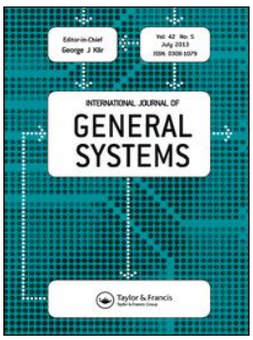

International Journal of General Systems

ISSN: 0308-1079 (Print) 1563-5104 (Online) Journal homepage: http://www.tandfonline.com/loi/ggen20

A MATHEMATICAL THEORY OF DESIGN Modeling the Design Process (Part II)

DAN BRAHA & ODED MAIMON

To cite this article: DAN BRAHA & ODED MAIMON (1999) A MATHEMATICAL THEORY OF DESIGN Modeling the Design Process (Part II), International Journal of General Systems, 27:4-5, 319-347, DOI: 10.1080/03081079908962069

To link to this article: http://dx.doi.org/10.1080/03081079908962069

Published online: 31 May 2007.

Int. J. General Systems, Vol. 27(4–5), pp. 319–347
Reprints available directly from the publisher
Photocopying permitted by license only

A MATHEMATICAL THEORY OF DESIGN

Modeling the Design Process (Part II)

DAN BRAHA[a,*] and ODED MAIMON[b]

[a]*Department of Industrial Engineering, Faculty of Engineering, Tel-Aviv University, Ramat-Aviv, Tel-Aviv 69978, Israel;* [b]*Manufacturing Engineering Department, Boston University, 44 Cummington St., MA 02215, USA*

(Received 12 August 1993; In final form 15 April 1996)

This study presents a *Formal General Design Theory* (*FGDT*), a mathematical theory of design. The scope and objectives of FGDT, as well as modeling design artifacts were examined in the first part of this study. Here, the entire design process (termed also *idealized design process*) is conceptualized by a single operator which has several properties that immanently characterize the design process. The concept of a *generating set* for the design solution space is also introduced, and its relation with the axiomatic model of the design process is explored. The FGDT will provide insight into design practice and guidelines for developing CAD systems.

Keywords: Theory of design; design space; design process; general systems; design methodology; knowledge representation

1. INTRODUCTION AND INFORMAL PRESENTATION

In the first part of this study (Maimon and Braha, 1994a), the authors have introduced Formal General Design Theory (FGDT), a mathematical theory of design. A general model for the representation of design artifacts (constituting the *design space*) was constructed.

In addition to the design space model, a design theory should also include the *design world* (*problem-domain*), the *design model process* and their interaction. Constructing generic models to design processes

* Corresponding author.

can be delineated by different abstraction levels. Our approach is to develop a logical framework for describing design processes (Maimon and Braha, 1994b). This will probably includes the construction of a logical language framework, and design form-specifications mutual adaptation scheme. On a more abstract epistemological level, we may explore the interface between a design world and a design process. We focus here on a special aspect of this interface, namely the *transformations* between these systems (also termed as the *idealized design process*).

Figure 1 describes a design meta-model process interacting with a design world, both constitute a *design system*. Both the design model space and the design world space are regarded as general systems (see Definition 1). The design model process is viewed as a design problem solver mechanism. The *design system triangular cycle* interaction is described as follows:

- The different entities, concepts and design problems included in the design world are translated to an abstract design space, which forms the artifacts space of the design model process. This operation is carried out via by the *abstraction channel*. This step conceptualizes the issue of design modeling.
- The translated design problems are solved by applying different *solution methods* (procedures, heuristics) constituting the design

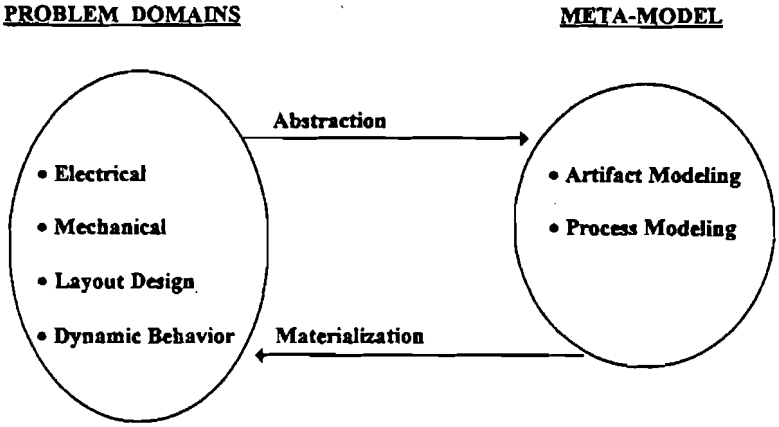

FIGURE 1 The triangular design process cycle.

model process. The design model process can be described as a goal directed derivation process from an initial set of specifications, using procedures for adaptively modifying pairs of design and requirements from one cycle (step) to the next. Future work will include a detailed model for the design model process. Here we conceptualize the entire design model process by a single operator, supplied by several properties which will *immanently* (so claimed) characterize *whatever* design model process.

- The terminal design solutions are translated backward, by the *materialization channel*, to the design world. This step conceptualizes the issue of implementing a design solution.

To illustrate the type of questions we shall be interested in, consider a scenario where the designer aims at solving a certain design problem (i.e., aims at finding an appropriate mapping between the design specifications and design solutions), given an initial set of partial designs expressed in the space of the design world. For solving the problem the designer has two alternatives. The first alternative is realized by first representing the set of partial designs in terms of the design model process. Then, by applying the design model process the designer obtains by the end of the process a set of design solutions, denoted by **A**. Finally, the set of design solutions is implemented by representing **A** in terms of the space of the design world space. Alternatively, the designer may directly solve the problem in terms of the design world process terminating with a set of design solutions, denoted by **B**. Now, among the three inclusion relationships between the sets **A** and **B**, which is the most 'appropriate' (or 'correct') one?

To illuminate this and other questions, we need a mathematical tool which applies set-theoretical topological techniques. It has been suggested that in certain occasions, standard general topologies are not suited to be applied to general systems and that suitable *generalized topologies* are needed (Kohout, 1990). Set-theoretical topological techniques employ subsets and mappings between them. It enables to work with sets of points instead of individuals points. For example, in the context of automata theory, it makes it possible to work with superstates, superinputs and superoutputs, instead of individuals states (see for example (Brave and Heymann, 1990), where the formulation of Hierarchical State Machines, HSM, is stated).

The description of this interface is simplified by introducing a set of axioms. These axioms deal with the relationships among three systems: the *design-world*, *design-process*, and the *designer's preferences* over partial solutions. The axioms are classified into two subgroups, termed as the *Internal* and *External* axioms. Each subgroup reflects a certain characteristic of the interface.

By utilizing the concept of generalized topology, the concept of a *generating set* of the design solutions space is introduced. Generating sets can be interpreted as a 'core' for generating the overall design solutions space. Besides its mathematical meaning, the engineering consequences of this notion would be to apply it in contexts, where an initial phase of off-line pre-processing, that involves finding a family of generating sets, can serve to reduce the complexity of the overall design process.

Organization Section 2 presents the basic concepts of FGDT. We formulate the *design space*, and identify the *design process* as a generalized topology. In Section 3 the overall design process is abstracted and axiomatized via two classes of '*design axioms*'. In Section 4 the concept of a *generating set* for the design solutions space is introduced. Finally, Section 5 concludes with some general remarks and directions for future work. For fluency of reading we present all the proofs in Appendix D.

2. PRELIMINARIES

A first step in formulating the design process is a general statement of design artifacts' representation. The following definition[1] presents only the essentials for understanding the forthcoming discussion:

DEFINITION 1 Artifact's Model—*An* artifact's model (*also termed the* design space) *is denoted by the tuple* $\langle M_0, C^0, M^* \rangle$. *Let us discuss each component in turn*,

Atomic Modules

M_0 denotes the set of *atomic modules*. *Atomic modules* cannot be defined in terms of other modules (except trivially by themselves). The

[1] For a detailed definition of a design system see 'Part I' of this study (Maimon and Braha, 1994a).

set M_0 represents primitive components which can be assembled to construct complex modules. Consider for example the design of a computer environment (e.g., communication networks): The basic modules can be programs, data files, subroutines. In the design of an analog circuit atomic modules represent resistors, capacitors, operational amplifiers, diodes.

Connections

C^0 denotes a preassigned *universal* set of *connections*. $c \in C^0$ denotes a collection of tuples, each represents a relation among the elements within the tuple. The relationships among modules may also be expressed in non-mathematical terms (e.g. in logical terms). Consider for example the design of an analog circuit; the connections among atomic modules may be represented by 'information' and 'physical' connections among capacitors, resistors etc.

