is commutative (the order of application is immaterial)

Rotation

$\Delta\mathbf{R}^{\mathbf{1}}[\ \{\bar{p},\bar{t},\alpha,n\}$ (shape)] or $\Delta\mathbf{R}^{\mathbf{0}}[\ \{\bar{p},\alpha,n\}$ (shape)])]

Rotation is an even isometry. Its coefficient matrix has a determinant of 1. Furthermore, a rotation matrix is an orthogonal matrix; therefore, its inverse is equal to its transpose. An improper rotation is a rotation that has a determinant of –1, and is actually a combination of a rotation and a reflection. Any rotation in space can be achieved by means of successive rotations along the three principal axes. All rotations form a group because of the following:

The inverse of a rotation is a rotation: $\mathbf{R}_{\phi}^{-1} = \mathbf{R}_{-\theta}$

The product of two rotations is a rotation. $\mathbf{R}_{\phi} \circ \mathbf{R}_{\theta} = \mathbf{R}_{\phi+\theta}$

Composition of rotations is not commutative; except for the product of two rotations with the same axis, which yields a rotation about the same axis. $\mathbf{R}_{\phi}\mathbf{R}_{\theta}\mathbf{R}_{\psi}$

About the **z** axis (yaw)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

About the **y** axis (pitch)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

About the **x** axis (roll)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi & 0 \\ 0 & \sin\psi & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Mirror (Reflection)

$\Delta\mathbf{M}^{0}[\ \{\overline{p},n\}$ (shape) $]\quad \Delta\mathbf{M}^{1}[\ \{\overline{p},\overline{t},n\}$ (shape) $]\quad \Delta\mathbf{M}^{2}[\ \{\overline{p},\overline{t},\overline{v},n\}$ (shape) $]$

Reflections are odd isometries, their determinant is –1.

A reflection is an involution, meaning it is its own inverse. $\mathbf{M}^{-1} = \mathbf{M}$ and $\mathbf{M} \circ \mathbf{M} = \mathbf{I}$

There are three types of reflections: (1) Reflection about a point, also called an inversion, (2) Reflection about a line, which in 3D space, is equivalent to a half turn about the line and is not really a reflection because the determinant is +1, and (3) Reflection about a plane. Reflection is considered the building block of all isometries: a translation can be described using two parallel reflections and a rotation can be described using two intersecting reflections.

- The product of two inversions is a translation.
- The product of three inversions is an inversion.
- The product of reflection of parallel planes is a translation
- The product of reflection of intersecting planes is a rotation about the line of intersection of the two planes.

Inversion about a point

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Reflection about the x-axis

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Reflection about the y-axis

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Reflection about the z-axis

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Reflection about the x-y plane

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Reflection about the x-z plane

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Reflection about the y-z plane

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

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

Screw rotation is a composition of rotation and translation along the rotation axis. It can be achieved by matrix multiplication. The translation factor is the pitch of the screw. It is also an even isometry meaning its determinant is 1.

Screw rotation matrix

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & t\phi \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

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

$\Delta\mathbf{T}^{\mathbf{1}}\Delta\mathbf{M}^{\mathbf{2}}[\ \{\bar{p},\bar{t},\bar{v},d,e,n\}\ \text{(shape)}\ ]$

Glide reflection is a composition of translation and reflection, with the translation vector on the reflection plane. It can be achieved by matrix multiplication.

Glide matrix

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

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

Isotopic Dilation is a similarity transformation, which has a uniform scaling factor k.

- If k>1 then its an expansion;
- If 0<k<1 then it is a contraction,
- If k < 0 then it is a scaled inversion (or half turn in 2D space).

Isotropic dilation fixing the origin

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} k & 0 & 0 & 0 \\ 0 & k & 0 & 0 \\ 0 & 0 & k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Anisotropic dilation is a non-uniform scaling. It is an affine transformation. If the scaling is unidirectional, (only in the x-direction for instance), it is referred to as strain. Negative factors in the dilation matrix's diagonal coefficients act as a scale coupled with a reflection.

Anisotropic dilation fixing the origin

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

The product of two dilations is a dilation.

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

Shear is an equiareal affine transformation, i.e. it preserves the area of shapes it transforms. In general an affine transformation multiplies the area by the absolute value of the determinant. In case of equiaffine or equiareal transformations, such as shear, the determinant of +1.

Shear fixing the y-z plane

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & s_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation can be expressed as the product of three shears. The product of two shears is a shear, and the inverse of a shear is a shear.

Two shears simultaneously

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & s_1 & 0 \\ 0 & 1 & s_2 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

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

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

	Parametric	Implicit
Circle	$x = a + r\cos\theta \quad y = b + r\sin\theta$	$x^2 + y^2 + z^2 + r^2 = 0$
Ellipse	$x = a\cos\theta \quad y = b\sin\theta$	$\frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0$
Parabola	$x = pt^2 \quad y = pt$	$x^2 - 4py = 0$
Hyperbola	$x = a\sec\theta$, $y = b\tan\theta$	$\frac{x^2}{a^2} - \frac{y^2}{b^2} - 1 = 0$
Trigonometric curves	$x = \theta, y = \cos\theta, y = \sin\theta, y = \tan\theta$	

Deformations (Nonlinear transformations)

Curves	Surfaces	Volumes
$x' = fx(u)$	$x' = fx(u,v)$	$x' = fx(u,v,w)$
$y' = fy(u)$	$y' = fy(u,v)$	$y' = fy(u,v,w)$
$z' = fz(u)$	$z' = fz(u,v)$	$z' = fz(u,v,w)$
univariate	bivariate	trivariate
C(t) = u(t)	C(t) = u(t), v(t)	C(t =u(t), v(t), w(t)

Tapering produces a global tapering about the z-axis

Twisting produces a global twist about the z-axis

Tapering

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} fx(z) & 0 & 0 & 0 \\ 0 & fy(z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Twisting

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos f(z) & -\sin f(z) & 0 & 0 \\ \sin f(z) & \cos f(z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

B.3. TRANSFORMING THE GEOMETRY OF REGULATORS

In order to apply a regulator about an arbitrary, point, axis, or plane, it is necessary to pre and post multiply by conjugate translations and rotation matrices. Below is an example of using this strategy to convert a reflection regulator M.

In order to reflect about a plane parallel to a principal axis, it is necessary to pre and post multiply the reflection by the translation matrix. The reflection matrix is translated to the origin then it is translated back to its position (The conjugate pair consists of a matrix and its inverse): $\mathbf{T}\ \mathbf{M}\ \mathbf{T}^{-1}$

$$\begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -tx \\ 0 & 1 & 0 & -ty \\ 0 & 0 & 1 & -tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In order to reflect through a rotated plane that passes through the origin, it is necessary to pre and post multiply the reflection by the three successive rotation matrices.

$$\mathbf{R}_{\psi}\ \mathbf{R}_{\theta}\ \mathbf{R}_{\phi}\ \mathbf{M}\ \mathbf{R}_{\phi}^{-1}\ \mathbf{R}_{\theta}^{-1}\ \mathbf{R}_{\psi}^{-1}$$

$$\begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi & 0 \\ 0 & \sin\psi & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\psi & \sin\psi & 0 \\ 0 & -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\phi & \sin\phi & 0 & 0 \\ -\sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If the rotated plane does not pass through the origin, it is necessary to pre and post multiply the above equation by a pair conjugate translations.

$$\mathbf{T}\ \mathbf{R}_{\psi}\ \mathbf{R}_{\theta}\ \mathbf{R}_{\phi}\ \mathbf{M}\ \ \mathbf{R}_{\phi}^{-1}\ \mathbf{R}_{\theta}^{-1}\ \mathbf{R}_{\psi}^{-1}\ \mathbf{T}^{-1}$$

$$\begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi & 0 \\ 0 & \sin\psi & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\psi & \sin\psi & 0 \\ 0 & -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\phi & \sin\phi & 0 & 0 \\ -\sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & -tx \\ 0 & 1 & 0 & -ty \\ 0 & 0 & 1 & -tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This is a commonly used strategy to define complex transformations though simpler matrices. It is applicable to the other regulators, such as Rotation and Dilation as well. For Dilation to fix any point in space, it needs to be pre and post multiplied by conjugate translation matrices $\mathbf{TDT}^{-1}$. For the Anisotropic dilation to be applied with respect to any three mutually orthogonal axes, the dilation matrix needs to be pre and post multiplied by the conjugate rotation matrices.

B.4. VARIATION REGULATORS

Variational regulators are composed with generative regulators to create a variation in the output shapes, by applying a formula to the shape attributes or regulator parameters.

Exception **ΞE** [$\{a, v\}$ $(\text{shape}_0 - \text{shape}_n)$]

The exception regulator allows a shape to be controlled differently from the rest of the output set. It gives this shape an exclusive formula.

Rhythm/Gradation **ΞG** [$\{a, \boldsymbol{f}, c\}$ $(\text{shape}_0 - \text{shape}_n)$]

The Rhythm regulator applies an additional formula to an attribute of the output set (or to an attribute of the regulator). The rhythm formula uses a coefficient, and ranges from simple to complex. Examples include $a_{new} = a_{old} + i \times c$ and $a_{new} = a_{old} + \sin(i \times c)$

Differential **ΞF** [$\{a, \boldsymbol{f}, c\}$ $(\text{shape}_0 - \text{shape}_n)$]

The Differential regulator an additional formula to the attribute of the regulator making vary across the different inputs. The differential formula uses a coefficient, and ranges from simple to complex. Examples include $a_{new} = a_{old} + i \times c$ and $a_{new} = a_{old} + \sin(i \times c)$

B.5. CONSTRAINT REGULATORS

Constraint regulators are based on an evaluation function or formula that determines whether or not the input element is within the geometric constraints. These are applied to input shapes.

Angle ΦL [$\{min, max, mod\}$ $(\text{shape}_1 - \text{shape}_k)$]

To derive the angle between two shapes, it is necessary to derive the angle between the direction-vectors of the defining regulators. Since these can be points, lines or planes, we use the following formulae to determine the angles between regulators.

The angle between two lines:

Dot product of the line vectors

Line1 : $l_1 = \bar{p}_1 + \bar{d}t$

Line1 : $l_2 = \bar{p}_2 + e\bar{v}$

$\bar{t} \bullet \bar{v} = |t||\bar{v}| \cos\theta$

$\theta = ar\cos(\bar{t} \bullet \bar{v} / |\bar{t}||\bar{v}|)$

The angle between two planes:

Dot product of the plane normals $\bar{m}$ and $\bar{n}$

$\bar{m} \bullet \bar{n} = |\bar{m}||\bar{n}| \cos\theta$

$\theta = \arccos(\bar{m} \bullet \bar{n} / |\bar{m}||\bar{n}|)$

The angle between a line and a plane:

The complementary angle to the line vector and the plane normal

line : $l = \bar{p} + d\bar{t}$

plane : $\bar{n}$

For complementary angles α and β

$\cos\beta = \cos(90 - \alpha) = \sin\alpha$

$\bar{n} \bullet \bar{t} = |\bar{n}||\bar{t}| \sin A$

$\alpha = \arcsin(\bar{n} \bullet \bar{t} / |\bar{n}||\bar{t}|)$

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

The proportion regulator controls a shape by means of its diagonal lines. The proportion can control volumetric shapes as well as planar ones. Although proportion regulator is intended for rectilinear shapes, it can also be used on non-rectilinear ones.

The diagonal is the diagonal vector, while the vectors defining the shape are the component vectors. For volumes there are additional surface diagonals that can be used for controlling the shape. The surface diagonals have two of the components of the volume diagonal. These are computed by means of the direction cosines.

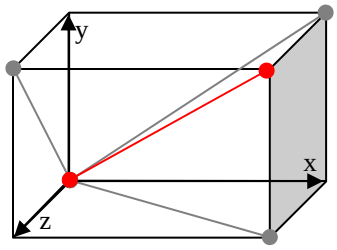
$$Dt_{xyz} = \overline{P}_0 + t_x + t_y + t_z$$

$$Dt_{xy} = \overline{P}_0 + t_x + t_y$$

$$Dt_{xz} = \overline{P}_0 + t_x + t_z$$

$$Dt_{yz} = \overline{P}_0 + t_y + t_z$$

$$Dt_{xyz} = \overline{P}_0 + Dt_{xy} + Dt_{yz} + Dt_{xz}$$



Equivalence $\Phi\mathbf{Q}$ [$\{a,v\}$ $(\text{shape}_0 - \text{shape}_n)$]

This is achieved by setting an equivalence relationship between a specific attribute of several shapes.

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

The dimension regulator restricts length, area and volume. The dimension regulator computes the dimensions of a shape based on the (n and d or θ) parameters of its defining regulators.

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

The boundary regulator controls the position of shapes within the allowable region of a boundary shape. It is based on the half space representation. The boundary regulators tests the shapes with the implicit function $f(x, y, z)$ of the boundary shape's defining regulators, in order to classify whether the bounded shapes are "in", "outside" or "on" the boundary shape.

$f(x, y, z) = 0 \Rightarrow$ on

$f(x, y, z) < 0 \Rightarrow$ in

$f(x, y, z) > 0 \Rightarrow$ out

Alignment $\Phi \mathbf{A}^0[\ \{\bar{p}\}\ (\text{shape}_0 - \text{shape}_k)\ ]$

$\Phi \mathbf{A}^1[\ \{\bar{p}, \bar{t}\}\ (\text{shape}_0 - \text{shape}_k)\ ]$

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

$\Phi \mathbf{A}^C[\ \{\bar{p}, \bar{t}, r\}\ (\text{shape}_0 - \text{shape}_k)\ ]$

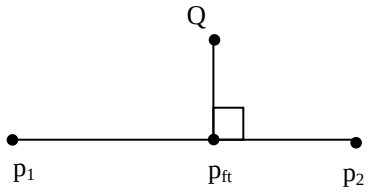
The alignment regulator restricts the position of shapes with respect to a point, line, plane or circle.

- If it's a point alignment, the xyz coordinates of the (starting point) of the shape is restricted to this point.
- If it's an orthogonal line alignment, for example parallel to the x-axis, y and z coordinates of the (starting point) of the shape is restricted, while the x coordinate is free. If it is an arbitrary line, it is necessary to determine the closest distance between the shape and the line. This is derived by the foot of the perpendicular between the initial shape and the line.
- If it's an orthogonal plane alignment, for example parallel to the x-plane, the z coordinates of the (starting point) of the shape is restricted, and the x and y coordinates are free. If it is an arbitrary plane, it is necessary to determine the closest distance between the shape and the plane. This is derived by the foot of the perpendicular between the initial shape and the plane.
- A circle alignment restricts elements by determining the closest distance between the

shape and the circle (or sphere), and computing the intersection point of circle (or sphere) with the line connecting the shape and the center of the circle or sphere.

The foot of perpendicular between a point and a line

$line\ \ l = (\overline{p}_1, \overline{p}_2)$



$\overline{p}_{ft} = \overline{p}_1 + d\overline{t}$

$\overline{p}_{ft} = \overline{p}_1 + d(\overline{p}_2 - \overline{p}_1)$

$(Q - \overline{p}_{ft}) \bullet (\overline{p}_2 - \overline{p}_1) = 0$

$(Q - (\overline{p}_1 + d(\overline{p}_2 - \overline{p}_1))) \bullet (\overline{p}_2 - \overline{p}_1) = 0$

The distance between Q and the line is the distance between Q and P2.

$$d = \frac{(Q_x - px_1)(px_2 - px_1) + (Qy - py_1)(py_2 - py_1) + (Q_z - pz_1)(pz_2 - pz_1)}{(px_2 - px_1)^2 + (py_2 - py_1)^2 + (pz_2 - pz_1)^2}$$

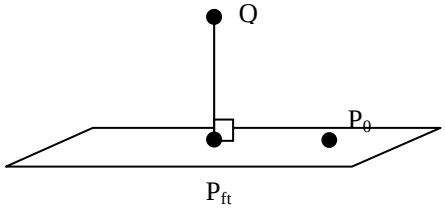
$px_{ft} = px_1 + d(px_2 - px_1)$
$py_{ft} = py_1 + d(py_2 - py_1)$
$pz_{ft} = pz_1 + d(pz_2 - pz_1)$

Foot of perpendicular between a point and a plane:

Plane defined by normal $N = \lfloor N_x, N_y, N_z \rfloor$

Point (away from the plane) $Q = \lfloor Q_x, Q_y, Q_z \rfloor$

Point (on the plane) $P_0 = \lfloor P_{0x}, P_{0y}, P_{0z} \rfloor$



The perpendicular projection of Q onto the plane is found by computing the perpendicular line (same direction as normal vector) that passes thought Q, then finding its intersection with the plane.

Perpendicular line that passes thought Q: $P_{ft} = Q + sN$

Line on the plane: $P_{ft} - P_0$

Perpendicular point: $N \bullet (P_{ft} - P_0) = 0$

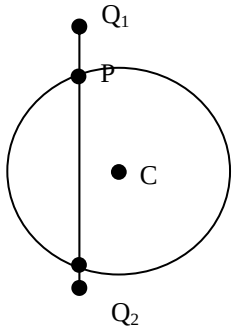
$$s = \frac{-N \bullet (Q - P_0)}{N \bullet N} = \frac{(N_x q_x + N_y q_y + N_z q_z)}{N_x{}^2 + N_y{}^2 + N_z{}^2}$$

$$P_{ft} = Q - \frac{(N_x q_x + N_y q_y + N_z q_z)}{N_x{}^2 + N_y{}^2 + N_z{}^2}|N|$$

Perpendicular distance $|Q - P_{ft}| = (Q - P_0)\cos\theta$

Intersection between a line and a circle

Line $P = Q + d(Q_2 - Q_1)$



$$Px = Q_1 x + d(Q_2 x - Q_1 x)$$
$$Py = Q_1 y + d(Q_2 y - Q_1 y)$$
$$Pz = Q_1 z + d(Q_2 z - Q_1 z)$$

Sphere centered at $C(Cx, Cy, Cz)$ with a radius r described by

$$(Px - Cx)^2 + (Py - Cy)^2 + (Pz - Cz)^2 = r^2$$

Substituting the equation of the line into the sphere gives a quadratic equation of the form

$$ad^2 + bd + c = 0$$

$$a = (Q_2 x - Q_1 x)^2 + (Q_2 y - Q_1 y)^2 + (Q_2 z - Q_1 z)^2$$
$$b = 2\left[(Q_2 x - Q_1 x)(Q_1 x - Cx) + (Q_2 y - Q_1 y)(Q_1 y - Cy) + (Q_2 z - Q_1 z)(Q_1 z - Cz)\right]$$
$$c = Cx^2 + Cy^2 + Cz^2 + Q_1 x^2 + Q_1 y^2 + Q_1 z^2 - 2\left[CxQ_1 x + CyQ_1 y + CzQ_1 z\right] - r^2$$

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

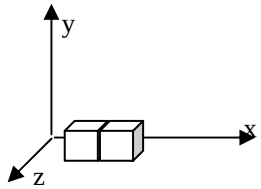
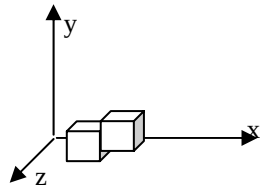
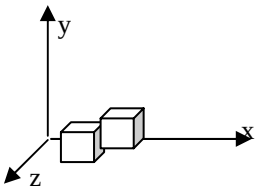
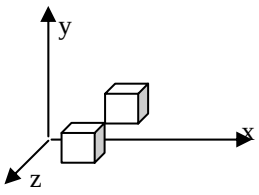
- if $b^2 - 4ac < 0$ then the line does not intersect the sphere
- if $b^2 - 4ac = 0$ then the line is tangent to the sphere
- if $b^2 - 4ac > 0$ then the line intersect the sphere in two places

TOPOLOGICAL REGULATORS

Topological regulators are also based on an evaluation function or formula that determines whether or not the input element is within the topological constraints.

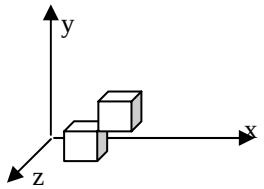
Distance $\Pi\mathbf{J}$ [$\{min, max, mod\}$ $(\text{shape}_1, \text{shape}_k)$]

Adjacency test: The distance is a binary relation that is determined by computing the position of each shape. The following shows the computation for adjacency along the x axis for rectilinear shapes. Non rectilinear shapes are tested thought their bounding volumes. The end point of the shape can be derived from the definition of the regulator.

Exactly face adjacent along x $s1.end.x = s2.start.x \wedge s1.start.x < s2.end.x$ $s1.start.y = s2.start.y \wedge s1.end.y = s2.end.y$ $s1.start.z = s2.start.z \wedge s1.end.z = s2.end.z$	
Face adjacent along x with shift along y $s1.end.x = s2.start.x \wedge s1.start.x < s2.end.x$ $s1.start.y >= s2.start.y \wedge s1.end.y < s2.end.y$ $s1.start.z = s2.start.z \wedge s1.end.z = s2.end.z$	
Face adjacent along x with shift along y and shift along z $s1.end.x = s2.start.x \wedge s1.start.x < s2.end.x$ $s1.start.y >= s2.start.y \wedge s1.end.y < s2.end.y$ $s1.start.z >= s2.start.z \wedge s1.end.z < s2.end.z$	
Line adjacent about x – y $s1.end.x = s2.start.x \wedge s1.start.x < s2.end.x$ $s1.end.y = s2.start.y \wedge s1.start.y < s2.end.y$ $s1.start.z = s2.start.z \wedge s1.end.z = s2.end.z$	

Point adjacent about x, y, z

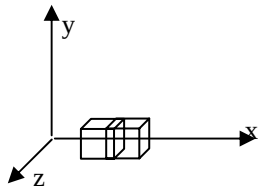
$s1.end.x = s2.start.x \wedge s1.start.x < s2.end.x$
$s1.end.y = s2.start.y \wedge s1.start.y < s2.end.y$
$s1.end.z = s2.start.z \wedge s1.start.z < s2.end.z$



Interlock Test: The interlock test determines whether the two shapes are interlocking or overlapping in 3D.

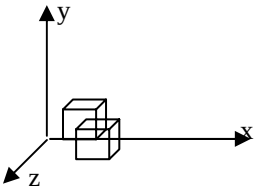
Interlock along the x

$s1.end.x > s2.start.x \wedge s1.start.x < s2.end.x$
$s1.start.y = s2.start.y \wedge s1.end.y = s2.end.y$
$s1.start.z = s2.start.z \wedge s1.end.z = s2.end.z$



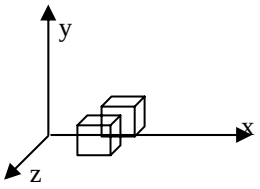
Interlock along the x and y

$s1.end.x > s2.start.x \wedge s1.start.x < s2.end.x$
$s1.start.y = s2.start.y \wedge s1.end.y = s2.end.y$
$s1.start.z = s2.start.z \wedge s1.end.z = s2.end.z$



Interlock along x, y, and z

$s1.end.x > s2.start.x \wedge s1.start.x < s2.end.x$
$s1.start.y > s2.start.y \wedge s1.end.y < s2.end.y$
$s1.start.z > s2.start.z \wedge s1.end.z < s2.end.z$



Connectedness $\Pi\mathbf{C}$ [{} ($shape_1$, $shape_k$)]

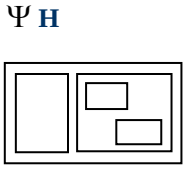
A shape is connected if and only if it is adjacent. The connectedness regulator ensures that connected shapes remain connected, by constraining their endpoints and freeing their other variables.

B.6. HIERARCHICAL REGULATORS

Hierarchical regulators define hierarchies of elements.

Containment

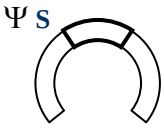
Ψ**H** [{} (container, constituent$_0$ – constituent$_n$)]



Defines a relationship between entities that is irrelevant of geometry. When it is composed with other constraint regulators, such as boundary of subdivision, it will have geometric implications.

Subshape

Ψ**S** [{} (supershape, subshape$_0$ – subshape$_n$)]



The subshape regulator ensures that the generative regulators of both shapes are equivalent and that the constraints of the supershape are maintained along the subshapes.

B.7. OPERATION REGULATORS

Operational regulators define shapes by means of discrete transformations. These are processing intersections of the inputs, in order to determine the output shapes.

B.7.1. INTERSECTION OF TWO LINES

Line $l_1 = (P_1, P_2)$; $Q_2 = P_1 + a(P_2 - P_1)$

Line $l_2 = (P_3, P_4)$; $Q_2 = P_3 + b(P_4 - P_3)$

$Q_1 = Q_2$ gives the intersection points with two unknowns

$$P_1 + a(P_2 - P_1) = P_3 + b(P_4 - P_3)$$

$$x_1 + a(x_2 - x_1) = x_3 + b(x_4 - x_3)$$

$$y_1 + a(y_2 - y_1) = y_3 + b(y_4 - y_3)$$

$$z_1 + a(z_2 - z_1) = z_3 + b(z_4 - z_3)$$

$$a = (x_4 - x_3)(y_1 - y_3) - (y_4 - y_3)(x_1 - x_3) / (y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)$$

$$b = (x_2 - x_1)(y_1 - y_3) - (y_2 - y_1)(x_1 - x_3) / (y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)$$

You can substitute either of these in their corresponding equations

$$x = x_1 + a(x_2 - x_1)$$

$$y = y_1 + a(y_2 - y_1)$$

$$z = z_1 + a(z_2 - z_1)$$

B.7.2. INTERSECTION OF TWO PLANES

$$N_1 \bullet p = d_1$$
$$N_2 \bullet p = d_2$$

The equation of the line of intersection is

$$p = c_1 N_1 + c_2 N_2 + e N_1 \otimes N_2$$

Substituting:

$$N_1 \bullet p = d_1 = c_1 N_1 \bullet N_1 + c_2 N_2 \bullet N_2$$
$$N_2 \bullet p = d_2 = c_1 N_1 \bullet N_1 + c_2 N_2 \bullet N_2$$

$$c_1 = (d_1 N_2 \bullet N_2 - d_2 N_1 \bullet N_2) / (N_1 \bullet N_1)(N_2 \bullet N_2) - (N_1 \bullet N_2)^2$$
$$c_2 = (d_2 N_1 \bullet N_1 - d_1 N_1 \bullet N_2) / (N_1 \bullet N_1)(N_2 \bullet N_2) - (N_1 \bullet N_2)^2$$

Note: also check if the planes are not parallel: if they are parallel then $N_1 \otimes N = 0$

B.7.3. INTERSECTION BETWEEN A LINE AND A PLANE

Line $Q_1 = p_1 + dt$

Plane $Q_2 = p_2 + av + bw$

Intersection is when $p_2 + av + bw = p_1 + dt$

$$d = \frac{(v \otimes w) \bullet p_2 - (v \otimes w) \bullet p_1}{(v \otimes w) \bullet t}$$

$$a = \frac{(w \otimes t) \bullet p_1 - (w \otimes t) \bullet p_2}{(w \otimes t) \bullet v}$$

$$b = \frac{(v \otimes t) \bullet p_1 - (v \otimes t) \bullet p_2}{(v \otimes t) \bullet w}$$

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

The subdivision regulator inputs a shape and generates many subshapes of this shape. The first version, $\Omega\mathbf{Z}$, operates by subdividing the generative regulators. It duplicates them and adjusts their parameters in order to determine their new position and sizes. It also acts as a super regulator that controls these sub regulators.

