Theory of Design

Michael Leyton

Center for Discrete Mathematics & Theoretical Computer Science (DIMACS),
Busch Campus, Rutgers University, New Brunswick, NJ 08904, USA.
mleyton@dimacs.rutgers.edu

Abstract

This paper gives an introduction to some of the basic concepts in the book, *A Generative Theory of Shape* (Michael Leyton, Springer-Verlag, 2001). The book develops a new mathematical theory of design, and supports this theory with lengthy analyses of mechanical CAD/CAM, architectural CAD, solid modeling, kinematics, etc., as well as the principal areas of perception, such as visual grouping. Of central concern is how complex design tasks are made *understandable* and structured *intelligently*. For this, new classes of algebraic structures are invented that embody understandability and intelligence. These structures are called unfolding groups. They unfold structure from a maximally collapsed version of that structure, in such a way that exploits transfer of existing structure and recoverability of generative operations. A principal aspect of the theory is that it develops an algebraic formalization of object-orientedness. The result is what we believe to be the first object-oriented theory of geometry.

Keywords: Shape, Geometry, Generative, Design, Robotics, Group theory, Symmetry, Wreath products.

1 Introduction

This paper gives an introduction to some of the basic concepts in the book *A Generative Theory of Shape* (Michael Leyton, Springer-Verlag, 2001). The book develops a new mathematical theory of design, and supports this theory with lengthy chapters on mechanical CAD/CAM, architectural CAD, solid modeling, etc. For instance, it gives an extensive mathematical theory of the main stages of MCAD/CAM: part-design, assembly and machining. And within part-design, it gives an algebraic analysis of sketching, alignment, dimensioning, resolution, editing, sweeping, feature-addition, and intent-management. Related also to CAD, the book develops an algebraic theory of robot manipulators and relative motion (for perception, animation and physics).

The central concern of the book is the analysis of *intelligent* shape-generation. Design should proceed intelligently, and software should support the intelligent needs of the designer as well as the moments of generative insight.

At the very basis of the theory, we define two criteria as fundamental to intelligent design:

(1) Maximization of transfer.
(2) Maximization of recoverability.

These two criteria will be introduced in the following two sections.

2 Maximization of Transfer

Superficially, design seems to involve the successive addition of structure. How else can one account for the fact that the design object appears to structurally grow? However, a careful analysis of how designers work, reveals that there is actually very little new structure added, as design proceeds. What actually happens is that designers tend to create each additional structural component as a *transfer* of an already existing component.

To illustrate, consider the way in which an architect draws the plan of an apartment building which will have studios, one-bedroom apartments and two-bedroom apartments. The following is the typical scenario, as found, for example, in a training manual for AutoCAD [24]. First the architect will generate the studio by copying and offsetting lines. For example, a wall can be created by drawing an initial line, and offsetting it some small parallel distance. Notice that an offset is the *transfer* of the initial line. Next, to create the opposite wall of the room, the architect copies this pair of lines *as a single unit* to the other position. That is, he *transfers the transfer*. When the drawing of the studio is complete, it is then saved in its own computer file. Next, rather than drawing the one-bedroom apartment from scratch, the architect takes the drawing for the studio apartment and modifies it until he obtains the one-bedroom apartment. This will involve copying the single room defining the studio, to make two rooms (living room and bedroom), adjusting the individual walls, copying and rearranging the closets, moving the kitchen units, etc. The drawing of the one-bedroom apartment is then saved in its own computer file. Finally, the two-bedroom apartment is created by applying the same kind of process to the one-bedroom apartment.

Thus, at all levels, the architect is essentially transferring existing structure. Remarkably, considering the fact that the studio was itself created merely by offsets of lines, the entire process of creating the successive apartments takes place by **transfer of transfer**.

We shall say that design exploits the following basic principle:

MAXIMIZATION OF TRANSFER. *Alternative statements of the principle:*

(1) Make one part of the generative sequence a transfer of another part of the generative sequence, whenever possible.

(2) Exploit existing structure rather than create new structure.

(3) Maximize re-usability.

A substantial part of our theory of intelligent design will be to develop an algebraic theory of transfer. For this, we invent a new class of (mathematical) groups called **unfolding groups**. As an example, such groups will *unfold* the studio apartment from a minimal set of primitive elements, then unfold the one-bedroom apartment from the studio, and then unfold the two-bedroom apartment from the one-bedroom apartment, etc. An unfolding group is a group that consists of a subgroup we will call an *alignment kernel* that expresses the minimal structural elements needed in the design. The remainder of the unfolding group consists of subgroups that will unfold the full complex structure outward from the alignment kernel, by *hierarchical transfer of transfer*.

3 Maximization of Recoverability

A generative theory of shape represents a given data set by a sequence of operations that generates the set. This sequence of operations must be *inferrable* from the set. We shall say that the operations are *recoverable* from the data set. It will be seen that recoverability of the generative sequence places strong constraints on the inference rules by which recovery takes place, and on the generative sequences that can be inferred. This, in turn, produces a theory of geometry that is very different from the current theories of geometry.

Essentially, the recoverability of generative operations from the data set means that the shape acts as a memory store for the operations. More strongly, we will argue that all memory storage takes place via geometry. In fact, a fundamental proposal of our theory is this:

$$\textbf{Geometry} \quad \equiv \quad \textbf{Memory Storage.}$$

As we shall see, this theory of geometry is fundamentally opposite to that of Klein's Erlanger Program, which has dominated most of 20th century geometry and physics. In Klein's program, geometric objects are defined as invariant under actions. However, if an object is invariant under actions, the actions are not recoverable from the object. Therefore Klein's theory of geometry concerns *memorylessness*, and ours concerns *memory retention*. We argue that the latter leads to a far more powerful theory of shape.

4 Complex Shape Generation

The primary goal of the book is to handle *complex* shape. Modern design software is used to generate enormously complex structures, e.g., the design of an aircraft can consist of several million objects. The design team is therefore faced with an overwhelmingly complex task that must be converted into an entirely understandable form. This exemplifies the general problem that we will investigate:

(1) The conversion of complexity into understandability: *The basic purpose is to give a generative theory of complex structure such that the complexity is entirely accounted for, and yet the structure is completely understandable.*

(2) Understandability and intelligence: *We shall see that understandability of a structure is achieved by maximizing transfer and recoverability.*

(3) The mathematics of understandability: *A significant portion of the book is the development of a mathematical theory of how understandability is created in a structure.*

5 Object-Oriented Theory of Geometry

Object-oriented programming is basic to geometric applications of computers, e.g., computer-aided design, solid modeling, kinematic simulation, etc. One of the general motivations for object-orientedness is *re-usability*; e.g., the decomposition of software into re-usable components called objects (classes), the decomposition of an object into data and re-usable actions on data, the passing of re-usable properties from parent to child in an inheritance hierarchy, etc.

According to section 2, our algebraic theory of transfer is an algebraic theory of re-usability. We will show, therefore, that the algebraic theory of transfer gives a powerful mathematical formalization of object-oriented programming.

Most crucially, we will be able to give a new theory of geometry that is object-oriented. To our knowledge, this is the first object-oriented theory of geometry to have been developed.

6 Transfer

Our two basic principles of intelligent design are maximization of transfer and maximization of recoverability. We will now begin our theory of transfer, and later go onto the theory of recoverability, showing how they inter-relate in the algebraic structure.

It will be argued that the appropriate formulation of transfer is as follows: A situation of transfer involves two levels: a *fiber group*, which is the group of actions to be transferred; and a *control group*, which is the group of actions that will transfer the fiber group. The justification for these structures algebraically being groups will be given later, but the theory of transfer will work equally for semi-groups, which is the most general case one would need to consider for generativity.

Now, one can think of transfer as the control group moving the fiber group around some space; i.e., *transferring* it. This is illustrated in Fig 1. The transferred versions of the fiber group are shown as the vertical copies, and will be called the *fiber-group copies*. The control group acts from above, and transfers the fiber-group copies onto

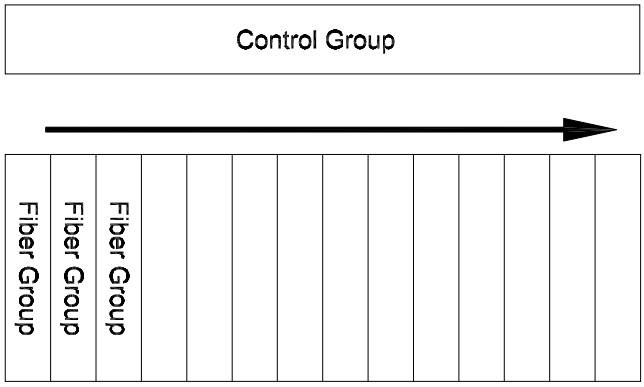

Figure 1: The control group transferring the fiber group.

each other, as indicated by the arrow. This will often be referred to as a structure of *nested control*.

A basic part of our approach is this:

(1) Give an algebraic theory of transfer.

(2) Reduce complex situations down to structures of transfer.

Transfer will be modeled by a group-theoretic construct called a *wreath product*, to be explained. Intuitively, a wreath-product is a group that contains the entire structure shown in Fig 1; that is, it has an upper subgroup that will be called a *control group*, and a system of lower subgroups that will be called the *fiber-group copies*. The control group sends the fiber-group copies onto each other. It does so simply by conjugating them onto each other.