Complex Modules (or A Behavioral Representation Scheme)

M^* is a *universal* set of *modules* e.g., $M^* = \{Atomic\ modules\} \cup \{Complex\ modules\}$, where '*Complex Modules*' is defined hierarchically in terms of predefined modules as follows. Let a system, $m \in M^*$, be defined by a set of $C \subseteq C^0$; such that $C = \{c_i\}_{i \in I}$, where $c_i = \{M_{i,k} = \langle m_{i1}, m_{i2}, \ldots, m_{in}\rangle_k$: $c_i(\langle m_{i1}, m_{i2}, \ldots, m_{in}\rangle)_k$ is true$\} \subseteq (M^*)^n$; with $c_i(\langle m_{i1}, m_{i2}, \ldots, m_{in}\rangle)_k$ being a formula (a rule or behavior) associated with the relation c_i, and $(M^*)^n$ is a Cartesian product of n copies of the set M^*. The k-th element $M_{i,k}$ of c_i, is termed as an *instantiation* of the relation c_i.

Hence, a system m is defined in terms of a set $\{M_{i,k}\}$ and a set C; such that $C \subseteq C^0$; and $M_{i,k}$ denotes an ordered set of modules. Denote $M = \cup_{i,k} M_{i,k}$ as the *carrier* set, and C as the *connection* set. For convenience the notation $m = \langle M, C\rangle$ is often used. Since this definition is hierarchical, the terms '*system*', '*complex module*' and '*module*' are used interchangeably.

The next definition identifies a design process as a generalized topology.

DEFINITION 2 A Design Process—*Defined as a topological space* $\langle M^*, U\rangle$, *where*

1. M^* denotes a set of modules representing partial designs. In topological terms (see Appendix A), it forms a carrier set.
2. $U: 2^{M^*} \rightarrow 2^{M^*}$ denotes a design process operator, representing the set of design solutions which is obtained (via the design process) given an initial set of partial designs. In topological terms, it forms a closure operator.

Remark 1.1 Tarski (1956) has formalized logical systems as a generalized topological space as follows:

- The set of all well-formed formulas of the logical system is taken as the carrier set.
- The consequence operator, which acts on the power set of all well-formed formulas to obtain new well-formed formulas, forms the closure operator.

Remark 1.2 A refined definition would be to define the design space as an abstract logic (Tarski, 1956). Definition 2 is sufficient for the following discussion on design-process axioms.

The operations described in Fig. 1 are formalized as follows:

DEFINITION 3 Design world process—*A generalized topology, $\langle M_p^*, U_p \rangle$ ('p' stands for a 'problem domain'), that represents the design world process. Similarly to Definition 2, M_p^* denotes the carrier set; U_p denotes the respective closure operator.*

Remark 2 The design model process is denoted, respectively, by $\langle M_m^*, U_m \rangle$ ('m' stands for a 'design model').

DEFINITION 4 Abstraction operator—*A mapping $\Upsilon: 2^{M_p^*} \rightarrow 2^{M_m^*}$. This mapping models the abstraction operation which is performed via the abstraction channel. It takes a set of modules (in the design world space) and transforms it into a compatible set of abstract modules (in the space of the design model).*

DEFINITION 5 Materialization operator—*A mapping $\Gamma: 2^{M_m^*} \rightarrow 2^{M_p^*}$. This mapping models the materialization operation which is performed via the materialization channel. It represents an (backward) operation from the design model into the design world. To express the essential nature of an abstraction, we assume throughout the discussion that Γ is related to Υ as follows: $\forall M_m \subseteq M_m^* : \Gamma(M_m) = \{m \in M_p^* : \Upsilon(m) \in M_m\}$.*

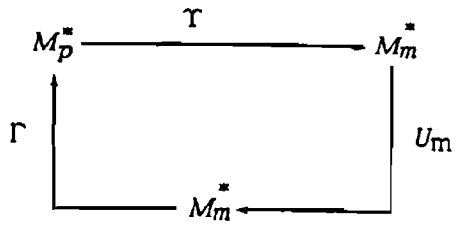

FIGURE 2 The operators' diagram.

The above definitions are summarized by Fig. 2.

These mappings are fairly expressive in general for introducing formal characteristic of the design world—design model interrelations. We next provide a more precise characterization of the abstraction mapping; and introduce a special type of abstraction mapping that is commonly encountered in practice. Let us first introduce several necessary definitions.

DEFINITION 6 Mapping on Relations—*Suppose that* $h: M_1^* \to M_2^*$ *is a mapping from* M_1^* *into* M_2^* *defined on the* same *universe of connections, and let* $m_1 = \langle M_1, C_1 \rangle \in M_1^*$. *For any relation* $c \in C_1$, $h(c)$ *indicates the relation on* M_2^* *defined by the following: suppose* $c = \{\langle m^0, m^1, \ldots, m^n \rangle \in M_1^n : c(\langle m^0, m^1, \ldots, m^n \rangle)$ is true$\}$, *then* $h(c) = \{\langle h(m^0), h(m^1), \ldots, h(m^n) \rangle \in (M_2^*)^n : c(\langle m^0, m^1, \ldots, m^n \rangle)$ is true$\}$, *and let* $h(C_1) = \{h(c) : c \in C_1\}$.

DEFINITION 7 Homomorphism—$m_2 = \langle M_2, C_2 \rangle \in M_2^*$ *is said to be homomorphic to* $m_1 = \langle M_1, C_1 \rangle \in M_1^*$, *if for each* $c \in C_1$, $h(c) \in C_2$.

Let us examine an abstraction mapping that satisfies the following properties:

1. *Additivity*—Let Υ be function from M_p^* onto M_m^* and $\Upsilon(S) = \cup_{m \in S} \Upsilon(m)$;
2. *Linearity*—$m_p = \langle M_p, C_p \rangle \in M_p^*$ and $m_m = \Upsilon(m_p)$, then $\Upsilon(\langle M_p, C_p \rangle) = \langle \Upsilon(M_p), \Upsilon(C_p) \rangle$.

We next present an important class of additive abstraction mappings which satisfy also the linearity property. The idea underlying the construction of this class is based on generating firstly the *basic modules* in M_m^* by partitioning the basic modules in M_p^*. Modules of higher-order are then defined recursively from this set, and M_p^*. This

procedure is often encountered in practice, where the domain of a design space is mapped and represented by some formal language. For example, Bondgraphs (Ulrich, 1988) and Flowgrams (Kim, 1978) are two languages for describing the exchange of energy in systems composed of a lumped parameter elements. In these languages generalized elements (a device or process) are identified (an element is a device or process which impresses a specified function on a quantity flowing through it, by inputting modules such as mass, energy, information, or a combination of the three).

DEFINITION 8 (Partition of M_p^*)—*Consider a regular*[2] *general system* M_p^*. *Let* $\pi(M_O)$ *be a partition of the set of basic modules, i.e. a family of non-empty subsets of* M_O *such that each element of* M_O *belongs to exactly one of these subsets. Formally,* $\pi(M_O) = \{\pi^i \mid \pi^i \in P(M_O),\ \pi^i \neq \phi, \cup_{i \in I} \phi^i = M_O,$ *and* $\pi^i \cap \pi^j = \phi$ *for all* $i, j \in I\}$. *We define the* Partition of M_p^* Under π *as a new general system* M_m^*, *which is defined recursively; and results with an abstraction mapping* $\Upsilon: M_p^* \rightarrow M_m^*$:

1. *Step 0*: (Basic modules $R_O \subseteq M_m^*$)—$\forall m$: $m \in M_O$ & $m \in \pi^i$, define a module $r_i \in R_O$; and let $\forall m_1, m_2 \in \pi^i$: $\Upsilon(m_1) = \Upsilon(m_2) = r_i$;
2. *Step k*: (Complex modules in M_m^*)—Let $m = \langle M, C \rangle$, if $\Upsilon(M)$ is already defined, define a module $r = \langle \Upsilon(M), \Upsilon(C) \rangle$; and let $\Upsilon(m) = r$.

Observation 1 π can be extended to M_p^* as follows: $\pi(M_p^*) = \{\pi^r = \Upsilon^{-1}(r) \mid \pi^r \in P(M_p^*),\ \pi^r \neq \phi,\ \cup_{r \in M_m^*} \pi^r = M_p^*,$ and $\pi^{r_1} \cap \pi^{r_2} = \phi$ for all $r_1, r_2 \in M_m^*\}$. Furthermore, Υ satisfies the linearity property.

The proof is clear and so is omitted.