The second version, $\Omega\mathbf{Z}^{\mathbf{P}}$, subdivides the shape according to a cutting plane. The intersections are computed, and then the sub-regulators are generated and adjusted to produce the subshapes defined by this plane. Differential sweeping regulators are used to define slanted of curved planes.

Boolean operations

Union	**ΩU** [{} ($shape_0 - shape_k$)]
Intersection:	**ΩI** [{} ($shape_0 - shape_k$)]
Difference:	**ΩD** [{} ($shape_0 - shape_k$)]
Symmetric Difference:	**ΩM** [{} ($shape_0 - shape_k$)]

The Boolean regulators input several shapes and output the union, difference or intersection of these. The Boolean regulators rely on the half space representation and to determine whether the key intersection points are “in”, “on” and “out” of the input shapes.

$f(x, y, z) = 0$ on the plane or curve

$f(x, y, z) < 0$ in the plane or curves

$f(x, y, z) > 0$ out of the plane or curve

Union = MIN ($f(x, y, z)$, $g(x, y, z)$)

Intersection = MAX ($f(x, y, z)$, $g(x, y, z)$)

Difference = MAX ($f(x, y, z)$,- $g(x, y, z)$)

B.8. COMPOSITION OF REGULATORS

The internal mechanism for composition of regulators depends on the regulator types. Composition of transformation regulators is achieved by means of matrix multiplication. Composition of the other types of regulators is achieved by means of the sequence of the evaluation functions.

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

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

In case of composing regulators of various types $\Delta\mathbf{M}^{\mathbf{1}}\Phi\mathbf{A}^{\mathbf{1}}$ **(s)** or $\Delta\mathbf{R}^{\mathbf{1}}\Omega\mathbf{U}$**(s)** the transformations are computed, then the operations are computed, then the constraints are evaluated.

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

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \text{ and then } \begin{bmatrix} x'' \\ y'' \\ z'' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

B.9. REGULATING REGULATORS

The formula for the regulator is affected when the regulator is regulated by other regulators. This is also achieved by matrix multiplication. For example, if a mirror is rotated, the formula for the translation will incorporate the mirror factors.

$\Delta\mathbf{R}^{\mathbf{1}}[\ \{\bar{p}, \bar{t}, \theta, n\}\ (\Delta\mathbf{T}^{\mathbf{1}})^{<0><1><2><3>}] \wedge \Delta\mathbf{T}^{\mathbf{1}}[\ \{\bar{p}, \bar{t}, d, n\}\ \text{(shape)}]$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & \sin\phi & 0 & 0 \\ -\sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

APPENDIX C

PATTERN REPRESENTATION AND TRANSFORMATION

In this appendix, I illustrate the capacity of the ICE notation to **represent** all symmetry group patterns, and to transform each pattern to every other pattern with the Cyclic, Dihedral, Frieze and Wallpaper groups. In the following examples, the notation is abbreviated to show only relevant parameters; and since the generation method is always discrete, it is not depicted. Furthermore, for the purpose of brevity, the following regulators will be encapsulated in shorter notations.

Horizontal translation	$\Delta\mathbf{T}\ [\{1,0\}\ ()] = \Delta\mathbf{T_H}$
Vertical translation	$\Delta\mathbf{T}\ [\{0,1\}\ ()] = \Delta\mathbf{T_V}$
30º translation	$\Delta\mathbf{T}\ [\{\sqrt{3},1\}\ ()] = \Delta\mathbf{T_{30}}$
60º translation	$\Delta\mathbf{T}\ [\{1,\sqrt{3}\}\ ()] = \Delta\mathbf{T_{60}}$
45º translation	$\Delta\mathbf{T}\ [\{1,1\}\ ()] = \Delta\mathbf{T_{45}}$
-45º translation	$\Delta\mathbf{T}\ [\{1,-1\}\ ()] = \Delta\mathbf{T_{-45}}$
Horizontal Mirror	$\Delta\mathbf{M}\ [\{1,0\}\ ()] = \Delta\mathbf{M_H}$
Vertical Mirror:	$\Delta\mathbf{M}\ [\{0,1\}\ ()] = \Delta\mathbf{M_V}$
Horizontal Glide (translation + Mirror):	$\Delta\mathbf{T}\Delta\mathbf{M}\ [\{1,0\}\ ()] = \Delta\mathbf{TM_H}$
Vertical Glide (translation + Mirror):	$\Delta\mathbf{T}\Delta\mathbf{M}\ [\{0,1\}\ ()] = \Delta\mathbf{TM_V}$

C.1. CYCLIC AND DIHEDRAL PATTERNS

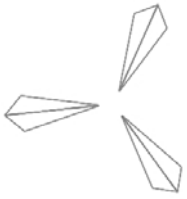
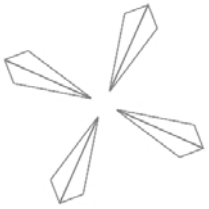
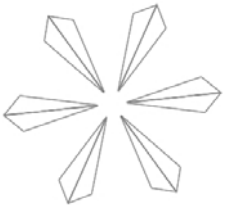
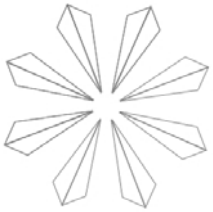
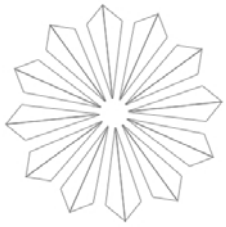
		
C(2)	**C(3)**	**C(4)**
ΔR [{*180*} (*s*)]	**ΔR** [{*120*} (*s*)]	**ΔR** [{*90*} (*s*)]
		
C(6)	**C(8)**	**C(12)**
ΔR [{*60*} (*s*)]	**ΔR** [{*45*} (*s*)]	**ΔR** [{*30*} (*s*)]
		
D(2)	**D(3)**	**D(4)**
ΔR [{*180*} (**ΔM** [{$\bar{t}$} (*s*)])]	**ΔR** [{*120*} (**ΔM** [{$\bar{t}$} (*s*)])]	**ΔR** [{*90*} (**ΔM** [{$\bar{t}$} (*s*)])]
		
D(6)	**D(8)**	**D(12)**
ΔR [{*60*} (**ΔM** [{$\bar{t}$} (*s*)])]	**ΔR** [{*45*} (**ΔM** [{$\bar{t}$} (*s*)])]	**ΔR** [{*30*} (**ΔM** [{$\bar{t}$} (*s*)])]

TABLE C.1 - CYCLIC AND DIHEDRAL PATTERNS

	 ΔR{*180*}	 ΔR{*120*}	 ΔR{*90*}	 ΔR{*60*}	 ΔR{*45*}	 ΔR{*30*}
 ΔR{*180*}		ΔR{*180*} ⇒ ΔR{*120*}	ΔR{*180*} ⇒ ΔR{*90*}	ΔR{*180*} ⇒ ΔR{*60*}	ΔR{*180*} ⇒ ΔR{*45*}	ΔR{*180*} ⇒ ΔR{*30*}
 ΔR{120}	ΔR{*120*} ⇒ ΔR{*180*}		ΔR{*120*} ⇒ ΔR{*90*}	ΔR{*120*} ⇒ ΔR{*60*}	ΔR{*120*} ⇒ ΔR{*45*}	ΔR{*120*} ⇒ ΔR{*30*}
 ΔR{*90*}	ΔR{*90*} ⇒ ΔR{*180*}	ΔR{*90*} ⇒ ΔR{*120*}		ΔR{*90*} ⇒ ΔR{*60*}	ΔR{*90*} ⇒ ΔR{*45*}	ΔR{*90*} ⇒ ΔR{*30*}
 ΔR{*60*}	ΔR{*60*} ⇒ ΔR{*180*}	ΔR{*60*} ⇒ ΔR{*120*}	ΔR{*60*} ⇒ ΔR{*90*}		ΔR{*60*} ⇒ ΔR{*45*}	ΔR{*60*} ⇒ ΔR{*30*}
 ΔR{*45*}	ΔR{*45*} ⇒ ΔR{*180*}	ΔR{*45*} ⇒ ΔR{*120*}	ΔR{*45*} ⇒ ΔR{*90*}	ΔR{*45*} ⇒ ΔR{*60*}		ΔR{*45*} ⇒ ΔR{*30*}
 ΔR{*30*}	ΔR{*30*} ⇒ ΔR{*180*}	ΔR{*30*} ⇒ ΔR{*120*}	ΔR{*30*} ⇒ ΔR{*90*}	ΔR{*30*} ⇒ ΔR{*60*}	ΔR{*30*} ⇒ ΔR{*45*}	
 ΔR{*180*}, ΔM	*DELETE* ΔM	*DELETE* ΔM ΔR{*180*} ⇒ ΔR{*120*}	*DELETE* ΔM ΔR{*180*} ⇒ ΔR{*90*}	*DELETE* ΔM ΔR{*180*} ⇒ ΔR{*60*}	*DELETE* ΔM ΔR{*180*} ⇒ ΔR{*45*}	*DELETE* ΔM ΔR{*180*} ⇒ ΔR{*30*}
 ΔR{*120*}, ΔM	*DELETE* ΔM ΔR{*120*} ⇒ ΔR{*180*}	*DELETE* ΔM	*DELETE* ΔM ΔR{*120*} ⇒ ΔR{*90*}	*DELETE* ΔM ΔR{*120*} ⇒ ΔR{*60*}	*DELETE* ΔM ΔR{*120*} ⇒ ΔR{*45*}	*DELETE* ΔM ΔR{*120*} ⇒ ΔR{*30*}
 ΔR{*90*}, ΔM	*DELETE* ΔM ΔR{*90*} ⇒ ΔR{*180*}	*DELETE* ΔM ΔR{*90*} ⇒ ΔR{*120*}	*DELETE* ΔM	*DELETE* ΔM ΔR{*90*} ⇒ ΔR{*60*}	*DELETE* ΔM ΔR{*90*} ⇒ ΔR{*45*}	*DELETE* ΔM ΔR{*90*} ⇒ ΔR{*30*}
 ΔR{*60*}, ΔM	*DELETE* ΔM ΔR{*60*} ⇒ ΔR{*180*}	*DELETE* ΔM ΔR{*60*} ⇒ ΔR{*120*}	*DELETE* ΔM ΔR{*60*} ⇒ ΔR{*90*}	*DELETE* ΔM	*DELETE* ΔM ΔR{*60*} ⇒ ΔR{*45*}	*DELETE* ΔM ΔR{*60*} ⇒ ΔR{*30*}
 ΔR{*45*}, ΔM	*DELETE* ΔM ΔR{*45*} ⇒ ΔR{*180*}	*DELETE* ΔM ΔR{*45*} ⇒ ΔR{*120*}	*DELETE* ΔM ΔR{*45*} ⇒ ΔR{*90*}	*DELETE* ΔM ΔR{*45*} ⇒ ΔR{*60*}	*DELETE* ΔM	*DELETE* ΔM ΔR{*45*} ⇒ ΔR{*30*}
 ΔR{*30*}, ΔM	*DELETE* ΔM ΔR{*30*} ⇒ ΔR{*180*}	*DELETE* ΔM ΔR{*30*} ⇒ ΔR{*120*}	*DELETE* ΔM ΔR{*30*} ⇒ ΔR{*90*}	*DELETE* ΔM ΔR{*30*} ⇒ ΔR{*60*}	*DELETE* ΔM ΔR{*30*} ⇒ ΔR{*45*}	*DELETE* ΔM

TABLE C.2 - TRANSFORMATION ACROSS CYCLIC AND DIHEDRAL PATTERNS

	 ΔR{180}, ΔM	 ΔR{120}, ΔM	 ΔR{90}, ΔM	 ΔR{60}, ΔM	 ΔR{45}, ΔM	 ΔR{30}, ΔM
 ΔR{180}	*INSERT* ΔM	*INSERT* ΔM ΔR{180} ⇒ ΔR{120}	*INSERT* ΔM ΔR{180} ⇒ ΔR{90}	*INSERT* ΔM ΔR{180} ⇒ ΔR{60}	*INSERT* ΔM ΔR{180} ⇒ ΔR{45}	*INSERT* ΔM ΔR{180} ⇒ ΔR{30}
 ΔR{120}	*INSERT* ΔM ΔR{120} ⇒ ΔR{180}	*INSERT* ΔM	*INSERT* ΔM ΔR{120} ⇒ ΔR{90}	*INSERT* ΔM ΔR{120} ⇒ ΔR{60}	*INSERT* ΔM ΔR{120} ⇒ ΔR{45}	*INSERT* ΔM ΔR{120} ⇒ ΔR{30}
 ΔR{90}	*INSERT* ΔM ΔR{90} ⇒ ΔR{180}	*INSERT* ΔM ΔR{90} ⇒ ΔR{120}	*INSERT* ΔM	*INSERT* ΔM ΔR{90} ⇒ ΔR{60}	*INSERT* ΔM ΔR{90} ⇒ ΔR{45}	*INSERT* ΔM ΔR{90} ⇒ ΔR{30}
 ΔR{60}	*INSERT* ΔM ΔR{60} ⇒ ΔR{180}	*INSERT* ΔM ΔR{60} ⇒ ΔR{120}	*INSERT* ΔM ΔR{60} ⇒ ΔR{90}	*INSERT* ΔM	*INSERT* ΔM ΔR{60} ⇒ ΔR{45}	*INSERT* ΔM ΔR{60} ⇒ ΔR{30}
 ΔR{45}	*INSERT* ΔM ΔR{45} ⇒ ΔR{180}	*INSERT* ΔM ΔR{45} ⇒ ΔR{120}	*INSERT* ΔM ΔR{45} ⇒ ΔR{90}	*INSERT* ΔM ΔR{45} ⇒ ΔR{60}	*INSERT* ΔM	*INSERT* ΔM ΔR{45} ⇒ ΔR{30}
 ΔR{30}	*INSERT* ΔM ΔR{30} ⇒ ΔR{180}	*INSERT* ΔM ΔR{30} ⇒ ΔR{120}	*INSERT* ΔM ΔR{30} ⇒ ΔR{90}	*INSERT* ΔM ΔR{30} ⇒ ΔR{60}	*INSERT* ΔM ΔR{30} ⇒ ΔR{45}	*INSERT* ΔM
 ΔR{180}, ΔM		ΔR{180} ⇒ ΔR{120}	ΔR{180} ⇒ ΔR{90}	ΔR{180} ⇒ ΔR{60}	ΔR{180} ⇒ ΔR{45}	ΔR{180} ⇒ ΔR{30}
 ΔR{120}, ΔM	ΔR{120} ⇒ ΔR{180}		ΔR{120} ⇒ ΔR{90}	ΔR{120} ⇒ ΔR{60}	ΔR{120} ⇒ ΔR{45}	ΔR{120} ⇒ ΔR{30}
 ΔR{90}, ΔM	ΔR{90} ⇒ ΔR{180}	ΔR{90} ⇒ ΔR{120}		ΔR{90} ⇒ ΔR{60}	ΔR{90} ⇒ ΔR{45}	ΔR{90} ⇒ ΔR{30}
 ΔR{60}, ΔM	ΔR{60} ⇒ ΔR{180}	ΔR{60} ⇒ ΔR{120}	ΔR{60} ⇒ ΔR{90}		ΔR{60} ⇒ ΔR{45}	ΔR{60} ⇒ ΔR{30}
 ΔR{45}, ΔM	ΔR{45} ⇒ ΔR{180}	ΔR{45} ⇒ ΔR{120}	ΔR{45} ⇒ ΔR{90}	ΔR{45} ⇒ ΔR{60}		ΔR{45} ⇒ ΔR{30}
 ΔR{30}, ΔM	ΔR{45} ⇒ ΔR{180}	ΔR{30} ⇒ ΔR{120}	ΔR{30} ⇒ ΔR{90}	ΔR{30} ⇒ ΔR{60}	ΔR{30} ⇒ ΔR{45}	

TABLE C.3 - TRANSFORMATION ACROSS CYCLIC AND DIHEDRAL PATTERNS

C.2. FRIEZE PATTERNS

p111

ΔT [{*1,0*} (*s*)]

p1m1

ΔT [{*1,0*} (ΔM [{*1,0*} (*s*)])]

pm11

ΔT [{*1,0*} (ΔM [{*0,1*} (*s*)])]

pmm2

ΔT [{1,0} (ΔM [{1,0} (

ΔM [{(0,1} (*s*)])])]

p112

ΔT [{*1,0*} (ΔR [{*180*} (*s*)])]

p1a1

ΔTΔM [{*1,0*} (*s*)]

pma2

ΔTΔM [{*1,0*} (ΔM [{*0,1*} (*s*)])]

TABLE C.4 - FRIEZE PATTERNS

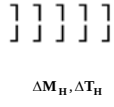
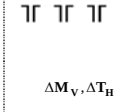
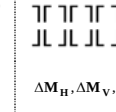
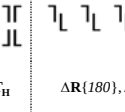
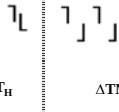
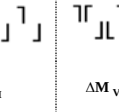
	 $\Delta\mathbf{T_H}$	 $\Delta\mathbf{M_H}, \Delta\mathbf{T_H}$	 $\Delta\mathbf{M_V}, \Delta\mathbf{T_H}$	 $\Delta\mathbf{M_H}, \Delta\mathbf{M_V}, \Delta\mathbf{T_H}$	 $\Delta\mathbf{R}\{180\}, \Delta\mathbf{T_H}$	 $\Delta\mathbf{TM_H}$	 $\Delta\mathbf{M_V}, \Delta\mathbf{TM_H}$
$\Delta\mathbf{T_H}$		*INSERT* $\Delta\mathbf{M_H}$	*INSERT* $\Delta\mathbf{M_V}$	*INSERT* $\Delta\mathbf{M_V}$ *INSERT* $\Delta\mathbf{M_H}$	*INSERT* $\Delta\mathbf{R}\{180\}$	*COMPOSE* $\Delta\mathbf{M_H}$	*INSERT* $\Delta\mathbf{M_V}$ *COMPOSE* $\Delta\mathbf{M_H}$
$\Delta\mathbf{M_H}, \Delta\mathbf{T_H}$	*DELETE* $\Delta\mathbf{M_H}$		$\Delta\mathbf{M_H} \Rightarrow \Delta\mathbf{M_V}$	*INSERT* $\Delta\mathbf{M_V}$	*REPLACE* $\Delta\mathbf{M_H} \Rightarrow \Delta\mathbf{R}\{180\}$	*DELETE* $\Delta\mathbf{M_H}$ *COMPOSE* $\Delta\mathbf{M_H}$	$\Delta\mathbf{M_H} \Rightarrow \Delta\mathbf{M_V}$ *COMPOSE* $\Delta\mathbf{M_H}$
$\Delta\mathbf{M_V}, \Delta\mathbf{T_H}$	*DELETE* $\Delta\mathbf{M_V}$	$\Delta\mathbf{M_V} \Rightarrow \Delta\mathbf{M_H}$		*INSERT* $\Delta\mathbf{M_H}$	*REPLACE* $\Delta\mathbf{M_V} \Rightarrow \Delta\mathbf{R}\{180\}$	*DELETE* $\Delta\mathbf{M_V}$ *COMPOSE* $\Delta\mathbf{M_H}$	*COMPOSE* $\Delta\mathbf{M_H}$
$\Delta\mathbf{M_H}, \Delta\mathbf{M_V}, \Delta\mathbf{T_H}$	*DELETE* $\Delta\mathbf{M_V}$ *DELETE* $\Delta\mathbf{M_H}$	*DELETE* $\Delta\mathbf{M_H}$	*DELETE* $\Delta\mathbf{M_H}$		*DELETE* $\Delta\mathbf{M_H}$ *REPLACE* $\Delta\mathbf{M_V} \Rightarrow \Delta\mathbf{R}\{180\}$	*DELETE* $\Delta\mathbf{M_V}$ *DELETE* $\Delta\mathbf{M_H}$ *COMPOSE* $\Delta\mathbf{M_H}$	*DELETE* $\Delta\mathbf{M_H}$ *COMPOSE* $\Delta\mathbf{M_H}$
$\Delta\mathbf{R}\{180\}, \Delta\mathbf{T_H}$	*DELETE* $\Delta\mathbf{R}\{180\}$	*REPLACE* $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{M_H}$	*REPLACE* $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{M_V}$	*INSERT* $\Delta\mathbf{M_H}$ *REPLACE* $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{M_V}$		*DELETE* $\Delta\mathbf{R}\{180\}$ *COMPOSE* $\Delta\mathbf{M_H}$	*REPLACE* $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{M_V}$ *COMPOSE* $\Delta\mathbf{M_H}$
$\Delta\mathbf{TM_H}$	*REMOVE* $\Delta\mathbf{M_H}$	*INSERT* $\Delta\mathbf{M_H}$ *REMOVE* $\Delta\mathbf{M_H}$	*INSERT* $\Delta\mathbf{M_V}$ *REMOVE* $\Delta\mathbf{M_H}$	*INSERT* $\Delta\mathbf{M_H}$ *INSERT* $\Delta\mathbf{M_V}$ *REMOVE* $\Delta\mathbf{M_H}$	*INSERT* $\Delta\mathbf{R}\{180\}$ *REMOVE* $\Delta\mathbf{M_H}$		*INSERT* $\Delta\mathbf{M_V}$
$\Delta\mathbf{M_V}, \Delta\mathbf{TM_H}$	*DELETE* $\Delta\mathbf{M_V}$ *REMOVE* $\Delta\mathbf{M_H}$	$\Delta\mathbf{M_V} \Rightarrow \Delta\mathbf{M_H}$ *REMOVE* $\Delta\mathbf{M_H}$	*REMOVE* $\Delta\mathbf{M_H}$	*INSERT* $\Delta\mathbf{M_H}$ *REMOVE* $\Delta\mathbf{M_H}$	*REPLACE* $\Delta\mathbf{M_V} \Rightarrow \Delta\mathbf{R}\{180\}$ *REMOVE* $\Delta\mathbf{M_H}$	*DELETE* $\Delta\mathbf{M_V}$	

TABLE C.5 - TRANSFORMATION ACROSS FRIEZE PATTERNS

C.3. WALLPAPER PATTERNS

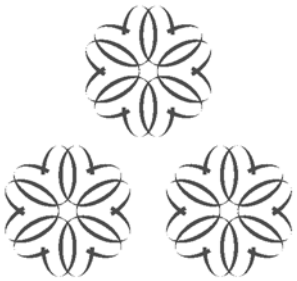
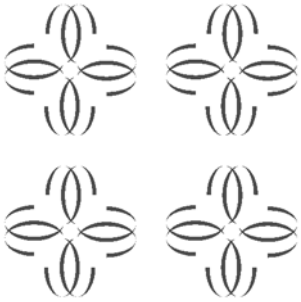
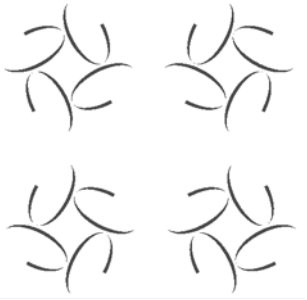
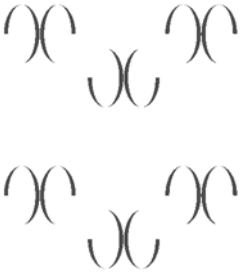
 **p6** $\Delta\mathbf{T}$ [{*1*, $\sqrt{3}$} ($\Delta\mathbf{T}$ [{*1,0*} ($\Delta\mathbf{R}$ [{*60*} (*s*)])])]	 **p6mm** $\Delta\mathbf{T}$ [{*1*, $\sqrt{3}$} ($\Delta\mathbf{T}$ [{*1,0*} ($\Delta\mathbf{R}$ [{*60*} ($\Delta\mathbf{M}$ [{*0,1*} (*s*)])])]	
 **p3** $\Delta\mathbf{T}$ [{*1*, $\sqrt{3}$} ($\Delta\mathbf{T}$ [{*1,0*} ($\Delta\mathbf{R}$ [{*120*} (*s*)])])]	 **p3m1** $\Delta\mathbf{T}$ [{*0,1*} ($\Delta\mathbf{T}$ [{$\sqrt{3}$,*1*} ($\Delta\mathbf{R}$ [{*120*} ($\Delta\mathbf{M}$ [{*0,1*} (*s*)])])]	 **p31m** $\Delta\mathbf{T}$ [{*1*, $\sqrt{3}$} ($\Delta\mathbf{T}$ [{*1,0*} ($\Delta\mathbf{R}$ [{*120*} ($\Delta\mathbf{M}$ [{*0,1*} (*s*)])])]
 **p4** $\Delta\mathbf{T}$ [{*0,1*} ($\Delta\mathbf{T}$ [{*1,0*} ($\Delta\mathbf{R}$ [{*90*} (*s*)])])]	 **p4mm** $\Delta\mathbf{T}$ [{*0,1*} ($\Delta\mathbf{T}$ [{*1,0*} ($\Delta\mathbf{R}$ [{*90*} ($\Delta\mathbf{M}$ [{*0,1*} (*s*)])])]	 **p4g** $\Delta\mathbf{TM}$ [{*0,1*} ($\Delta\mathbf{TM}$ [{*1,0*} ($\Delta\mathbf{R}$ [{*90*} (*s*)])])]

TABLE C.6 - WALLPAPER PATTERNS

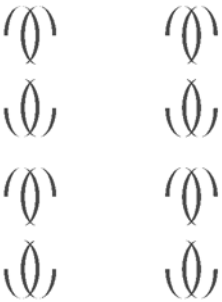
p2

ΔT [{*0,1*} (ΔT [{*1,0*} (
ΔR [{*180*} (*s*)]
)])]