The entire group, the wreath product, will be notated in the following way:

$$\text{Fiber Group} \ \textcircled{w} \ \text{Control Group}.$$

It should be emphasized that the upper group, the control group, is shown to the right of the lower group, the fiber group.

Formulating transfer in terms of wreath products has enormous advantages, as follows: (1) A wreath product is a group that contains all the transferred versions of the fiber group. Thus, rather than thinking of the transferred versions as separate algebraic entities, they are all integrated within a single algebraic structure. (2) This single algebraic structure also encompasses the control group. Thus the wreath product contains the network of algebraic connectivity that relates the control group to the fiber group. This algebraic connectivity will explain an enormous amount in design.

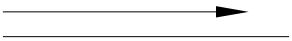

Figure 2: The generation of a side, using translations.

7 Initial Examples

Although we will look at highly complex shape later, it is best to begin with a simple example of how wreath products can be used to model transfer: Let us consider the generation of a square. The primary scenario for drawing a square, on a sheet of paper, is to draw the sides in sequence around the figure. The intelligent thing is to notice that this scenario can be broken down into one of transfer, as follows:

The first side is generated by starting with a corner point, and applying translations to trace out the side, as shown in Fig 2. Next, this translational structure is *transferred* from one side to the next - rotationally around the square. In other words, there is transfer of translations by rotations. This is illustrated in Fig 3.

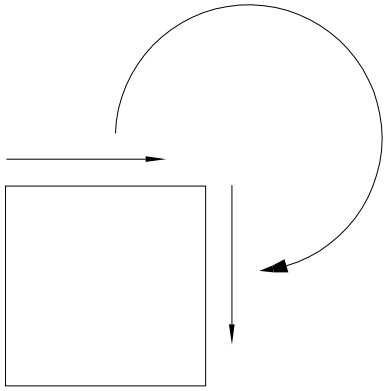

Figure 3: Transfer of translation by rotation.

Therefore, we argue that the transfer structure is defined as the wreath product:

$$\text{Translations} \wr \text{Rotations}$$

where Translations is the fiber group and Rotations is the control group. Recall that, in any transfer situation, the control group moves the fiber group around.

The translation group will be denoted by the additive group $\mathbb{R}$. The rotation group is $\mathbb{Z}_4$, the cyclic group of order 4, represented as

$$\mathbb{Z}_4 \;=\; \{\; e, \quad r_{90}, \quad r_{180}, \quad r_{270} \;\}$$

where r_θ means clockwise rotation by θ degrees. Thus the transfer structure illustrated in Fig 3, is this:

$$\mathbb{R} \;\text{ⓦ}\; \mathbb{Z}_4. \tag{1}$$

Now observe that the group at (1) gives **generative coordinates** to the square, as follows. Any point on the square can be described by a pair of coordinates:

$$(t, r) \;\; \in \;\; \mathbb{R} \;\text{ⓦ}\; \mathbb{Z}_4$$

where $t \in \mathbb{R}$ and $r \in \mathbb{Z}_4$. The first coordinate gives the generative (translational) distance along a side, and the second coordinate gives the generative (rotational) distance of a side from the first generated side. Therefore a point is given a complete generative description from the origin.

Fig 4 illustrates this by giving the coordinates of four of the points.

The crucial thing to observe is that the coordinates *maximize transfer*. Fig 5 illustrates this by showing that the coordinates on one side are a transfer of the coordinates of another side.

Now, *deformed* shapes are handled in our system by adding extra layers of transfer. For example, to obtain a parallelogram, one adds the general linear group $GL(2, \mathbb{R})$ onto the two-level group of the square thus:

$$\mathbb{R} \;\text{ⓦ}\; \mathbb{Z}_4 \;\text{ⓦ}\; GL(2, \mathbb{R}). \tag{2}$$

Notice that the operation used to add $GL(2, \mathbb{R})$ on to the lower structure $\mathbb{R}\text{ⓦ}\mathbb{Z}_4$ is, once again, the wreath-product ⓦ which means that $GL(2, \mathbb{R})$ acts by *transferring* $\mathbb{R}\text{ⓦ}\mathbb{Z}_4$, as follows: Since the fiber group $\mathbb{R}\text{ⓦ}\mathbb{Z}_4$ represents the structure of the square, this means that $GL(2, \mathbb{R})$ transfers the structure of the square onto the parallelogram. In particular, it transfers the *generative coordinates* of the square onto the parallelogram. For example, $GL(2, \mathbb{R})$ transfers the four points on the square in Fig 4 onto the corresponding four points on the parallelogram, as shown in Fig 6.

More deeply still, the fiber group $\mathbb{R}\text{ⓦ}\mathbb{Z}_4$ in expression (2) is itself a transfer structure, as seen in Fig 5, where rotation transferred the translation process from the top side onto the right side. This transfer structure is itself transferred, by $GL(2, \mathbb{R})$, onto the parallelogram, as shown in Fig 7. That is, we have **transfer of transfer**. This recursive transfer is encoded by the successive ⓦ operations in expression (2).

The theory is equally applicable to 3-dimensional shape. For example, consider the structure of a cylinder. In computer vision and graphics, cylinders are described generatively as the sweeping of the circular cross-section along the axis, as shown in Fig 8. To our knowledge, the group of this sweeping structure has never been given. We propose that the appropriate group is:

$$SO(2) \;\text{ⓦ}\; \mathbb{R}. \tag{3}$$

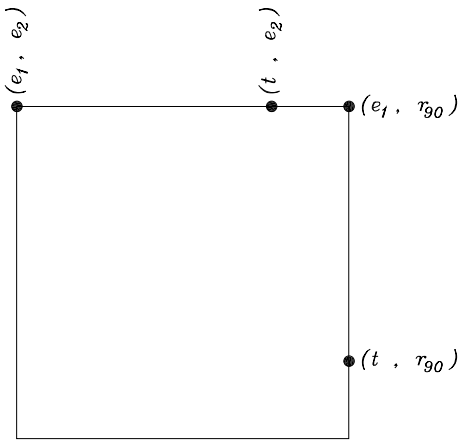

Figure 4: The coordinates of four points.

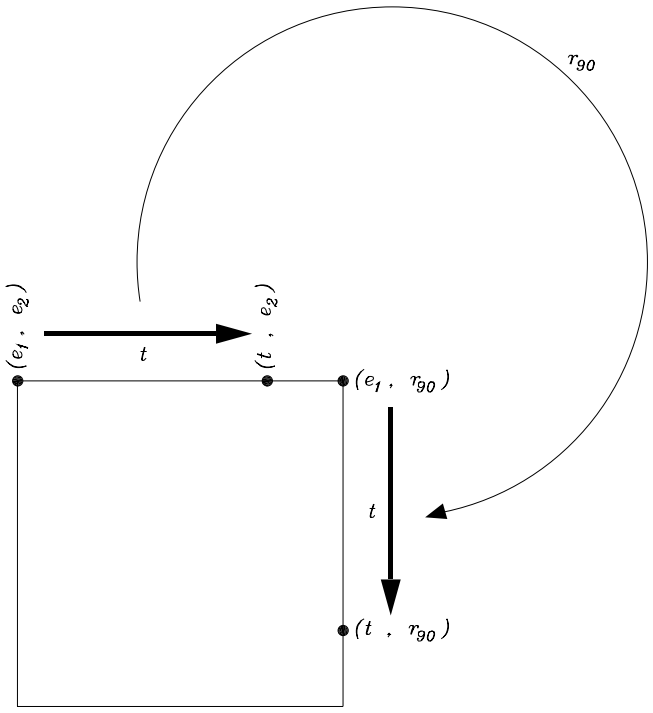

Figure 5: The control-nested structure of those coordinates.

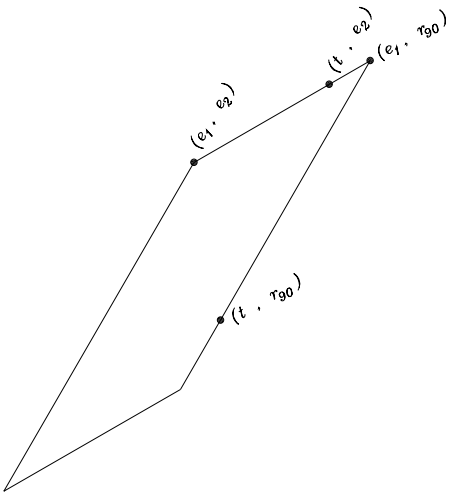

Figure 6: The transferred coordinates from a square.