Example 1 Let M_p^* be stated as: $M_O = \{m_1, m_2, m_3, m_4\}$, $C_O = \{c\}$, $m_5 = c(\langle 1, 3 \rangle)$, $m_6 = c(\langle 1, 2 \rangle)$, $m_7 = c(\langle m_5, m_6 \rangle)$, and $\pi(M_O) = \{\{m_1, m_2\}, \{m_3, m_4\}\}$. Applying Definition 8, M_m^* will be defined as follows: $\Upsilon(m_1) = \Upsilon(m_2) = r_1$; $\Upsilon(m_3) = \Upsilon(m_4) = r_2$; $\Upsilon(m_5) = r_3 = c(\langle r_1, r_2 \rangle)$; $\Upsilon(m_6) = r_4 = c(\langle r_1, r_1 \rangle)$; $\Upsilon(m_7) = r_5 = c(\langle r_3, r_4 \rangle)$.

It would be a useful property if enlarging the initial set of potential designs would not alter the current terminal set of design solutions,

[2] See Maimon and Braha (1994a).

viz., if the design operator will satisfy the *monotonicity* property: $M_1 \subseteq M_2 \subseteq U \Rightarrow U(M_1) \subseteq U(M_2)$. The following lemma shows that any topological space $\langle M^*, U\rangle$, whereby the design process operator satisfies the monotonicity property, has a fixed-set of designs, viz., a set of partial designs such that when given as an input to the design process operator will produce an *identical* set of design solutions (that is, $U(M) = M$).

LEMMA 1 *Let $\langle M^*, U\rangle$ be a topological space, and U be a monotonous design operator. Then U has a fix-point in $P(M^*)$ i.e., there exists $M^\circ \subseteq M^*$ such that $U(M^\circ) = M^\circ$.*

We term a design solution as '*successful*' if it satisfies the initial specifications set. A design process operator is termed '*locally useful in M*' if the initial set of designs M is included in the terminal set of design solutions i.e., if $M \subseteq U(M)$. U is termed '*useful*' when U is locally useful in M iff M is a set of successful designs. A direct consequence of these definitions and Lemma 1 will be:

COROLLARY 1 *Let $\langle M^*, U\rangle$ be a topological space. U is* monotonous *and* useful. *Then M° is the maximal (in terms of set inclusion) set of successful designs.*

We assume the designer has preference relation among the different complex modules. The definition of a designer's preferences assumes the existence of a utility function translating a preference order over the reals. The designer's preferences are modeled thanks to a reflexive and transitive relation, defined as follows:

DEFINITION 9 Designer Preferences—*Assume M^*_m can be ordered by a binary relation of "preference or indifferent" denoted $\succeq_m$ which satisfies:*

1. $\forall m_1, m_2 \in M^*_m$: $m_1 \succeq_m m_2$ or $m_2 \succeq_m m_1$. It is interpreted as "the designer strictly prefers or indifferent between the modules m_1 and m_2 relative to a goal set of specifications."
2. The relation $\succeq_m$ is transitive, i.e., $(m_1 \succeq_m m_2 \;\&\; m_2 \succeq_m m_3) \Rightarrow m_1 \succeq_m m_3$.

Remark 3 Similarly, denote by $\succeq_p$ the designer's preferences over M^*_p.

The following generalizes the previous definition to account for preferences over sets of modules.

DEFINITION 10 Strong-Pareto: *We denote* $M_1 \succeq_m M_2$ *iff* $\forall m_1 \in M_1$, $m_2 \in M_2$: $m_1 \succeq_m m_2$. *This relation is not necessarily complete.*

Remark 4 Similarly define *Strong-Pareto* over sets of modules in M_p^*.

Example 2 Design of Mechanical Fasteners.

To illustrate some of the abstract concept introduced above, let us examine a typical example characterizing a wide range of techniques in Artificial Intelligence (AI) known as *Cased Based Design*. Case-based problem solving is based on the premise that a machine problem solver makes use of its experiences (cases) in solving new problems instead of solving every new problem from scratch (Kolodner *et al.*, 1985). It is only very recently that this aspect of the design process, the use of past cases, is beginning to be recognized in the design automation literature (Freeman and Newell, 1971; Lenat, 1984; Mostow, 1985; Gero, 1987; Huhns and Acosta, 1987; Maher and Zhao, 1987; Mostow and Barley, 1987; Ulrich, 1988). The following example of mechanical fasteners design, inspired by (Ulrich, 1988), will serve to illustrate some of the terminology introduced in this section.

The Knowledge-Base and Task

The design of mechanical fasteners is a common design problem in the realm of mechanical engineering, and is chosen to illustrate the concepts introduced. The function of a fastener is to hold two or more parts together. There are numerous types and numbers of existing mechanical fasteners. The proposed problem is to design a new fastener according to certain given specifications based on the knowledge that we already have from the existing fasteners.

The first step is to build a data base of this knowledge. The existing designs are represented in a set of design descriptions that include structural, functional and the causal relationships between function and structure. The structural description of the device and its components are listed. In the case of fasteners, the general hierarchy is composed of the five components that are common to all fasteners; drive, head, body, tail and a tip as shown in Fig. 3.

The behavioral abstraction of a design is usually determined by the structural hierarchy. For instance, the function of a fastener is to hold

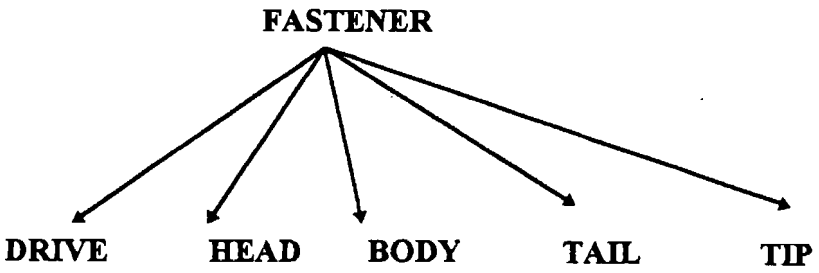

FIGURE 3 Hierarchy structure of a fastener.

two or more parts together. This is done by the fastener which transmits the load to the fastened parts which is the entity behavior. The relation is usually in the form of a causal one governed by the physical laws. The behavior of the components follows the same principle. For an illustration of the relationships; the strength of the fastener is determined by the static fatigue, stress rupture, and impact strength. The fatigue is determined by the material, head size, thread size, threaded length, fillet, fabrication, and surface structure. Those properties, in turn, are determined by the physical attributes of the components and entity. The same would apply for the rest of the properties of the fastener. This would be captured and represented as the causal and factual knowledge of the system in a causal network and as an if–then structure. These structures would link the functions to the attributes and the attributes to the components.

Now this knowledge can be used to synthesize new knowledge. To perform this synthesis task, the system is designed to transform a functional specification to a structural description. A typical input to a design system is a conjunction of functional attributes. The program task is to propose a fastener design which will meet a particular set of functional requirements. Consequently, the result of applying the above design procedure to an input specification is a set of potential designs, each of which consists of a set of five structural attributes corresponding to the five parts of the fastener structure.

A Design Process Model Perspective

Following our definitions and notations we note the following:

- *Design World Space* (M_p^*)—The design world space consists of numerous types of existing mechanical fasteners e.g., roll-pin, dowell

pin, hex bolt, socket-head screw, barbed fastener and cotter pin (see Fig. 4).

