Solid Freeform Fabrication and Parametric Engineering

J.C. Boudreaux
NIST/Advanced Technology Program

Abstract. Solid freeform fabrication (SFF) is based on a part-centric process model: create a *solid model* of the part, form *planar slices*, and *fabricate* the part by producing all of the polyhedra by any of several methods. A class of applications is emerging which will pull SFF from the part-centric model to a new paradigm in which parts are seen as components of interactive networks. This is precisely the context for which parametric engineering has been proposed. In this paper, a computational framework will be developed that consists of a finitary topological representation of parts as *3-manifolds* and representation of the evaluative context by means of a *symbolic environment*. Parametric engineering is interpreted as a controlled evaluation of parameters within the context of the symbolic environment.

1. Introduction

Solid freeform fabrication (SFF) has had significant impact on the production of engineering prototypes and more recently on the manufacture of functional parts, especially the direct fabrication of molds, dies, and other tooling. These applications have pushed SFF toward a part-centric process model: (1) create a *solid geometric model* of the part, defining surface boundaries and identifying interior (material) regions from the exterior (void) regions; (2) select a part orientation and then form *planar slices*, decomposing the solid into a sequence of thin cross-sectional *polyhedral* layers; and (3) *fabricate* the part by producing all of the polyhedra by any of several methods, including: stereolithography, selective laser sintering, laminated object manufacturing, fused deposition modeling, and three dimensional printing. Given the current emphasis on the need to significantly reduce the time it takes to bring industrial goods to market and the practice of shifting component design to supply chains, a class of applications is emerging which will pull SFF from the part-centric model to a new paradigm in which parts are seen as *components of interactive networks and the capabilities of the system as a whole depend upon a cross-supply-chain ability to develop and use specifications for shared interfaces between (often remote) components.*

If a group of suppliers are assigned linked components, then the interfaces between these components must be specified in sufficient detail to allow each supplier to develop test and validation procedures even while the parts are being designed. To further complicate matters, these interfaces are often multimodal in the sense that they may require geometric, mechanical, electrical, thermal, fluidic, and/or chemical specifications. This is precisely the context for which parametric engineering has been proposed. In the next section, I describe a finitary topological encoding for parts based on Morse theory and Reeb graphs **[1]**, **[8]**, **[9]**. In Section 3 and 4, I use directed graphs to model representations of engineered artifacts **[7]**. In the Section 5, I sketch a computational framework **[2]** for parametric engineering in the context of SFF.

2. Finitary Representations of 3-Manifolds

This paper is an attempt to construct a qualitative parametric engineering model of the product design process in the context of solid freeform fabrication. It seems natural to suppose that *engineered artifacts* may be resolved into components parts that can be usefully represented as *compact connected 3-manifolds* (definitions of the topological terms used in this paper are collected in the **Appendix**, see also **[3]**, **[4]**, **[5]**) It is often useful to consider *3-manifolds with boundary.* The boundary consists of a possibly disconnected set of compact smooth 2-manifolds (*topological surfaces*). For example, the 2-sphere is the single boundary of the closed 3-ball. In the general case, the class of structures includes more complicated objects that have enclosed *voids* and *tunnels*. The collection of all 3-manifolds (with or without boundaries) that have these topological features will be called *nice 3-manifolds*. Every 3-manifold *M* can be *triangulated* by an abstract simplicial complex *K* such that a geometric realization of *K* in Euclidean 3-space is homeomorphic to *M.*

Let *(M, K)* be a nice triangulated 3-manifold, then *solid shapes* may be assigned to it by mapping *M* homeomorphically into an open 3-ball, *i.e.*, by developing a geometric realization of *M*. Once a solid shape is available, and then it is a straightforward exercise to apply the SFF process by forming planar sections of the solid. True enough, but this idea is really useless without some way to encode the mappings, specifically some plausible method to select a reasonable number of candidate realizations from among the huge number of possible homeomorphisms. *What might be proposed is that there is a finitary description of M that reduces the set of all homeomorphisms to a finite (or at most denumerable) set such that an optimal solid shape is near the image of a candidate.*