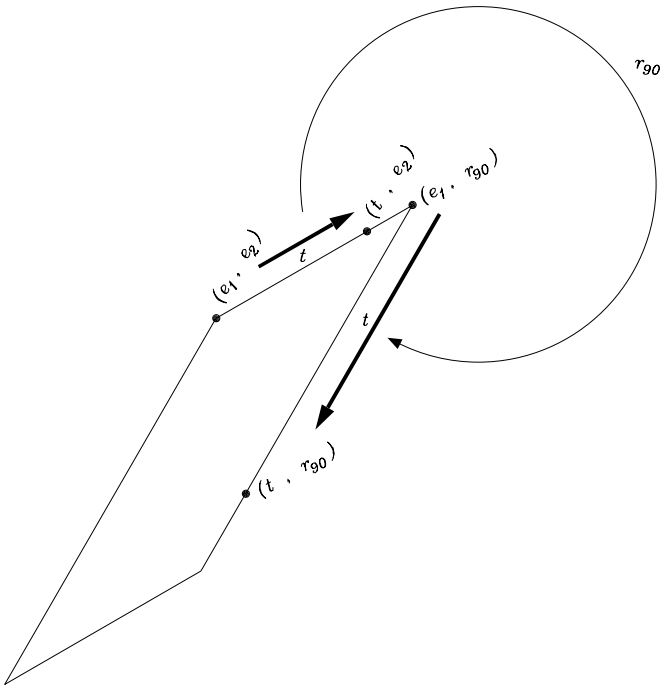

Figure 7: The transfer of transfer.

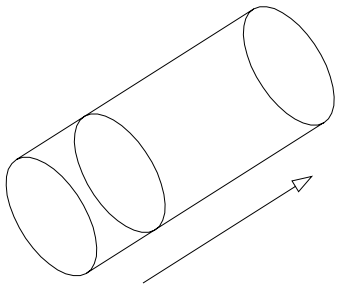

Figure 8: The sweep structure of a cylinder.

Notice that it uses the wreath-product operation $\textcircled{w}$ rather than the direct product $\times$ (which is the classical definition). The operation $\textcircled{w}$ means that this new group has a *fiber-control* structure, in which $SO(2)$ is the fiber group and $\mathbb{R}$ is the control group. This is exactly what is seen in the sweeping structure shown in Fig 8. The cross-section is generated first as a fiber, and then its position is controlled by translation.

Now consider deformations of the cylinder. According to our theory, these are obtained merely by adding a deforming group, e.g., a group of spline tensors, $\mathcal{H}$, onto the group of the cylinder, as a higher level of transfer, thus:

$$SO(2) \textcircled{w} \mathbb{R} \textcircled{w} \mathcal{H}. \tag{4}$$

The operation used to add $\mathcal{H}$ onto the lower structure $SO(2)\textcircled{w}\mathbb{R}$ is, once again, the wreath-product $\textcircled{w}$. This means that the structure of the straight cylinder is transferred onto the structure of the bent cylinder. It is this transfer that allows us to see the bent cylinder as a distorted version of the straight one, i.e., ensuring recoverability.

As another example of this approach to deformation, consider the projectively distorted grid of squares in Fig 9. We propose that the undistorted grid of squares is generated by the wreath product:

$$\mathbb{R} \textcircled{w} \mathbb{Z}_4 \textcircled{w} \mathbb{Z}^H \textcircled{w} \mathbb{Z}^V$$

where the first two levels $\mathbb{R}\textcircled{w}\mathbb{Z}_4$ is the square, as before; and the next level $\mathbb{Z}^H$ is the group of horizontal translations; and finally $\mathbb{Z}^V$ is the group of vertical translations ("vertical" and "horizontal" mean intrinsic to the grid). Then, to get the projective distortion, one merely adds the projective group as an extra level of control thus:

$$\mathbb{R} \textcircled{w} \mathbb{Z}_4 \textcircled{w} \mathbb{Z}^H \textcircled{w} \mathbb{Z}^V \textcircled{w} PGL(3, \mathbb{R}). \tag{5}$$

Thus the projective group *transfers* the undistorted grid onto the distorted one; i.e., projection is described as transfer. The important thing to notice is that the structure is

exhaustively generative, from the lowest to the highest level, and that this generativity has been entirely described as a hierarchy of transfer.

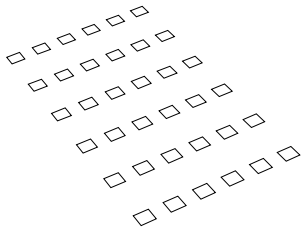

Figure 9: A grid of squares distorted by projection.

8 The Designer's Detection of Transfer

Now, in the design process, the designer must *detect* the fact that parts of the generative sequence can be transferred onto other parts of the generative sequence. Our theory allows us to algebraically formulate the designer's detection of transfer.

Let us illustrate this first with a square. Initially, the designer notices that each side can individually be generated by applying the translation group $\mathbb{R}$. However, these detections are in *parallel*, because the designer has not yet noticed that the sides can be transferred to each other by rotation. Therefore the four translational groups $\mathbb{R}$ are initially independent. We will model this independence, or parallelism, by saying that the four groups have a *direct product* relation thus:

$$\mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}.$$

Next, the designer *notices* that the four groups $\mathbb{R}$ can be *rotated* onto each other. We will model this *detection* by adding $\mathbb{Z}_4$ onto the direct product via a *splitting extension*, thus:

$$[\mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}] \, \textcircled{S} \, [\mathbb{Z}_4].$$

The splitting extension symbol $\textcircled{S}$ means that $\mathbb{Z}_4$ acts as a symmetry group on the group to its left, i.e., on the product of fiber groups $\mathbb{R}$. The particular symmetry action we choose here is to have $\mathbb{Z}_4$ rotate the fiber groups $\mathbb{R}$ onto each other. Therefore:

> **Detection of transfer is modeled by a splitting extension that identifies a symmetry in a direct product below**.

Some terminological comments: The full group,

$$[\mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}] \, \textcircled{S} \, [\mathbb{Z}_4] \tag{6}$$

is an example of what one calls a wreath product. It can be written either as shown in (6), where all the fiber-group copies are explicit - and the splitting extension symbol Ⓢ is used. Or it can be written like this:

$$\mathbb{R} \, \textcircled{W} \, \mathbb{Z}_4$$

where only a single fiber group is shown, and one uses the wreath-product symbol Ⓦ. The two notations mean exactly the same thing.

9 Shape Generation by Group Extensions

One can see from the above discussion that the concept of *group extension* is basic to our generative theory. A group extension takes a group G_1 and adds to it a second group G_2 to produce a third, more encompassing, group G, thus:

$$G_1 \, \textcircled{E} \, G_2 \;\; = \;\; G$$

where Ⓔ is the extension operation. See our book on group extensions, Leyton [22].

It is clear, looking back over the examples given so far, that according to our theory:

Shape generation proceeds by a sequence of group extensions.

That is, shape generation starts with a base group and successively adds groups obtaining a structure of this form:

$$G_1 \, \textcircled{E} \, G_2 \, \textcircled{E} \, \ldots \textcircled{E} \, G_n.$$

This approach to shape-generation differs substantially from standard shape-grammar approaches, which are based on the application of production rules. In our approach, structural elements correspond to groups, and the addition of structural elements corresponds to group extensions.

Structural elements $\longrightarrow$ Groups.

Addition of structural elements $\longrightarrow$ Group extensions.

Furthermore, imposing the condition of *maximization of transfer* demands that the structure $G_1 \textcircled{E} G_2 \textcircled{E} \ldots \textcircled{E} G_n$ is a wreath product.

10 Recoverability

Recall that this generative theory of shape is founded on two principles of intelligence: maximization of transfer, and maximization of recoverability. The previous sections began to examine transfer, and it is now necessary to bring in recoverability. By *recovery*, we mean the following problem:

> **Given a data set, recover or infer a sequence of operations that generate the set.**

Our previous book [19] was a 600-page analysis of this problem, and one of the main conclusions of this analysis was the following:

ASYMMETRY PRINCIPLE. *The only recoverable operations are symmetry-breaking ones. That is, a generative program is recoverable only if it is symmetry-breaking on each of the successively generated states.*

In order to build a theory of design, it is worth considering a psychological example of the above principle. It is extremely useful, when viewing a design, to understand how the human visual system recovers from it the design operations that were used in creating it. Psychological results, such as the following, give insight into this: In a series of experiments [16] [17], we found that, when subjects are presented with a parallelogram oriented in the picture plane as shown in Fig 10a, they see it as a rotated version of the parallelogram in Fig 10b, which they then see as a sheared version of the rectangle in Fig 10c, which they then see as a stretched version of the square in Fig 10d. The remarkable thing is that the only data that the subjects are actually given is the first figure, the rotated parallelogram. The experiments found that, on being presented with this figure, their minds recovered the generative history shown.

The most important thing is that, to carry out this recovery of successively previous states, their minds were successively removing asymmetries and recovering symmetries. Thus, subjects conjectured that the generative history was *symmetry-breaking in the forward-time direction*, from square to rotated parallelogram.

Now let us consider design, generally. We argue that the most fundamental fact about design - a fact that has (to our knowledge) never been pointed out - is that design is inherently a process of symmetry-breaking. For example, the designer in CAD starts with shape primitives that are symmetric, and successively breaks their symmetry by deformation and concatenation.

Understanding design as successive symmetry-breaking allows us to develop a deep theory of the design process, as follows.