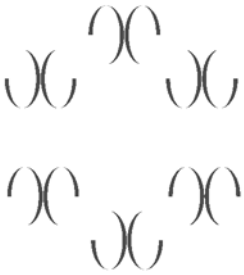
cmm

ΔT [{*0,1*} (ΔT [{*1,0*} (
ΔR [{*180*} (ΔM [{*0,1*} (*s*)])]
)])]



pmm

ΔT [{*0,1*} (ΔT [{*1,0*} (
ΔM [{*1,0*} (ΔM [{*0,1*} (s)])]
)])]



pmg

ΔTM [{*0,1*} (ΔT [{*1,0*} (
ΔR [{*180*} (ΔM [{*0,1*} (*s*)])]
)])]

pgg

ΔTM [{0,1} (ΔTM [{1,0}
)])]

pg

ΔTM [{0,1} (ΔT [{1,0}
)])]

p1

ΔT [{*0,1*} (ΔT [{*1,0*}
)])]

pm

ΔT [{*0,1*} (ΔT [{*1,0*} (
ΔM [{*0,1*} (*s*)]
)])]

cm

ΔT [{*1,–1*} (ΔT [{*1,1*} (
ΔM [{*0,1*} (*s*)]
)])]

TABLE C.7 - WALLPAPER PATTERNS

	 p6 $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	 p6m $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	 p3 $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	 p3m1 $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_{30}}, \Delta\mathbf{T_V}$	 p31m $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	
 p6 $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$		*INSERT* $\Delta\mathbf{M_V}$	$\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{120\}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{30}}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{120\}$	
 p6m $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	*DELETE* $\Delta\mathbf{M_V}$		*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{120\}$	$\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{30}}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	$\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{120\}$	
 p3 $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	$\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{60\}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{60\}$		*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{30}}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*INSERT* $\Delta\mathbf{M_V}$	
 p3m1 $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_{30}}, \Delta\mathbf{T_V}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_{30}} \Rightarrow \Delta\mathbf{T_H}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	$\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_{30}} \Rightarrow \Delta\mathbf{T_H}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{T_{30}} \Rightarrow \Delta\mathbf{T_H}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$		$\Delta\mathbf{T_{30}} \Rightarrow \Delta\mathbf{T_H}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	
 p31m $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{60\}$	$\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{60\}$	*DELETE* $\Delta\mathbf{M_V}$	$\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{30}}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$		

TABLE C.8 - TRANSFORMATIONS ACROSS WALLPAPER PATTERNS

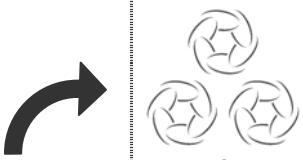
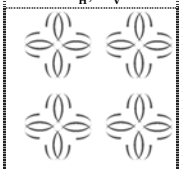
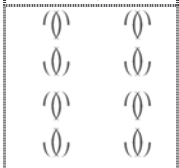
	 p6 $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	 p6m $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	 p3 $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	 p3m1 $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_{30}}, \Delta\mathbf{T_V}$	 p31m $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	
 p4 $\Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	$\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	$\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{30}}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	
 p4m $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	$\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	$\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{30}}$	$\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	
 p4g $\Delta\mathbf{R}\{90\}$ $\Delta\mathbf{TM_H}, \Delta\mathbf{TM_V}$	$\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$ *REMOVE* $\Delta\mathbf{M_H}$ *REMOVE* $\Delta\mathbf{M_V}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$ *REMOVE* $\Delta\mathbf{M_H}$ *REMOVE* $\Delta\mathbf{M_V}$	$\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$ *REMOVE* $\Delta\mathbf{M_H}$ *REMOVE* $\Delta\mathbf{M_V}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{30}}$ *REMOVE* $\Delta\mathbf{M_H}$ *REMOVE* $\Delta\mathbf{M_V}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$ *REMOVE* $\Delta\mathbf{M_H}$ *REMOVE* $\Delta\mathbf{M_V}$	
 p2 $\Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	$\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	$\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{30}}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	
 cmm $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	$\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	$\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{30}}$	$\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	
 pmm $\Delta\mathbf{M_V}, \Delta\mathbf{M_H}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	*DELETE* $\Delta\mathbf{M_V}$ *REPLACE* $\Delta\mathbf{M_H} \Rightarrow \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	*REPLACE* $\Delta\mathbf{M_H} \Rightarrow \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	*DELETE* $\Delta\mathbf{M_V}$ *REPLACE* $\Delta\mathbf{M_H} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	*REPLACE* $\Delta\mathbf{M_H} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{30}}$	*REPLACE* $\Delta\mathbf{M_H} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	

TABLE C.9 - TRANSFORMATIONS ACROSS WALLPAPER PATTERNS

	p6 $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	p6m $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	p3 $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	p3m1 $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_{30}}, \Delta\mathbf{T_V}$	p31m $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	
p1 $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	*INSERT* $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	*INSERT* $\Delta\mathbf{R}\{60\}$ *INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	*INSERT* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	*INSERT* $\Delta\mathbf{R}\{120\}$ *INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{30}}$	*INSERT* $\Delta\mathbf{R}\{120\}$ *INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	
pm $\Delta\mathbf{M_V}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	*DELETE* $\Delta\mathbf{M_V}$ *INSERT* $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	*INSERT* $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	*DELETE* $\Delta\mathbf{M_V}$ *INSERT* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	*INSERT* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{30}}$	*INSERT* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$	
cm $\Delta\mathbf{M_V}$ $\Delta\mathbf{T_{45}}, \Delta\mathbf{T_{-45}}$	*DELETE* $\Delta\mathbf{M_V}$ *INSERT* $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_{45}} \Rightarrow \Delta\mathbf{T_H}$ $\Delta\mathbf{T_{-45}} \Rightarrow \Delta\mathbf{T_{60}}$	*INSERT* $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_{45}} \Rightarrow \Delta\mathbf{T_H}$ $\Delta\mathbf{T_{-45}} \Rightarrow \Delta\mathbf{T_{60}}$	*DELETE* $\Delta\mathbf{M_V}$ *INSERT* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_{45}} \Rightarrow \Delta\mathbf{T_H}$ $\Delta\mathbf{T_{-45}} \Rightarrow \Delta\mathbf{T_{60}}$	*INSERT* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_{45}} \Rightarrow \Delta\mathbf{T_{30}}$ $\Delta\mathbf{T_{-45}} \Rightarrow \Delta\mathbf{T_V}$	*INSERT* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_{45}} \Rightarrow \Delta\mathbf{T_H}$ $\Delta\mathbf{T_{-45}} \Rightarrow \Delta\mathbf{T_{60}}$	
pmg $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{TM_V}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$ *REMOVE* $\Delta\mathbf{M_V}$	$\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$ *REMOVE* $\Delta\mathbf{M_V}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$ *REMOVE* $\Delta\mathbf{M_V}$	$\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{30}}$ *REMOVE* $\Delta\mathbf{M_V}$	$\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$ *REMOVE* $\Delta\mathbf{M_V}$	
pgg $\Delta\mathbf{R}\{180\}$ $\Delta\mathbf{TM_H}, \Delta\mathbf{TM_V}$	$\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$ *REMOVE* $\Delta\mathbf{M_V}$ *REMOVE* $\Delta\mathbf{M_H}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$ *REMOVE* $\Delta\mathbf{M_V}$ *REMOVE* $\Delta\mathbf{M_H}$	$\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$ *REMOVE* $\Delta\mathbf{M_V}$ *REMOVE* $\Delta\mathbf{M_H}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{30}}$ *REMOVE* $\Delta\mathbf{M_V}$ *REMOVE* $\Delta\mathbf{M_H}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$ *REMOVE* $\Delta\mathbf{M_V}$ *REMOVE* $\Delta\mathbf{M_H}$	
pg $\Delta\mathbf{T_H}, \Delta\mathbf{TM_V}$	*INSERT* $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$ *REMOVE* $\Delta\mathbf{M_V}$	*INSERT* $\Delta\mathbf{M_V}$ *INSERT* $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$ *REMOVE* $\Delta\mathbf{M_V}$	*INSERT* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$ *REMOVE* $\Delta\mathbf{M_V}$	*INSERT* $\Delta\mathbf{M_V}$ *INSERT* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{30}}$ *REMOVE* $\Delta\mathbf{M_V}$	*INSERT* $\Delta\mathbf{M_V}$ *INSERT* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{60}}$ *REMOVE* $\Delta\mathbf{M_V}$	

TABLE C.10 - TRANSFORMATIONS ACROSS WALLPAPER PATTERNS

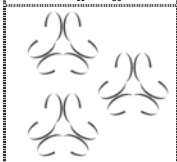
	p4 $\Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	p4m $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	p4g $\Delta\mathbf{R}\{90\}$ $\Delta\mathbf{TM_H}, \Delta\mathbf{TM_V}$	p2 $\Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	cmm $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	pmm $\Delta\mathbf{M_V}, \Delta\mathbf{M_H}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$
 p6 $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	$\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	$\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$ *COMPOSE* $\Delta\mathbf{M_V}$ *COMPOSE* $\Delta\mathbf{M_H}$	$\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*INSERT* $\Delta\mathbf{M_V}$ *REPLACE* $\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{M_H}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$
 p6m $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	$\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$ *COMPOSE* $\Delta\mathbf{M_V}$ *COMPOSE* $\Delta\mathbf{M_H}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	$\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*REPLACE* $\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{M_H}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$
 p3 $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	$\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	$\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$ *COMPOSE* $\Delta\mathbf{M_V}$ *COMPOSE* $\Delta\mathbf{M_H}$	$\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*INSERT* $\Delta\mathbf{M_V}$ *REPLACE* $\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{M_H}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$
 p3m1 $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_{30}}, \Delta\mathbf{T_V}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_{30}} \Rightarrow \Delta\mathbf{T_H}$	$\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_{30}} \Rightarrow \Delta\mathbf{T_H}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_{30}} \Rightarrow \Delta\mathbf{T_H}$ *COMPOSE* $\Delta\mathbf{M_V}$ *COMPOSE* $\Delta\mathbf{M_H}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{30}} \Rightarrow \Delta\mathbf{T_H}$	$\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{30}} \Rightarrow \Delta\mathbf{T_H}$	*REPLACE* $\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{M_H}$ $\Delta\mathbf{T_{30}} \Rightarrow \Delta\mathbf{T_H}$
 p31m $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	$\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$ *COMPOSE* $\Delta\mathbf{M_V}$ *COMPOSE* $\Delta\mathbf{M_H}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	$\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*REPLACE* $\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{M_H}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$

TABLE C.11 - TRANSFORMATIONS ACROSS WALLPAPER PATTERNS

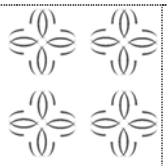
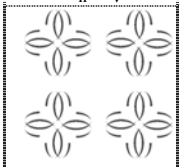
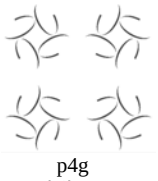
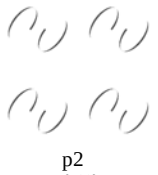
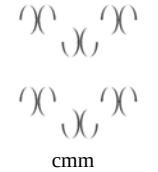
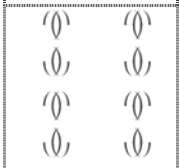
	 p4 $\Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T}_\mathbf{H}, \Delta\mathbf{T}_\mathbf{V}$	 p4m $\Delta\mathbf{M}_\mathbf{V}, \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T}_\mathbf{H}, \Delta\mathbf{T}_\mathbf{V}$	 p4g $\Delta\mathbf{R}\{90\}$ $\Delta\mathbf{TM}_\mathbf{H}, \Delta\mathbf{TM}_\mathbf{V}$	 p2 $\Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T}_\mathbf{H}, \Delta\mathbf{T}_\mathbf{V}$	 cmm $\Delta\mathbf{M}_\mathbf{V}, \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T}_\mathbf{H}, \Delta\mathbf{T}_\mathbf{V}$	 pmm $\Delta\mathbf{M}_\mathbf{V}, \Delta\mathbf{M}_\mathbf{H}$ $\Delta\mathbf{T}_\mathbf{H}, \Delta\mathbf{T}_\mathbf{V}$
 p4 $\Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T}_\mathbf{H}, \Delta\mathbf{T}_\mathbf{V}$		*INSERT* $\Delta\mathbf{M}_\mathbf{V}$	*COMPOSE* $\Delta\mathbf{M}_\mathbf{V}$ *COMPOSE* $\Delta\mathbf{M}_\mathbf{H}$	$\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{180\}$	*INSERT* $\Delta\mathbf{M}_\mathbf{V}$ $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{180\}$	*INSERT* $\Delta\mathbf{M}_\mathbf{V}$ *REPLACE* $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{M}_\mathbf{H}$
 p4m $\Delta\mathbf{M}_\mathbf{V}, \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T}_\mathbf{H}, \Delta\mathbf{T}_\mathbf{V}$	*DELETE* $\Delta\mathbf{M}_\mathbf{V}$		*DELETE* $\Delta\mathbf{M}_\mathbf{V}$ *COMPOSE* $\Delta\mathbf{M}_\mathbf{V}$ *COMPOSE* $\Delta\mathbf{M}_\mathbf{H}$	*DELETE* $\Delta\mathbf{M}_\mathbf{V}$ $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{180\}$	$\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{180\}$	*REPLACE* $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{M}_\mathbf{H}$
 p4g $\Delta\mathbf{R}\{90\}$ $\Delta\mathbf{TM}_\mathbf{H}, \Delta\mathbf{TM}_\mathbf{V}$	*REMOVE* $\Delta\mathbf{M}_\mathbf{V}$ *REMOVE* $\Delta\mathbf{M}_\mathbf{H}$	*INSERT* $\Delta\mathbf{M}_\mathbf{V}$ *REMOVE* $\Delta\mathbf{M}_\mathbf{V}$ *REMOVE* $\Delta\mathbf{M}_\mathbf{H}$		$\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{180\}$ *REMOVE* $\Delta\mathbf{M}_\mathbf{V}$ *REMOVE* $\Delta\mathbf{M}_\mathbf{H}$	*INSERT* $\Delta\mathbf{M}_\mathbf{V}$ $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{180\}$ *REMOVE* $\Delta\mathbf{M}_\mathbf{V}$ *REMOVE* $\Delta\mathbf{M}_\mathbf{H}$	*INSERT* $\Delta\mathbf{M}_\mathbf{V}$ *REPLACE* $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{M}_\mathbf{H}$ *REMOVE* $\Delta\mathbf{M}_\mathbf{V}$ *REMOVE* $\Delta\mathbf{M}_\mathbf{H}$
 p2 $\Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T}_\mathbf{H}, \Delta\mathbf{T}_\mathbf{V}$	$\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{90\}$	*INSERT* $\Delta\mathbf{M}_\mathbf{V}$ $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{90\}$	$\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{90\}$ *COMPOSE* $\Delta\mathbf{M}_\mathbf{V}$ *COMPOSE* $\Delta\mathbf{M}_\mathbf{H}$		*INSERT* $\Delta\mathbf{M}_\mathbf{V}$	*INSERT* $\Delta\mathbf{M}_\mathbf{V}$ *REPLACE* $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{M}_\mathbf{H}$
 cmm $\Delta\mathbf{M}_\mathbf{V}, \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T}_\mathbf{H}, \Delta\mathbf{T}_\mathbf{V}$	*DELETE* $\Delta\mathbf{M}_\mathbf{V}$ $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{90\}$	$\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{90\}$	*DELETE* $\Delta\mathbf{M}_\mathbf{V}$ $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{90\}$ *COMPOSE* $\Delta\mathbf{M}_\mathbf{V}$ *COMPOSE* $\Delta\mathbf{M}_\mathbf{H}$	*DELETE* $\Delta\mathbf{M}_\mathbf{V}$		*REPLACE* $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{M}_\mathbf{H}$
 pmm $\Delta\mathbf{M}_\mathbf{V}, \Delta\mathbf{M}_\mathbf{H}$ $\Delta\mathbf{T}_\mathbf{H}, \Delta\mathbf{T}_\mathbf{V}$	*DELETE* $\Delta\mathbf{M}_\mathbf{V}$ *REPLACE* $\Delta\mathbf{M}_\mathbf{H} \Rightarrow \Delta\mathbf{R}\{90\}$	*REPLACE* $\Delta\mathbf{M}_\mathbf{H} \Rightarrow \Delta\mathbf{R}\{90\}$	*DELETE* $\Delta\mathbf{M}_\mathbf{V}$ *REPLACE* $\Delta\mathbf{M}_\mathbf{H} \Rightarrow \Delta\mathbf{R}\{90\}$ *COMPOSE* $\Delta\mathbf{M}_\mathbf{V}$ *COMPOSE* $\Delta\mathbf{M}_\mathbf{H}$	*DELETE* $\Delta\mathbf{M}_\mathbf{V}$ *REPLACE* $\Delta\mathbf{M}_\mathbf{H} \Rightarrow \Delta\mathbf{R}\{180\}$	*REPLACE* $\Delta\mathbf{M}_\mathbf{H} \Rightarrow \Delta\mathbf{R}\{180\}$	

TABLE C.12 - TRANSFORMATIONS ACROSS WALLPAPER PATTERNS

	p4 $\Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T}_H, \Delta\mathbf{T}_V$	p4m $\Delta\mathbf{M}_V, \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T}_H, \Delta\mathbf{T}_V$	p4g $\Delta\mathbf{R}\{90\}$ $\Delta\mathbf{TM}_H, \Delta\mathbf{TM}_V$	p2 $\Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T}_H, \Delta\mathbf{T}_V$	cmm $\Delta\mathbf{M}_V, \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T}_H, \Delta\mathbf{T}_V$	pmm $\Delta\mathbf{M}_V, \Delta\mathbf{M}_H$ $\Delta\mathbf{T}_H, \Delta\mathbf{T}_V$
p1 $\Delta\mathbf{T}_H, \Delta\mathbf{T}_V$	*INSERT* $\Delta\mathbf{R}\{90\}$	*INSERT* $\Delta\mathbf{M}_V$ *INSERT* $\Delta\mathbf{R}\{90\}$	*INSERT* $\Delta\mathbf{R}\{90\}$ *COMPOSE* $\Delta\mathbf{M}_V$ *COMPOSE* $\Delta\mathbf{M}_H$	*INSERT* $\Delta\mathbf{R}\{180\}$	*INSERT* $\Delta\mathbf{M}_V$ *INSERT* $\Delta\mathbf{R}\{180\}$	*INSERT* $\Delta\mathbf{M}_V$ *INSERT* $\Delta\mathbf{M}_H$
pm $\Delta\mathbf{M}_V$ $\Delta\mathbf{T}_H, \Delta\mathbf{T}_V$	*REPLACE* $\Delta\mathbf{M}_V \Rightarrow \Delta\mathbf{R}\{90\}$	*INSERT* $\Delta\mathbf{R}\{90\}$	*REPLACE* $\Delta\mathbf{M}_V \Rightarrow \Delta\mathbf{R}\{90\}$ *COMPOSE* $\Delta\mathbf{M}_V$ *COMPOSE* $\Delta\mathbf{M}_H$	*REPLACE* $\Delta\mathbf{M}_V \Rightarrow \Delta\mathbf{R}\{180\}$	*INSERT* $\Delta\mathbf{R}\{180\}$	*INSERT* $\Delta\mathbf{M}_H$
cm $\Delta\mathbf{M}_V$ $\Delta\mathbf{T}_{45}, \Delta\mathbf{T}_{-45}$	*REPLACE* $\Delta\mathbf{M}_V \Rightarrow \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T}_{45} \Rightarrow \Delta\mathbf{T}_H$ $\Delta\mathbf{T}_{-45} \Rightarrow \Delta\mathbf{T}_V$	*INSERT* $\Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T}_{45} \Rightarrow \Delta\mathbf{T}_H$ $\Delta\mathbf{T}_{-45} \Rightarrow \Delta\mathbf{T}_V$	*REPLACE* $\Delta\mathbf{M}_V \Rightarrow \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T}_{45} \Rightarrow \Delta\mathbf{T}_H$ $\Delta\mathbf{T}_{-45} \Rightarrow \Delta\mathbf{T}_V$ *COMPOSE* $\Delta\mathbf{M}_V$ *COMPOSE* $\Delta\mathbf{M}_H$	*REPLACE* $\Delta\mathbf{M}_V \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T}_{45} \Rightarrow \Delta\mathbf{T}_H$ $\Delta\mathbf{T}_{-45} \Rightarrow \Delta\mathbf{T}_V$	*INSERT* $\Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T}_{45} \Rightarrow \Delta\mathbf{T}_H$ $\Delta\mathbf{T}_{-45} \Rightarrow \Delta\mathbf{T}_V$	*INSERT* $\Delta\mathbf{M}_H$ $\Delta\mathbf{T}_{45} \Rightarrow \Delta\mathbf{T}_H$ $\Delta\mathbf{T}_{-45} \Rightarrow \Delta\mathbf{T}_V$
pmg $\Delta\mathbf{M}_V, \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T}_H, \Delta\mathbf{TM}_V$	*DELETE* $\Delta\mathbf{M}_V$ $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{90\}$ *REMOVE* $\Delta\mathbf{M}_V$	$\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{90\}$ *REMOVE* $\Delta\mathbf{M}_V$	*DELETE* $\Delta\mathbf{M}_V$ $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{90\}$ *COMPOSE* $\Delta\mathbf{M}_H$	*DELETE* $\Delta\mathbf{M}_V$ *REMOVE* $\Delta\mathbf{M}_V$	*REMOVE* $\Delta\mathbf{M}_V$	*REPLACE* $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{M}_H$ *REMOVE* $\Delta\mathbf{M}_V$
pgg $\Delta\mathbf{R}\{180\}$ $\Delta\mathbf{TM}_H, \Delta\mathbf{TM}_V$	$\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{90\}$ *REMOVE* $\Delta\mathbf{M}_V$ *REMOVE* $\Delta\mathbf{M}_H$	*INSERT* $\Delta\mathbf{M}_V$ $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{90\}$ *REMOVE* $\Delta\mathbf{M}_V$ *REMOVE* $\Delta\mathbf{M}_H$	$\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{R}\{90\}$	*REMOVE* $\Delta\mathbf{M}_V$ *REMOVE* $\Delta\mathbf{M}_H$	*INSERT* $\Delta\mathbf{M}_V$ *REMOVE* $\Delta\mathbf{M}_V$ *REMOVE* $\Delta\mathbf{M}_H$	*INSERT* $\Delta\mathbf{M}_V$ *REPLACE* $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{M}_H$ *REMOVE* $\Delta\mathbf{M}_V$ *REMOVE* $\Delta\mathbf{M}_H$
pg $\Delta\mathbf{T}_H, \Delta\mathbf{TM}_V$	*INSERT* $\Delta\mathbf{R}\{90\}$ *COMPOSE* $\Delta\mathbf{M}_V$	*INSERT* $\Delta\mathbf{M}_V$ *INSERT* $\Delta\mathbf{R}\{90\}$ *REMOVE* $\Delta\mathbf{M}_V$	*INSERT* $\Delta\mathbf{R}\{90\}$ *COMPOSE* $\Delta\mathbf{M}_H$	*INSERT* $\Delta\mathbf{R}\{180\}$ *REMOVE* $\Delta\mathbf{M}_V$	*INSERT* $\Delta\mathbf{M}_V$ *INSERT* $\Delta\mathbf{R}\{180\}$ *REMOVE* $\Delta\mathbf{M}_V$	*INSERT* $\Delta\mathbf{M}_V$ *INSERT* $\Delta\mathbf{M}_H$ *REMOVE* $\Delta\mathbf{M}_V$

TABLE C.13 -TRANSFORMATIONS ACROSS WALLPAPER PATTERNS

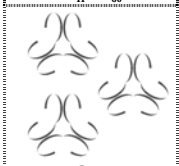
	p1 $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	pm $\Delta\mathbf{M_V}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	cm $\Delta\mathbf{M_V}$ $\Delta\mathbf{T_{45}}, \Delta\mathbf{T_{-45}}$	pmg $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{TM_V}$	pgg $\Delta\mathbf{R}\{180\}$ $\Delta\mathbf{TM_H}, \Delta\mathbf{TM_V}$	pg $\Delta\mathbf{T_H}, \Delta\mathbf{TM_V}$
 p6 $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	*DELETE* $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*INSERT* $\Delta\mathbf{M_V}$ *DELETE* $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*INSERT* $\Delta\mathbf{M_V}$ *DELETE* $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{45}}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_{-45}}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$ *COMPOSE* $\Delta\mathbf{M_V}$	$\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$ *COMPOSE* $\Delta\mathbf{M_H}$ *COMPOSE* $\Delta\mathbf{M_V}$	*DELETE* $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$ *COMPOSE* $\Delta\mathbf{M_V}$
 p6m $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	*DELETE* $\Delta\mathbf{M_V}$ *DELETE* $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*DELETE* $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*DELETE* $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{45}}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_{-45}}$	$\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$ *COMPOSE* $\Delta\mathbf{M_V}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{60\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$ *COMPOSE* $\Delta\mathbf{M_H}$ *COMPOSE* $\Delta\mathbf{M_V}$	*DELETE* $\Delta\mathbf{M_V}$ *DELETE* $\Delta\mathbf{R}\{60\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$ *COMPOSE* $\Delta\mathbf{M_V}$
 p3 $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	*DELETE* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*INSERT* $\Delta\mathbf{M_V}$ *DELETE* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*INSERT* $\Delta\mathbf{M_V}$ *DELETE* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{45}}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_{-45}}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$ *COMPOSE* $\Delta\mathbf{M_V}$	$\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$ *COMPOSE* $\Delta\mathbf{M_H}$ *COMPOSE* $\Delta\mathbf{M_V}$	*DELETE* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$ *COMPOSE* $\Delta\mathbf{M_V}$
 p3m1 $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_{30}}, \Delta\mathbf{T_V}$	*DELETE* $\Delta\mathbf{M_V}$ *DELETE* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_{30}} \Rightarrow \Delta\mathbf{T_H}$	*DELETE* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_{30}} \Rightarrow \Delta\mathbf{T_H}$	*DELETE* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{45}}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_{-45}}$	$\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{30}} \Rightarrow \Delta\mathbf{T_H}$ *COMPOSE* $\Delta\mathbf{M_V}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{30}} \Rightarrow \Delta\mathbf{T_H}$ *COMPOSE* $\Delta\mathbf{M_H}$ *COMPOSE* $\Delta\mathbf{M_V}$	*DELETE* $\Delta\mathbf{M_V}$ *DELETE* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_{30}} \Rightarrow \Delta\mathbf{T_H}$ *COMPOSE* $\Delta\mathbf{M_V}$
 p31m $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_{60}}$	*DELETE* $\Delta\mathbf{M_V}$ *DELETE* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*DELETE* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$	*DELETE* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{45}}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_{-45}}$	$\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$ *COMPOSE* $\Delta\mathbf{M_V}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{120\} \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$ *COMPOSE* $\Delta\mathbf{M_H}$ *COMPOSE* $\Delta\mathbf{M_V}$	*DELETE* $\Delta\mathbf{M_V}$ *DELETE* $\Delta\mathbf{R}\{120\}$ $\Delta\mathbf{T_{60}} \Rightarrow \Delta\mathbf{T_V}$ *COMPOSE* $\Delta\mathbf{M_V}$