The basic insight of **Morse theory** is that that there is a very close connection between the topology of a smooth manifold and the *critical points* of smooth functions on the manifold (points at which all partial derivatives of the function are null) **[1]**. Critical points consist of *maxima*, *minima*, and *saddle points.* First, assume that *M* is a nice 3-mainfold and that ∂M, the boundary of *M,* is the disconnected union (possibly empty) of nice manifolds *V* and *W.* Intuitively, think of V and W as planes which sandwich *M* between them. Then let *f* be a smooth mapping of *M* into a real closed interval *[a,b]* such that $f^{-1}(a) = V$ and $f^{-1}(b) = W$. Second, generate slices through *M*, that is, if *x* in *[a,b]*, then $f^{-1}(x)$ is such slice. Not all such functions are useful. What we want to do is to restrict our attention to *Morse functions* whose critical points on *M* are *nondegenerate* (the Hessian matrix of partial second derivatives is non-singular at the point). On nice manifolds, these functions are plentiful. The point of this exercise is to use *f* to scan *M* to build a CW-complex that has the same homotopy type as *M.* The cell complex is built up by a sequence cell attachments, which is completely determined by the sequence of critical points. As *f* slides between critical points, the topology of the manifold does not change, that is, no cells need to be attached. When *f* reaches a critical point, the topology of the manifold changes in way that requires the cell complex to be updated by attaching a *k*-cell, where *k* is the *index of the critical point* (the number of negative eigenvalues of the Hessian matrix). Since nice manifolds are compact, there are at most finitely many critical points, which implies that the sequence of critical points p_i and the associated critical slices $f^{-1}(x_i)$ completely encode the intrinsic topology of *M*.

Once a Morse function *f* has been defined on *M*, a **Reeb graph** can be used to generate a *skeleton* of *M* **[8]**, **[9]**. A quotient space is formed on $M \times \mathbf{R}$ by identifying points, which belong to the same connected component of a slice. That is, if *p* and *q* points, then *(p, f(p))* is identified with *(q, f(q))* if *f(p) = (f(q)* and *p* and *q* are in the same connected component of $f^{-1}(f(q))$. The nodes of the Reeb graph will be the critical points of *M* and the edges will be given by the connected components between critical slices.

3. Artifacts and Their Component Trees

Engineered artifacts may be represented by combinatorial objects called *directed graphs,* or *digraphs*, which are abstract structures of the form (*V*, **R**), consisting of a finite, unempty, but otherwise arbitrary set V of nodes and an arbitrary set **R** of ordered pairs *(x,y)* of nodes, called *links.* Two projection functions are associated with digraphs, namely, *hd* and *tl* mapping links to nodes such that for all links *p = (x,y)*, *hd(p) = x* and *tl(p) = y*. A sequence of links *P(n)* is a *path* if all adjacent links are *composable*, that is, *tl(P(i)) = hd(P(i+1).* The projection functions *hd* and *tl* can be extended to finite paths *P* by assigning *hd(P) = hd(P(1))* and *tl(P) = tl(P(n))*, where *n* is the largest index, that is, *P* is a path from *P(1)* to *P(n)*. Given any node *x*, *in(x)* is the set of all links *p* such that *tl(p) = x*, and *out(x)* is the set of all links *p* such that *hd(p) = x*. As a first approximation, artifacts may be represented by specific digraphs called *(rooted) trees*: there is a single root node, say *r,* such that in*(x)* is the empty set, and given any non-root node, say *y*, there is one any only one path P from *r* to y. The set of all nodes *x* of a tree such that *out(x)* is the empty set are called *leaves*. Clearly, if a tree has finitely many nodes, then every path must terminate in a leaf.

The use of trees to represent artifacts, to be called *component trees (c-trees)*, is based on the following interpretation: (1) leaf nodes are labeled by *fabricated (unassembled) parts*; (2) non-leaf nodes are labeled by *industrial processes* which assemble (and join) their inputs into an output; and (3) links *p* are labeled by *transfer operations* which *receive* inputs on *hd(p)* and *send* outputs to *tl(p).*

Each non-leaf node can be interpreted both as a place in which one or more manufacturing processes are performed on the components which are passed up by the node's immediate descendants and also as a source of (partially) finished components. The interweaving of product and process data in a single structure is useful. For example, if the tree is read in terms of the processes to be performed, then it contains precisely the information needed to define of the layout of the factories needed to produce the product and as the source of the master production schedule which, together with process time standards, controls the flow of production. If the tree is given the product reading, then the process details are suppressed and the nodes are paired with the components of the artifact. Thus, the nodes are labeled with, and interpreted by; their outputs, and the edges show the part relation. In effect, the product interpretation provides a detailed structural map of the components of the artifact.