11 Shape Primitives

It is first necessary to understand what the shape primitives are with which the design process starts. No one has precisely defined the nature of primitives, or systematically

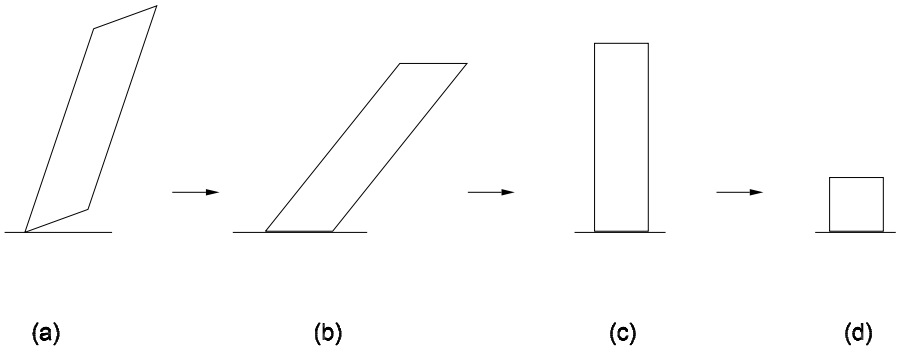

Figure 10: Psychological results found in Leyton [16] [17].

classified them. In the book, we give a rigorous theory of primitives. The basic claim is this:

STRUCTURAL AND FUNCTIONAL THEORY OF PRIMITIVES. *Primitives are determined by the maximization of transfer and the maximization of recoverability. To achieve these maximizations, they are structured by iso-regular groups.*

An iso-regular group is a structure that was invented particularly for this generative theory, and is defined thus:

ISO-REGULAR GROUP. *This is a group satisfying the following three conditions:*

(1) It is an n-fold wreath product, $G_1 \textcircled{w} G_2 \textcircled{w} \ldots \textcircled{w} G_n$.
(2) Each level G_i is either a cyclic group or a connected 1-parameter Lie group.
(3) Each level G_i is represented as an isometry group.

This section deals with the transfer aspect of primitives; and section 13 deals with the recoverability aspect.

Table 1 gives our systematic classification of surface primitives. Each is given by an iso-regular group. The top half of the table shows what we call the Level-Continuous Primitives. In these, each level is continuous. Since there are only two connected 1-parameter Lie groups, $SO(2)$ and $\mathbb{R}$, and since we want to maximize transfer, these primitives are generated simply by taking all possible 2-level wreath products using $SO(2)$ and $\mathbb{R}$.

LEVEL-CONTINUOUS	
Plane	$\mathbb{R} \,\textcircled{w}\, \mathbb{R}$
Sphere	$SO(2) \,\textcircled{w}\, SO(2)$
Cross-Section Cylinder	$SO(2) \,\textcircled{w}\, \mathbb{R}$
Ruled Cylinder	$\mathbb{R} \,\textcircled{w}\, SO(2)$
LEVEL-DISCRETE	
Cube	$\mathbb{R} \,\textcircled{w}\, \mathbb{R} \,\textcircled{w}\, \mathbb{Z}_2 \,\textcircled{w}\, \mathbb{Z}_3$
Cross-Section Block	$\mathbb{R} \,\textcircled{w}\, \mathbb{Z}_n \,\textcircled{w}\, \mathbb{R}$
Ruled or Planar-Face Block	$\mathbb{R} \,\textcircled{w}\, \mathbb{R} \,\textcircled{w}\, \mathbb{Z}_n$

Table 1: Classification of primitives by maximizing transfer and recoverability

As an example, consider what the table calls a cross-section cylinder. This was the cylinder given earlier in Fig 8, that is, the sweeping of a circular cross-section along its axis. The group we gave for that is:

$$SO(2) \,\textcircled{w}\, \mathbb{R}.$$

The table shows also an alternative generation of a cylinder, which it calls the ruled cylinder. This time, a straight line within the cylinder surface is transferred around the cross-section. The transfer again is modeled by a wreath product. This reverses the fiber-control roles of $SO(2)$ and $\mathbb{R}$ from the previous case. Therefore, the wreath-product hierarchy is this:

$$\mathbb{R} \,\textcircled{w}\, SO(2).$$

Notice now that, by the maximization of recoverability, a cone is not included because, by the removal of asymmetries, it leads back to a cylinder; i.e., it is not primitive. The relation between recoverability and primitives is given in section 13.

The lower half of the table gives what we call the level-discrete primitives. These are primitives in which one level of transfer is a discrete group. The cross-section block and ruled block correspond to the two cylinder cases just discussed, where the continuous rotation group $SO(2)$ is replaced by the discrete rotation group $\mathbb{Z}_n$, and an extra fiber level $\mathbb{R}$ is wreath sub-appended below the rotation group to correspond to the side.

The remaining entry in the table is the cube, which will be dealt with in the next section.

Clément, Rivière, & Temmerman, [2], and Srinivasan, [30] [31], have developed a different group-theoretic method for characterizing primitives. Their approach begins with the full Euclidean group and successively restricts this to subgroups until they reach subgroups corresponding to primitive surfaces. One can therefore characterize their approach as top-down. In contrast, the approach developed in this paper, being

generative, is bottom-up, successively adding 1-generator groups as layers. We believe that both the approaches have value in different circumstances.

12 Theory of Reference Objects

Reference objects are an essential part of design. For example, in MCAD, one standardly begins by selecting a construction plane from the three coordinate planes of the Cartesian frame, and one then draws a 2D sketch on the selected plane and sweeps the drawing out from that plane, to create a 3D object. Let us now understand the role of such reference objects.

First, for convenience, let us introduce the following two simple terms, which will be useful in the theory. Let us define a *process* to be an action of a cyclic or 1-parameter Lie group; and let us define a *phase* as a sequence of processes.

It is obviously the case that a reference object *begins* a phase of the shape-generation procedure. This is because it is used as a *reference* for the phase.

The crucial factor to now bring into consideration is the Asymmetry Principle which says that the generation of a shape is recoverable from a data set only if it is symmetry-breaking on successively generated states. Thus recoverable shape-generation must proceed from a symmetry ground-state to an asymmetrization of that ground-state.

Thus by using the Asymmetry Principle, we conclude that any *phase* of shape-generation must be a progression from symmetry to asymmetry. This leads us to propose the following:

THEORY OF REFERENCE OBJECTS. *Any reference object corresponds to the symmetry ground-state of a phase. When there is a sequence of phases, a reference object occurs at a phase-transition. It corresponds to the point of maximal symmetry (start point) of the new phase and the point of maximal asymmetry (end point) of the previous phase.*

Using this theory, let us now understand the role of construction planes: First observe that Table 1 gives a cube essentially as the group $\mathbb{Z}_2 \circledS{w} \mathbb{Z}_3$ where the fiber $\mathbb{Z}_2$ represents the reflectional symmetry between two opposite faces, and the control $\mathbb{Z}_3$ transfers this reflectional symmetry between the three pairs of opposite faces of the cube. In fact, for convenience, consider the slightly larger group $\mathbb{Z}_2 \circledS{w} \Sigma_3$, where Σ_3 is the full permutation group on three elements; i.e., this will permute the three pairs of opposite sides of the cube. We will call $\mathbb{Z}_2 \circledS{w} \Sigma_3$ the *hyperoctahedral wreath group*. The structure of the group is this:

$$[\mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_2] \circledS{s} \Sigma_3 \tag{7}$$

which explicitly shows the three reflection fibers $\mathbb{Z}_2$ to the left of the extension symbol Ⓢ. These three fibers correspond to the three pairs of opposite faces of the cube. The

group Σ_3 to the right of the symbol Ⓢ will transfer the three reflectional pairs onto each other.

This group is particularly important in the generative theory: It is our claim that it gives the main structure of the 3D Cartesian frame as a datum object, and that any construction plane is the selection of one of the reflection fibers. Thus, consider again the standard procedure of part-design in MCAD: The software presents the designer with the 3D Cartesian frame, as shown in Fig 11. The designer then selects one of the three planes as the construction plane, and draws a 2D sketch on the selected plane. He then sweeps the drawing out from that plane, to create a 3D object.

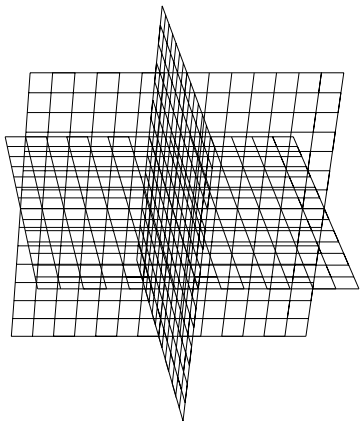

Figure 11: The designer chooses one of the 3 Cartesian planes as a construction plane.

According to our theory, what is going on here is as follows: The initial 3D Cartesian frame, i.e., with three planes, has a symmetry structure given by the hyperoctahedral wreath group, where the three planes correspond to the three reflection fibers. The designer then selects one of the three reflection fibers as construction plane, makes the drawing within the fiber, and then sweeps the fiber in the perpendicular direction. This breaks the reflectional symmetry of the fiber as well as the full hyperoctahedral wreath group. Therefore, these actions accord with our theory that reference objects are symmetry-ground states of phase transitions, and that the phases break the symmetries of those ground-states.

13 Externalization Principle

Let us now return to the problem of *recoverability*; i.e., inferring from a given data set (such as a design) the past generative operations that produced the data set. It is necessary to distinguish between two types of inference:

External inference. *The data set contains a record of only one state (e.g., only a single photograph). Therefore the "spatial arrangement" in the data set represents part of the structure of an individual state.*

Internal inference. *The data set contains records of several states (e.g., several photographs taken along the generative history). Therefore some of the "spatial arrangement" in the data set, represents structure between successive states.*

We will now give a rule that turns out to be fundamental to the entire process of recovering generative history:

EXTERNALIZATION PRINCIPLE. *In a generative sequence inferred by external inference, the inferred starting state is structured by an iso-regular group.*

The Externalization Principle gives the profound basis of CAD, as follows: Computer-aided design proceeds by a sequence of symmetry-breaking actions. The designer selects primitives which then undergo deformation and Boolean combination. In accord with the Externalization Principle, the primitives are characterized by iso-regular groups. The subsequent deformation and Boolean combination breaks the iso-regularity, as we shall now see.