TABLE C.14 -TRANSFORMATIONS ACROSS WALLPAPER PATTERNS

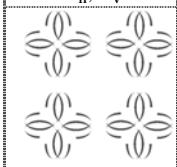
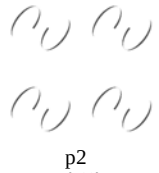
	p1 $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	pm $\Delta\mathbf{M_V}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	cm $\Delta\mathbf{M_V}$ $\Delta\mathbf{T_{45}}, \Delta\mathbf{T_{-45}}$	pmg $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{TM_V}$	pgg $\Delta\mathbf{R}\{180\}$ $\Delta\mathbf{TM_H}, \Delta\mathbf{TM_V}$	pg $\Delta\mathbf{T_H}, \Delta\mathbf{TM_V}$
 p4 $\Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	*DELETE* $\Delta\mathbf{R}\{90\}$	*REPLACE* $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{M_V}$	*REPLACE* $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{M_V}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{45}}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{-45}}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{180\}$ *COMPOSE* $\Delta\mathbf{M_V}$	$\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{180\}$ *COMPOSE* $\Delta\mathbf{M_H}$ *COMPOSE* $\Delta\mathbf{M_V}$	*DELETE* $\Delta\mathbf{R}\{90\}$ *COMPOSE* $\Delta\mathbf{M_V}$
 p4m $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	*DELETE* $\Delta\mathbf{R}\{90\}$ *DELETE* $\Delta\mathbf{M_V}$	*DELETE* $\Delta\mathbf{R}\{90\}$	*DELETE* $\Delta\mathbf{R}\{90\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{45}}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{-45}}$	$\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{180\}$ *COMPOSE* $\Delta\mathbf{M_V}$	*DELETE* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{180\}$ *COMPOSE* $\Delta\mathbf{M_H}$ *COMPOSE* $\Delta\mathbf{M_V}$	*DELETE* $\Delta\mathbf{M_V}$ *DELETE* $\Delta\mathbf{R}\{90\}$ *COMPOSE* $\Delta\mathbf{M_V}$
 p4g $\Delta\mathbf{R}\{90\}$ $\Delta\mathbf{TM_H}, \Delta\mathbf{TM_V}$	*DELETE* $\Delta\mathbf{R}\{90\}$ *REMOVE* $\Delta\mathbf{M_H}$ *REMOVE* $\Delta\mathbf{M_V}$	*REPLACE* $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{M_V}$ *REMOVE* $\Delta\mathbf{M_H}$ *REMOVE* $\Delta\mathbf{M_V}$	*REPLACE* $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{M_V}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{45}}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{-45}}$ *REMOVE* $\Delta\mathbf{M_H}$ *REMOVE* $\Delta\mathbf{M_V}$	*INSERT* $\Delta\mathbf{M_V}$ $\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{180\}$ *REMOVE* $\Delta\mathbf{M_H}$	$\Delta\mathbf{R}\{90\} \Rightarrow \Delta\mathbf{R}\{180\}$	*DELETE* $\Delta\mathbf{R}\{90\}$ *REMOVE* $\Delta\mathbf{M_H}$
 p2 $\Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	*DELETE* $\Delta\mathbf{R}\{180\}$	*REPLACE* $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{M_V}$	*REPLACE* $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{M_V}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{45}}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{-45}}$	*INSERT* $\Delta\mathbf{M_V}$ *COMPOSE* $\Delta\mathbf{M_V}$	*COMPOSE* $\Delta\mathbf{M_H}$ *COMPOSE* $\Delta\mathbf{M_V}$	*DELETE* $\Delta\mathbf{R}\{180\}$ *COMPOSE* $\Delta\mathbf{M_V}$
 cmm $\Delta\mathbf{M_V}, \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	*DELETE* $\Delta\mathbf{M_V}$ *DELETE* $\Delta\mathbf{R}\{180\}$	*DELETE* $\Delta\mathbf{R}\{180\}$	*DELETE* $\Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{45}}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{-45}}$	*COMPOSE* $\Delta\mathbf{M_V}$	*DELETE* $\Delta\mathbf{M_V}$ *COMPOSE* $\Delta\mathbf{M_H}$ *COMPOSE* $\Delta\mathbf{M_V}$	*DELETE* $\Delta\mathbf{M_V}$ *DELETE* $\Delta\mathbf{R}\{180\}$ *COMPOSE* $\Delta\mathbf{M_V}$
 pmm $\Delta\mathbf{M_V}, \Delta\mathbf{M_H}$ $\Delta\mathbf{T_H}, \Delta\mathbf{T_V}$	*DELETE* $\Delta\mathbf{M_V}$ *DELETE* $\Delta\mathbf{M_H}$	*DELETE* $\Delta\mathbf{M_H}$	*DELETE* $\Delta\mathbf{M_H}$ $\Delta\mathbf{T_H} \Rightarrow \Delta\mathbf{T_{45}}$ $\Delta\mathbf{T_V} \Rightarrow \Delta\mathbf{T_{-45}}$	*REPLACE* $\Delta\mathbf{M_H} \Rightarrow \Delta\mathbf{R}\{180\}$ *COMPOSE* $\Delta\mathbf{M_V}$	*DELETE* $\Delta\mathbf{M_V}$ *REPLACE* $\Delta\mathbf{M_H} \Rightarrow \Delta\mathbf{R}\{180\}$ *COMPOSE* $\Delta\mathbf{M_H}$ *COMPOSE* $\Delta\mathbf{M_V}$	*DELETE* $\Delta\mathbf{M_V}$ *DELETE* $\Delta\mathbf{M_H}$ *COMPOSE* $\Delta\mathbf{M_V}$

TABLE C.15 - TRANSFORMATIONS ACROSS WALLPAPER PATTERNS

	p1 $\Delta\mathbf{T}_H, \Delta\mathbf{T}_V$	pm $\Delta\mathbf{M}_V$ $\Delta\mathbf{T}_H, \Delta\mathbf{T}_V$	cm $\Delta\mathbf{M}_V$ $\Delta\mathbf{T}_{45}, \Delta\mathbf{T}_{-45}$	pmg $\Delta\mathbf{M}_V, \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T}_H, \Delta\mathbf{TM}_V$	pgg $\Delta\mathbf{R}\{180\}$ $\Delta\mathbf{TM}_H, \Delta\mathbf{TM}_V$	pg $\Delta\mathbf{T}_H, \Delta\mathbf{TM}_V$
p1 $\Delta\mathbf{T}_H, \Delta\mathbf{T}_V$		*INSERT* $\Delta\mathbf{M}_V$	*INSERT* $\Delta\mathbf{M}_V$ $\Delta\mathbf{T}_H \Rightarrow \Delta\mathbf{T}_{45}$ $\Delta\mathbf{T}_V \Rightarrow \Delta\mathbf{T}_{-45}$	*INSERT* $\Delta\mathbf{M}_V$ *INSERT* $\Delta\mathbf{R}\{180\}$ *COMPOSE* $\Delta\mathbf{M}_V$	*INSERT* $\Delta\mathbf{R}\{180\}$ *COMPOSE* $\Delta\mathbf{M}_V$ *COMPOSE* $\Delta\mathbf{M}_H$	*COMPOSE* $\Delta\mathbf{M}_V$
pm $\Delta\mathbf{M}_V$ $\Delta\mathbf{T}_H, \Delta\mathbf{T}_V$	*DELETE* $\Delta\mathbf{M}_V$		$\Delta\mathbf{T}_H \Rightarrow \Delta\mathbf{T}_{45}$ $\Delta\mathbf{T}_V \Rightarrow \Delta\mathbf{T}_{-45}$	*INSERT* $\Delta\mathbf{R}\{180\}$ *COMPOSE* $\Delta\mathbf{M}_V$	*REPLACE* $\Delta\mathbf{M}_V \Rightarrow \Delta\mathbf{R}\{180\}$ *COMPOSE* $\Delta\mathbf{M}_V$ *COMPOSE* $\Delta\mathbf{M}_H$	*DELETE* $\Delta\mathbf{M}_V$ *COMPOSE* $\Delta\mathbf{M}_V$
cm $\Delta\mathbf{M}_V$ $\Delta\mathbf{T}_{45}, \Delta\mathbf{T}_{-45}$	*DELETE* $\Delta\mathbf{M}_V$ $\Delta\mathbf{T}_{45} \Rightarrow \Delta\mathbf{T}_H$ $\Delta\mathbf{T}_{-45} \Rightarrow \Delta\mathbf{T}_V$	$\Delta\mathbf{T}_{45} \Rightarrow \Delta\mathbf{T}_H$ $\Delta\mathbf{T}_{-45} \Rightarrow \Delta\mathbf{T}_V$		*INSERT* $\Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T}_{45} \Rightarrow \Delta\mathbf{T}_H$ $\Delta\mathbf{T}_{-45} \Rightarrow \Delta\mathbf{T}_V$ *COMPOSE* $\Delta\mathbf{M}_V$	*REPLACE* $\Delta\mathbf{M}_V \Rightarrow \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T}_{45} \Rightarrow \Delta\mathbf{T}_H$ $\Delta\mathbf{T}_{-45} \Rightarrow \Delta\mathbf{T}_V$ *COMPOSE* $\Delta\mathbf{M}_V$ *COMPOSE* $\Delta\mathbf{M}_H$	*DELETE* $\Delta\mathbf{M}_V$ $\Delta\mathbf{T}_{45} \Rightarrow \Delta\mathbf{T}_H$ $\Delta\mathbf{T}_{-45} \Rightarrow \Delta\mathbf{T}_V$ *COMPOSE* $\Delta\mathbf{M}_V$
pmg $\Delta\mathbf{M}_V, \Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T}_H, \Delta\mathbf{TM}_V$	*DELETE* $\Delta\mathbf{M}_V$ *DELETE* $\Delta\mathbf{R}\{180\}$ *REMOVE* $\Delta\mathbf{M}_V$	*DELETE* $\Delta\mathbf{R}\{180\}$ *REMOVE* $\Delta\mathbf{M}_V$	*DELETE* $\Delta\mathbf{R}\{180\}$ $\Delta\mathbf{T}_H \Rightarrow \Delta\mathbf{T}_{45}$ $\Delta\mathbf{T}_V \Rightarrow \Delta\mathbf{T}_{-45}$ *REMOVE* $\Delta\mathbf{M}_V$		*DELETE* $\Delta\mathbf{M}_V$ *COMPOSE* $\Delta\mathbf{M}_H$	*DELETE* $\Delta\mathbf{M}_V$ *DELETE* $\Delta\mathbf{R}\{180\}$
pgg $\Delta\mathbf{R}\{180\}$ $\Delta\mathbf{TM}_H, \Delta\mathbf{TM}_V$	*DELETE* $\Delta\mathbf{R}\{180\}$ *REMOVE* $\Delta\mathbf{M}_H$ *REMOVE* $\Delta\mathbf{M}_V$	*REPLACE* $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{M}_V$ *REMOVE* $\Delta\mathbf{M}_H$ *REMOVE* $\Delta\mathbf{M}_V$	*REPLACE* $\Delta\mathbf{R}\{180\} \Rightarrow \Delta\mathbf{M}_V$ $\Delta\mathbf{T}_H \Rightarrow \Delta\mathbf{T}_{45}$ $\Delta\mathbf{T}_V \Rightarrow \Delta\mathbf{T}_{-45}$ *REMOVE* $\Delta\mathbf{M}_H$ *REMOVE* $\Delta\mathbf{M}_V$	*INSERT* $\Delta\mathbf{M}_V$ *REMOVE* $\Delta\mathbf{M}_H$		*DELETE* $\Delta\mathbf{R}\{180\}$ *REMOVE* $\Delta\mathbf{M}_H$
pg $\Delta\mathbf{T}_H, \Delta\mathbf{TM}_V$	*REMOVE* $\Delta\mathbf{M}_V$	*INSERT* $\Delta\mathbf{M}_V$ *REMOVE* $\Delta\mathbf{M}_V$	*INSERT* $\Delta\mathbf{M}_V$ $\Delta\mathbf{T}_H \Rightarrow \Delta\mathbf{T}_{45}$ $\Delta\mathbf{T}_V \Rightarrow \Delta\mathbf{T}_{-45}$ *REMOVE* $\Delta\mathbf{M}_V$	*INSERT* $\Delta\mathbf{M}_V$ *INSERT* $\Delta\mathbf{R}\{180\}$	*INSERT* $\Delta\mathbf{R}\{180\}$ *COMPOSE* $\Delta\mathbf{M}_H$	

TABLE C.16 -TRANSFORMATIONS ACROSS WALLPAPER PATTERNS

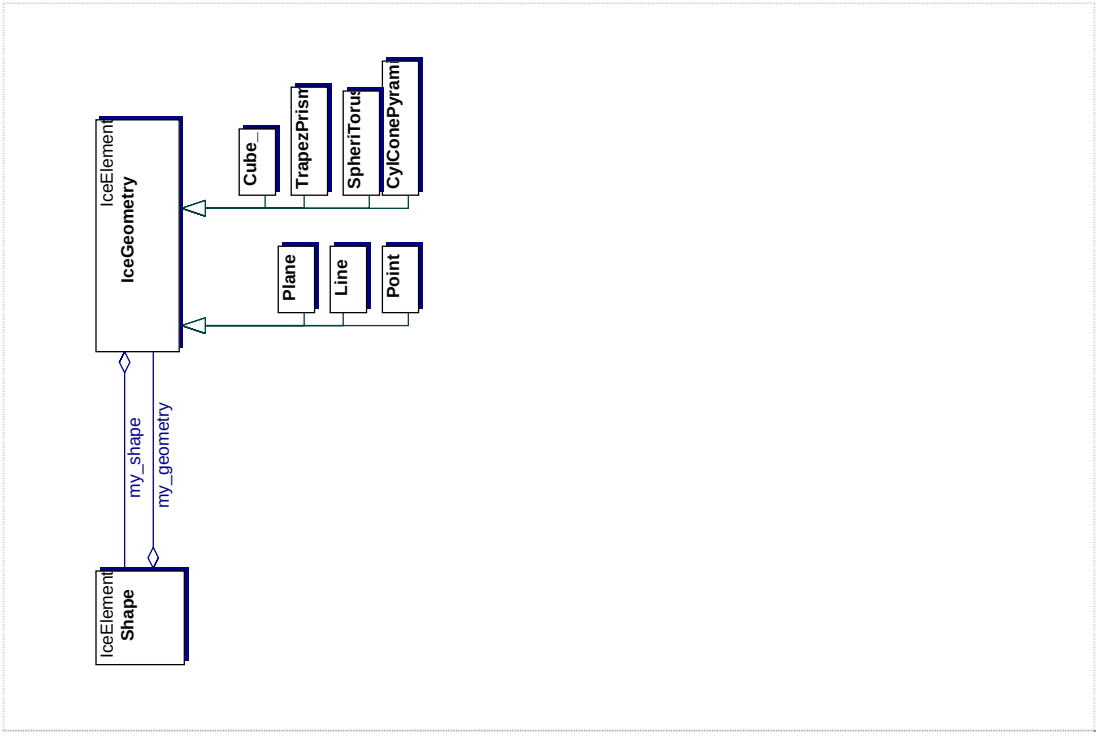
APPENDIX D

THE ENGINEERING OF THE ICE IMPLEMENTATION

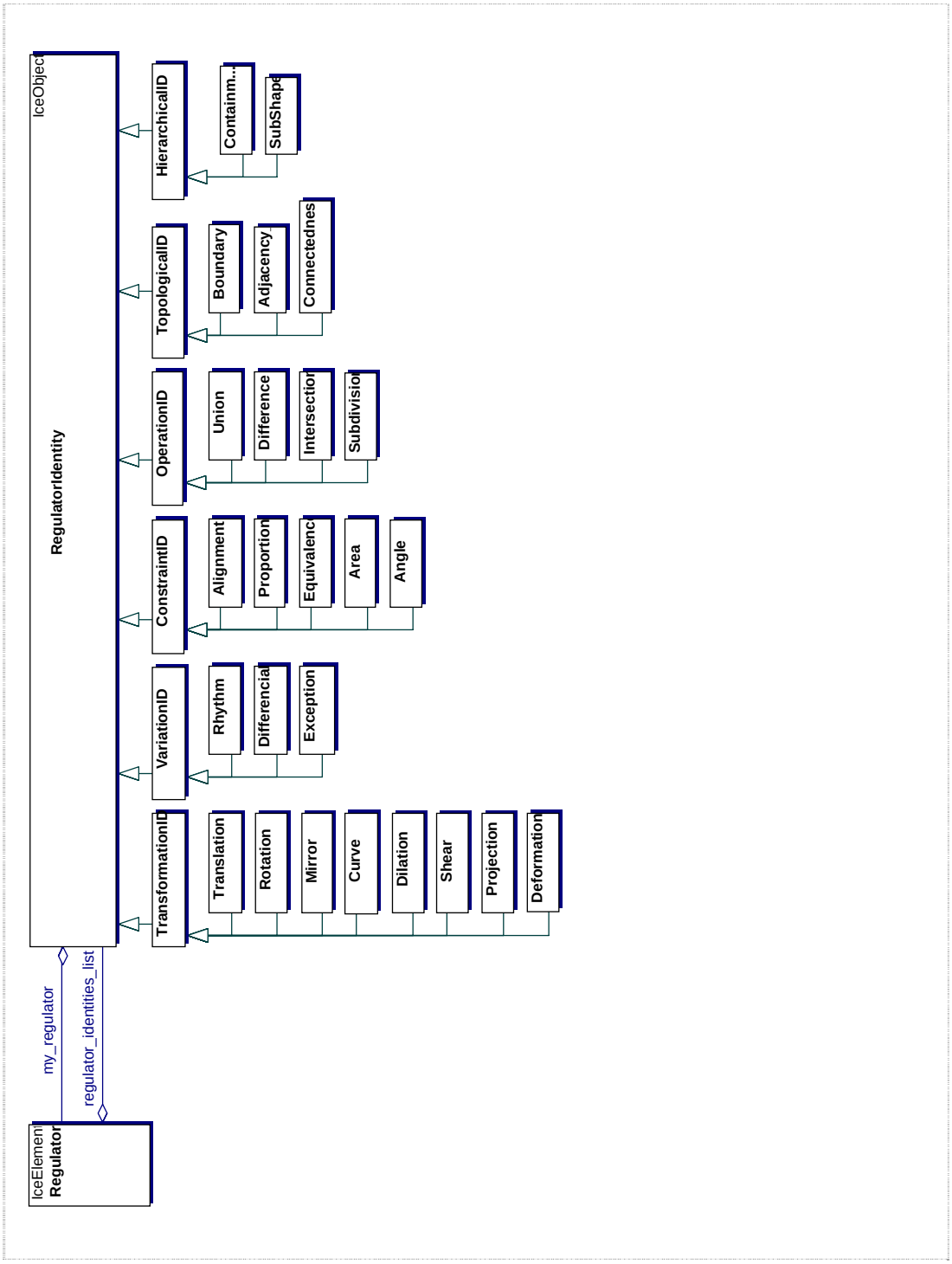
This appendix illustrates the UML diagrams used to engineer the ICE implementation. Section D.1 shows the object models, and section D.2 shows use cases and interaction diagrams for the various functionalities of regulators.