An example might help. Begin with a finite list of labels that are taken to be the unassembled components of the artifact being modeled. Assume at the being that each label is free. Then the bottom-up construction of a tree proceeds by repeating the following procedure: select any number of items *X* from the free list, remove each selected item from the free list, and then add the set *{X}* to the free list. This procedure may terminate whenever only one item remains. The termination condition is *ambiguous*. Consider this derivation from three labels:

(1) $A, B, C \Rightarrow$ (2) $\{A, C\}, B \Rightarrow$ (3) $\{A, C\}, \{B\} \Rightarrow$ (4) $\{\{A, C\}, \{B\}\} \Rightarrow$ *stop*

Unlike the earlier steps, step (4) is a possible termination point, but not an obligatory one. This process could be extended indefinitely:

(5) $\{\{\{A, C\}, \{B\}\}\} \Rightarrow$ (6) $\{\{\{\{A, C\}, \{B\}\}\}\} \Rightarrow \ldots$

This process can be applied at any point in the derivational process:

(3) $\{A, C\}, \{B\} \Rightarrow$ (3°) $\{\{A, C\}\}, \{B\} \Rightarrow$ (3°°) $\{\{\{A, C\}\}\}, \{B\} \Rightarrow \ldots$

To make the tree structure explicit notice that the root node is the final item and that labels are leaf nodes that represent fabricated parts. Given a root node, all of the other nodes are generated by reversing the derivation: if *x* is already a node, then add the elements of *x* as nodes and add links from *x* to each of its elements.

4. Multilayered Structures

Up to now, the focus has been on the static aspects of artifacts, but it is obviously necessary to understand them from an operational point of view. Artifacts are *systems* that are composed of interacting parts. Systems, and system components, are *stratified* in the sense that single components at one level may be further resolved into components at the next lower level. The product/process interpretation clearly establishes that an artifact as an *organizational unity*, quite distinct from the much less interesting notion of a nested set of components. That is what assembly and joining actually do for us. What must be added to this model is *the principle that the overt behavior of an artifact depends upon the operational capacity of its components.*

To develop this idea, we really need interpretation in which each node *x* is paired it with a *transfer function*. Given a time-sequence of *inputs* acquired from *in(x)* and information about the current *state* of *x* yields (after some delay) a time-sequence of *outputs* released to *out(x)* and then updating the state of *x*. Saying that artifacts are systems is a recognition that artifacts are distributed networks of processors whose states are coupled in complicated, self-reflexive ways. That is, an observer able to examine simultaneously the state of every one of an artifact's components would be able to discover behavioral regularities. The fact that one component, say *X,* is in state *s* more or less tightly constrains the observed states of other components linked to X, *and conversely*.

But at this point the links between components (and the first step is an important part of the definition of this idea) are much more complicated. This has to do with the factors affecting the behavior of the artifact. Clearly, to address this need, a more elaborate *multilayered structure* needs be defined on top of the c-tree. First, we need see the operational aspects of the artifact in terms of a network, or communication model, which was suggested above. I imagine that the components are able to send and receive signals. To begin with, let's assume that the interpreted c-tree is the base layer, say *layer 0*. Then in order to reflect network features of the situation, we need to drop the tree-defining restrictions on the links. The purpose of higher-level links is to function as wires in a network, that is, if p is such a link, then a signal is sent from hd(p) and received at tl(p). Links do not need to be consistent with the 0-layer tree. Artifacts have potentially unbounded stratification. Even though there are at most finitely many nodes, there is no *apriori* bound on the number of layers into which an artifact can be resolved. Since stratification is intended to support a *semantic analysis* of the behavior of artifacts, the new layers will be unrestricted *digraphs* of the form $(V, \mathbf{R})$, where V is once again the set of the components, but $\mathbf{R}$ is an arbitrary set of ordered pairs (x,y) of nodes.