- *Design Model Space*—The design model space consists of a knowledge base of network-like representation of fasteners. *Basic modules* represent structural and functional properties of a fastener, whereas *Complex modules* represent the causal relationships between function and structure. Complex modules are represented by a network- and -graph scheme (i.e., a node with several antecedents is caused by their conjunction) where nodes represent functional or structural

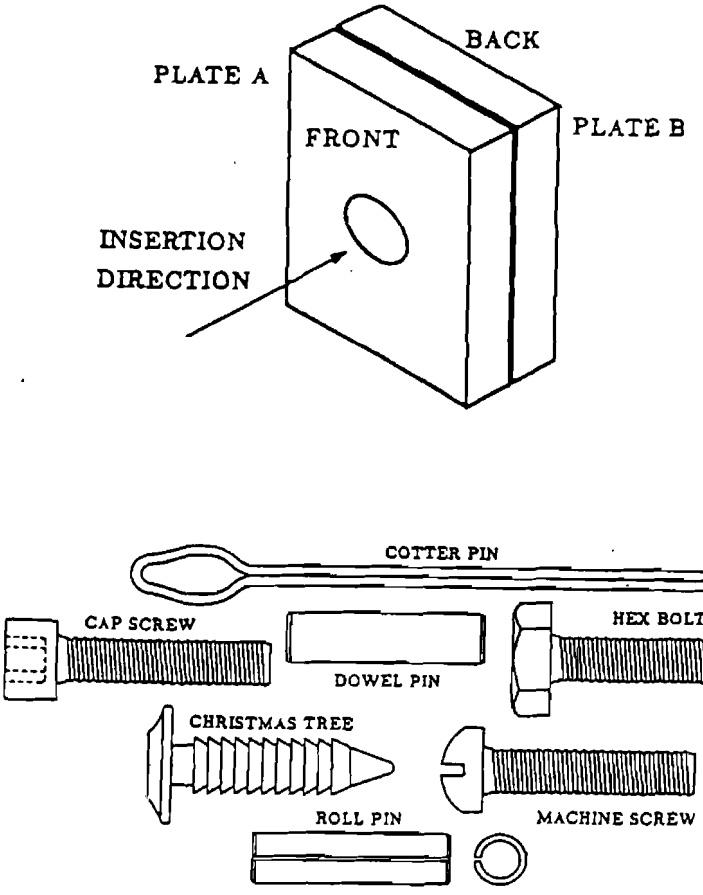

FIGURE 4 Fasteners in the knowledge base.

attributes and links (*connections*) represent causal relations. For example, AND ([HEAD_SIZE LARGE, BODY_SIZE LARGE, BODY_LENGTH LARGE, STRENGTH_MATERIAL HIGH], STRENGTH HIGH) represents a factual knowledge of the system which links physical attributes (basic modules HEAD_SIZE, BODY_SIZE, . . .) of a fastener with its strength function (basic module STRENGTH).

- *Abstraction Operator* (Υ)—The Abstraction operator takes a set of fasteners (in the design world space) and transforms it into a compatible set of network-like representation of fasteners (in the design model space).
- *Materialization Operator* (Γ)—The Materialization operator takes a set of network-like representation of fasteners (in the design model space) and transforms it into a compatible set of fasteners.
- *Design Model Process Operator* (U_m)—The design model process operator takes as input, given a conjunction of functional attributes, a set of network-like representation of existing fasteners (S). The result of applying U_m, is a set of candidate designs ($U_m(S)$).

3. DESIGN PROCESS AXIOMS

In this section, the interface between the design world and the design model is examined. We shall introduce several 'natural' axioms, which are believed as characterizing any design system. This group of axioms may or may not be satisfactory or complete. Although we expect that they are reasonable in some situations, we take no stance on this here.

The axioms are classified into two groups. The first group, termed as the set of *Internal Axioms*, is characterized by internal properties relating to the *design world—design model* relationships. A design system which satisfies the *Internal Axioms* will be termed *well-defined*. The second group, termed the set of *External Axioms*, will take the *designer's preferences* into considerations.

Group of Internal Axioms

DEFINITION 11 Axiom of Validity (AV)—*The generalized topology* $\langle M_m^*, U_m \rangle$ *is assumed to satisfy the following condition:* $\forall S \subseteq M_p^* : \Gamma(U_m(\Upsilon(S))) \subseteq U_p(S)$.

The axiom of validity implies that any design solution that can be obtained by the design model process can also be obtained by the design world process, viz. the design model process is *weaker* than the design world process. In other words, no module that is not part of the problem domain space should be constructed by the design model process. In general, a proper inclusion is expected because of the abstraction operation.

DEFINITION 12 Axiom of Completeness (*AC*)—*The generalized topology* $\langle M_m^*, U_m \rangle$ *is said to be complete (see Appendix A) if it satisfies* $\forall S \subseteq M_p^* : \Upsilon(U_p(S)) \subseteq U_m(\Upsilon(S))$.

If both the *Validity* and *Completeness* axioms hold, we term the design system as *well-defined*.

The design world process operator, U_m, can be simulated by the design world process operator, U_p, in the following way. First, the problem is solved in the design world, and afterwards the derived solutions are expressed in terms of the design model space by applying the abstraction operator Υ. The *AC* claims that this simulation cannot perform better than solving the problem initially by the design model process. This property also stress the importance in applying a model—it provides a necessary condition for applying a model as a mean for solving design problems, thus accentuates the importance of design models.

Remark 5 *AC* can be reduced (thus dependent) to *AV* in the following special case. Let Υ be a (1-1) correspondence $\Upsilon : M_p^* \leftrightarrow M_m^*$ such that $m_1 \in U_p(S) \Longleftrightarrow \Upsilon(m_1) \in U_m(\Upsilon(S))$. That is, the generalized topologies $\langle M_p^*, U_p \rangle$ and $\langle M_m^*, U_m \rangle$ are *isomorphic*. Hence, by the Validity property we obtain: $\forall \tilde{S} \subseteq M_m^* : (\Gamma)^{-1}(U_p(\Gamma(\tilde{S}))) \subseteq U_m(\tilde{S})$. Designating $S = \Gamma(\tilde{S})$ and substituting in the previous expression, we obtain $\forall \tilde{S} \subseteq M_m^* : \Upsilon(U_p(S)) \subseteq U_m(\Upsilon(S))$ which is exactly *AC*.

Remark 6 The above concepts and models can also be valuable in developing models that describe the activity of an organism in the external environment. That is to say, the activity of an organism can be viewed as a transformation of input stimuli which produces behavior.

Group of External Axioms

DEFINITION 13 Axiom of Isomorphism (*AI*)—*The partially-ordered sets* $(M_p^*, \succeq_p)$ & $(M_m^*, \succeq_m)$ *are* isomorphic *under the mapping* Υ, *i.e.* $M_1 \succeq_p M_2$ *iff* $\Upsilon(M_1) \succeq_m \Upsilon(M_2)$.

The *AI* is rather natural; the preference relation over complex modules is preserved under the abstraction operator, despite the fact the design space M_p^* is mapped to another space M_m^*.

DEFINITION 14 Strong-Pareto Under Problem-Domain Closure (*SPDC*)—*If* $M_1 \succeq_p M_2$ *then* $\sim[U_p(M_2) \succeq_p U_p(M_1)]$.

That is, if each module in M_1 dominates every module in M_2, the designer cannot yield from M_2 better solutions (in terms of 'strong' dominance). Note, however, that the designer can obtain one (or more) solutions in M_2 that are preferable to some solution(s) in M_1.

Similarly we define:

DEFINITION 15 Strong-Pareto Under Model-Space Closure (*SPMC*)—*If* $M_1 \succeq_m M_2$ *then* $\sim[U_m(M_2) \succeq_m U_m(M_1)]$.

The *SPMC* is interpreted similarly to the *SPDC*.

THEOREM 1 (CONSISTENCY):

1. *The AC, AI & SPMC imply the SPDC.*
2. *The AV, AI & SPDC imply the SPMC.*

Observation 2 Let Υ be a function from M_p^* onto M_m^* and $\Upsilon(S) = \cup_{m \in S}\Upsilon(m)$. If *AV & AC* are satisfied, then $\forall S \subseteq M_p^*: \Upsilon(U_p(S)) = U_m(\Upsilon(S))$.

We next show that the existence of a fix point is preserved under homeomorphism between the design world and the design model, viz., it is a topological property.

LEMMA 2 *Let* $\langle M_p^*, U_p\rangle$ *and* $\langle M_m^*, U_m\rangle$ *be two generalized topologies representing respectively the design world and the design model. If* U_p *is monotonous,* Υ *is additive and AC & AV are satisfied, then* U_m *is monotonous.*

COROLLARY 2 *If* M_p° *is a fix-set of designs of* $\langle M_p^*, U_p\rangle$, *then* $\Upsilon(M_p^\circ)$ *is a fix-set of designs of* $\langle M_m^*, U_m\rangle$.

The construction of 'well-defined' design systems requires in part the correctness of the *Validity* and the *Completeness* properties, thus leads us naturally to inquire the inherent complexity in verifying these properties. For ease of exposition, we consider in the sequel a slightly different version of the above properties, which is done by replacing

the universal quantifier '$\forall S$' with the existential one '$\exists S$'. This modification reflects also the satisficing nature of design (Simon, 1981)—in place of verifying the correctness of the *Validity* property for *every* subset of the problem domain space, the designer may be satisfied by verifying the *existence* of such a set. The precise computational problem is expressed as follows:

DEFINITION 16 *The Quasi-Validity problem (QVP)—Given a design system (including the design world, design model, Abstraction operator and Materialization operator), the decision problem, termed the Quasi-Validity problem, concerns the existence of the following property:* '$\exists S \subseteq M_{\mathrm{p}}^{*} : \Gamma(U_{\mathrm{m}}(\Upsilon(S))) \subseteq U_{\mathrm{p}}(S)$'.