D.1. OBJECT MODELS

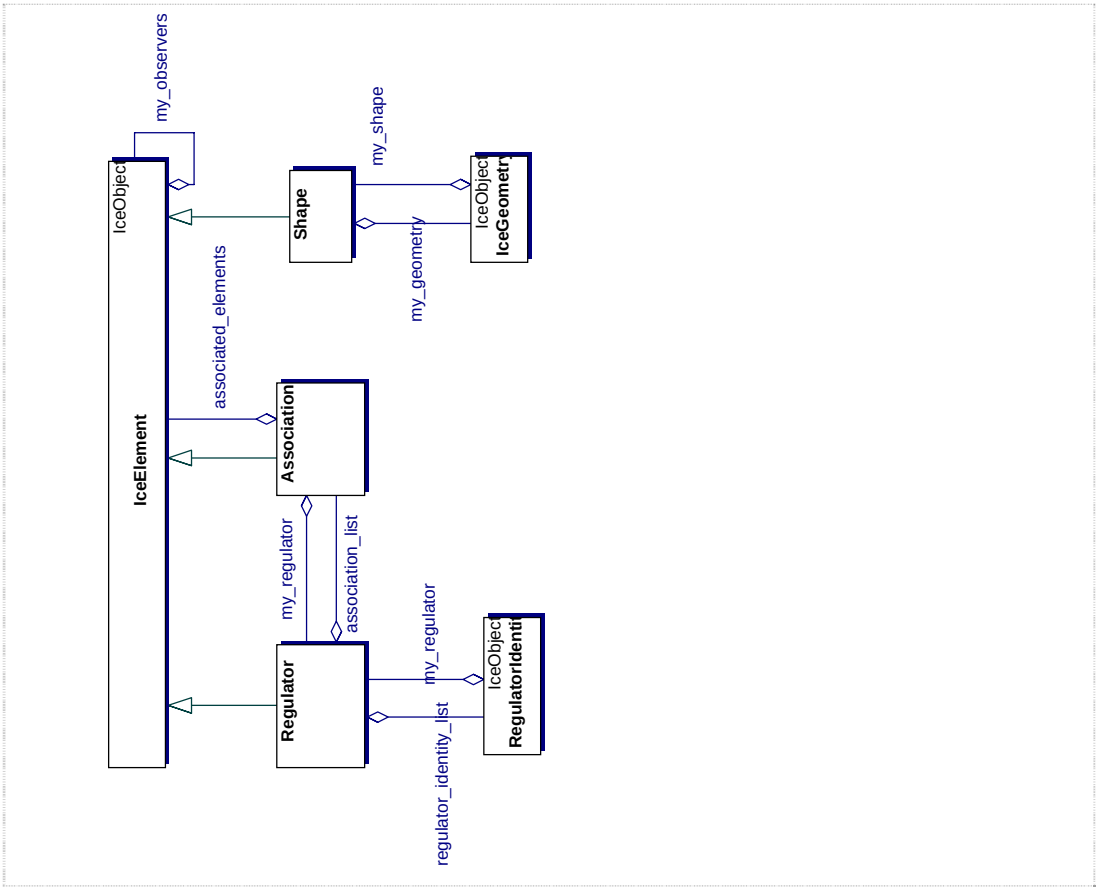
D.1.1. Shape Bridge Object model



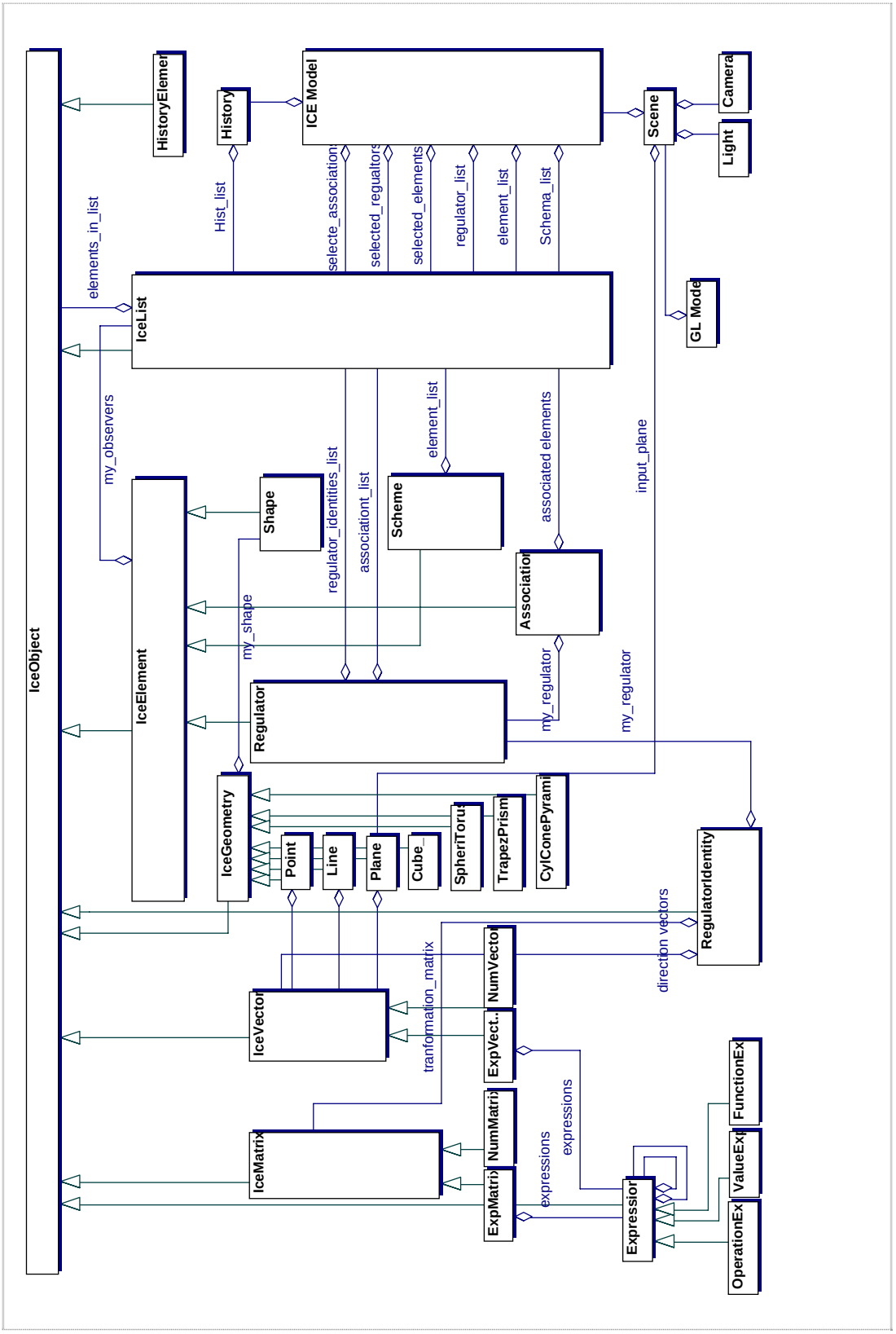
D.1.2. Regulator Bridge Object model



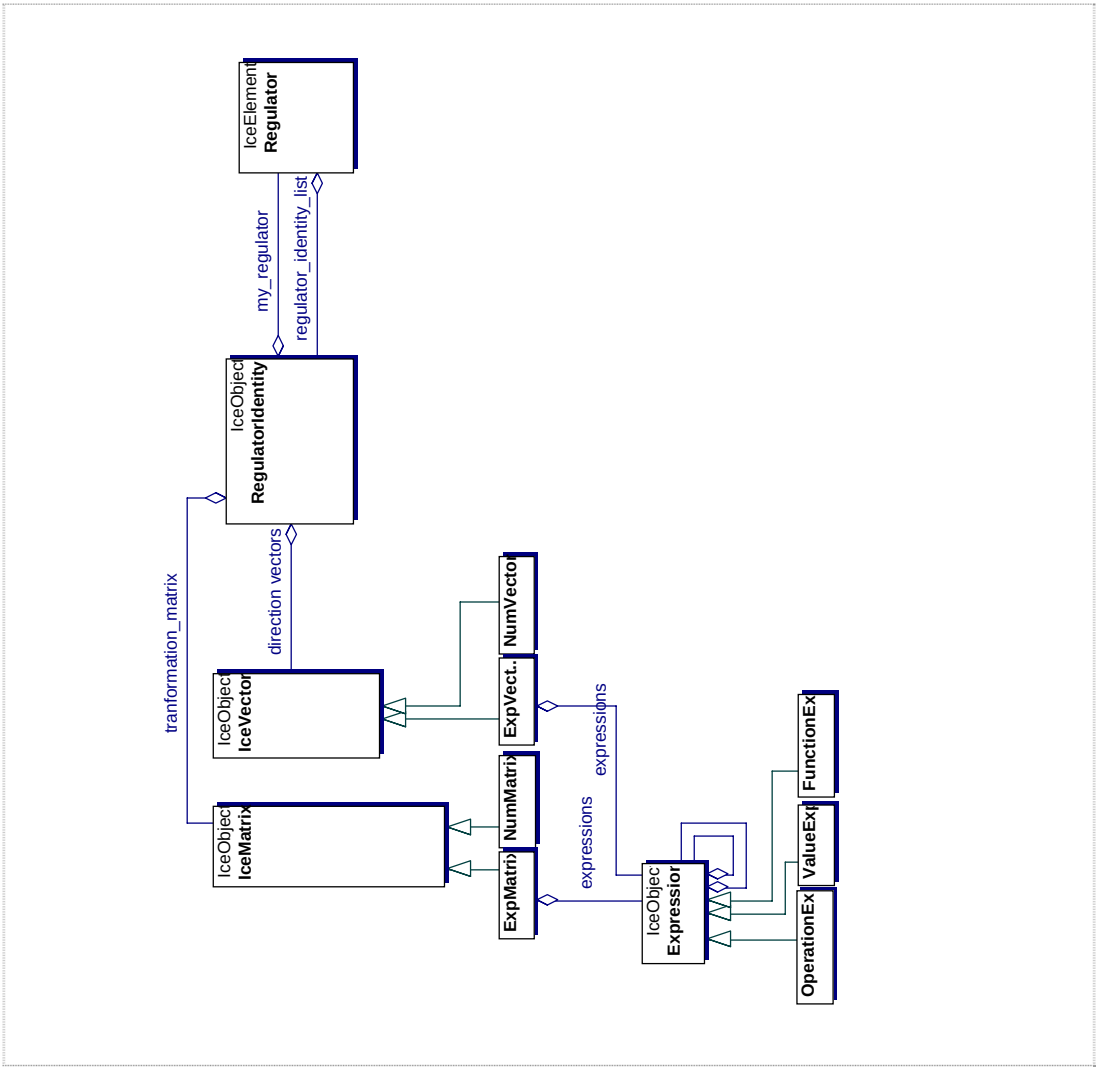
D.1.3. Observer Object Model



D.1.4. ICE Implementation Object Model



D.1.5. Matrices Object Model



D.2. USE-CASES AND INTERACTION DIAGRAMS

All use-cases are designed for the developer as the main actor, in the future theses will be refined and intended for designers. All use-cases assume the presence of shape and scheme libraries, a history list, a graph view, and a string view, which are not currently implemented. The following use-cases include are for instantiation, deletion, selection, shape modification, associations, and regulator modification.

D.2.1. Instantiation Use-cases

D.2.1.1 Instantiate a new shape

Entry condition: The model view is available, the shape controls are available

Flow of events:

The user chooses the shape from a shape library

The system sets the interaction mode for the chosen shape

or

The user ***defines a new shape by composing regulators***

The system sets the interaction mode as “define new shape’

The selected or defined shape appears in the shape pre-view

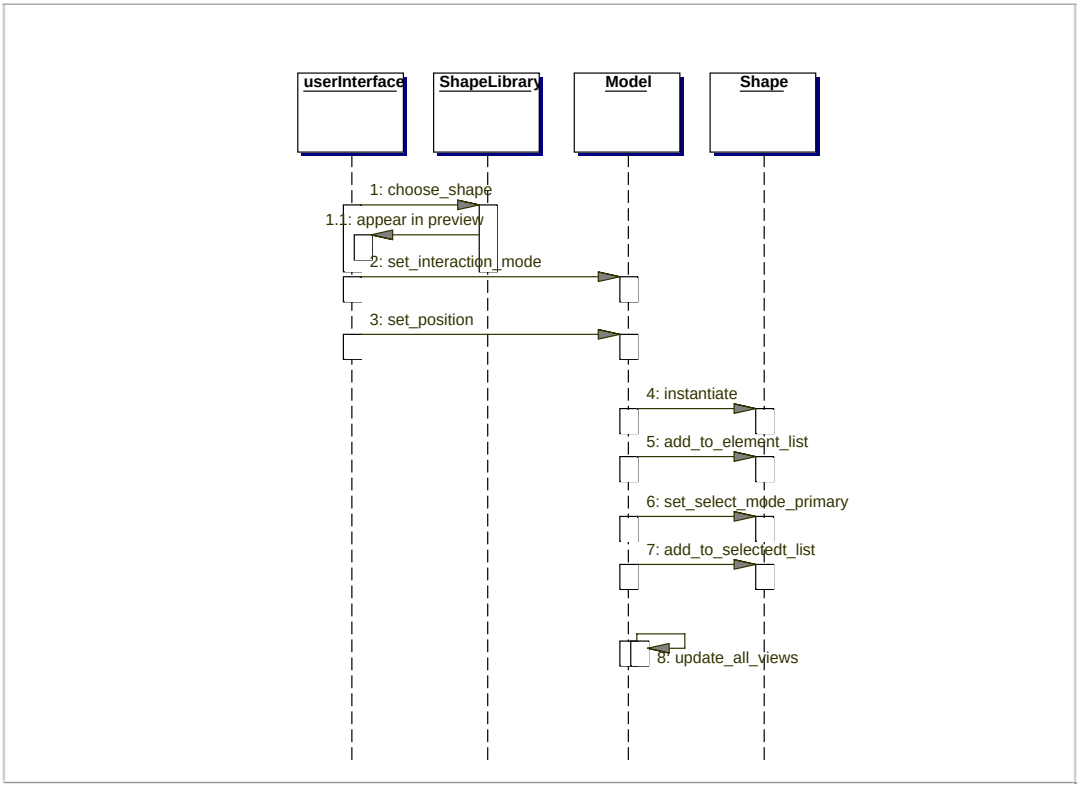
The user positions the cursor clicks the model view

A new shape is instantiated and added to the model’s element list.

The new shape ***is selected***

The system updates the history

Exit condition: The new shape appears on the model view, graph view and the string view.



D.2.1.2 Define a new shape by composing regulators

Entry condition: The model view and shape controls are available

Flow of events:

The user selects the start vertex (or start shape)

The user chooses to define a new shape

The user chooses a regulator (from a list) and

The system ***associates it*** to the start vertex (or shape)

The shape preview displays the resulting shape (in progress)

Optional steps:

The user manipulates the parameters of the regulator

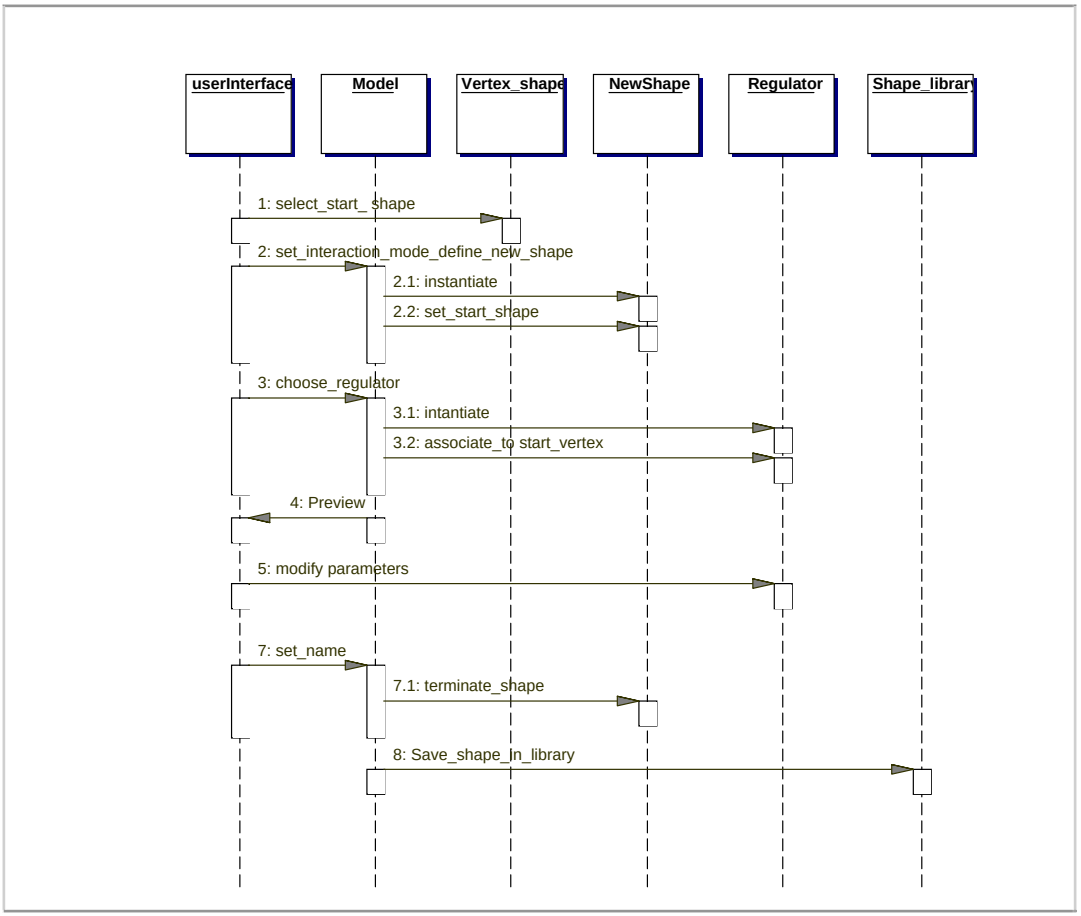
The user chooses the subset of the shapes to generate

The user chooses the subset of generates shapes to be applied to the next regulator

The user can repeat choosing a regulator and associating it to the vertices as many times as needed to define the new shape

Exit condition: The user types the name and terminates the definition of the new shape

The system saves the new shape in the shape library



D.2.1.3 Instantiate a new regulator

Entry condition: The model view is available, the regulator controls are available

Flow of events:

The user chooses the regulator (from the list of possible regulator types)

The system sets the create-regulator mode

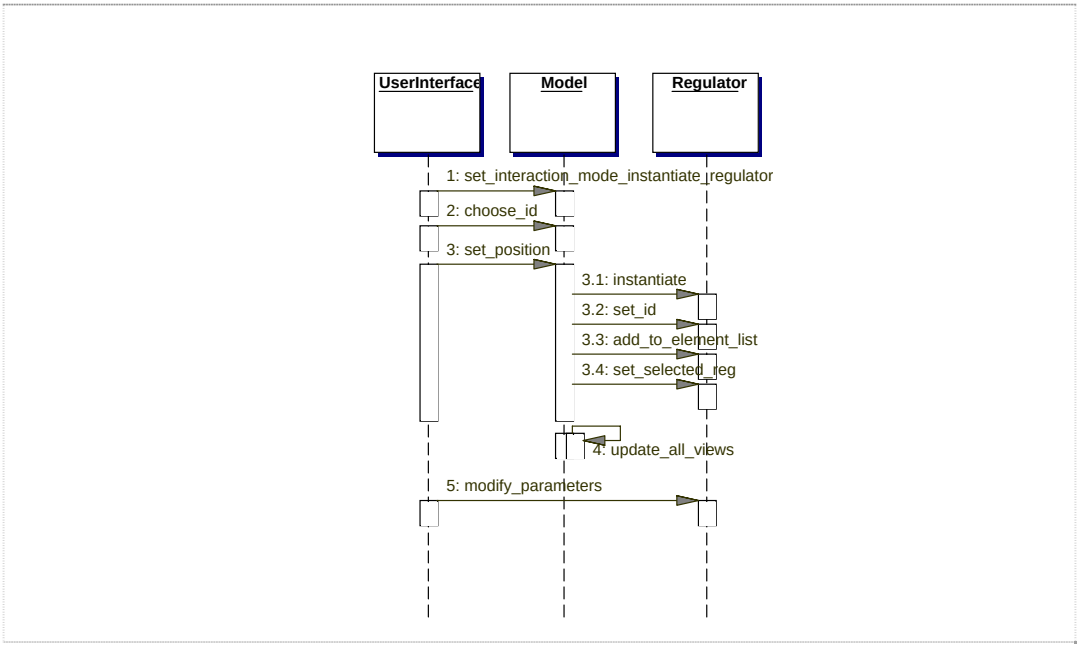
The user points and clicks on the model view

A new regulator is instantiated (with the chosen identities) and added to the model's element list.

The new regulator is selected

Exit condition: The new regulator appears on the model view, graph view and the string view.

Special requirements: The user can set parameters and identities before or after the instantiation of the regulator.



D.2.2. Selection Use-cases

D.2.2.1 Select a shape or a regulator

Entry condition: There exits at least one shape or regulator in the model

Flow of events:

The system default interaction mode is “select an element “

The user points and clicks the mouse over the shape/regulator on the model view.

The system matches the coordinates, identifies the shape or regulator and

If it’s a regulated element it is placed in the model’s element list.

If it’s a regulator it becomes the selected regulator.

Shift clicking allows the user to select multiple shapes

The system ***traverses the regulator tree*** to identify all associated shapes and regulators.

The system updates the history

Exit condition:

The selected shape/regulator will appear highlighted (with primary highlights) on the model-view, the graph view, and the string view.

The shapes and regulators associated directly and those associated by transitivity will also be highlighted in secondary and ternary highlights respectively.

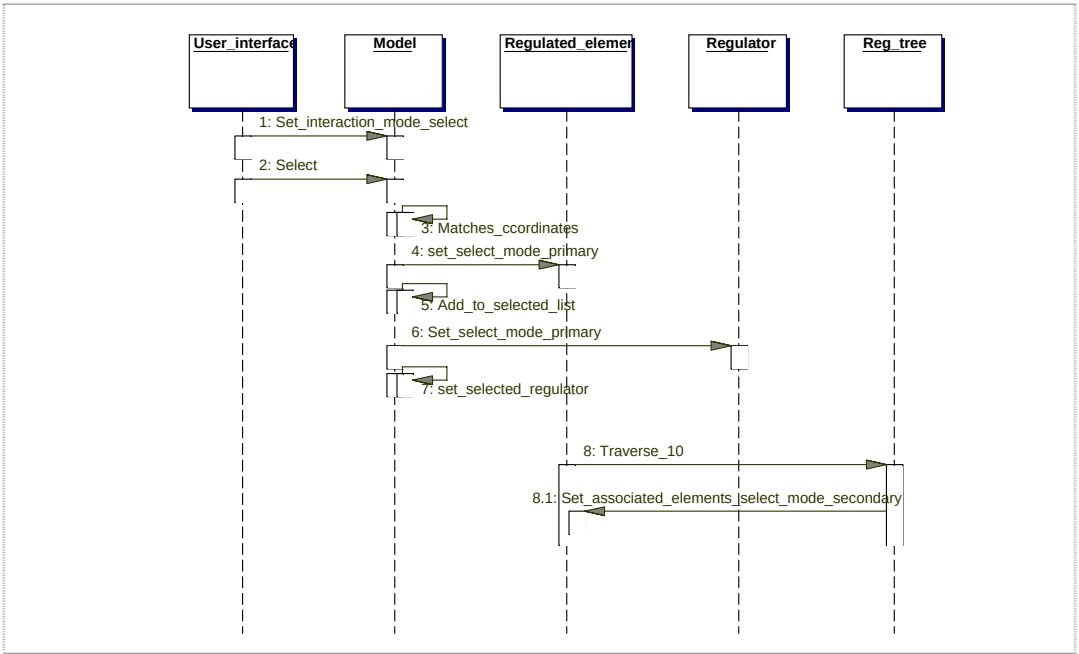
Associations are highlighted on the graph view.

The shapes or regulator controls become available

Special requirements:

The system relies on OpenGL selection mechanism to match the coordinates with the shape/vertex and return a shape/vertex ID

Variations of “select a regulator” include (i) selecting all the shapes of a regulator by clicking on the regulator, (ii) selecting all associations of a regulator by clicking on the regulator



D.2.2.2 Select an association

Entry condition:

A shape or a regulator is selected and its direct associations are highlighted on the model. The associations are depicted and highlighted on the graph view.

Flow of events:

The user points and clicks the mouse over the association on the graph view

The system identifies it as the selected association

Shift clicking allows the user to select multiple shapes

The system updates the history

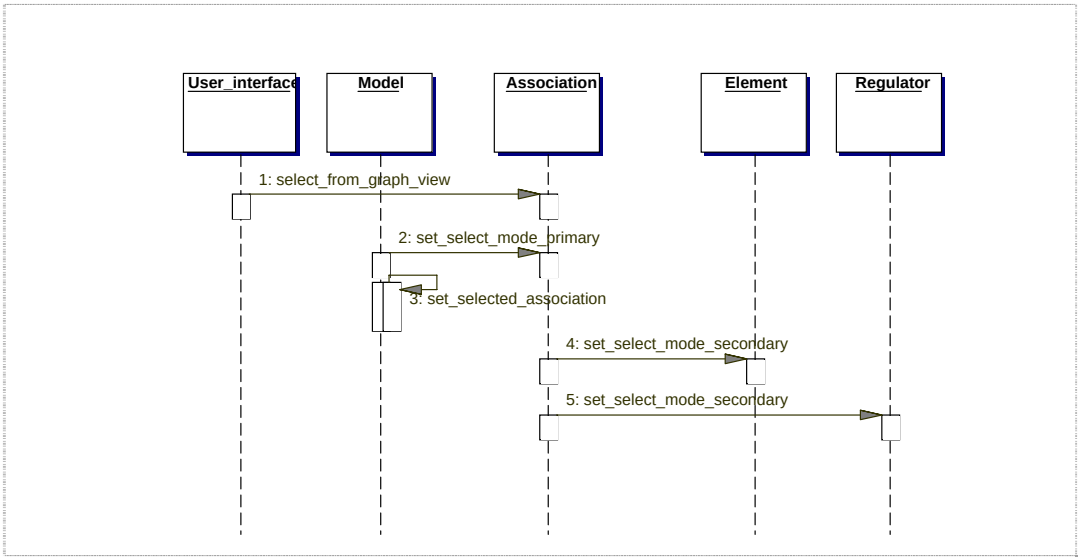
Exit condition:

The shapes and the regulator of the selected association will appear highlighted (with primary highlights) on the model-view and the string view.

The association controls become available.

Special requirements:

A variation of “select an association” include selecting all shapes of a specific association by clicking on the association.



D.2.2.3 Select a vertex, an internal-regulator, or an internal-association

Entry condition: A shape or regulator is selected

Flow of events:

The user makes a second click on the shape

All the vertices and internal regulators and internal associations appear on the model, and the graph view and the string view.

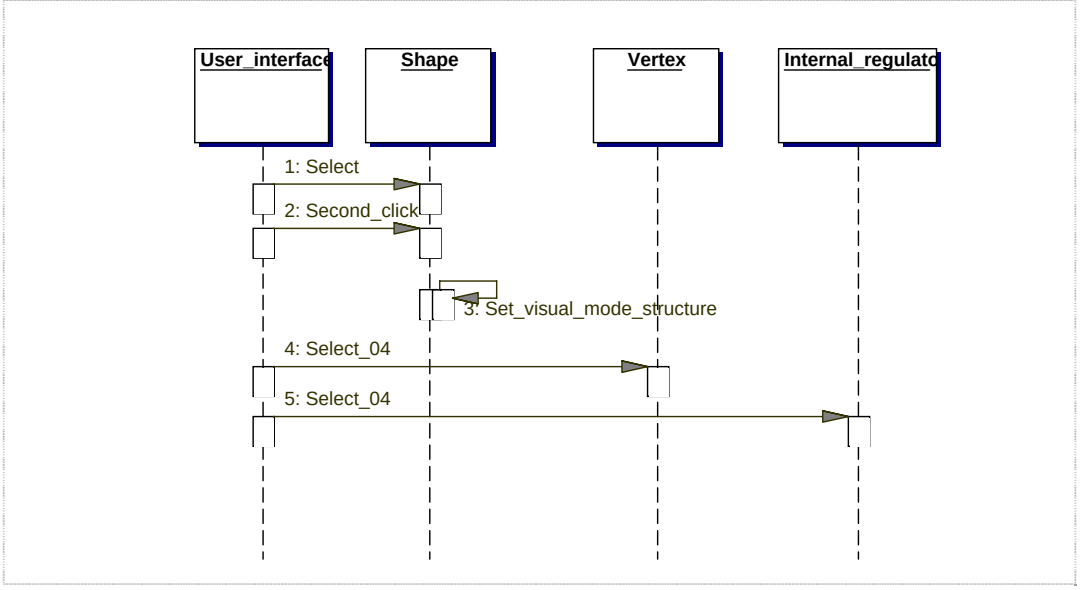
The user selects the vertex or internal regulator in the same way as ***select a shape or regulator***

The user selects a subshape-association in the same was as ***select an association***

The system updates the history

Exit condition:

The user deselects the shape.



D.2.3. Viewing Usecases

D.2.3.1 Show/hide/emphasize shapes and regulators

Entry condition: There are shapes and regulators in the model

Flow of events:

The user selects one of the following

a shape

a regulator

an association

all-shapes

all regulators

all associations

The user turns on/off the visibility of the selected item

The system sets the visibility flag for the selected item

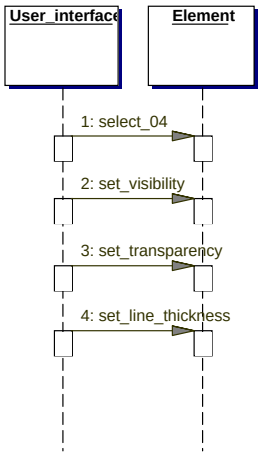
The user adjusts the transparency level of the selected item

The system sets the alpha value for the selected item

The user adjusts the line thickness of the selected item

The system sets the line thickness for the selected item

The system updates the history



D.2.4. Deletion Use-cases

D.2.4.1 Delete a vertex or a shape

Entry condition: A shape or vertex is selected

Flow of events:

The user chooses to delete the selected element

The system sets the interaction mode as “delete element”

The system deletes the shape or vertex and

The ***system traverses the regulator tree*** to deletes the image-shapes of this shape/vertex

For deleted vertices, the ***system traverses the external regulator tree*** to propagate change to associated image shapes

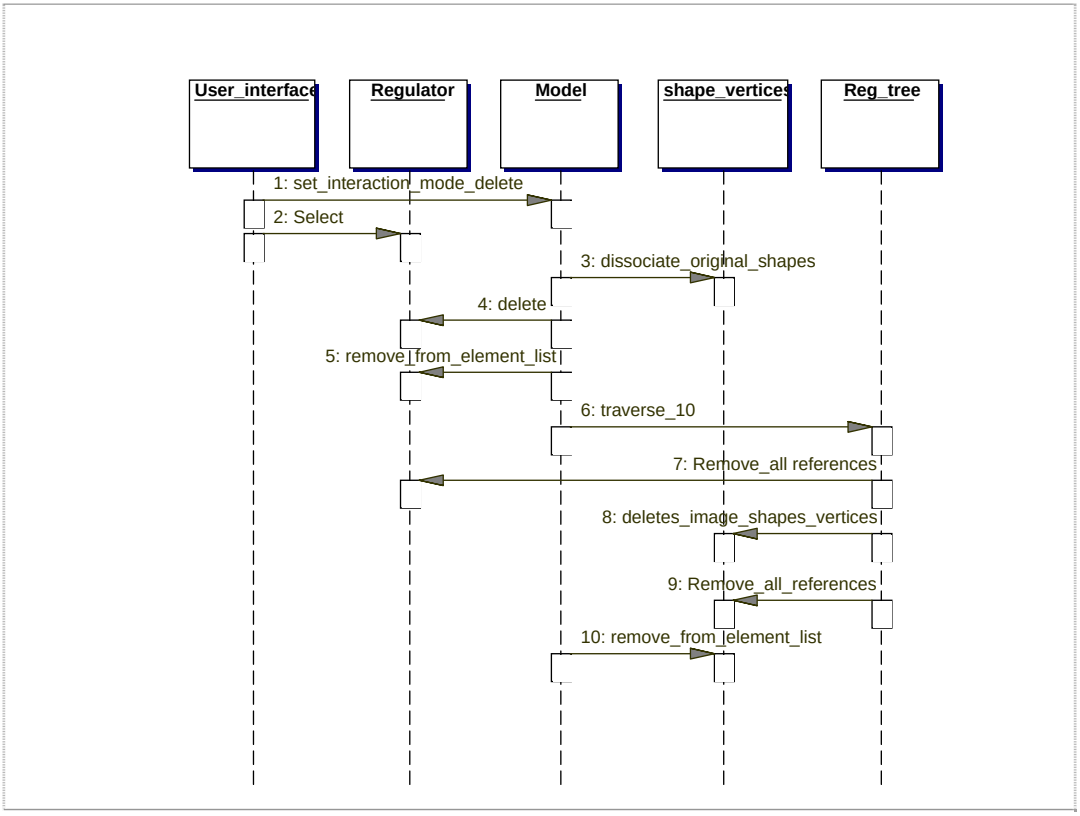
The system removes all references to it (and its image-shapes) from regulators and associations

The system removes it from the models element list

The system updates the history

Exit condition: The model view, graph and string views are updated.