Let us first illustrate the fact that Boolean combination destroys iso-regularity. The left-most diagram in Fig 12 shows two primitives, a cylinder and a cube, both of which are characterized by iso-regular groups. The three figures to the right show the effects of applying the three Boolean operations (to the primitives as solids). One can see that the resulting solid, in each of the three cases, cannot be characterized by an iso-regular group. Thus the effect of Boolean combination is to destroy iso-regularity.

Section 23 will show how to formulate Boolean combination in more detail. The next section will show how to formulate deformation.

14 The Meaning of Deformation

Standardly, in mathematics, one describes deformation as a smooth action on some object. However, for us, the fault of this definition is that it does not ensure recoverability. For example, *mathematically*, a straight sheet of paper can be obtained from a twisted sheet of paper by deformation. However, no one would describe a straight sheet of paper as a deformed object. The reason is that the deformation is not recoverable from the straight sheet. This is crucial for CAD: When presented with a design involving straight surfaces, one does not want to conjecture that they were the result of twisting. In contrast, when presented with a design such as Frank Gehry's Guggenheim Museum, one wants to conjecture that the surfaces were the result of deformation. Thus we need a definition of deformation that ensures recoverability.

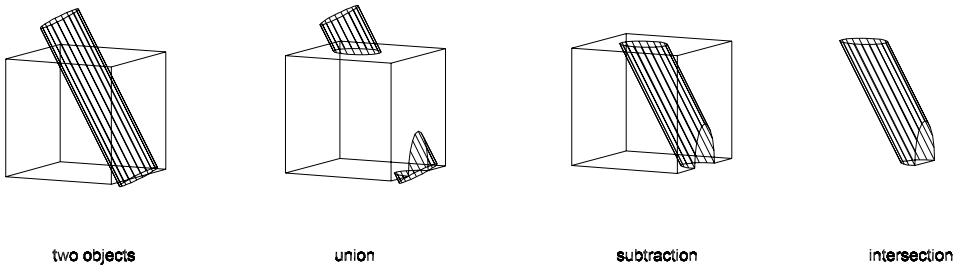

Figure 12: The Boolean operations.

Note that the basis of this must be the Asymmetry Principle (section 10), which states that the only recoverable operations are symmetry-breaking ones. In particular, to fully characterize deformation, one needs the Externalization Principle (section 13) which is the use of the Asymmetry Principle in all cases of external inference. That is, we have:

DEFORMATION. *Deformation is the smooth breaking of an iso-regular group.*

For instance, a parallelogram cannot be described by an iso-regular group. However, it is generated by breaking the iso-regular group of the square. Similarly, a bent pipe cannot be described by an iso-regular group. However, it is generated by breaking the iso-regular group of the straight pipe.

The important thing to understand now is that, in our theory, the breaking of iso-regularity is itself given by adding levels of transfer onto the existing iso-regular group. For example, recall from section 7 that a parallelogram is obtained by adding $GL(2,\mathbb{R})$ onto the iso-regular group $\mathbb{R}\circledcirc\mathbb{Z}_4$ of a square, as a higher level of transfer, thus:

$$\mathbb{R} \circledcirc \mathbb{Z}_4 \circledcirc GL(2,\mathbb{R}).$$

Notice that this group sequence is no longer an iso-regular group, since the final control group $GL(2,\mathbb{R})$ is not an isometry group.

Similarly, a deformed cylinder is obtained by adding a spline tensor group $\mathcal{H}$ onto

the iso-regular group $SO(2)\textcircled{w}\mathbb{R}$ of a straight cylinder, as a higher level of transfer, thus:

$$SO(2) \textcircled{w} \mathbb{R} \textcircled{w} \mathcal{H}.$$

Notice that this is no longer an iso-regular group, since the final control group $\mathcal{H}$ is not an isometry group.

Similarly, a projectively distorted grid of squares is obtained by adding the projective group $PGL(3,\mathbb{R})$ onto the iso-regular group $\mathbb{R}\textcircled{w}\mathbb{Z}_4\textcircled{w}\mathbb{Z}^H \textcircled{w} \mathbb{Z}^V$ of the regular grid, as a higher level of transfer, thus:

$$\mathbb{R} \textcircled{w} \mathbb{Z}_4 \textcircled{w} \mathbb{Z}^H \textcircled{w} \mathbb{Z}^V \textcircled{w} PGL(3,\mathbb{R}).$$

Notice that this is no longer an iso-regular group, since the final control group $PGL(3,\mathbb{R})$ is not an isometry group.

The important thing to observe is that the operation used to add the extra group onto the iso-regular group is, once again, the wreath-product $\textcircled{w}$. This means that the non-deformed structure is transferred onto the deformed structure. It is this transfer, and the fact that it is symmetry-breaking, that allows us to see the deformed structure as a version of the non-deformed one; i.e., ensuring recoverability.

15 Theory of Symmetry-Breaking

A basic factor emerges from the above discussion: In order to ensure recoverability, the control group must be symmetry-breaking on its fiber. Notice that this gives a far more powerful theory of symmetry-breaking than the conventional one that underlies physics and chemistry.

CONVENTIONAL VIEW OF SYMMETRY-BREAKING. *Symmetry-breaking is a reduction of symmetry group.*

Thus the transition from a square to a parallelogram is conventionally given by the following reduction in symmetry group:

$$D_4 \longrightarrow \mathbb{Z}_2.$$

However, according to our view, this is inherently weak because it means a loss of algebraic structure. In our approach, symmetry-breaking actually preserves the original group. The breaking of a symmetry group G_1 is carried out by extending G_1 by another symmetry group G_2 via a wreath product thus: $G_1\textcircled{w}G_2$. The original symmetry is given by the fiber copy of G_1 which corresponds to the identity element in the control group G_2. Non-identity elements in G_2 break the symmetry of the fiber group. Most crucially, in our view, symmetry-breaking corresponds to an increase in symmetry group!

NEW VIEW OF SYMMETRY-BREAKING. *Symmetry-breaking is extension via a wreath product. The extending group is the symmetry group of the asymmetrizing action.*

16 Algebraic Theory of Inheritance

We shall now go more deeply into the process of design. For this, it is necessary to introduce some of our algebraic theory of inheritance.

The term *inheritance,* in object-oriented programming, refers to the passing of properties from a parent to a child, [23]. The child incorporates these parent properties, but also adds its own.

This kind of structure covers two types of situation. The first is class inheritance, which is a static software concept, and the second is a type of dynamic linking created at run-time. The book gives an algebraic theory of both types of inheritance, but we will have time to deal here with only the latter.

This *run-time inheritance* is fundamental to all computer-aided design, assembly, robotics, animation, etc. A typical example is a child object inheriting the transform of a parent object, and adding its own. For example, in architectural CAD, a door is defined as a child of a wall, and moves with the wall if the designer decides to change the position of the wall. However, the door can also open and close with respect to its attached position in the wall. This means that the door inherits the movement of the wall, but adds its personal movement with respect to the latter.

Now for the basic statement of our algebraic theory of inheritance:

ALGEBRAIC THEORY OF INHERITANCE. *Inheritance arises from a wreath product:*

$$\begin{array}{lcl} \textit{Parent} & \longleftrightarrow & \textit{Control group} \\ \textit{Child} & \longleftrightarrow & \textit{Fiber group} \end{array}$$

Notice that this means that the basis of inheritance is *transfer*. The enormous power of this theory is that it explains inheritance in all of CAD, robotics, assembly, animation, and so on, as will be illustrated.

Before going on to applications, let us first consider diagrammatic aspects of current programs. Because run-time inheritance is created by the designer, it is usually represented by diagrams that the designer can view. It will be useful for us to show how these diagrams can be converted into algebra. A good diagrammatic representation is used by *3D Studio Max*, as illustrated in Fig 13. Here, inheritance is represented by indentation - i.e., an indented object is a child of the next object above with respect to which it is indented. Each object, except the World object, has a transform shown just below it. The transform relates the coordinate frame of the object to the coordinate frame of its parent. This transform is the "personal" transform of the object. In addition,

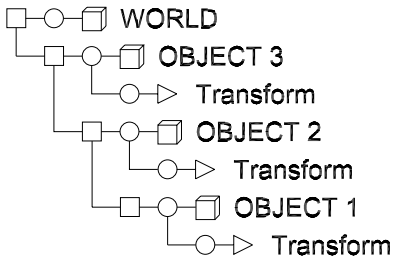

Figure 13: The representation of parent-child relations in *3D Studio Max*.

the object inherits the transform of its parent. The object therefore adds its personal transform to its inherited transform. This means, of course, that via its parent, it inherits the transform of its parent's parent, and so on. We shall now show how to convert such diagrams into algebra.