Remark 7 Similarly to Definition 16, one can define the *Quasi-Completeness* problem as the decision problem concerning the existence of the following property: '$\exists S \subseteq M_{\mathrm{p}}^{*}$: $\Upsilon(U_{\mathrm{p}}(S)) \subseteq U_{\mathrm{m}}(\Upsilon(S))$'. To simplify the presentation we refer solely in the following to the *Quasi-Validity* problem.

The following result concerns the computational complexity of the *Quasi-Validity* problem. We assume that the number of modules and connections is finite, and that their representation is efficient (i.e., the size of the module's and connection's representation is polynomial). We also assume that the *Abstraction* and *Materialization* operators are efficiently computed.

THEOREM 2 *The Quasi-Validity problem is* Np-complete (*in the number of modules and connections*).

If no upper bound is placed on k—the length of the sequence of integers $i_1, i_2, \ldots, i_k$—the *BPCP* becomes undecidable. Undecidable problems arise in a variety of areas. This problem, known as Post's Correspondence Problem, is used as a valuable tool in establishing other problems to be undecidable. In particular, following the proof of Theorem 2, we conclude that there are instances for which the *Quasi-Validity* problem is undecidable. Specifically, the *QVP* becomes undecidable for instances that have *unbounded* space of complex modules. The engineering implication might be that a design process cannot be verified for validation and completeness without the intervention of a human-being designer. It also confines our desiderata for devising an

artificial and completely automatic design system problem solver. However, it is well known that the practical complexity limits of this verification problem (e.g. of existing expert systems for design) are considerably lower. Some currently design systems cannot be in fact completely verified. The focus is thus on developing verification methods that can be practically implemented and guarantee only that well over a certain percent of all possibilities be verified (that is, the design process be almost valid and complete).

4. GENERATING SETS

Consider a scenario in which the design model process which is applied (that is, the design space representation, and design process operator) is fixed, and we know beforehand that the designer will to be asked to solve many design problems with respect to this model. The designer can improve her performance in either two ways. Intuitively, it is clear that given such a model, the designer can try to devise special purpose, domain dependent procedures for solving design problems with respect to this particular model, instead of using general design problem solving scheme. However, this approach is not very helpful to the designer, since this approach may be rather costly and time consuming; and moreover the designer should be provided with a way in which she can obtain such special purpose algorithms. Another strategy is to consider the latter problem of how an initial phase of off-line preprocessing, performed on the design model process, can serve to reduce the complexity of the on-line design process. This strategy is especially relevant in design systems where designers might be encountered with many potential design problems during their on-line activity. In such cases performing rather extensive preprocessing phase might be computationally profitable.

We next introduce the concept of generating sets for a design process. Generating sets play a similar role to the concept of topological bases appearing in general topology. Consequently, we are led to consider the engineering relevance of generating sets in contexts where an initial phase of off-line pre-processing, that involves finding a family of generating sets, can considerably improve the on-line complexity of the overall design process. More precisely, any set of design solutions

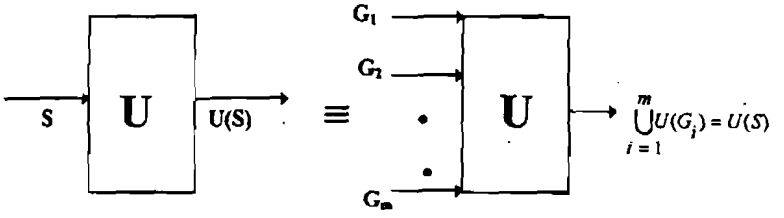

FIGURE 5 A schematic description of generating sets.

$U(S)$, which is obtained during the on-line activity, can be determined as the union of previously derived solutions $U(G_i)$, obtained during the initial phase of off-line pre-processing. The concept is illustrated schematically in Fig. 5.

DEFINITION 17 Generating Set—*A family $\Phi = \{G_i\}_{i \in I}$ is termed a generating set for a generalized topology $\langle M_p, U_p \rangle$, if there exists a subfamily $\{G_i\}_{i \in \Omega}$ $(\Omega \subseteq I)$ such that $\forall S \subseteq M_p^*: U_p(S) = \cup_{i \in \Omega} U_p(G_i)$.*

Remark 8 Similarly, we define a generating set for $\langle M_m, U_m \rangle$.

DEFINITION 18 Design System Character—*Denote* $Ca(M^*) = \inf\{|\Phi^j| : \Phi^j = \{G_i^j\}_{i \in I}$ *is a generating set for a generalized system* $\langle M^*, U \rangle\}$. *$Ca(M^*)$ is properly defined, since the set of cardinal numbers is well-ordered by the natural relation* $\leq$.

Example 3 **Mechanical Fasteners Design** (continuing Example 2)—Consider a scenario where the fasteners are grouped in accordance with the measurable properties of their *materials* such as geometrical tolerances, density, tensile strength, or hardness. In particular, the fasteners are made up from two types of materials: *mild steel* and *cast iron*. The tensile constraint prevents a fastener from being constructed from different materials. Consequently, this constraint partitions the knowledge base of past cases into two disjoint sets (see Fig. 6):

$$\underbrace{\{F_1, F_2, F_3, F_4\}}_{\text{Mild Steel}}$$

and

$$\underbrace{\{F_5, F_6, F_7, F_8\}}_{\text{Iron Cast}}.$$

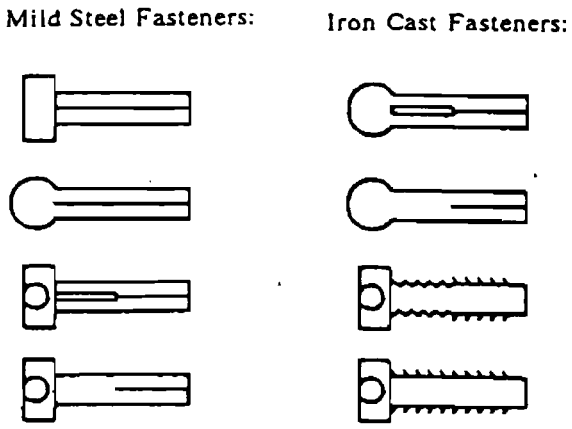

FIGURE 6 Partition of the fasteners' data-base.

Accordingly the generating sets are:

Mild Steel Fasteners: $G_1 = \{F_1\}; G_2 = \{F_2\}; \ldots, G_{15} = \{F_1, F_2, F_3, F_4\}$;
Iron Cast Fasteners: $G_{16} = \{F_5\}; G_{17} = \{F_6\}; \ldots, G_{30} = \{F_5, F_6, F_7, F_8\}$.

Consider the design problem is to construct a new fastener which has the properties of *high strength, high precision and* being *retractable*; and the initial set of partial solutions (the knowledge base of existing fasteners) is $S = \{F_1, F_2, F_5, F_6\}$. Applying the design process, U, yields $U(S) = U(F_1, F_2) \cup U(F_5, F_6) = U(G_5) \cup U(G_{20})$. In this case, the terminating design solutions set is determined by the generating set $G_5 = \{F_1, F_2\}$ and $G_{20} = \{F_5, F_6\}$.

Note that the family of generating sets consists of 30 elements, whereas the family of all possible subsets of fasteners has 255 elements $(2^8 - 1)$. Thus (following Definition 18) the *design system character* equals $Ca(M^*) = 30$.

Let us present some general properties characterizing generating sets and their relations with the axiomatic model of the design process.

LEMMA 3 *Let* $G_1, G_2 \in \Phi$. *Then, if* $m \in U_p(G_1) \cap U_p(G_2)$, *there exists* $G \in \Phi$ *such that* $m \in U_p(G) \subseteq U_p(G_1) \cap U_p(G_2)$.

LEMMA 4 *Assume* $Ca(M^*) \leq t$. *Then for every collection* $\{G_i\}_i \in I$, *there exists a set* $I_0 \subseteq I$ such that $|I_0| \leq t$ and $\cup_{i \in I_0} U_p(G_i) = \cup_{i \in I} U_p(G_i)$.