Special requirements:



D.2.4.2 Delete a regulator

Entry condition: A regulator is selected

Flow of events:

The user chooses to delete the selected element

The system sets the interaction mode as “delete element”

If it is a non-generative regulator, all shapes are dissociated from the regulator but remain as independent shapes

If it’s a generative regulator, Its input shapes are dissociated and Its output shapes are deleted

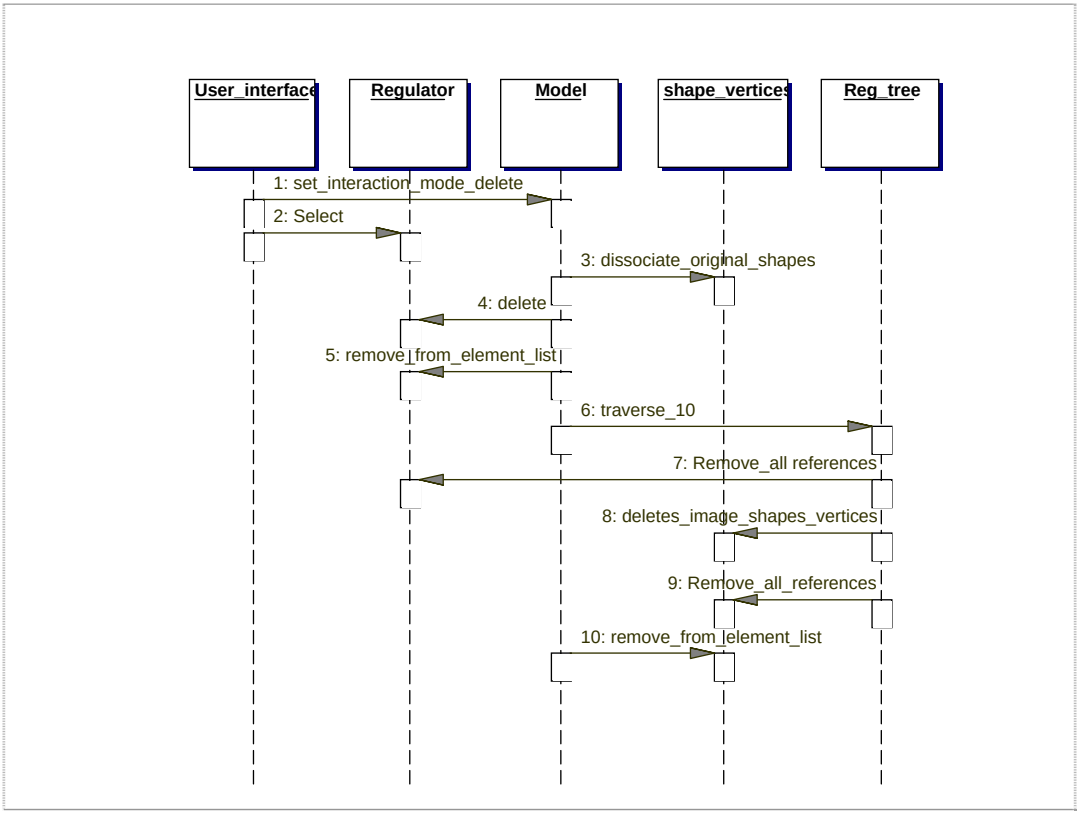
If the regulator is in a sequence, the original shapes of this regulator become the original shapes of the succeeding regulator

The system deletes regulator from the regulator tree and removes all references to it.

The system ***traverses the regulator tree***, and regenerates it starting from the deleted regulator.

The system updates the history

Exit condition: The model view, graph and string views are updated.



D.2.5. Tree Traversal Use-cases

D.2.5.1 ***Traverse a regulator tree (System use-case)***

Entry condition: The user selects or changes an element in the regulator tree

Flow of events:

Traversal begins from a shape

The shape notifies its observers (associations or regulators)

The observers (associations or regulators) update their *other* shapes (based on the new values and the regulator's formula)

These other shapes notify their *other* observers (associations or regulators)

Traversal begins from a regulator

The regulator goes through its regulated elements (shapes or associations)

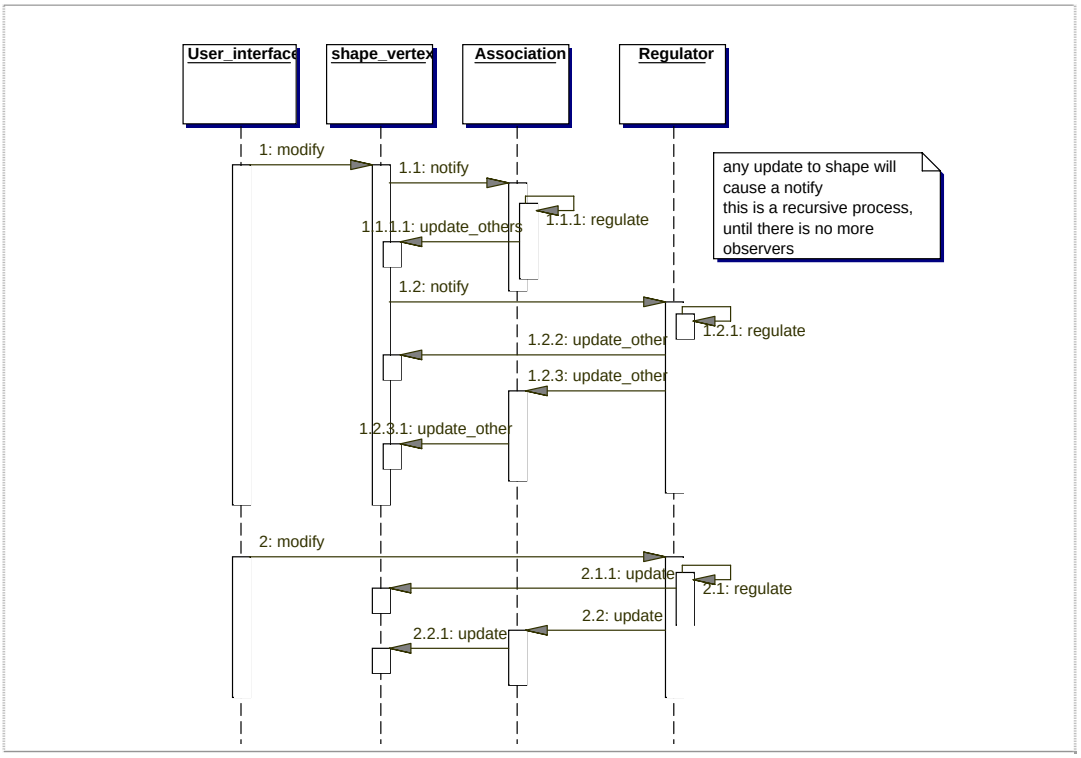
The regulator updates its direct elements

The associations update their shapes (based on the new values and the regulator's formula)

The shapes notify their *other* observers (associations or regulators)

Exit condition:

The notification-update process recursively continues until there are no more observers (associations or regulators)



D.2.6. Shape Modifications Use-cases

D.2.6.1 Modify a shape

Entry condition: A shape is selected in the model, and shape controls are available

Flow of events:

The user modifies an attribute of a the shape (attributes = color, alpha, line thickness)

The system sets the new values for this shape

or

The user ***modifies a vertex of the shape***

or

The user ***modifies the internal regulator of the shape***

or

The user ***replaces the shape***

or

The user ***subdivides the shape***

or

The user moves or rotates the shape

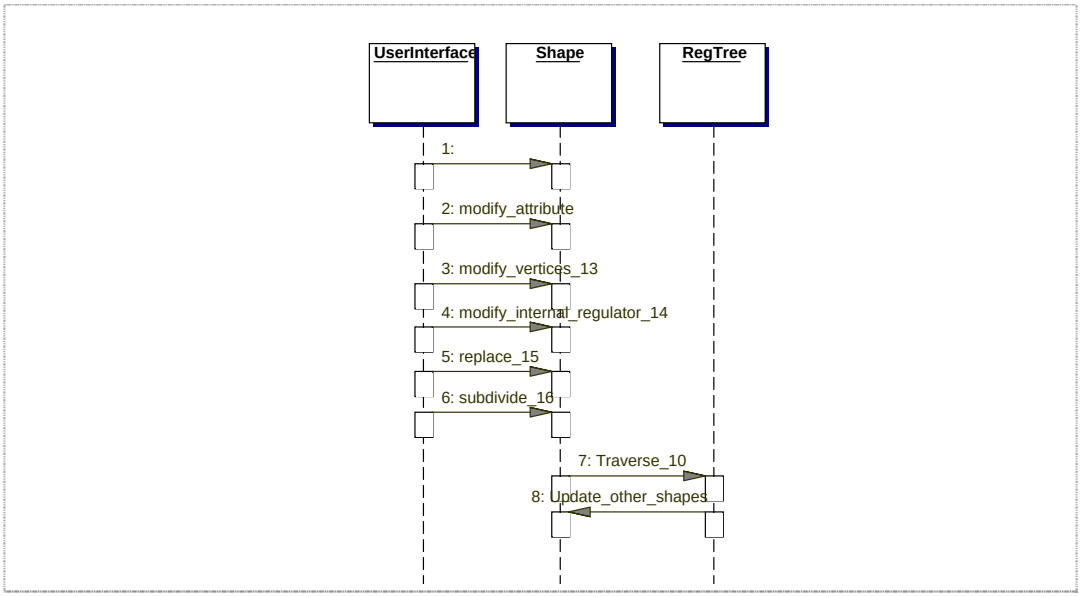
The system adjusts the position or rotation parameters

The system ***traverses the external regulator tree*** of the configuration and testing for the constraints and propagating the replacement to all associated shapes

The system updates the history

Exit condition:

The drawing view, graph views, and the string view are updated to show the modified shape



D.2.6.2 *Modifying the shapes controlled by constraint regulators*

Entry condition: The ***user modifies the shape***

Flow of events:

The system tests the new shape configuration with the regulator's formula

If the new shape configuration is conforming with the regulator's constraints

The system updates the shape

Otherwise

If the system is in "keep constraint mode" (left mouse)

The shape remains in its initial configuration

If the system is in "remove constraint mode" (right mouse)

The shape is dissociated from the regulator and

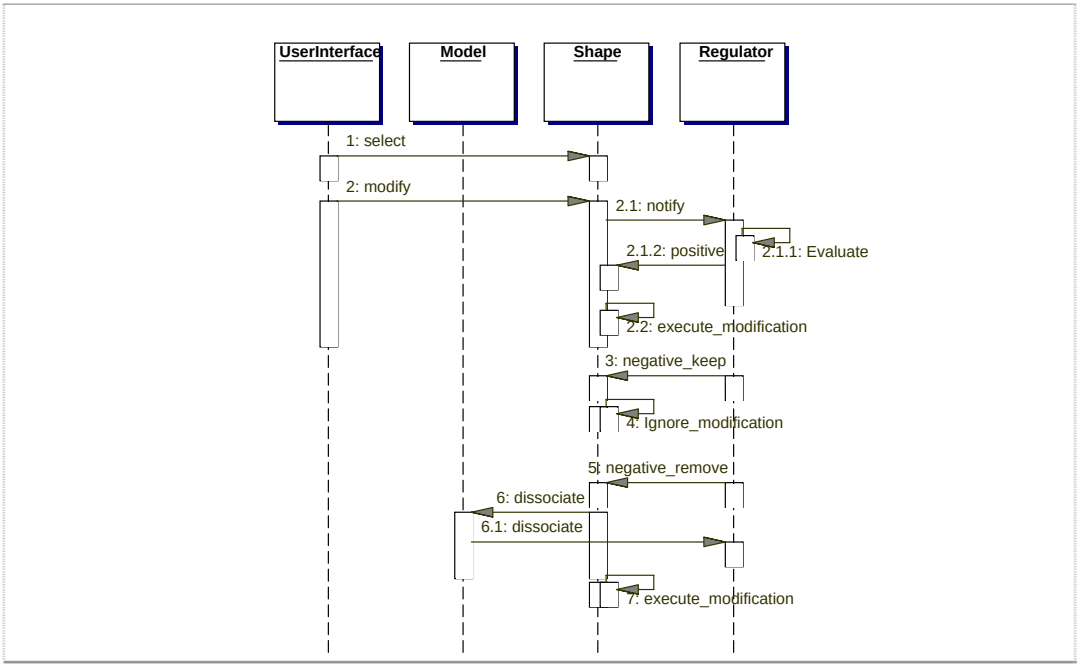
Its configuration is updated.

Special requirements:

Each regulator will have its specific test formula and significant modifications

This use-case is applicable to the following regulators.

- ALIGNMENT/ BOUNDARY: Moving/rotating
- PROPORTION/ DIMENSION/ AREA-VOLUME: Resizing
- ANGLE : Rotating
- DISTANCE/ ADJACENCY/ OVERLAP: moving
- CONTAINMENT: moving/ rotating/ resizing



D.2.6.3 Modify (move) a vertex in *a* shape

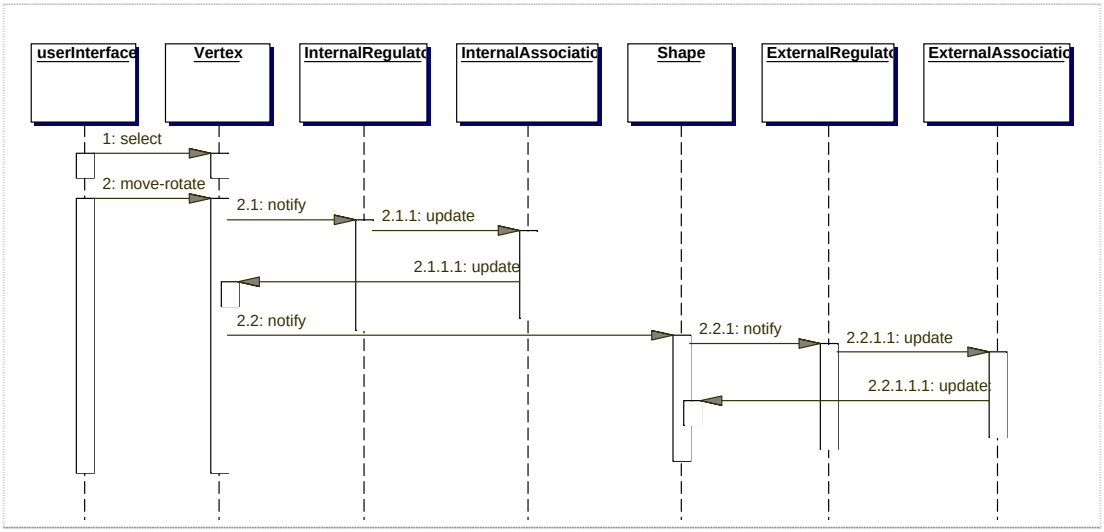
Entry condition: A vertex within a shape is selected in the model

Flow of events:

The user moves the vertex (by dragging the mouse or by adjusting the coordinate values)

The system sets the new vertex position

The system ***traverses the internal regulator tree***, and propagates the vertex move, therefore updating the whole shape. (This can cause a change in form, dimension, and/or size of the shape)



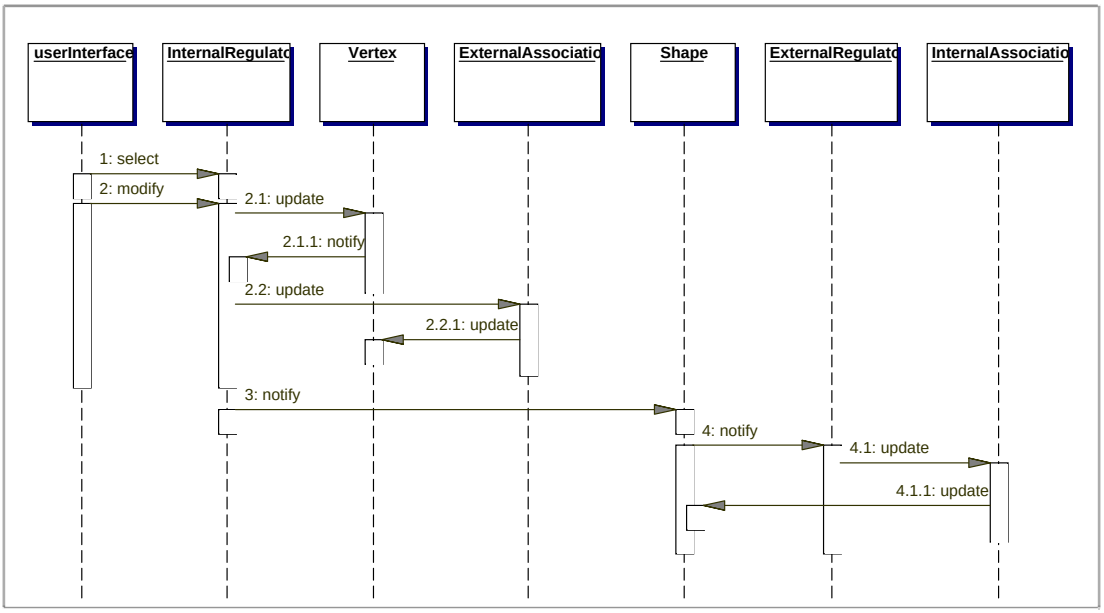
D.2.6.4 Modify the Internal Regulators of a Shape

Entry condition: An internal regulator within a shape is selected in the model

Flow of events:

The user ***manipulates the regulators parameters*** (geometry, parameters, number, and variations)

The system ***traverses the internal regulator tree*** and propagates these changes to the regulated vertices, as well as to all associated vertices, therefore updating the shape.



D.2.6.5 Replace a shape

Entry condition:

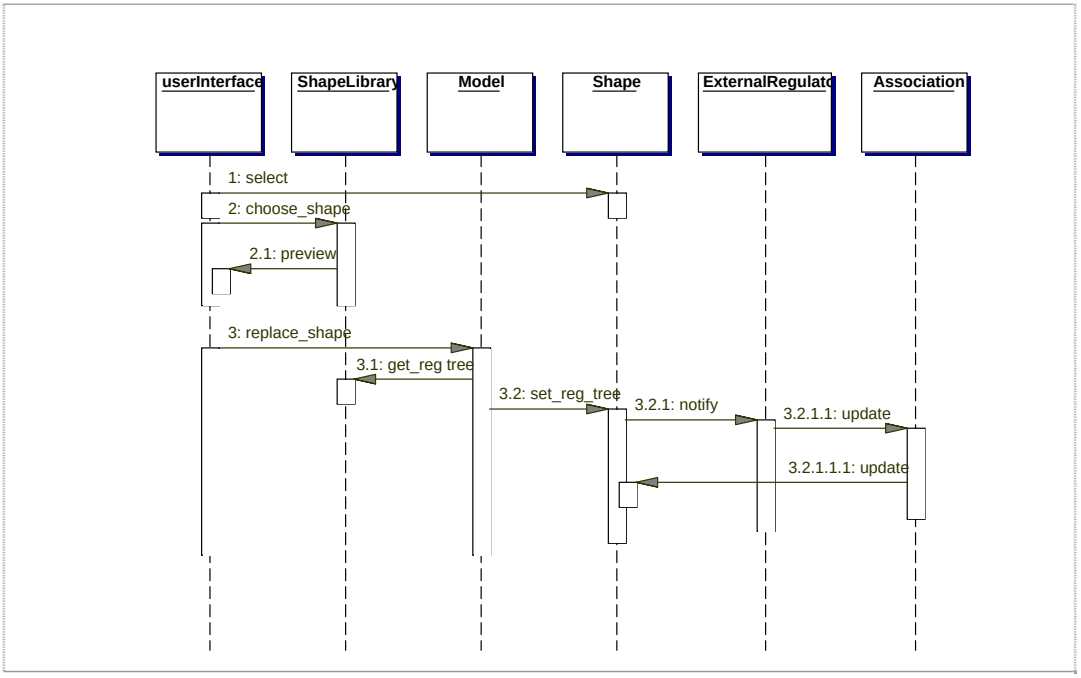
A shape is selected in the model and shape controls are available

Flow of events:

The user selects another shape from the shape library, which appears on the shape preview

The user chooses to replace the selected shape on the model with the one in shape library

The system replaces the internal regulator tree of the shape, while keeping its location (first vertex), and other attributes intact



D.2.7. Association Use-cases

D.2.7.1 *Associate a shape, vertex, or regulator to a regulator*

Entry condition:

There is at least one shape or item and one regulator in the model

Flow of events:

The user selects an item(s) to be regulated (this can be a shape, vertex, or a regulator)

The user selects a regulator

The user chooses to associate

(press a button or drag the right mouse and drop it on the regulator)

If it's a non generative regulator:

The system adds the item to the list of regulated elements pertaining to the selected regulator.

The item will have a reference to the regulator

If it's a generative regulator:

The regulator instantiates an association which assigns this item as the original,

The regulator generates n number of images-items and adds them to the image list of the associations.

The association will be added to the list of regulated elements pertaining to the regulator.

The item will have references to the association and to the regulator.

The system updates the history

Exit condition:

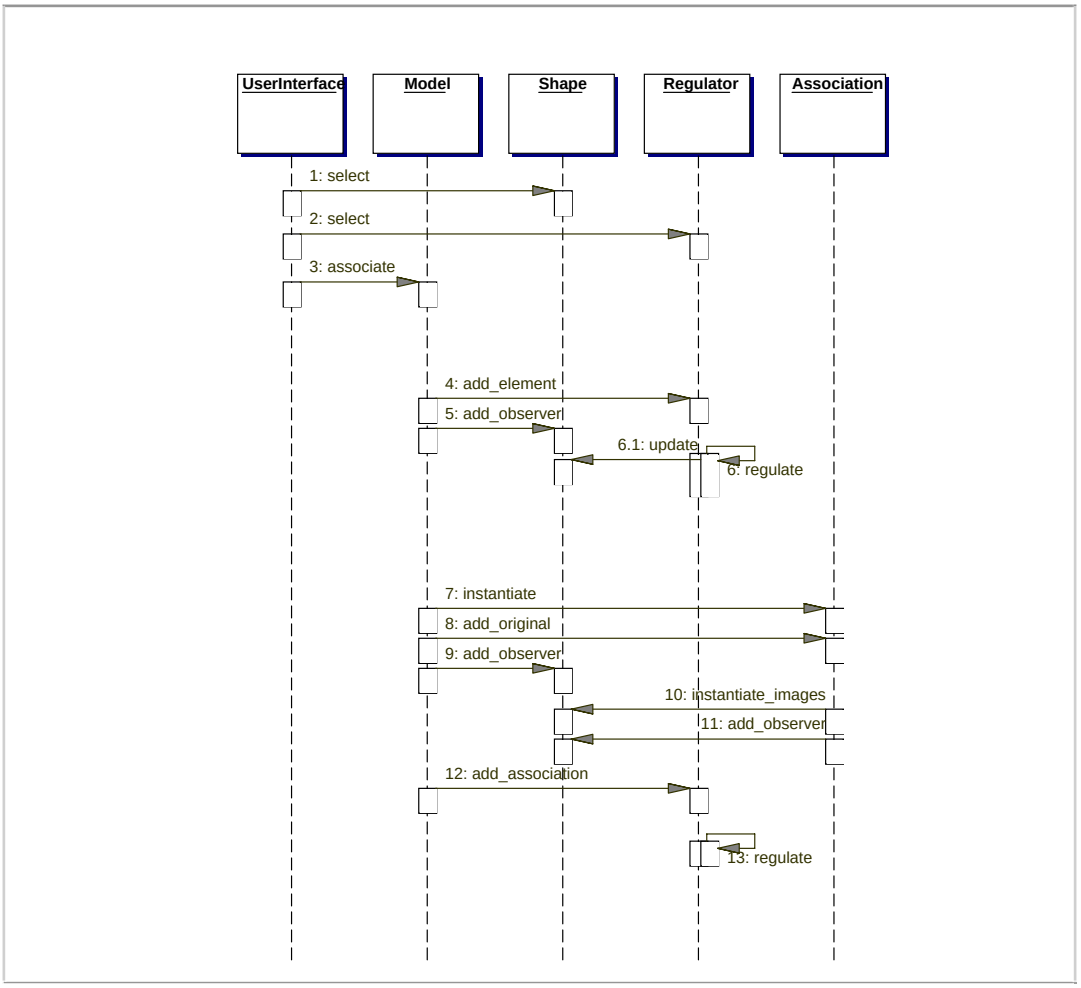
The regulator and the item form an associated branch in the regulator tree.

The regulator will regulate this item for the first time indicating that the association succeeded.

Special requirements:

The regulator will regulate this item (image-shapes/vertices/regulators) every time the regulator tree is traversed.