GROUP OF ENTIRE TRANSFORM STRUCTURE. *Consider a set of $n+1$ objects: Object 1 to n, and the World. Suppose that they are linked such that Object i is the child of Object $i+1$, and Object n is the child of the World. Then the group of the entire transform structure is the wreath product:*

$$ {}_{F_1}G_1^{F_2} \mathbin{\textcircled{w}} {}_{F_2}G_2^{F_3} \mathbin{\textcircled{w}} \ldots \mathbin{\textcircled{w}} {}_{F_n}G_n^{W} $$

where:

> *(1) Object i has personal transform group G_i and frame F_i.*
> *(2) Personal transform group G_i relates frame F_{i+1} of the parent, upper index, to the personal frame F_i, lower index. (The world frame F_{i+1} is written as W.)*

Notice that the subscript i of the group G_i is the same as the subscript i of its lower index F_i; that is, G_i and F_i are both personal to the Object i defining that level.

17 Theory of Relative Motion

Relative motion is basic to the design of articulated kinematic systems. As an example of our algebraic theory of inheritance, we are going to give a theory of relative motion in human perception, classical and quantum mechanics, robotics, and computer animation. The theory is simply this:

ALGEBRAIC THEORY OF RELATIVE MOTION. *A relative motion system corresponds to a wreath product in which the relative motion is given by the fiber group*

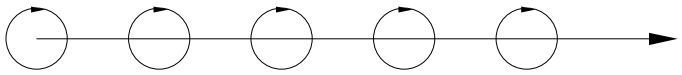

Figure 14: A relative motion system.

and the absolute motion, to which it is judged, is given by the control group:

$$relative\ motion \ \textcircled{w}\ absolute\ motion.$$

As an example, consider Fig 14, which shows circular motion, which itself is moving along a line. This occurs, for instance, when a wheel is moving along the ground. Observe that the circular motion is the *relative* motion; and the motion along the line is the *absolute* motion. The circular motion is given, of course, by the group $SO(2)$, and the linear motion is given by the translation group $\mathbb{R}$. Most crucially, one can see from the diagram that the entire system is given by a wreath product in which the circular motion is the fiber group and the absolute motion is the control group. The reason is that there is one copy of the fiber group $SO(2)$ for each point in the control group $\mathbb{R}$. That is, one has the wreath product:

$$SO(2) \ \textcircled{w}\ \mathbb{R}.$$

Our book shows that this theory explains all the main motion examples of Gestalt perceptual psychology - induced motion, separation of systems, the Johansson motion phenomena, etc. The rule is this:

> **Decompose the motion into two symmetry groups, such that one group** *transfers* **the other.**

This gives a wreath product where the transferring symmetry group is the control group and the transferred symmetry group is the fiber group.

This also formulates the standard decompositional procedures in classical and quantum mechanics. For example, it describes the well-known result that the total angular momentum of a system of particles about an origin O is decomposable into two components - the angular momentum of the particles with respect to the *center of mass*, and the angular momentum of the center of mass with respect to the *origin* O. Furthermore, it gives the well-known re-description of a two-particle system in terms of fictitious masses - the total mass and the reduced mass.

As illustrations of our theory, let us now consider serial-link manipulators, and then animation.

18 Serial-Link Manipulators

Standardly in a serial-link manipulator (such as the human arm), one says that the frames of two successive links are related by a special Euclidean transformation A_i, and thus the overall relationship between the hand coordinate frame and the base coordinate frame is given by the product of matrices

$$A_1 A_2 \ldots A_n \tag{8}$$

corresponding to the succession of links.

Now, in setting up the object-oriented structure of such manipulators, one usually stipulates that a distal link is a child of the next proximal link, and so on, successively along the manipulator. Our argument is that this arises from the *transfer* structure: The distal link has a space of actions that is transferred through the environment by the next proximal link. This exemplifies our claim that *the basis of inheritance is the deeper notion of transfer*. It is this that allows us to formulate inheritance algebraically in terms of wreath products. Thus, we argue that the group of a serial-link manipulator has the following wreath-product structure:

$$SE(3)_1 \mathbin{\textcircled{w}} SE(3)_2 \mathbin{\textcircled{w}} \ldots \mathbin{\textcircled{w}} SE(3)_n \tag{9}$$

where each level $SE(3)_i$ is isomorphic to the special Euclidean group $SE(3)$, and the succession from left to right corresponds to the succession from hand to base (distal to proximal).

The entire group we have given in (9) for the serial-link manipulator, is very different from the group that is normally given in robotics for serial-link manipulators. Standardly, it is assumed that, because one is multiplying the matrices in (8) together, and therefore producing an overall Euclidean motion T between hand and base, the group of such motions T is simply $SE(3)$. However, we argue that this is not the case. The group is the much more complicated group given in expression (9). The conventional group $SE(3)$ necessarily models the arm as a rigid structure, whereas the wreath product (9) models the arm as a structure we call *semi-rigid*: a group where rigidity breaks down at a discrete set of points. Most crucially the wreath product models the object-oriented structure, which is basic to all computation concerning the kinematics.

19 Animation

Now let us consider animation. A basic aspect of animation is the setting up of reference frames and inheritance relations between those frames. In order to illustrate our algebraic theory, consider a solar system consisting of the earth revolving around the sun, and the moon revolving around the earth. The most powerful method of simulating this is to use not only a frame fixed within each planet, but a dummy frame (within the planet) that rotates relative to the frame of the planet. Fig 15 shows the inheritance diagram that would be set up in *3D Studio Max* [5].

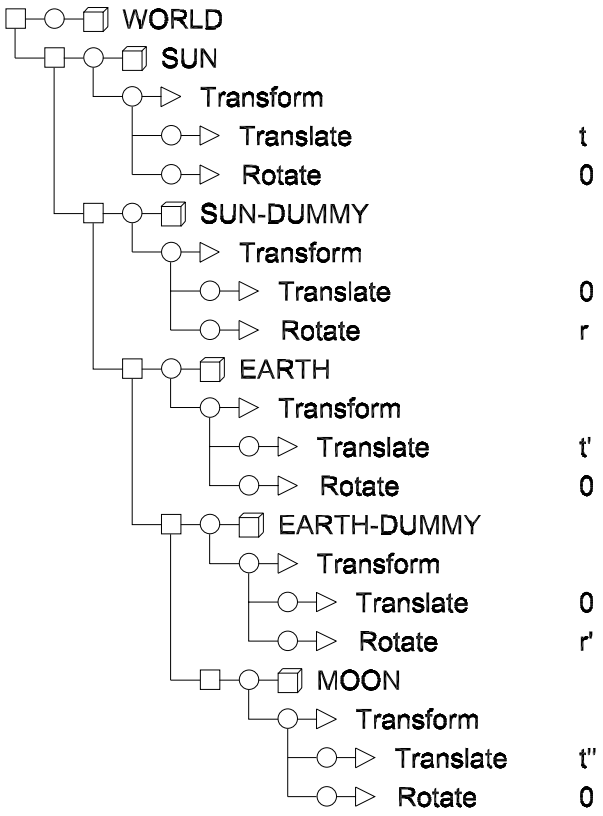

Figure 15: Inheritance hierarchy for a solar system.

Now using our method for converting inheritance diagrams into algebra (section 16), we see that the group of this inheritance structure is the following wreath product:

$$ {}_mT^{ed} \wr {}_{ed}SO(2)^e \wr {}_eT^{sd} \wr {}_{sd}SO(2)^s \wr {}_sT^w. \tag{10} $$

The symbols are as follows: There are two groups involved: T, the 2D translation group, and $SO(2)$, the rotation group. These two groups alternate down the sequence. Also, going down the sequence, the respective frame-symbols are $w =$ world; $s =$ sun; $sd =$ sun-dummy; $e =$ earth; $ed =$ earth-dummy; $m =$ moon.

20 Complex Shape

Let us now turn to the main purpose of the book: the representation of complex shape. As stated, the goal is to develop a systematic analysis of complex shape, such that

complexity is converted into understandability. To solve this problem, we develop a symmetry group for a complex object. Organized in accord with the theory of transfer and recoverability, this group will contain all the required information for designing the object, manufacturing it, manipulating it, navigating with respect to it, etc.

It is worth considering what our research procedure was to create the following group theory of complex objects: We worked through every single operation in each of several main CAD, solid modeling, assembly, and animation programs, including AutoCAD, Architectural Desktop, Mechanical Desktop, ProEngineer, 3D Studio Max, etc., as well as all the major manuals on each of the programs - approximately 15,000 pages of text. Each individual situation was characterized by a group, and a new class of groups was invented for any situation that could not be formalized in terms of any previously created class of groups. Proceeding in this manner, it was eventually found that three classes of groups could handle any newly created situation:

(1) Telescope groups.
(2) Super-local unfoldings.
(3) Sub-local unfoldings.

Thus our current assumption is that these groups can handle any complexity in shape generation.

Besides formulating these three classes of groups, we formulated the notion of an *unfolding group*, which is the over-arching class that contains these three types.

The basic idea of an unfolding group is that any complex structure such as a design in CAD, or a scene in computer vision, is *unfolded* from a maximally collapsed version of itself, which we call an **alignment kernel**. The full structure is unfolded outwards by applying *transfer* from different parts of the alignment kernel. The following gives a basic description of unfolding groups:

UNFOLDING GROUPS

Unfolding groups are characterized by the following two properties:

SELECTION: The control group acts *selectively* **on only part of its fiber.**

MISALIGNMENT: The control group acts by *misalignment*.