THEOREM 3 *Let t be a cardinal number, and suppose $Ca(M^*) \leq t$. Then, for every generating set Φ (for M^*), there exists a generating set Φ_0 such that $|\Phi_0| \leq t$ and $\Phi_0 \subseteq \Phi$.*

THEOREM 4 *Consider AV & AC are satisfied, $\Phi = \{G_i\}$ a generating set for the generalized topology $\langle M_p, U_p \rangle$ and Υ satisfies the conditions set out in Observation 2. Then $\{\Upsilon(G_i)\}$ is a generating set for the generalized topology $\langle M_m, U_m \rangle$.*

An essential nature of the abstraction principle is that this mapping contributes to reduce the complexity of solving the design problem. This beneficial property has two facets:

- *Computational complexity*—saving in time and space resources. We will return to this issue latter on.
- *Descriptive (or Systems) complexity*—saving in the amount of information required to describe the design problem. For example, Suh *et al.* (1978) have suggested a measure of information complexity which is based on the concept of entropy prevailing in information systems theory.

The latter type of measure links between the structure of the design representation space and the design problem-solver (or design process algorithm). Different abstractions of design systems give the notion of complexity different meanings; each of which requires a special discussion. Despite the differences in complexities of the various systems types, two general principles of systems complexity can be recognized (Lõfgren, 1977; Cornacchio, 1977). These principles can be applied as guidelines for a comprehensive study of systems complexity:

1. *Principle of Information*—According to this principle the complexity of a system should be proportional to the amount of information (The term "information" can be used in three meanings; syntactic, semantic or pragmatic aspects) required to describe the system. One of the simplest means for expressing a *descriptive complexity*), is to measure it by the number of entities involved in the system (e.g., modules and connections) and the variety of interdependence among the entities (represented by instantiations). Indeed, everything else being the same, our ability to cope or comprehend with a system tends to decrease when the number of

entities involved or the variety of their interconnections increase. There are many different ways in which descriptive complexity can be expressed. However, each of them must satisfy some general axioms (requirements) described as follows (Casti, 1979; Gottinger, 1975):

Let M^* denotes the universal set of modules; let $P(M^*)$ denote the power set of M^*. Then a function $W_{M^*}: P(M^*) \to R$ that satisfies the following axioms will denote a *measure of descriptive complexity* within the set M^*.

A1. $W_{M^*}(\phi) = 0$ (Normalization);

A2. $A \subseteq B \Rightarrow W_{M^*}(A) \leq W_{M^*}(B)$ (Monotonicity);

A3. *A is a homomorphic image of* $B \Rightarrow W_{M^*}(A) \leq W_{M^*}(B)$ (Abstraction); Axioms A1–A2 ensure that descriptive complexity measures will be non-negative. Axioms A2–A3 express the expectancy for a reduction in the complexity measure as a result of a system abstraction. Axiom A4 is also intuitive; it says that relabelling the entities in one system does not change the corresponding complexity measure.

2. *Principle of Uncertainty*—According to this principle, systems complexity should be proportional to the amount of information needed to resolve any uncertainty associated with the system involved. Measures that deal with this aspect of complexity are termed *Uncertainty measures.*

We suggest, here, that the design system character $Ca(\cdot)$ may be an appropriate *representative* of *structural complexity measures*. That is, it complies with the general descriptive complexity Axioms A1–A3. Moreover, since it is invariant (in form) of *any* problem domain it is adequately suited as a *generic* measure. The following corollary strengthen this supposition, since it is consistent with our expectancy for a reduction in the corresponding complexity measure due to abstraction (i.e., in accordance with Axiom A3 above).

COROLLARY 3 *If* Υ *is an additive mapping and AV & AC hold, then* $Ca(M_m^*) \leq Ca(M_p^*)$.

It was suggested earlier that in special cases an efficient strategy would be to perform an initial phase of off-line preprocessing that

involves finding a family of generating sets—prior to carrying out the design process (on-line). Obviously the computational complexity of applying this strategy is intimately related to finding a generating sets family (which has predefined properties such as minimum cardinality), for the design process. The general problem of finding a generating sets family is shown here to be intractable. Intractability puts limits on finding efficient algorithms for a large class of design problems. Nevertheless, it may become tractable by restricting to domains having special structures and properties. The general problem and result are formulated as follows:

DEFINITION 19 Generating Sets Problem (*GSP*)—*Given a Design Process* $\langle M, U \rangle$, *is there a collection* $\Phi = \{G_i\}_{i \in I}$ *with* $Ca(M^*) = k$?

For simplicity we consider that the number of modules and connections is finite, and that their representation is efficient. Moreover it is assumed that the design process operator U is efficiently computed. The latter assumption presupposes the use of efficient heuristics and strategies that look for satisfying—in contrast to optimal—design solutions.

LEMMA 5 *The* GSP *is* NP-complete.

5. CONCLUDING REMARKS

This study aims at the *gradual* development of a theory, based on a careful analysis of the ordinary everyday interpretation of design facts. This preliminary stage of transition from unmathematical plausibility considerations to the formal procedure of mathematics is necessarily *heuristic*. Its first applications (which serve to corroborate the theory) are necessarily to elementary design situations where no theory is required for them.

This undertaking can provide a common framework for various engineering design problems, consequently to rectify the semantic pollution prevailing in this area. The more engineering sciences breaks into subgroups, and a less communication is possible among the disciplines. However, the greater chance there is that the total growth of knowledge is being slowed by the loss of relevant communications. It is one of the main objectives of a mathematical general design theory to develop a

framework which enable one specialist to obtain relevant communications from others and build up expert knowledge.

As shown before, FGDT is capable of representing design processes. In particular, it supports the representation of models, and the mappings required by the correspondence model. The design process in the real knowledge (termed also the *'real' design process*) will be viewed as a goal-directed evolutionary process which starts with an initial set of design specifications. By adaptively modifying pairs of designs and specifications from one cycle (step) to the next, the designer derives a design solution (Maimon and Braha, 1994b). FGDT can therefore be used to manually construct design processes. However, the mapping between the function (specifications) and the attribute properties (modules) cannot be easily manually coded. Therefore, future research need to address the creation of the mapping between the function and attribute topologies.

The next stage develops when the theory will be applied to somewhat more complicated, less obvious situations. Beyond this lies the field of real success: genuine prediction by theory, and genuine applications in the form of intelligent CAD tools for the design engineer. It is well known that all mathematized sciences have gone through these successive phases of evolution.

APPENDIX A

Basic Notions of Generalized Topological Spaces

1. *Generalized topology*—Let 2^S be the power set of an arbitrary set S. Define the mapping $U: 2^S \rightarrow 2^S$, then the mapping U is called the *closure* of the topology and the tuple $\langle S, U \rangle$ designate the topological space without axioms, formed by the closure U and the *carrier* set S.
2. *Continuous mapping*—A mapping f from a topological space $\langle S_1, U_1 \rangle$ to a topological space $\langle S_2, U_2 \rangle$ which satisfies: $X \subseteq S_1 \Rightarrow f[U_1(X)] \subseteq U_2[f(X)]$.
3. *Homeomorphism*—A bijective mapping f for subspaces of two topological spaces such that both f and f^{-1} are continuous functions.
4. *Topological property*—A property which is preserved under homeomorphism.

An illustrative example—Tarski (1956) has formalized logical system as a generalized topological space as follows,

1. The set of all well-formed formulas of the logical system is taken as the carrier set.
2. The consequence operator, which acts on the power set of all well-formed formulas to obtain new well-formed formulas, forms the closure operator.

APPENDIX B

Bounded Post Correspondence Problem (BPCP) (Garey and Johnson, 1979)

The BPCP is expressed in terms of the following:

- Finite alphabet Σ.
- Two sequences $a=(a_1, a_2, \ldots, a_n)$ and $b=(b_1, b_2, \ldots, b_n)$ of strings from Σ^*.
- Positive integer $K \leq n$.

Now, BPCP can be formulated as: Is there a sequence $i_1, i_2, \ldots, i_k$ where $k \leq K$, positive integers less equal to n, such that the two strings $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ and $b_{i_1}, b_{i_2}, \ldots, b_{i_k}$ are identical.

APPENDIX C

SET BASIS Problem (Garey and Johnson, 1979)

The SET BASIS problem is expressed in terms of the following:

- INSTANCE: Collection C of subsets of a finite set S. Positive integer $K \leq |S|$.

Now, SET BASIS can be formulated as: Is there a collection B of subsets of S with $|B| = K$ such that, for each $c \in C$, there is a subcollection of B whose union is exactly c?

APPENDIX D (PROOFS)

Proof of Lemma 1 Let us examine the set $B=\{M \in M^* : M \subseteq U(M)\}$. B is not empty since $\phi \in B$. Now, define $M^o = \cup_{M \in B} M$. Let

us show that $U(M^\circ) = M^\circ$.

Step 1: $M^\circ \subseteq \cup_{M \in B} U(M) - M \subseteq U(M)$ for every $M \in B$, hence $M^\circ = \cup_{M \in B} M \subseteq \cup_{M \in B} U(M)$.