These separate layers must be *fused* to form a single digraph representation. Since each layer has the same set of nodes, we need only consider the manner in which the fused digraph is connected. Following Mack's approach in **[7]**, engineered artifacts are to be represented as *complex systems*. The connections between nodes in the system are called *arrows*:

(1) if p is a k-layer link (x,y) then $p = (x,y,k)$ is an arrow ;
(2) (*adjoints*) if f is an arrow, then there is an associated arrow f^* in the "opposite direction" such that $f^{**} = f$ and for all arrows f and g, $(fg)^* = g^*f^*$;
(3) (*path closure*) all paths of arrows are arrows;
(4) (*identity*) every node x has unique identity arrow id_x that maps the node identically onto itself.

It can be shown that every arrow is a path of links or the adjoints of links. Because of the way in which the layers have been fused, arrows can pass from one layer to another with no difficulty at all. On each layer the total number of links was limited by the number of nodes: if there are m nodes then there can be at most m^2 links on every layer. Unlike links, arrows can form *loops*, that is, we can begin at a node x and then walk along a path as far as we like and then find our way back to x, freely using adjoints if necessary. Unlike the simple connectivities provided by c-trees, arrows enable the multimodal analysis of artifacts required by geometric, mechanical, electrical, thermal, fluidic, and/or chemical specifications.

5. A Sketch of a Computational Framework

Artifacts are being interpreted over a domain of abstract structures whose inherent syntax is compositional: the leaf components are atoms and the non-leaf components are compound entities constructed from simpler ones below them. This observation suggests that the operational interpretation of artifact trees could be based upon a *functional model of computation*, the most familiar example of which is embodied in Lisp. The atoms include such customary values as integers, floats, and strings, but they also include a potentially infinite set of *symbols*. There are several styles of compound elements, including arrays and files, but the most important are finite sequences of expressions, or *lists*. The list constructor is cons. That is, if *x* and *y* are any Lisp expressions, then cons*(x,y)* is the list whose first element is *x* and whose remaining elements are *y*. The only list is not constructed in this manner is the empty list, called nil which has no elements. This model is based on an interpreter that repeat the following sequence of operations: (1) *read* an expression from an input stream; (2) *evaluate* the expression in the context of the current *symbolic environment* (a list of *symbol/value pairs*); and then (3) *write* the resulting expression to an output stream. When a symbol is read, the evaluator tries to match the symbol with an entry in the symbolic environment. If a match is found, then the corresponding value is returned.

Computational domains are constructed with the Lisp framework by building all of the necessary structural elements into the symbolic environment. Thus, the constructed domain is spread over a list of symbol/value pairs which are dynamically updated by the evaluation of a special class of functions which cause these updates to take place. The structural elements of the artifact domain, which may be implemented in the Lisp by using the digraph processor, defined in **[2]**. The digraph processor includes constructors, which add and delete both nodes and links. These operators may be applied dynamically, that is, during process of evaluating a digraph object, the shape of a digraph can be modified by the addition and deletion of nodes and links.

Digraphs are combinatorial objects, but the point of introducing this type is that digraphs can be used to provide places to store useful values, called *contents*, on both nodes and links. Since contents can be any syntactically well-formed expression, when evaluated, the result could not only modify the combinatorial structure of the digraph and could also have side effects on the symbolic environment in respect to which the evaluation of the digraph is taking place **[2]**.

6. Concluding Remarks.

This paper has really been a series of conjectures. The artifact structure that was built on top of c-trees is a kind of automaton whose memory cells consist of the node of the c-tree. This method of computing is in some respects analogous to cellular automata. Each cell is connected to its neighbors in a very precisely defined way. Where cellular automata are based on lattice neighborhoods with simple connectivity (cells usually have either 4 or 8 cell neighborhoods), *artifact automata* have much more complicated neighborhoods. If *x* is cell of an artifact automaton, then its neighborhood consists of all the nodes with which it is linked. Thus, the structure of these neighborhoods may differ radically from one cell to another. But like classical cellular automata, all updates happen by a *Jacobi sweep* **[7]**. The contents associated with the nodes and links are evaluated at time *t* and their value at time *t+1* is determined. After all of these evaluations have been completed, the current configuration of the artifact automata is updated by swapping in the *t+1* values.

Once an artifact automaton has been wired up, the initial states of all nodes are set to some predetermined initial value and each leaf node is assigned it an appropriate Reeb graph, say one which encodes the topology of an already approved design. After a *coherent root state* attained, one may retrieve *solid shapes* from the leaf nodes. Repeat as necessary. One can imagine at least one trivial solution: return the solid shape of the already approved design. To do any better than this will require a substantial deepening of our undestanding of role of topology in the optimal design of physical shapes subject to engineering constraints.