The user can set the regulators parameters before or after the association


UserInterface
Model
Shape
Regulator
Association
1: select
2: select
3: associate
4: add_element
5: add_observer
6.1: update
6: regulate
7: instantiate
8: add_original
9: add_observer
10: instantiate_images
11: add_observer
12: add_association
13: regulate

D.2.7.2 Associate a regulator to all the shapes in a branch of the regulator tree

Entry condition:

There are several generative regulators (with image shapes) in a sequence

Flow of events:

The user ***instantiates a new regulator*** or ***select a regulator***

The user ***selects a shape*** in the regulator tree

The user ***chooses to associate the new regulator to all*** *shapes* branching from this shape

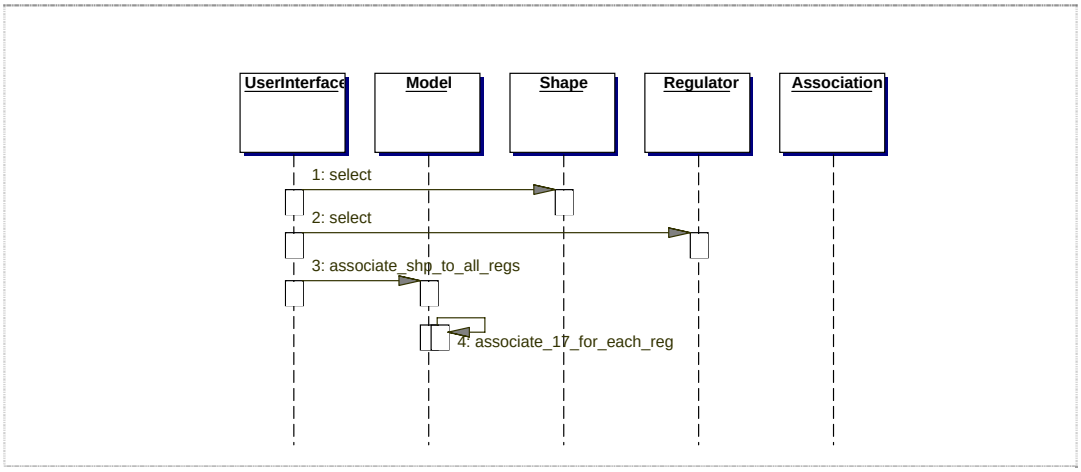
The system updates the history

Exit condition:

The system traverse the regulator tree (from the selected shape) ***associating each*** shape (in this branch) to the new regulator

Special requirements:

To associate all shapes in the tree associate, the user needs to select the first shape.



D.2.7.3 Associate a subset of image-shapes to a regulator

Entry condition:

There is at least one generative regulator with images shapes

Flow of events:

The user ***instantiates a new regulator*** or ***select a regulator***

The user ***selects the desired images shapes*** (shift click)

or

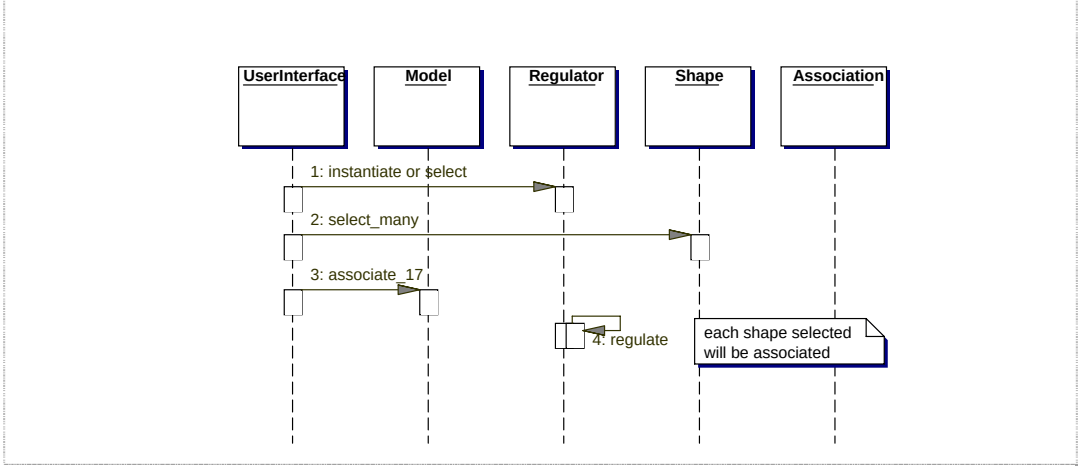
The user ***selects an association***

The user chooses to ***associate these shapes to the new regulator***

The system updates the history

Exit condition

The system ***associates each of these selected shapes to the new regulator***



D.2.7.4 *Dissociate a shape from a regulator*

Entry condition:

There is at least one shape or item and one regulator (associated to each other) on the model.

Flow of events:

The user selects the shape or regulated item

The user selects the regulator

The user chooses to dissociate these

If it's a non generative regulator:

The shape is removed from the list of regulated elements pertaining to the regulator

The shape's reference to regulator is deleted

If it's a generative regulator:

All image shapes remain but are dissociated from the regulator

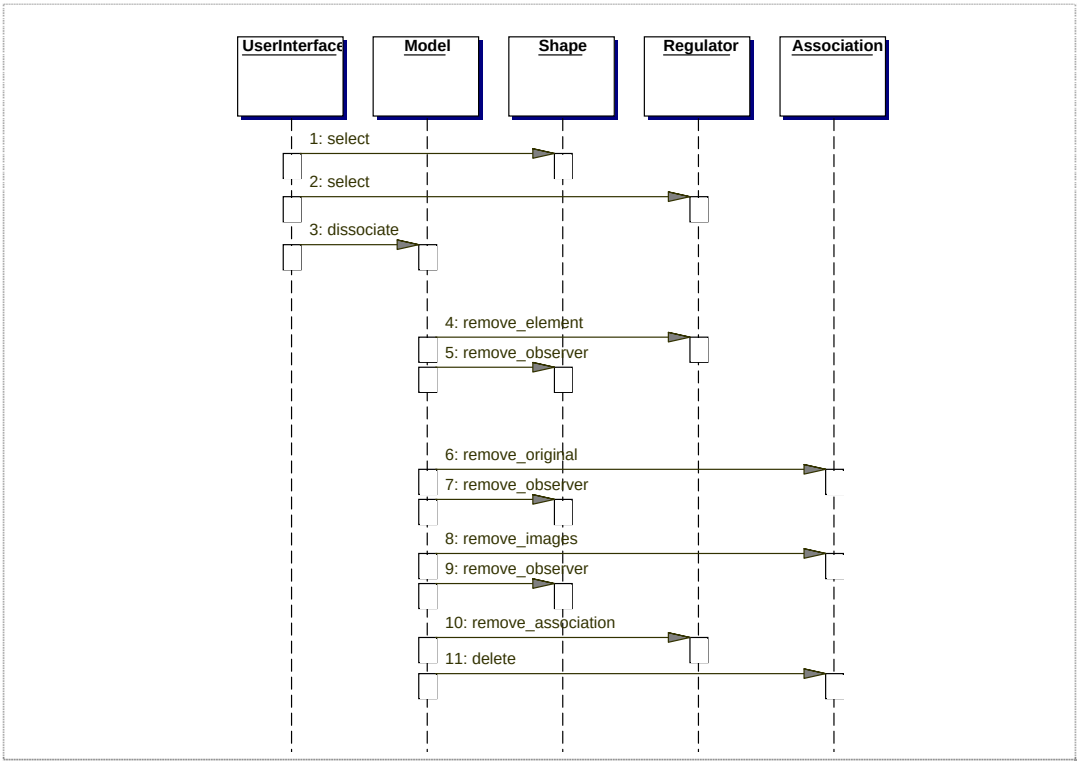
The original shape's reference and image shape's reference to regulator and to the association are deleted.

The association is deleted

These image shapes become independent shapes on the model

Exit condition:

The shapes are no longer regulated by this regulator



D.2.7.5 *Relate a set of shapes*

Entry condition:

There are shapes and regulators in the model,

Flow of events:

RELATE a set of shapes

The user selects the shapes

The user chooses to relate them

The system generates a new regulator

The system puts these two shapes in its association

The user chooses the identity for this new regulator

Or

RELATE a shape to an existing association

The user selects a shape

The user selects an association

The user chooses to relate them

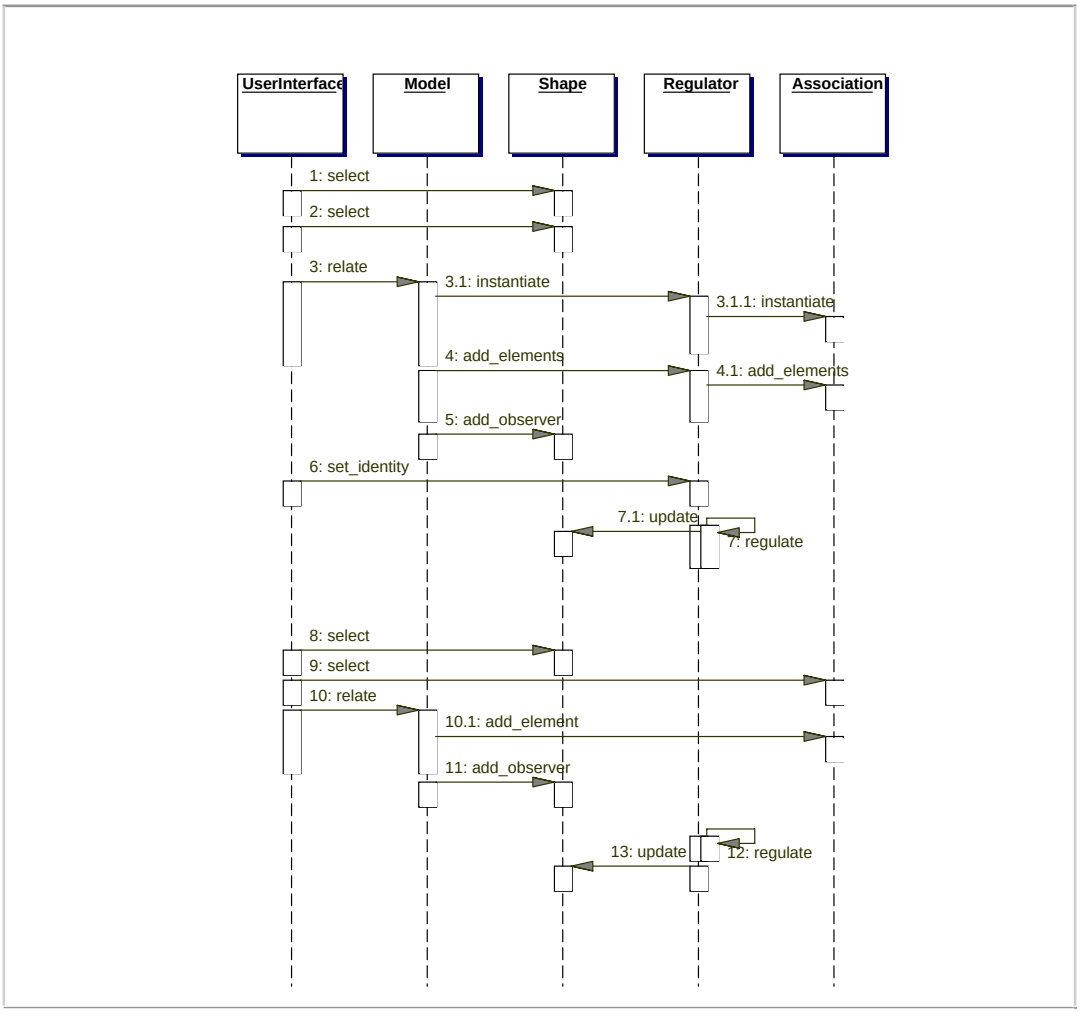
The system puts the selected shape in the selected association

The system traverses the regulator tree and updates the newly related shapes to fit the regulator's parameters

The system updates the history

Special requirements:

This use-case occurs when the user generates a configuration, then realizes that some objects have a relation that can benefit from regulation


UserInterface
Model
Shape
Regulator
Association
1: select
2: select
3: relate
3.1: instantiate
3.1.1: instantiate
4: add_elements
4.1: add_elements
5: add_observer
6: set_identity
7.1: update
7: regulate
8: select
9: select
10: relate
10.1: add_element
11: add_observer
13: update
12: regulate

D.2.7.6 *Modify the number/density of elements generated by the regulator*

Entry condition:

An association is selected

Flow of events:

The user modifies the number of shapes/vertices generated by the regulator

If the number of shapes is increased

The system ***instantiates new shapes/vertices*** and ***associate them to the regulator.***

If the number of shapes is decreased

The system ***deletes*** the extra shapes and removes reference to them

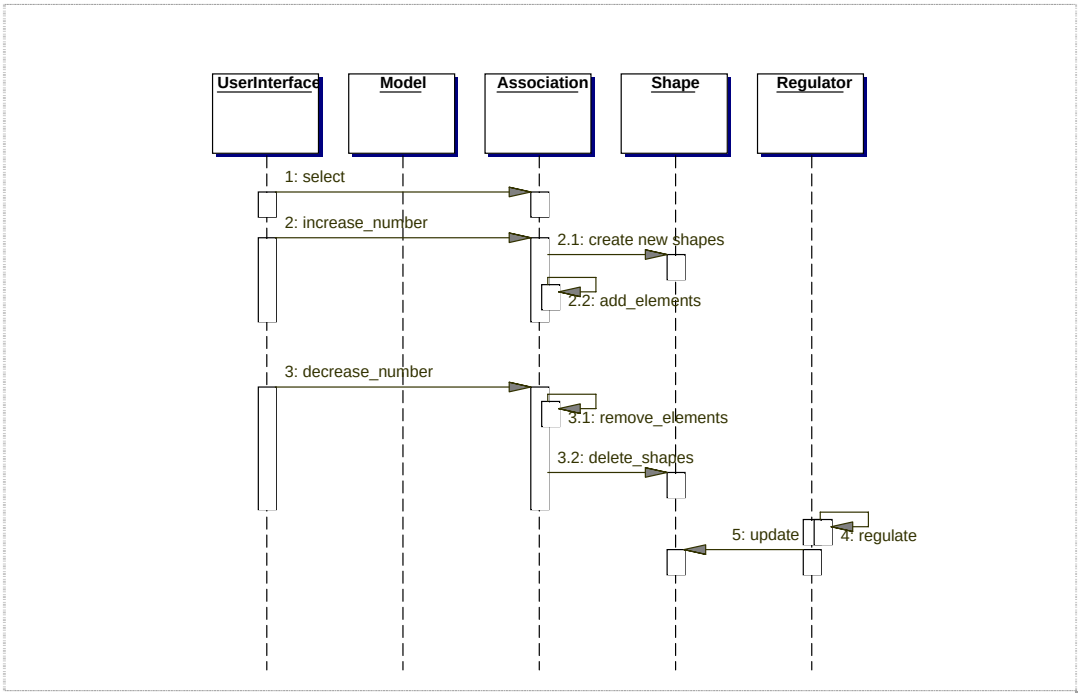
The user modifies the density of shapes/vertices generated by the regulator

The system adjusts the number as well as the factor, so that the extent remains the same.

Special requirements:

If a regulator is selected, all its associations are updated

Only applicable to generative regulators



D.2.7.7 *Choose the subset of image-shapes to generate*

Entry condition:

An association is selected and the association controls are available: A list of image shapes become available for editing

Flow of events:

The user chooses the images to be generated (checks them)

or

The user chooses to use the pattern mode of generation and

The user enters the number of pattern (on-off)

The system updates the generated images according to the users choices

it ***deletes*** some images-shape and

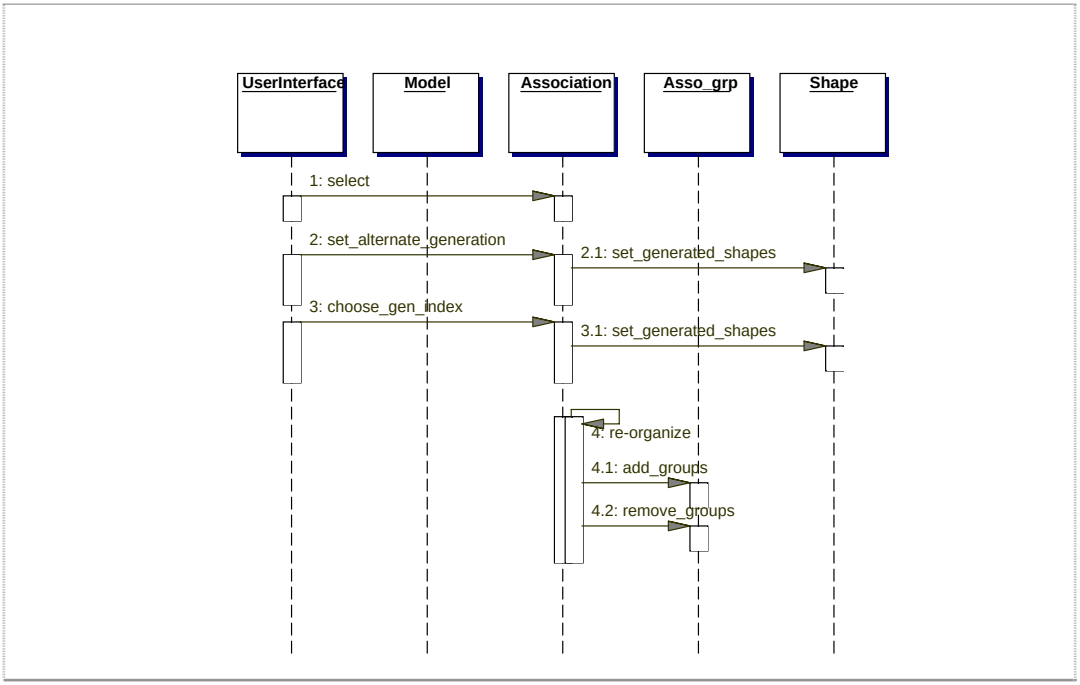
it ***instantiate*** other-image shapes

The association is grouped according to the adjacent generated shapes

Special requirements:

If a regulator is selected, all its associations are updated

Only applicable to generative regulator



D.2.7.8 *Set/modify the discrete/continuous/combination factors of an association*

Entry condition:

An association is selected and the association controls are available: A list of image shapes become available for editing

Flow of events:

The user chooses the continuous/discrete/ parameters

or

If the user prefers the combined mode (partly continuous and partly discrete)

The user chooses the indices that are continuous

The system organizes the association in groups according to the indices in the combined mode

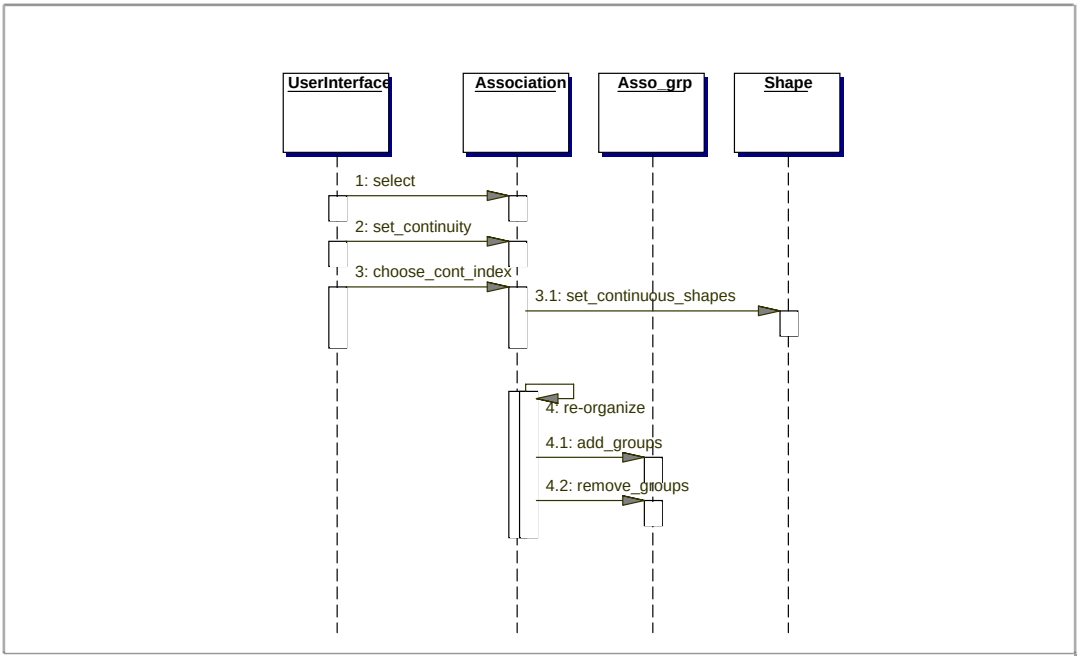
Special requirements:

The default is discrete for external regulators

The default is continuous for internal regulators

If a regulator is selected, all its associations are updated

Only applicable to generative regulators



D.2.7.9 *Set exceptions and variations within the generated set*

Entry condition:

An association is selected and the association controls are available: A list of image shapes become available for editing

Flow of events:

The user chooses the image-shapes to be exceptions

These will not be regulated as the other images, and it will behave as an exception.

The user modifies the transformation factor for this image shape.

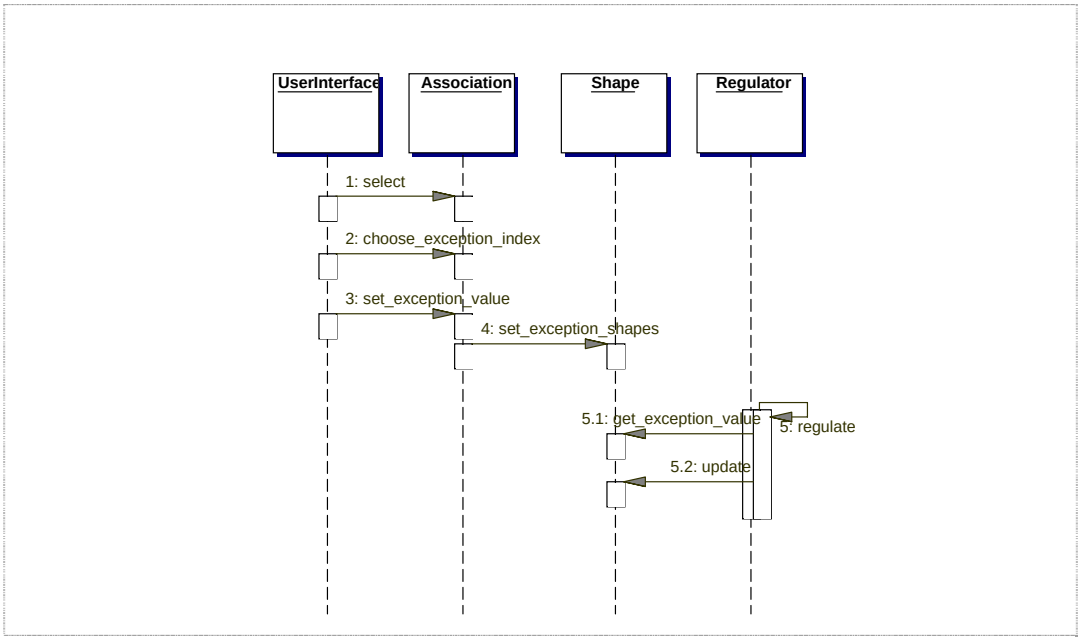
The system will flag this as an exception and store its user defined factor vector.

This factor will override the common transformation factor and will behave as exception.

Special requirements:

If a regulator is selected, all its associations are updated

Only applicable to generative regulator



D.2.8. Regulator Modification Use-Cases

D.2.8.1 *Modify the regulator*

Entry condition:

A regulator is selected

Flow of events:

The user ***moves or rotates the regulator***

or

The user ***manipulates the regulator's endpoints***

or

The user ***modifies the direction vector for the regulator***

or

The user ***modifies the major parameters of the regulator***

or

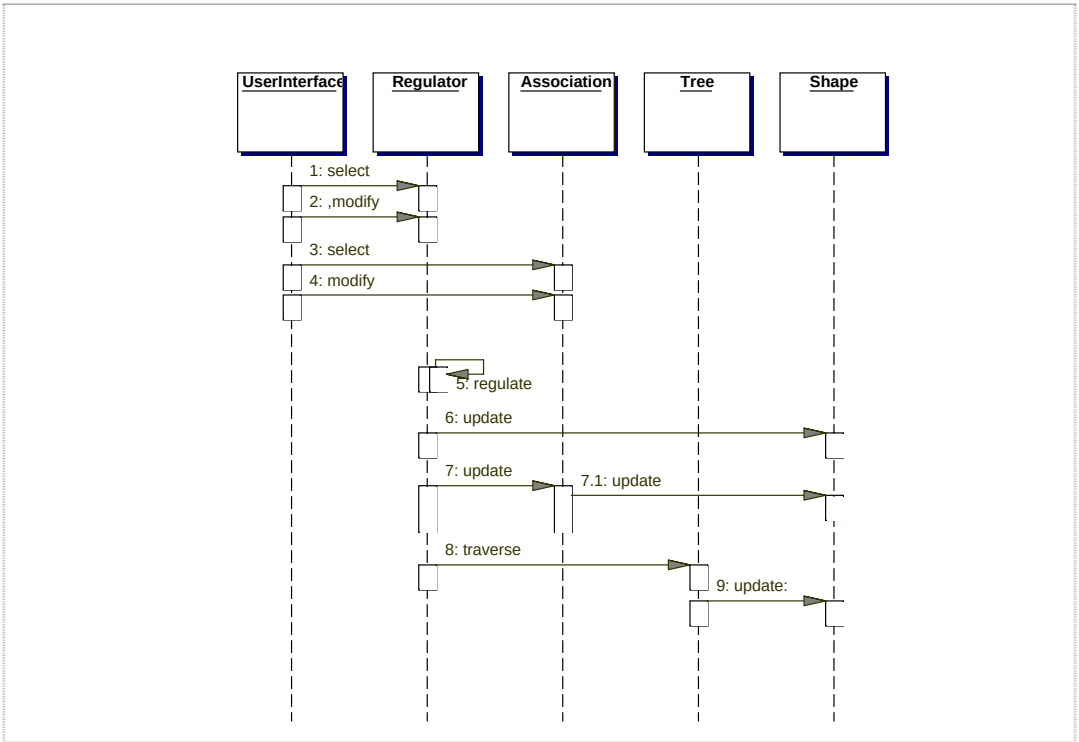
The user ***modifies the dimension of the regulator***

The regulator re-regulates its dependent shapes/vertices to accommodate the change in geometry or number or parameters of the regulator

The system ***traverses the regulator tree*** to update all the associated shapes and vertices.