Major classes of unfolding groups are structured by starting with a configuration in which n primitives are maximally aligned. This configuration will be called the *alignment kernel.* **The unfolding causes successive misalignment of the primitives. Because this works by transfer, the unfolding action maps the alignment kernel onto misaligned versions of itself. We formalize this in the following way:**

(1) There are n objects which have symmetry groups $G_1, \ldots, G_n$. These correspond to the *primitives*. They are given by iso-regular groups.

(2) One forms the direct product, $G_1 \times \ldots \times G_n$, and makes this the fiber group of a wreath product, with control group $G(C)$, thus:

$$[G_1 \times \ldots \times G_n] \text{ Ⓦ } G(C).$$

The direct product $G_1 \times \ldots \times G_n$ should not be confused with the product of fiber-group copies. It is a single fiber group.

(3) In the above wreath product, any fiber-group copy, i.e., any copy of $G_1 \times \ldots \times G_n$, corresponds to an *object configuration*.

(4) Let the fiber-group copy in which the object symmetry groups $G_1, \ldots, G_n$ are maximally aligned with each other, be called the *alignment kernel*. Choose this to be the fiber-group copy corresponding to the identity element of the control group.

(5) The control group transfers object-configurations $[G_1 \times \ldots \times G_n]_g$ onto object-configurations $[G_1 \times \ldots \times G_n]_h$. In doing so, it pulls the objects out of alignment with each other. The control group is therefore symmetry-breaking on the alignment kernel, by creating misalignment.

21 Telescope Groups

The remainder of the paper gives a basic description of the three crucial types of unfolding groups and their role in design.

To get an intuitive sense of telescope groups, think of an ordinary telescope. In an ordinary telescope, you have a set of rings that are initially maximally coincident. Then you pull them successively out of alignment with each other. A telescope group is a group structured like this.

Let us look at this in more detail. In a telescope group, the initially coincident objects are n primitives. The fiber group is a direct product of the symmetry groups of those primitives:

$$[G_1 \times \ldots \times G_n]_{\mathcal{T}}$$

where $\mathcal{T}$ denotes "telescope". The configuration in which these primitives are maximally aligned is a copy of this fiber group, called the alignment kernel. Next, the control group is itself a wreath product $G(C)_1 \text{Ⓦ} \ldots \text{Ⓦ} G(C)_{n-1}$ of order $n-1$, and hence, with the alignment kernel, the entire unfolding group is a wreath product of order n.

$$[G_1 \times \ldots \times G_n]_{\mathcal{T}} \text{ Ⓦ } G(C)_1 \text{ Ⓦ } \ldots \text{ Ⓦ } G(C)_{n-1}. \tag{11}$$

We now stipulate that each control group $G(C)_i$ acts on only the left-subsequence $G_1 \times \ldots \times G_i$ of the alignment kernel. Therefore the successive control groups downward act on successively shorter left-subsequences. This gives the *telescope opening effect*.

As an example, we argue that the serial-link manipulator is best modeled by a telescope group. Recall that, in section 18, it was proposed that the group of a serial-link manipulator is the following wreath product:

$$SE(3) \wr \ldots \wr SE(3) \wr SE(3).$$

Let us now incorporate more structure into this. If one considers carefully the standard way of encoding the successive link frames along the manipulator (i.e., as related by successive Euclidean transformations [4], [25]), one realizes that, in fact, the structure of frames corresponds to an opening telescope effect. That is, the succession of Euclidean transformations allows one to consider the set of frames as having come from an initial state (the group identity element in each level) in which the frames were all coincident at the base; and they were successively pulled out of alignment by those transformations. Now, according to our theory of reference objects, section 12, we encode any Cartesian frame by the hyperoctahedral wreath group $\mathbb{Z}_2 \wr \Sigma_3$ which expresses the symmetry of the frame.

Thus, the group of a serial-link manipulator is actually the following telescope group:

$$\begin{aligned} &[[\mathbb{Z}_2 \wr \Sigma_3]_1 \times \ldots \times [\mathbb{Z}_2 \wr \Sigma_3]_n \times [\mathbb{Z}_2 \wr \Sigma_3]_{n+1}]_{\mathcal{T}} \\ &\wr SE(3)_1 \wr \ldots \wr SE(3)_n. \end{aligned} \tag{12}$$

The first line in this expression is the alignment kernel, and the second line is the control group which will unfold the kernel as an opening telescope. Notice that the successive misalignments of the hyperoctahedral wreath groups $\mathbb{Z}_2 \wr \Sigma_3$ in the alignment kernel shows that the essense of frame assignment in robotics is unfolding via symmetry breaking.

22 Super-Local Unfoldings

A frequent design situation is this: The designer selects part of the existing design, copies it, and then drags the copy to some other region of the design, perhaps with modification; e.g., walls are created not by drawing a new wall each time but by copying, moving, and modifying existing walls. We will model this by an unfolding group structured in the following way:

$$[G_1 \ldots G_j] \wr G_n^X.$$

The group in brackets represents a shape, e.g., a design, up to the current state. Then, one wreath appends a group G_n above this, which acts selectively on only some part X of the structure below. The wreath product operation here indicates that G_n acts by *transferring* X in some way. The entire group is called *super-local* because it is created by wreath-appending a control group *above* an existing structure, such that

the added control group acts selectively on only part of its fiber. Such groups model situations, for example, in *AutoCAD*, where one freezes part of the existing structure and manipulates some unfrozen cross-hierarchy selection of elements; or conversely, situations, for example, in *3D Studio Max*, where the cross-hierarchy selection is locked and manipulated over a sequence of steps.

23 Sub-Local Unfoldings

Valuable as super-local unfolding groups are, this is nothing compared with the final class of groups: sub-local unfoldings. We argue that these characterize what is the most powerful design process in the world today: Constructive Solid Geometry (CSG). This is the process of generating complex shape by moving, deforming and combining, primitives.

In a sense, sub-local unfoldings are generalized versions of telescope groups. Intuitively, if we regard a telescope group as an octopus with one arm, we can think of a super-local unfolding group as an octopus with several arms; indeed the arms can themselves have arms, and so on.

In fact, what creates the arm structure is the object-oriented inheritance hierarchy. Standardly, the design process proceeds by creating main objects, and appending child objects, and so on. According to our theory of inheritance, parents are control groups and children are fibers, in a wreath product. Therefore the process of creating a design, or any complex shape, is a process of adding groups *below* existing groups, via a wreath product. This is an essential aspect of sub-local unfolding. It corresponds to the prefix "sub" in the term "sub-local unfolding". However, there is also the other crucial term "local". And we now have to understand this.

The power of sub-local unfoldings is that they handle anomalies. We argue:

Complex shape generation is the generation of anomalies.

Anomalies are what create complexity. They are the essense for example of linguistics, i.e., the phenomenon of markedness; and they are the essense of crystal physics, i.e., you can only really see a crystal via its defects. Any environment - e.g., a complex scene in computer vision, an airplane design in mechanical engineering - is, according to this theory, a hierarchy of anomalies. It is in order to handle this, that we created the theory of sub-local unfoldings.

Before describing some of the mathematics of sub-local unfolding, it is necessary to present more of our algebraic theory of object-oriented programming. First of all, within that theory, there is an analysis of class structure which says that each geometric class consists of an internal symmetry group, specified often in the invariants clauses of the software text for the class, and an external group consisting of command operations, such as deformations, specified in the feature clauses of the class text. A principle claim of the theory is that the relation between the internal symmetry group and command structure, in the software text, is a wreath product, thus:

$$G_{sym} \wr G(C)$$

where G_{sym} is the internal symmetry group and $G(C)$ is the group of command operations. This has been a basic proposal of all our previous research, e.g., Leyton [14], [15], [16], [17], [18], [19].

Now let us turn to cloning. It is important to notice that, when one clones an object, one is producing a copy with the same instance values. This means that one is essentially creating a copy that is *aligned* with the original, as can be seen in such programs as 3D Studio Max and Viz. This copy can then be manipulated via its command operations, which will then pull the clone out of alignment, i.e., break the symmetry of the object-clone pair.

With this in mind, let us now return to design. We want to consider object creation and feature attachment. Feature attachment is the term used in mechanical design for the successive addition of structural units and components. It is, of course, the main process in design, [1], [3], [11], [28], [29], [34].

THEORY OF FEATURE ATTACHMENT

When one creates objects and attaches them in the design structure, one is entering new instances into the alignment kernel, and positioning the command group for each new instance in the appropriate wreath position within the unfolding group corresponding to the inheritance hierarchy of the structure.

The basic rule for positioning the command group is this: Parallel objects are linked by a direct product, and dependent objects by a wreath product.[1]

Let us now show how this works for sub-local unfolding. It is best to use an illustration.[2] Fig 16 shows an eight-room apartment. In this apartment, there is a central main room, as well as six surrounding rooms, and an anomaly: an extension room off the 5th surrounding room (bottom right).