Step 2: $\cup_{M \in B} U(M) \subseteq M^\circ$—For every $M \in B$, $M \subseteq M^\circ$. Hence the monotonicity of U implies $U(M) \subseteq U(M^\circ)$ for every $M \in B$. Thus, it is concluded that $U_{M \in B} U(M) \subseteq U(M^\circ)$.

Step 3: $M^\circ \subseteq U(M^\circ)$—A direct result from Steps 1 and 2.

Step 4: $U(M^\circ) \subseteq M^\circ$—From Step 3 we obtain $M^\circ \subseteq U(M^\circ)$ and the monotonicity of U implies $U(M^\circ) \subseteq U(U(M^\circ))$, hence $U(M^\circ) \in B$, viz., $U(M^\circ) \subseteq M^\circ$.

Step 5: $U(M^\circ) = M^\circ$—A direct consequence of Steps 3 and 4.

Proof of Theorem 1

Part 1 Let $M_1 \succeq_p M_2$, and infer the following: $M_1 \succeq_p M_2 \overset{(AI)}{\Leftrightarrow} \Upsilon(M_1) \succeq_m \Upsilon(M_2) \overset{(SPMC)}{\Leftrightarrow} \sim[U_m(\Upsilon(M_2)) \succeq_m U_m(\Upsilon(M_1))]$. Since $\Upsilon(U_p(M_1)) \subseteq U_m(\Upsilon(M_1))$ & $\Upsilon(U_p(M_2)) \subseteq U_m(\Upsilon(M_2))$ and by applying the *strong Pareto* definition, we conclude $\sim[\Upsilon(U_p(M_2)) \succeq_m \Upsilon(U_p(M_1))]$. Finally, by the *AI* axiom, we obtain $\sim[U_p(M_2) \succeq_p U_p(M_1)]$, which concludes the proof. ■

Part 2 Let $M_1 \succeq_m M_2$ and infer the following: $M_1 \succeq_m M_2 \overset{(AI)}{\Leftrightarrow} \Gamma(M_1) \succeq_p \Gamma(M_2) \overset{(SPMC)}{\Leftrightarrow} \sim[U_p(\Gamma(M_2)) \succeq_p U_p(\Gamma(M_1))]$. Since $\Gamma(U_m(\Upsilon(\Gamma(M_1))) \subseteq U_p(\Gamma(M_1))$ & $\Gamma(U_m(\Upsilon(\Gamma(M_2)))) \subseteq U_p(\Gamma(M_2))$ and by applying the *strong Pareto* definition, we conclude $\sim[\Gamma(U_m(M_2)) \succeq_p \Gamma(U_m(M_2))]$. Finally, by the *AI* axiom, we obtain $\sim[U_m(M_2) \succeq_m U_m(M_1)]$, which concludes the proof. ■

Proof of Observation 2 By the *Validity* property $\forall S \subseteq M_p^*$: $\Gamma(U_m(\Upsilon(S))) \subseteq U_p(S)$. Since Υ is onto we infer: $\forall S \subseteq M_p^*$: $\Upsilon[\Gamma(U_m(\Upsilon(S)))] \subseteq \Upsilon(U_p(S)) \Rightarrow U_m(\Upsilon(S)) \subseteq \Upsilon(U_p(S))$. Hence, the result is inferred, by applying the latter term and the *Completeness* property *AC*. ■

Proof of Lemma 2 Let $\tilde{S}_1 \subseteq \tilde{S}_2 \subseteq M_m^*$. Since Υ is onto and additive $\exists S_1, \exists S_2$: $(S_1 \subseteq S_2 \subseteq M_m^*$ & $\Upsilon(S_1) = \tilde{S}_1$ & $\Upsilon(S_2) = \tilde{S}_2)$. By Observation 2, we obtain $\Upsilon(U_p(S_1)) = U_m(\Upsilon(S_1))$ and $\Upsilon(U_p(S_2)) = U_m(\Upsilon(S_2))$. The monotonicity of U_p implies $U_p(S_1) \subseteq U_p(S_2)$, and hence $\Upsilon(U_p(S_1)) \subseteq \Upsilon(U_p(S_2))$. Thus $U_m(\Upsilon(S_1)) \subseteq U_m(\Upsilon(S_2))$, viz., $U_m(\tilde{S}_1) \subseteq U_m(\tilde{S}_2)$ as required. ■

Proof of Corollary 2 The existence of a fix-point of $\langle M_m^*, U_m \rangle$ is resulted by the Additivity of U_m. Let M_p° be a fix-point of $\langle M_p^*, U_p \rangle$, viz., $U_p(M_p^\circ) = M_p^\circ$. Now, by Observation 2 $\Upsilon(U_p(M_p^\circ)) = U_m(\Upsilon(M_p^\circ))$ which also means $\Upsilon(M_p^\circ = U_m(\Upsilon(M_p^\circ))$, thus $\Upsilon(M_p^\circ)$ is a fix-point of $\langle M_m^*, U_m \rangle$. ■

Proof of Theorem 2 It is easy to see that the $QVP \in NP$. Let us transform *BPCP* (see Appendix B) to the *Quasi-Validity* problem. Let a finite alphabet Σ, sequences $a = \{a_1, a_2, \ldots, a_n\}$ and $b = \{b_1, b_2, \ldots, b_n\}$ of strings from Σ^*, and a positive integer $K \leq n$ form an arbitrary instance of *BPC*. We shall construct sets M_p^* and M_m^*, operators Υ, U_p and U_m, such that $\exists S \subseteq M_p^* : \Gamma(U_m(\Upsilon(S))] \subseteq U_p(S)$ iff the reply to *BPCP* is the affirmative:

1. Let N be the alphabet $N = \{1, 2, 3, \ldots, n\}$ and $\tilde{\Sigma} = \{a_1, a_2, \ldots, a_n\} \cup \{b_1, b_2, \ldots, b_n\} \cup N$, and define the following sets: $M_p^* = \tilde{\Sigma}_K^*$ and $M_m^* = \tilde{\Sigma}_K^*$, where $\tilde{\Sigma}_K^*$ is the set of all strings of length less or equal K.
2. The Closure operators U_p and U_m are partial mappings from $2^{\tilde{\Sigma}_K^*}$ to $2^{\tilde{\Sigma}_K^*}$, where their domain is restricted to sets of numerals. Formally,

$$U_p(S) = \begin{cases} \{a_{i_1} a_{i_2} \ldots a_{i_k}\} & \text{if } S = \{i_1, i_2, \ldots, i_k\}, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

$$U_m(S) = \begin{cases} \{b_{i_1} b_{i_2} \ldots b_{i_k}\} & \text{if } S = \{i_1, i_2, \ldots, i_k\}, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

3. The abstraction mapping, Υ, is defined as the identity operator, i.e.

$$\forall S \in \tilde{\Sigma}_K^* : \Upsilon(S) = S.$$

Now, one easily verifies that $\exists S \subseteq M_p^* : \Gamma[U_m(\Upsilon(S))] \subseteq U_p(S) \iff$ the reply to *BPCP* is affirmative. ■

Proof of Lemma 3 Let $W = U_p(G_1) \cap U_p(G_2)$, then $m \in W = \cup_{i \in \Omega} U_p(G_i)$ for some subset $\Omega \subseteq I$. Now, $\exists i^* \in \Omega$: $m \in U_p(G_{i^*})$; and since $\forall i \in \Omega$: $U_p(G_i) \subseteq W$, we conclude $m \in U_p(G_{i^*}) \subseteq U_p(G_1) \cap U_p(G_2)$. ■

Proof of Lemma 4 Take a generating set $\Phi = \{F_i\}$ for M^* such that $|\Phi| \leq t$, and denote by Φ_0 the collection of all $F \in \Phi_0$, such that for some $i \in I$ we have $U_p(F) \subseteq U_p(G_i)$. To every $F \in \Phi_0$, define the set

$\Pi(F) \subseteq I$ as $\Pi(F) = \{i: U_p(F) \subseteq U_p(G_i)\}$. By the axiom of choice, we can define a function $\pi(.)$ such that to every $F \in \Phi_0$: $\pi(F) \in \Pi(F)$. Let us show that $I_0 = \pi(\Phi_0) \subseteq I$ satisfies the conditions of the lemma. First, observe that $|I_0| = |\pi(\Phi_0)| \leq |\Phi_0| \leq t$. Trivially, $\cup_{i \in I_0} U_p(G_i) \subseteq \cup_{i \in I} U_p(G_i)$. Let us show the inverse inclusion. Take $m \in \cup_{i \in I} U_p(G_i)$, then since Φ is a generating set, $\exists i^* \in I \exists F \in \Phi$: $m \in U_p(F) \subseteq U_p(G_{i^*})$. That is, $F \in \Phi_0$ and $\pi(F) \in I_0$, i.e. $m \in U_p(F) \subseteq U_p(G_{\pi(F)}) \subseteq \cup_{i \in I_0} U_p(G_i)$, which concludes the proof. ■