Exit condition:

The regulator and the model is updated on the model, graph and string view



D.2.8.2 *Activate/ de-activate a regulator or a category of regulators*

Entry condition:

The model view is available

There is at least one regulator (associated to shapes) in the model

Flow of events:

The user selects a regulator or selects all-transformation or all constraint regulators

The user chooses to activate or deactivate the selected items

For activation:

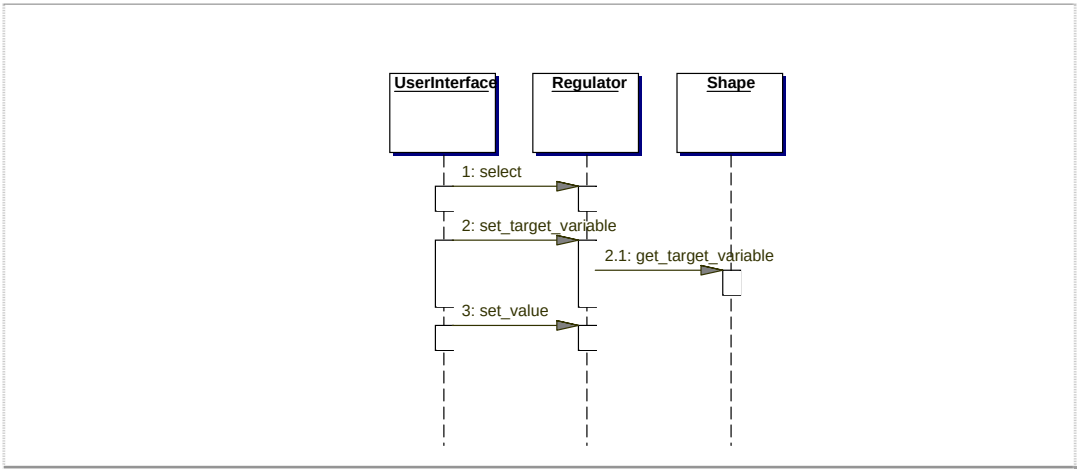
The active attribute of the selected regulators is set to true

For deactivation

The active attribute of the selected regulators is set to false

Special requirements:

When a regulator is deactivated, changes to this regulator will not affect the shape/vertices it controls; similarly changes to shapes associated though this regulator will not affect the other shapes/vertices in the association.



D.2.8.3 *Move a regulator*

Entry condition:

A regulator is selected

Flow of events:

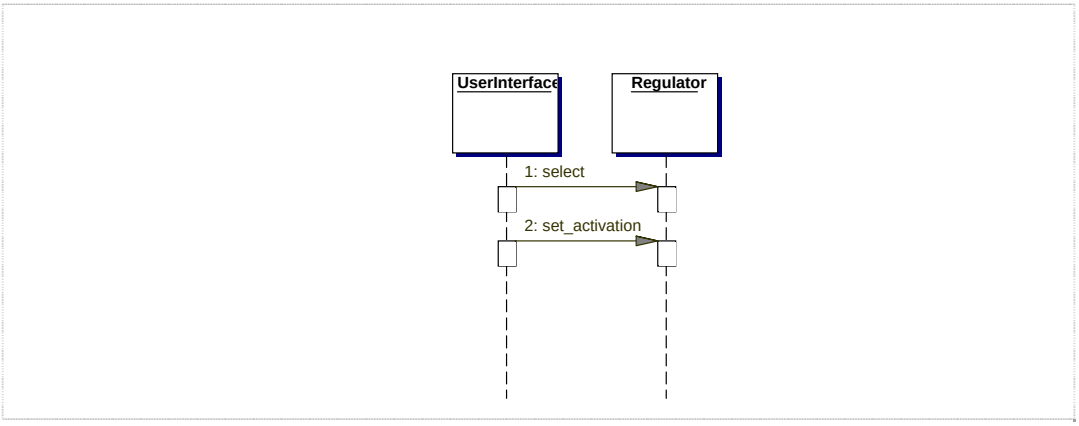
The user drags the regulator on the model view

or

The user enters the x, y, z coordinates for the regulator's position

For point regulators the system resets the point's coordinates to the new position.

For line (p+dt) and plane (p+dt+er) regulators the system resets the starting point's (p) coordinates to the new position.



D.2.8.4 *Rotate a regulator*

Entry condition:

A regulator is selected

Flow of events:

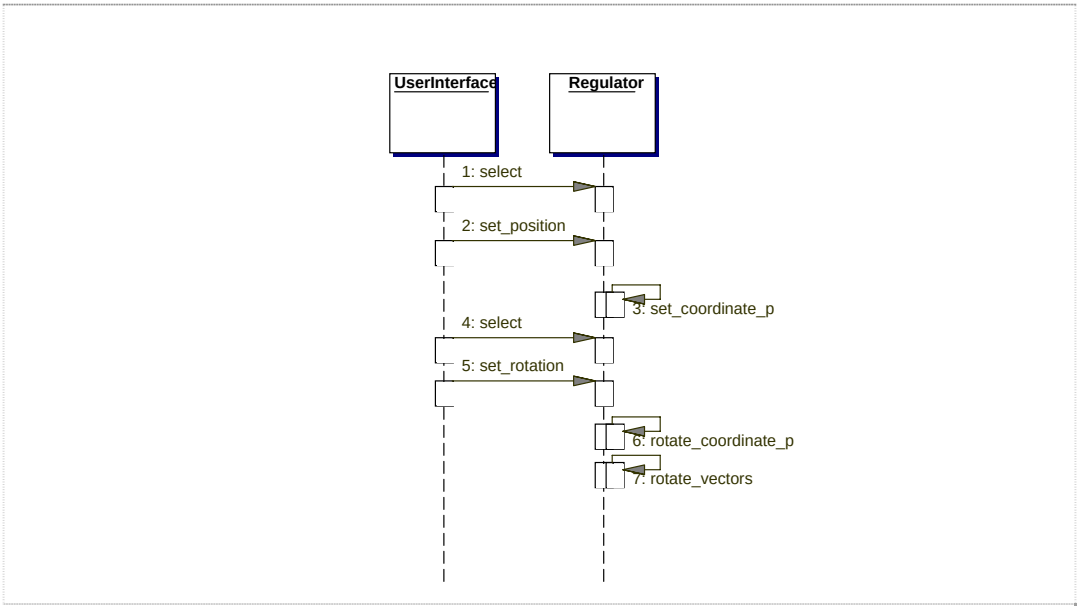
The user enters the rotation along the x, y, z axis to set the regulator's orientation

or

The user sets the arbitrary axis (x, y, z) and its rotation degree.

For point regulators the system resets the point's coordinates by multiplying the point p by the rotation matrix defined by the new user-defined orientation

For line and plane regulators the system rotates the line/plane by multiplying the starting point and the direction vector/vectors by the rotation matrix defined the orientation sliders: R(p+dt) and R(p+dt+er)



D.2.8.5 *Manipulate the regulator's endpoints*

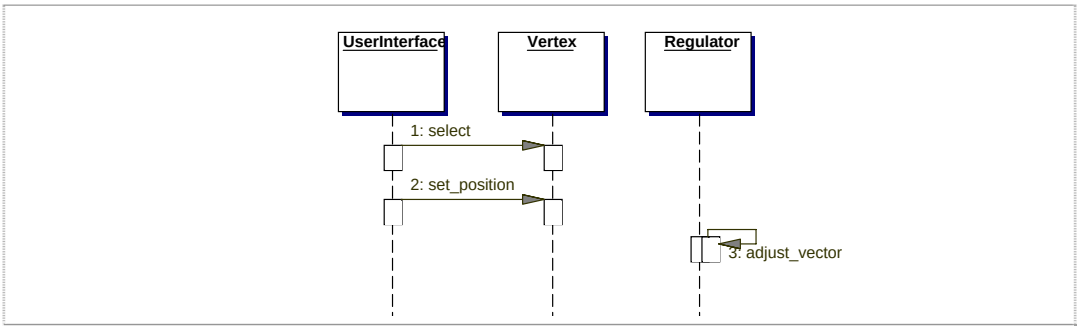
Entry condition: A regulator's endpoint is selected

Flow of events:

The user moves the endpoint (by dragging the mouse or by adjusting the coordinates values)
The system moves the endpoint to the specified position and revaluates the direction vector/s.

Special requirements:

To move an edge two endpoints must be selected



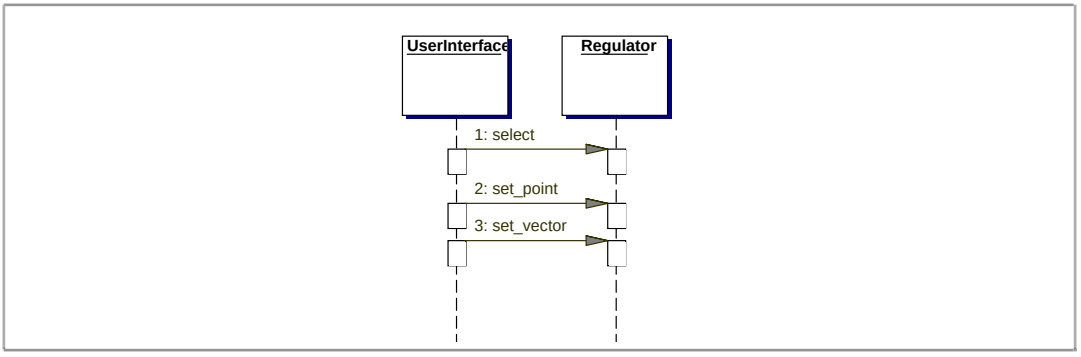
D.2.8.6 *Modify the direction vectors defining the regulators*

Entry condition: A regulator is selected

Flow of events:

The user enters the coordinates for
- the start point
- the line vector
- the plane vectors
- the volume vectors

The system updates the regulator definition and displays the new regulator in the model view



D.2.8.7 *Set (add/remove) an identity to a regulator*

Entry condition: The regulator controls are available

Flow of events:

The user selects a regulator

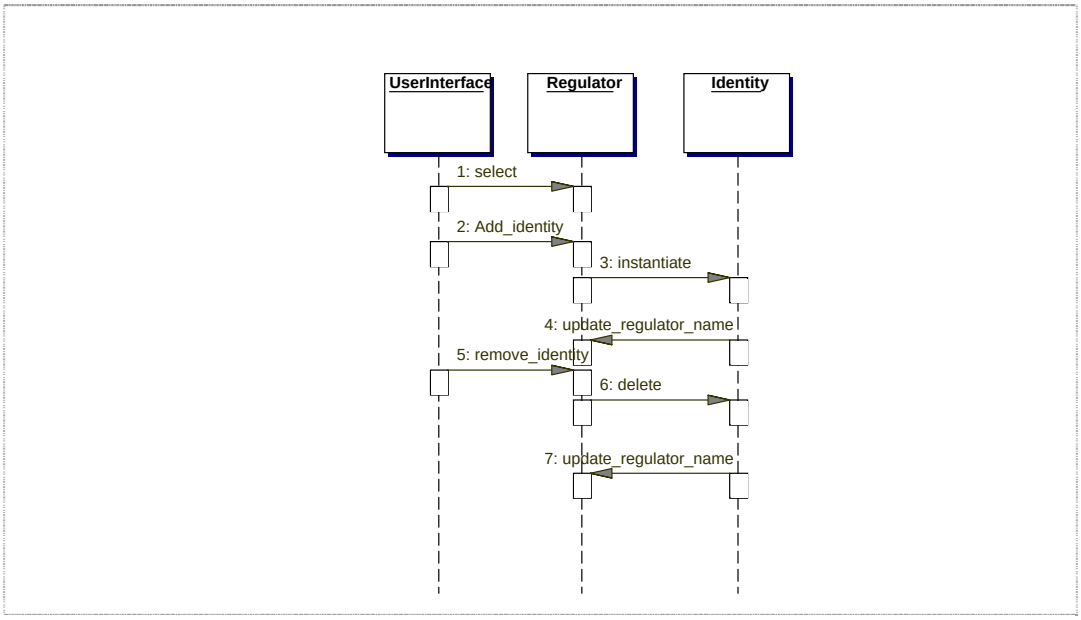
The user chooses the desired identity (turns on or off)

The system adds/removes the chosen identity into/form the regulator's identity list

The system updates the name of the regulator

Special requirements:

Each regulator may have one instance of a specific identity



D.2.8.8 *Set the dimension of the regulator*

Entry condition: The regulator controls are available

Flow of events:

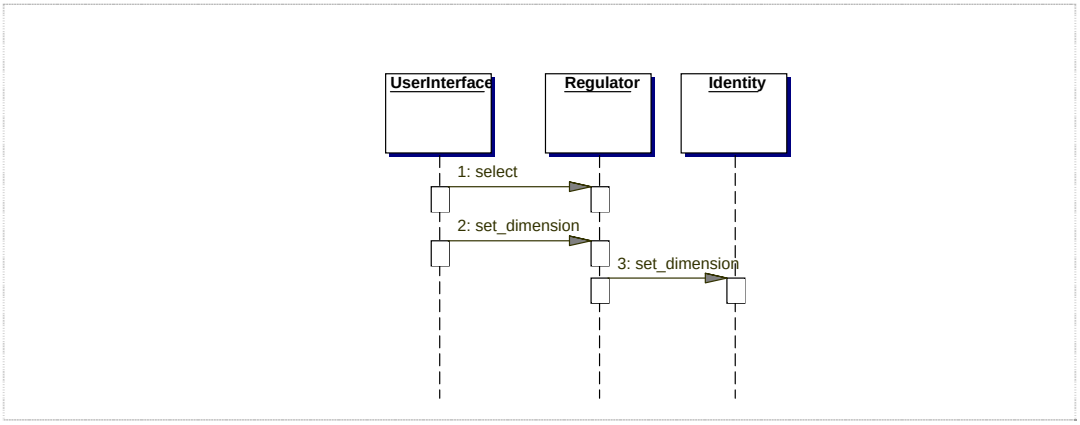
The user selects a regulator

The user chooses the desired dimension (point, line, plane, or circle if applicable)

The system the regulator's dimension

Special requirements:

Each regulator may have one instance of a specific identity



D.2.8.9 *Modify the major parameter for regulators*

Entry condition: A regulator is selected

Flow of events:

The user adjusts the transformation factor of the regulator

translation distance
rotation degree
curve factors
dilation factors
shear factor

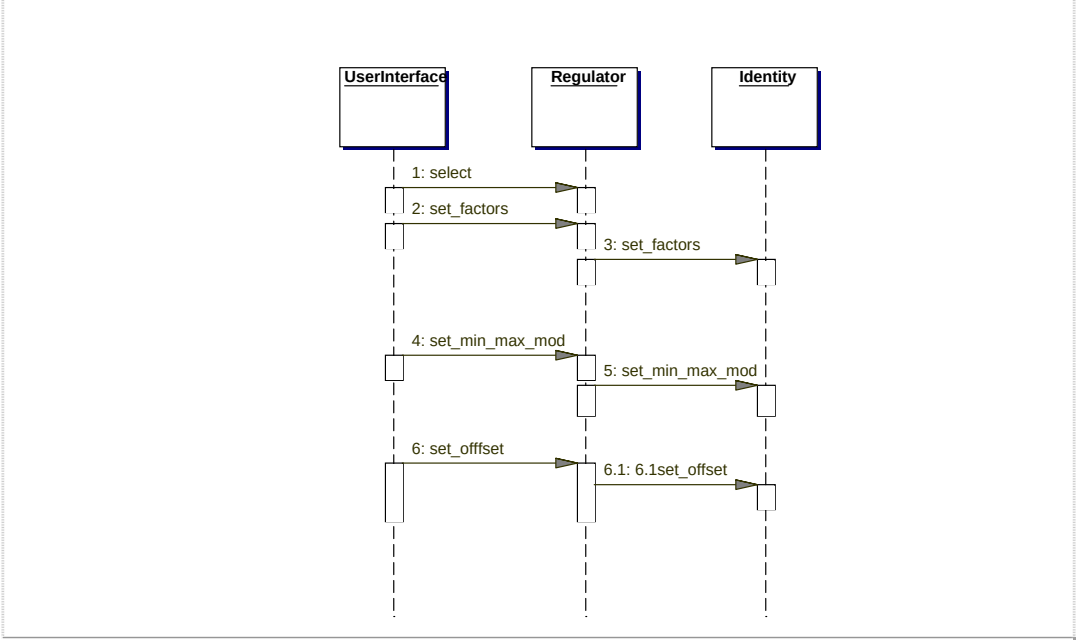
The user adjusts the min and max or module values for the parameters or the constraint regulators.
ANGLE/ DISTANCE/ DIMENSION/ AREA-VOLUME/ OVERLAP

The user adjusts the attribute, types and cycles values for the variation regulators.

The system set the transformation parameters

Special requirements:

Note: reflection has no specific parameters other than geometry



D.2.8.10 *Insert a regulator in a sequence*

Entry condition: A regulator is selected

Flow of events:

The user invokes the regulator controls

The user chooses to insert a regulator

The system creates a new regulator

The system inserts the new regulator in the regulator tree (before the selected regulator) by updating the pointers

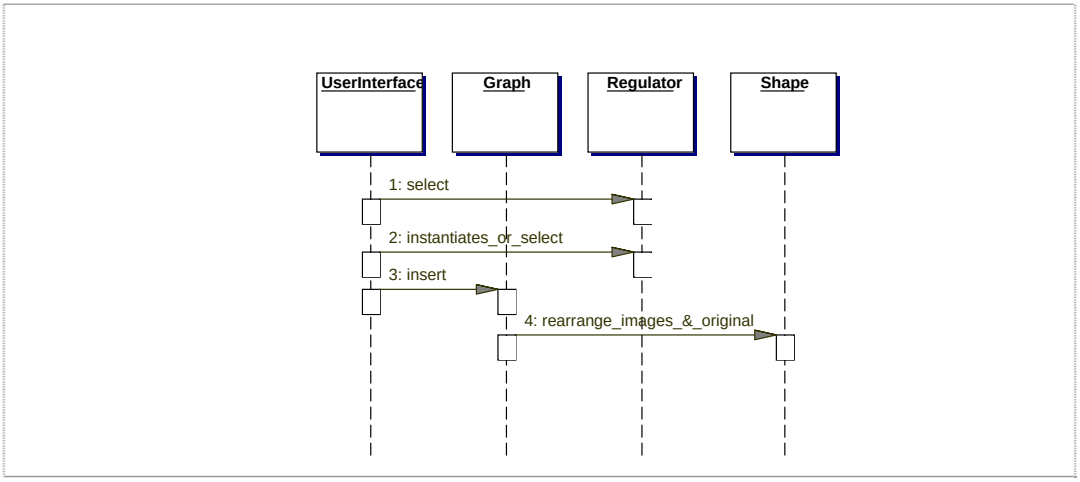
The original shapes of the succeeding regulator become the original shapes of the new regulator

The original and images shapes of the new regulator become the original shapes of the succeeding regulator

The regulator tree is re-generated (starting from the new regulator)

The ***user sets the identity*** of the new regulator

The ***user sets the parameters*** of new the regulator



D.2.8.11 *Delete a regulator from a sequence*

Entry condition:

A regulator is selected

Flow of events:

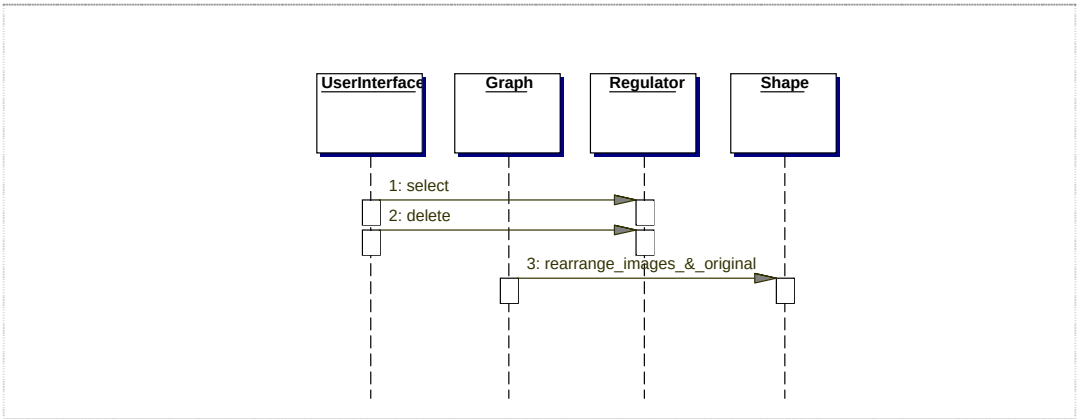
The user chooses to delete it

The system deletes the regulator from the regulator tree updating the pointers

The original shapes of this regulator become the original shapes of the succeeding regulator

The images shapes of this regulator are deleted

The regulator tree is re-generated (starting from the deleted regulator)



D.2.8.12 *Swap the order of two regulators*

Actors: Developer

Entry condition:

Two regulators are selected

The regulator controls are available

Flow of events:

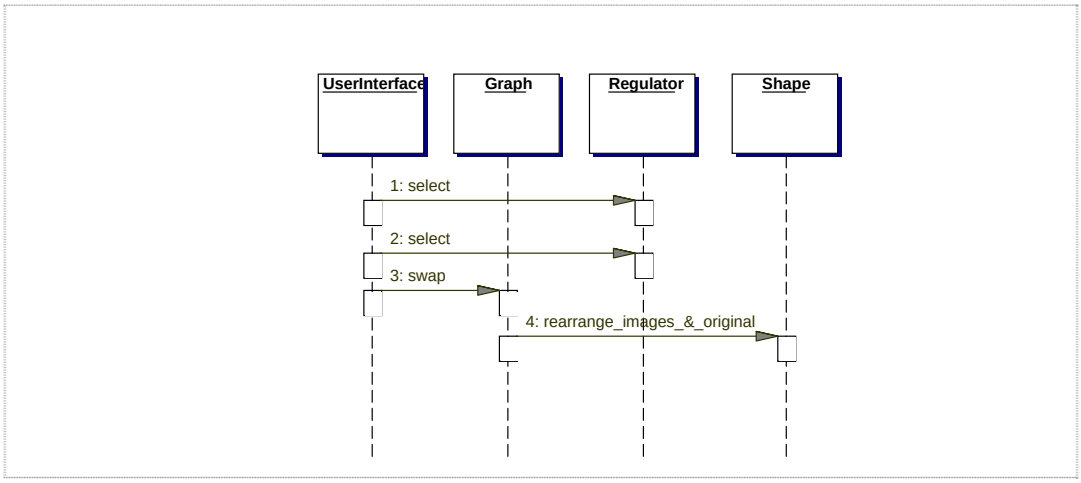
The user chooses to swap the order of these two regulators

The system swaps the order of the two regulators in the regulator tree by updating pointers

The original shapes of the first regulator become the original shapes of the second regulator

The original and image shapes of the second regulator will become the original shapes of the first regulator

The regulator tree is re-generated (starting from the swapped regulators)



D.2.8.13 *Subdivide a regulator*

Entry condition:

A regulator is selected

The regulator controls are available

Flow of events: The user chooses to subdivide the regulator

The system generates a subdivision regulator

The initial–selected regulator is set as the original of the subdivision regulator (and it is rendered invisible).

The user enters the number of subdivisions (let say k)

The user chooses by factor or by number

The user chooses whether the subdivisions are to be connected or independent shapes.

The user can also set the formula for the subdivision regulator (the default is a form in the direction of the selected regulator)

The system generate k new regulators (the images for the subdivision regulator)

The system divides the factor/number of the selected regulator by k

The systems sets the factor/number for each new k regulators

The system sets the differential factor for the new regulators based on the formula of the subdivision regulator.

Each new regulator will have its set of output shapes.

If connected, each new regulator will have "as the original shape" (the last shape generated of the preceding regulator in the subdivision)

If the selected regulator was in a sequence, (it is an insert)

The system replaces the initial regulator with the subdivision regulator in the sequence.

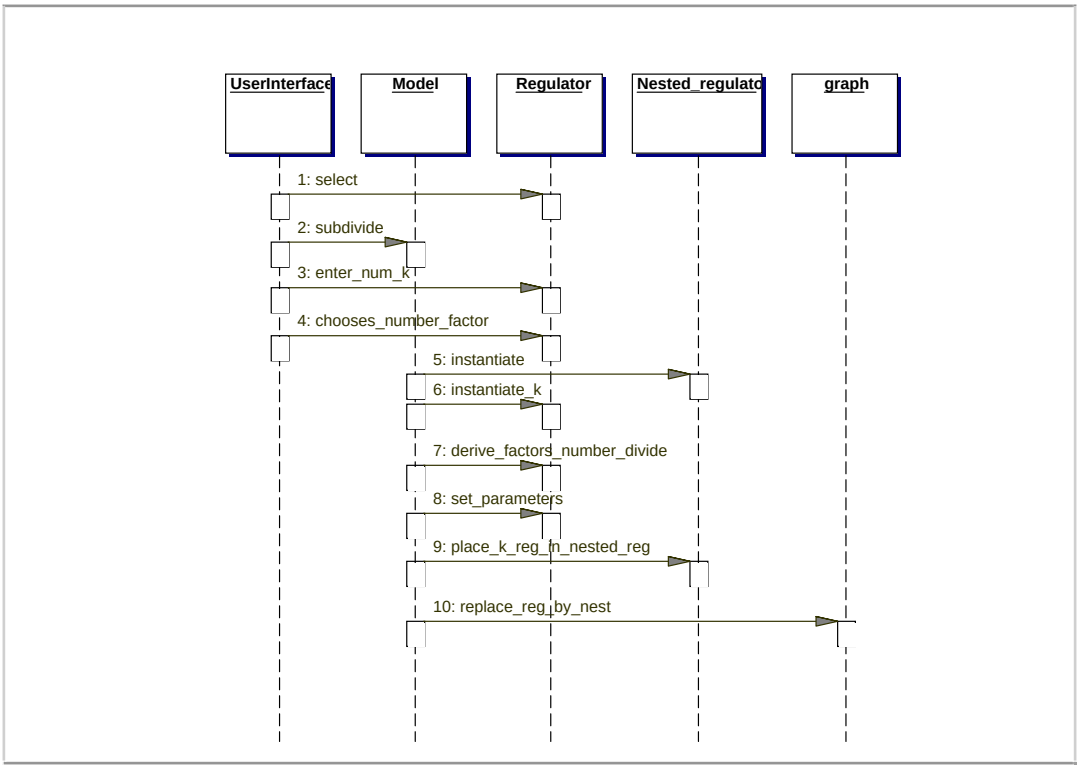
The output shapes of all the new regulators will become originals for the succeeding regulators

Special requirements:

Subdivision applies to all factors of the active identities of the initial regulators and all numbers in the regulator's associations.

Merging is achieved by deleting the subdivision regulator. (The original is restored)

Number, formula, connection-flag, and subdivision type can be modified anytime.


UserInterface
Model
Regulator
Nested_regulato
graph
1: select
2: subdivide
3: enter_num_k
4: chooses_number_factor
5: instantiate
6: instantiate_k
7: derive_factors_number_divide
8: set_parameters
9: place_k_reg_in_nested_reg
10: replace_reg_by_nest