First observe that the parent-child structure is this: The central main room is the ultimate *parent* within the apartment. The six surrounding rooms are its *children*, which are in parallel, and the extension is a *child* of the 5th surrounding room. Now, let us give the group of the apartment *without* the anomalous extension:[3]

$$
\begin{aligned}
&[\mathbb{R} \circledcirc{w} \mathbb{Z}_4]_{\mathcal{U}} \\
&\circledcirc{w}[AGL(2,\mathbb{R}) \times AGL(2,\mathbb{R}) \times AGL(2,\mathbb{R}) \times AGL(2,\mathbb{R}) \times AGL(2,\mathbb{R}) \times AGL(2,\mathbb{R})] \\
&\circledcirc{w} AGL(2,\mathbb{R}).
\end{aligned} \tag{13}
$$

The affine groups $AGL(2,\mathbb{R})$ are the *command* groups associated with the object instances. They are arranged in accord with the algebraic theory of inheritance; that is, a parent-child relationship is given by a wreath product, and a parallel relationship is given by a direct product. Thus to interpret this group: The affine group on the bottom

[1] Our book [21] shows that the same theory applies to the feature-recognition process, e.g., in machining, [6], [7], [8], [9], [10], [12], [13], [26], [27], [32], [33]. For example, the standard features produced by a cylindrical cutter - i.e., the hole, slot, and pocket - are given by iso-regular groups. These are entered into the alignment kernel, and successively mis-aligned by the unfolding control groups.

[2] Purely for ease of exposition, the structures in this example will be described only in plan-view.

[3] Expression (13) is only a schematic abbreviation of the group. The reader should see [21] for full details.

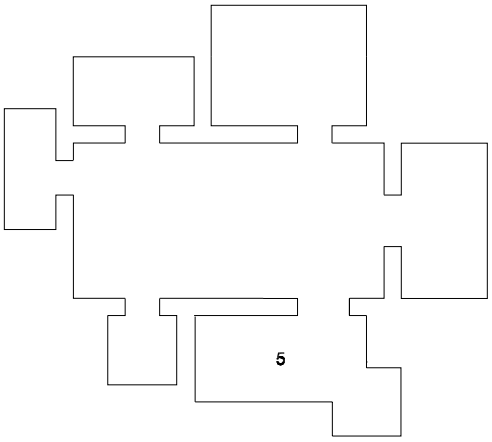

Figure 16: An eight-room apartment.

line gives the relation between the main room and the world frame. The six affine groups on the middle line give the relation between the six surrounding rooms and the main room. The top line represents the alignmnent kernel, which consists of a direct product of as many instances of the primitive $\mathbb{R} \circledcirc \mathbb{Z}_4$, as are required.

Now observe that the wreath product symbol at the beginning of the bottom line says that the relation between the main room in the bottom line, and the six rooms in the middle line, is that of parent to children. More properly, this is a relation between the *command groups* of those object instances. The direct products in the middle line says that the relation between the command groups of the six surrounding rooms is parallel. Finally, the wreath product at the beginning of the middle line denotes the relation between the internal symmetry group and command group in the software text of the object class.

Now, to add the *anomaly*, i.e., the extension room off Room 5, we wreath sub-append an extra affine group to the 5th member of the direct product in the second line.

Thus we see illustrated here how feature creation and attachment is handled: The command groups of the object instances are moved to the required algebraic position in the unfolding structure.

The crucial thing is that, in the succession through the lines in the above expression, one is creating symmetry-breaking by successive misalignment.

Generally, our theory of complex shape can therefore be given a summary description in the following way:

> **Complex shape generation proceeds by a series of symmetry-breaking phase-transitions, that occur by** *selective misalignment.* **This is given by a wreath-product hierarchy in which the fiber groups, representing the symmetry ground-states, are the alignment states, and the successive control groups create the selective misalignments by transfer. We call this process,** *unfolding*.

The substantial algebraic theory developed for this in the book is applied at length to mechanical design and manufacturing, architectural design, and robotics (as well as perception). Using the theory of unfolding groups, we work in detail through the main stages of mechanical CAD/CAM: part-design, assembly and machining. For example, in part-design, we give an extensive algebraic analysis of sketching, alignment, dimensioning, resolution, editing, sweeping, feature-addition, and intent-management. The equivalent analysis is also done for architectural design. The structure of robot manipulators and assembly is also a central concern.

References

[1] Chen, X., & Hoffmann, C.M (1995). On editability of feature-based design. *Computer-Aided Design*, **27(12)**, 905-914.

[2] Clément, A., Rivière, A., & Temmerman, M. (1994). *Cotation Tridimensionnelle des Systèmes Méchaniques.* Ivry-Sur-Seine Cedex: PYC Edition.

[3] Cunningham, J.J., & Dixon, J.R. (1988). Designing with features: the origin of features. *ASME computers in engineering,*San Fransisco, USA, July/August, p237-43.

[4] Denavit, J., & Hartenberg, R.S. (1955). A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*, ASME, **22**, 215-221.

[5] Elliot, S., Miller, P., Abouaf, J., Espinosa-Aguilar, D., Alexander, S., Barnard, D., Kakert, P., Kalwick, D., Kelm, K., Koch, M., Williamson, M. (1998). *Inside 3D Studio Max 2, Volume I.*. Indianapolis, IN: New Riders Publishing.

[6] Falcidieno, B., & Giannini, F. (1987). In: Marechaled, editor. *Extraction and organization of form features into a structured boundary model*. EUROGRAPHICS'87. Amsterdam, Elsevier. p249-259.

[7] Falcidieno, B., & Giannini, F. (1989). Automatic recognition and representation of shape based feature in a geometric modeling system. *Computer Vision, Graphics and Image Processing*, **48**, 93-123.

[8] Han, J., Regli, W.C., & Brooks, S. (1998). Hint-based reasoning for feature recognition: status report. *Computer-Aided Design*, **30(13)**, 1003-1007.

[9] Han, J., & Requicha, A.A.G. (1995). Integration of feature based design and feature recognition. In: ASME International Computers in Engineering Conference, Boston.

[10] Henderson, M.R., & Anderson, D.C. (1984). Computer recognition and extraction of form features: a CAD/CAM link. *Computers in Industry*, **5**, 329-339.

[11] Hoffmann, C.M., & Juan, R. (1993). Erep - an editable, high-level representation for geometric design and analysis. In: Wilson, P.R., Wozny, M.J. & Pratt, M.J., editors. Elsevier, Amsterdam. p129-164.

[12] Joshi, S. & Chang, T.C. (1988). Graph-based heuristics for recognition of machined features from a 3D solid model. *Computer-Aided Design*, **20(2)**, 58-66.

[13] Kyprianu, L.K. (1980). *Shape classification in computer-aided design.* PhD thesis, Univesity of Cambridge, UK.

[14] Leyton, M. (1984). Perceptual organization as nested control. *Biological Cybernetics*, **51**, 141-153.

[15] Leyton, M. (1986a). Principles of information structure common to six levels of the human cognitive system. *Information Sciences*, **38**, 1-120. Entire journal issue.

[16] Leyton, M. (1986b). A theory of information structure I: General principles. *Journal of Mathematical Psychology*, **30**, 103-160.

[17] Leyton, M. (1986c). A theory of information structure II: A theory of perceptual organization *Journal of Mathematical Psychology*, **30**, 257-305.

[18] Leyton, M. (1987a). Nested structures of control: An intuitive view. *Computer Vision, Graphics, and Image Processing*, **37**, 20-53.

[19] Leyton, M. (1992). *Symmetry, Causality, Mind.* Cambridge, Mass: MIT Press.

[20] Leyton, M. (1999). New foundations for perception. In Lepore, E. (Editor). *Invitation to Cognitive Science.* Blackwell, Oxford. p121 - 171.

[21] Leyton, M. (2001). *A Generative Theory of Shape.* Berlin: Springer-Verlag.

[22] Leyton, M. (2001). *Group Extensions.* Book submitted.

[23] Meyer, B. (1997). *Object-Oriented Software Construction*. New Jersey: Prentice Hall.

[24] Omura, G. (1999). *Mastering AutoCAD 2000.* San Fransisco: Sybex.

[25] Paul, R.P. (1981). *Robot manipulators.* Cambridge, Mass: MIT Press.

[26] Pratt, M.J. (1993). Applications of feature recognition in the product life cycle. *Intl. J. Computer Integrated Manufact.*, **6(1-2)**, 13-19.

[27] Regli, W.C. (1995). *Geometric algorithms for recognition of features from solid models.* PhD thesis, Univesity of Maryland.

[28] Rossignac, J.R. (1990). Issues on feature based editing and interrogation of solid models. *Computer and Graphics*, **14(2)**, 149-172.

[29] Shah, J.J. (1991). Conceptual development of form features and feature modelers. *Research in Engineering Design*, **2**, 93-108.

[30] Srinivasan, V. (1999). A geometrical product specification language based on a classification of symmetry groups. *Computer Aided Design, 31 (11),* 659-68.

[31] Srinivasan, V. *Theory of Dimensioning.* Book, in preparation.

[32] Vandenbrande, J.H. (1990). *Automatic recognition of machinable features in solid models.* PhD thesis, Univesity of Rochester, Rochester, NY.

[33] Vandenbrande, J.H. & Requicha, A.A.G., (1993). Spatial reasoning for automatic recognition of machinable feature in solid models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **15(12)**, 1-17.

[34] Whitney, D.E., Mantripragada, R., Adams, J.D., & Rhee, S.J. (1999). Towards a theory of design of kinematically constrained mechanical assemblies. *International Journal of Robotics Research,* **18(12)**, 1235-1248.