References

[1] Bott, R, "Lectures on Morse Theory, Old and New," *Bulletin (new series) AMS,* vol 7 (1982); 331-358.
[2] Boudreaux, J.C. "Concurrency, device abstraction, and real-time processing in AMPLE," in W.A. Gruver and J.C. Boudreaux (eds.), *Intelligent Manufacturing: Programming Environments for CIM*, Springer-Verlag, 1993; 31-91.
[3] Bourbaki, N. *Theory of Sets,* Herman, 1968.
[4] Dey, T.K., H. Edelsbrunner and S. Guha, "Computational Topology," in B. Chazzle, J.E. Goodman, and R. Pollak (eds.), *Advances in Discrete and Computational Geometry,* AMS, 1999.
[5] Hocking, J.G. and G.S.Young, *Topology*, Dover, 1961.
[6] Kunii, T.L. "Research Issues in Modeling Complex Object Shapes," *IEEE Computer Graphics and Applications*, vol 14(1994); 80-83.
[7] Mack, G., "Universal Dynamics, a Unified Theory of Complex Systems. Emergence, Life and Death," *Commun. Math. Phys.*, vol 219 (2001); 141-178.
[8] Shinagawa, Y. and T.L. Kunii, "Constructing a Reeb Graph Automatically from Cross Sections," *IEEE Computer Graphics and Applications*, vol 11(1991); 44-51.
[9] Shinagawa, Y., T.L. Kunii Y.L. Kergosien, "Surface Coding Based on Morse Theory," *IEEE Computer Graphics and Applications*, vol 11(1991); 66-78.
[10] Sorkin, R.D. "Finitary Substitute for Continuous Topology," *Int. J. Theor. Phy.,* vol 30(1991); 923-947.

Appendix

A topological space is a pair (S, **O**) where S is an unempty set of points and **O** is a set of subsets of S, called the *open sets*, which satisfies the following conditions: **O** contains both S and the empty set $\varnothing$, and is closed under finite intersection and arbitrary union. A *closed set* is the set-theoretical complement of an open set. The collection **C** of all closed sets satisfies the *dual* conditions: **C** contains both $\varnothing$ and S, and is closed under finite unions and arbitrary intersections. Let Y be an arbitrary subset of S and p be a point, then p is a *limit point* of Y only if every open set of (S, **O**) contains a point of Y distinct from p. That is, elements of Y get arbitrarily close to p. For example, the *(Euclidean) n-space* is the set of all real n-tuples $\boldsymbol{x} = (x(1),\ldots, x(n))$. If $\boldsymbol{x}$ is a point, then the *norm* of $\boldsymbol{x}$ is $\|\boldsymbol{x}\|$ is defined by the Pythagorean distance from the origin $\boldsymbol{0}$. The induced metric $|\boldsymbol{xy}|$ is $\|\boldsymbol{x} - \boldsymbol{y}\|$. For any point $\boldsymbol{x}$, the *open n-ball* with center $\boldsymbol{x}$ and radius r is the set of points $\boldsymbol{y}$ such that $|\boldsymbol{xy}| < r$. The *Euclidean topology* is generated by forming finite intersections and arbitrary unions of open n-balls. There are several important subspaces: the *n-halfspace* is $\{\boldsymbol{x} : x(1) \geq 0\}$, the *closed n-ball* is $\{\boldsymbol{x} : \|\boldsymbol{x}\| \leq 1\}$, and the *(n-1)-sphere* is $\{\boldsymbol{x} : \|\boldsymbol{x}\| = 1\}$.

If U is a collection of subsets of S whose union is equal to S, then U *covers* S. Any subset of U which also covers S is called a *subcover* of U. A topological space is *compact* if every cover has a finite subcover. A set of points X is said to be dense in S if every point of S is either a point of a limit point of X, that is, S is the *closure* of X. A space is *separable* just in case it has a countable dense subset.

Let S and T be topological spaces, then a function f from S to T is *continuous*, or a *map*, provided it is an *into* function and for every subset X of S and point p, if p is a limit point of X, then $f(p)$ is a limit point of $f(X)$, the image of X under f. Equivalently, f is continuous if it is *into* and the *inverse* image under f of every open set of T is an open set of *S*. A function from spaces *S* to T is a *homeomorphism* provided it is a one-to-one onto function (a *bijection*) and both the function and its inverse are continuous.