Proof of Theorem 3

Part 1 Suppose that $t \geq \omega_0$. Take a generating set $\Phi_1 = \{F_j\}_{j \in J}$ for the system M^* such that $|J| \leq t$. Let $\Phi = \{G_i\}_{i \in I}$, and for any $j \in J$ let $I(j) = \{i \in I: U_p(G_i) \subseteq U_p(F_j)\}$. Since Φ is a generating set for the system M^*, we have $U_p(F_j) = \cup \{U_p(G_i): i \in I(j)\}$; and by Lemma 4, there exists a set $I_0(j) \subseteq I(j)$ such that $|I_0(j)| \leq t$ and $U_p(F_j) = \cup\{U_p(G_i): i \in I(j)\} = \cup\{U_p(G_i): i \in I_0(j)\}$. Let $\Phi_0 = \{G_i: i \in I_0(j)\ \&\ j \in J\}$. Since $|J| \leq t$, from $|I_0(j)| \leq t$ and the equality $t^2 = t$, it follows that $|\Phi_0| \leq t$. Let us show that Φ_0 is a generating set for M^*. Since Φ_1 is a generating set, we have—for an arbitrary set $M \subseteq M^* - U_p(M) = \cup_{j \in \Omega} U_p(F_j)$. Since $U_p(F_j) = \cup \{U_p(G_i): i \in I(j)\}$, we finally conclude $U_p(M) = \cup_{j \in \Omega} \cup_{i \in I(j)} U_p(G_i)$, which proves that Φ_0 is a generating set for M^*.

Part 2 Suppose $t < \omega_0$. Let $\Phi_1 = \{F_i\}_{i \in I}$ be a generating set for M^* such that $|\Phi_1| = Ca(M^*) = k \leq t$. For any $G_i \in \Phi_1$, there exists a subset $I_i \subseteq I$ such that $U_p(F_i) = \cup_{j \in I_i} U_p(G_j)$. Let us show that $\exists j_i \in I_i$ such that $U_p(F_i) = U_p(G_{j_i})$. If it is not true, then $\forall j \in I_i$: $U_p(G_j) \subsetneq U_p(F_i)$. Since Φ_1 is a generating set for M^*, we conclude there exists a subset $\Phi_0 \subset \Phi_1$ such that $U_p(F_i) = \{\cup U_p(F_j)$: $F_j \in \Phi_0$ & $F_i \notin \Phi_0\}$. Thus, $\Phi_1 - \{F_i\}$ is a generating set for M^* and $Ca(M^*) = k-1$, in contradiction to the assumption that $Ca(M^*) = k$. ■

Proof of Theorem 4 We have to show that $\forall \tilde{S} \subseteq M_m^*$, for some subfamily $\{\tilde{G}_i\}_{i \in \Omega}$: $U_m(\tilde{S}) = \cup_{i \in \Omega} U_m(\Upsilon(G_i))$. Since Υ is onto and additive $\exists S$: $\tilde{S} = \Upsilon(S)$. Applying Observation 2, we obtain $U_m(\Upsilon(S)) = \Upsilon(U_p(S))$. By definition, for some subfamily $\{G_i\}_{i \in \Omega}$: $U_p(S) = \cup_{i \in \Omega} U_p(G_i)$.

Substituting and applying the Additivity of Υ, we obtain, $\Upsilon(U_p(S)) = \Upsilon(\cup_{i \in \Omega}(U_p(G_i)) = (\cup_{i \in \Omega} \Upsilon(U_p(G_i))$. Finally, by applying the fact that (M_m, U_m) is *well-defined*, we obtain $\cup_{i \in \Omega} \Upsilon(U_p(G_i)) = \cup_{i \in \Omega} U_m(\Upsilon(G_i))$. Denote $\forall i \in \Omega$: $\tilde{G}_i = \Upsilon(G_i)$ and conclude the proof.

Proof of Corollary 3 Suppose $Ca(M_p^*) = t$, and Let $\Phi = \{G_i\}_{i \in I}$ be a generating set for M_p^* such that $|\Phi| = Ca(M_p^*) = t$. By Theorem 4, $\Phi_1 = \{\Upsilon(G_i)\}$ is a generating set for M_m^*, thus $|\Phi_1| \leq |\Phi| = Ca(M_p^*) = t$. Now, by Definition 18, $Ca(M_p^*) \leq |\Phi_1| \leq |\Phi| = Ca(M_p^*) = t$ which concludes the proof. ■

Proof of Lemma 5 It is easy to see that the *GSP* belongs to NP. Let us transform *SET BASIS* (see Appendix C) to the *Generating Sets* problem. Let C be a collection of subsets of a finite set S, and $K \leq |S|$ be a positive integer, which form an arbitrary instance of *SET BASIS*. It is easy to see that by letting U to be the identity operator the result is concluded.

References

Brave, Y. and Heymann, M. (1990) "Reachability in Discrete Event Systems Modeled as Hierarchical State Machines," Technical Report 90, Technion, Israel.

Casti, J.L. (1979) *Connectivity, Complexity and Catastrophe in Large Scale Systems*. Wiley-Interscience, New York and London.

Cornacchio, J.V. (1977) "System Complexity—A Bibliography," *International Journal of General Systems*, 3, 267–271.

Freeman, P. and Newell, A. (1971) "A Model for Functional Reasoning in Design." In *Proc. of the 2nd Int. Joint Conf. on Artificial Intelligence*, pp. 621–633.

Garey, M.R. and Johnson, D.S. (1979) *Computers and Intractability: A guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco.

Gero, J.S. (1987) "Prototypes: A New Schema for Knowledge Based Design." *Technical Report*. Architectural Computing Unit, Department of Architectural Science.

Gottinger, H.W. (1975) "Complexity and Information Technology in Dynamic Systems." *Kybernetes*, 4, 129–141.

Huhns, M.H. and Acosta, R.D. (1987) "Argo: An Analogical Reasoning System for Solving Design Problems." *Technical Report AI/CAD-002-87*. Microelectric and Computer Technology Corporation, March.

Kim, H.S. (1978) *Manufacturing Axiomatics with Special Applications to Air Compressor Design*. Master Thesis, MIT.

Kohout, L. (1990) *A Perspective on Intelligent Systems: A Framework for Analysis and Design*. Chapman and Hall Computing, London.

Kolodner, J.L., Simpson, R.L. and Sycara, K. (1985) "A Process Model of Case-based Reasoning in Problem Solving." *Proceedings of IJCAI-85*, Los Angeles, pp. 284–290.

Lenat, D. (1984) "Why AM and EURISKO appear to work." *Artificial Intelligence*, 23, 269–294.

Löfgren, L. (1977) Complexity of descriptions of system: a foundational study. *International Journal of General Systems*, 3, 197–214.

Maher, M.L. and Zhao, F. (1987) "Using Experience to Plan the Synthesis of New Designs." *In Expert Systems in Computer Aided Design*. Gero, J.S. (ed.). North-Holland, Amsterdam.

Maimon, O. and Braha, D. (1994a) "A Mathematical Theory of Design: Representation of Design Knowledge (Part I)." *Technical Report*. Department of Manufacturing Systems, Boston University.

Maimon, O. and Braha, D. (1994b) "On the Design Process." *Technical Report*, Department of Industrial Engineering. Tel-Aviv University, Israel.
Mostow, J. (1985) "Toward Better Models of the Design Process." *The AI Magazine*, Spring, pp. 44–57.
Mostow, J. and Barley, M. (1987) "Automated Reuse of Design Plans." *Proceedings of the International Conference on Engineering Design*, February.
Simon, H.A. (1981) *The Science of the Artificial*. MIT Press, Cambridge, MA.
Suh, N.P., Bell, A.C. and Gossard, D.C. (1978) "On Axiomatic Approach to Manufacturing and Manufacturing Systems." *Journal of Engineering for Industry*, **100**(5), 127–130.
Tarski, A. (1956) *Logic, Semantic, Metamathematics*, Oxford University Press, Oxford.
Ulrich, K.T. (1988) Computation and *Pre-Parametric Design*. Technical Report 1043, MIT Artificial Intelligence Laboratory.