A space is *connected* if it is not the union of two disjoint open sets otherwise it is *separated*. A space is a T_0 *space* if given any two points at least one is contained in an open set not containing the other. A space is a T_1 *space* if given any two points, each is contained in an open set not containing the other. A space is a T_3 (*Hausdorff*) *space* if given any two points, there are disjoint open sets, each containing just one of the points.

An *abstract simplicial complex* is a pair (V, Σ) where *V* is a set of *vertices*, and Σ is a collection of finite subsets of *V* such that each vertex lies in some element of Σ and if σ is any element of Σ, then every subset of σ is also an element of Σ. The *dimension of a simplex* is one less than its cardinality and the *dimension of a simplicial complex* is the maximum dimension of the elements of Σ. A *simplicial mapping* of one simplicial complex onto another is an into function from the vertices of one to the vertices of the other such that the images of the simplexes of one are simplexes of the other. Two simplicial complexes are *isomorphic* if there is a bijective simplicial mapping from one to the other such that the inverse mapping is also simplicial. Given an *n*-space of suitably high dimension, the *convex hull* every set of $(k+1)$ pointwise independent points is a *geometric k-simplex*. It is an easy exercise to show that every finite abstract simplicial complex is isomorphic to a geometric simplicial complex, called a *geometric realization*.

The boundary of a closed *d*-ball B^d is a *(d-1)*-sphere. A *d-cell* is any topological space *Q* homeomorphic to a closed d-ball. The image of the *(d-1)*-sphere is ∂Q. The basic operation in the formation of *CW-complexes* is *attaching a d-cell*: let *X* be a space, *Q* a *d*-cell and $f{:}\partial Q \rightarrow X$ a continuous map, then the attachment of *Q* to *X* is the quotient of the disjoint union of *X* and *Q* formed by identifying *q* in *Q* with *f(q)* in *X*. A *CW-complex* is a sequence X_i of topological spaces such that $X_0 = \emptyset$ and X_{i-1} arises from X_i by attaching a *d*-cell. Simplicial complexes are special cases of CW-complexes.

An *n-dimensional manifold (n-manifold) M* is a separable Hausdorff topological space such that for every point *p*, there is a neighborhood of *p, which* is homeomorphic to Euclidean *n*-space or to Euclidean *n*-halfspace. Let *U* be an open set for some point p of *M* and let *h* be a homeomorphism from *U* into *n*-space, or equivalently, any open n-ball, then the pair (*U*, *h*) is called a *chart of dimension n.* An *atlas* **A** on *M* is a collection of charts whose open sets cover *M*. Suppose that *(U, h)* and *(V, k)* are charts whose open sets have a nonempty intersection, say *W*, then *h* and *k* given rise to a homeomorphism on the *n*-space images of *W* , specifically, $(kh^{-1}){:}\ h(W) \rightarrow k(W)$, and which establishes functional relationships between the *n*-tuples in the respective images of *W.* If all of the functions so established are differentiable (in class C^k or C^{∞}) then *M* is said to be a *differentiable (smooth) manifold*. The collection of all points, if any, with neighborhoods homeomorphic to the *n*-halfspace define the *boundary* of the manifold, ∂M.

Given two topological space X and Y, let $f,g{:}\ X \rightarrow Y$ be continuous functions. Then *f* and *g* are *homotopic* iff there is a continuous mapping $h{:}\ X \times [0,1] \rightarrow Y$ such that for all *x* in *X*, $h(x,0) = f(x)$ and $h(x,1) = g(x)$. The mapping *h* is a *homotopy between f and g* and the product space $X \times [0,1]$ is the *homotopy cylinder*. The existence of a homotopy between functions establishes that the functions can be continuously deformed into one another. The homotopy relation is an equivalence relation on the set Y^X of all continuous mappings from *X* to *Y* that partitions the set into *homotopy classes*. Two spaces *X* and *Y* are *homotopically equivalent*, or *of the same homotopy type*, if there are mappings $f{:}X \rightarrow Y$ and $g{:}X \rightarrow Y$ such that the composite mappings $fg{:}Y \rightarrow Y$ and $gf{:}X \rightarrow X$ are homotopic, respectively, to the identity mappings $id{:}Y \rightarrow Y$ and $id{:}X \rightarrow X$.