Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

T. Tomiyama, H. Yoshikawa

Extended general design theory

Department of Computer Science Report CS-R8604 January

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

Extended General Design Theory

Tetsuo Tomiyama
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Hiroyuki Yoshikawa
Department of Precision Machinery Engineering
Faculty of Engineering, the University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113, Japan

Computer Aided Design (CAD) systems are getting more and more popular in many industries. Because designing is an intellectual process, CAD systems of next generation are supposed to have intelligence. This is why many researchers are recently working on knowledge based CAD systems. However, despite of these efforts, it is now becoming clear that *ad hoc* approaches are unsuccessful and that we need a guiding principle for implementing such CAD systems. In this paper, we propose a design theory to formalize design processes and design knowledge. GENERAL DESIGN THEORY is based on axiomatic set theory, and it clarifies what designing is, how to formalize a design process, and how to describe design knowledge. We begin with three axioms which define an ideal state. Then, we think about a real problem by adding a hypothesis about the physical aspect of our world, so that we can deduce theorems about real design processes and discuss theoretical problems of the data description method of future CAD systems.

1980 *Mathematics Subject Classification:* 03E30, 54A05, 54D35, 69K14, 69L60
1982 *CR Categories:* H.2.1., I.2.4., J.6
Key Words & Phrases: Design theory, axiomatic set theory, machine design, data model, knowledge representation, CAD.
Note: This paper was originally presented at IFIP Working Group 5.2 Working Conference, 'Design Theory for CAD,' held in Tokyo, Japan, from October 1, 1985, to October 3, 1985. It is to appear in H. Yoshikawa (ed.), *Design Theory for CAD*, to be published from North-Holland, Amsterdam, the Netherlands. This report is its revised version.

1. INTRODUCTION

Design methodology has been studied steadily (e.g., see [1]). However, most of the studies are discussing *how to design* only phenomenologically. They do not give sufficient and scientific explanations on basic problems such as *what is design.* Thus, this style of study would be done in an inductive way. Although this is an important attitude in scientific researches and the results are practically valuable for designers, at the same time it may yield several problems. For instance, the observational accuracy of a design process at the moment is insufficient and we cannot avoid ambiguity or vagueness of the theories. Furthermore, there is no way to provide logical verifications of the theory, since it would be sometimes too phenomenological and empirical.

Generally speaking, design theories are no more able to change real design activities than linguistic theories are able to change our language life. This implies that a design theory might be able to give an advice about what to do next during a designing process but not about what a future CAD system, e.g., an integrated and intelligent CAD system [2], should be. From these two reasons, design theories

Report CS-R8604
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

seem almost useless.

On the other hand, definitely we still need some tough guiding principles used for building such an advanced system from the following reason. Clearly, we need innovative methods of computer science for the implementation of these systems. Knowledge engineering is one of them, and it seems promising (e.g., see [3]). Among the problems found in applying knowledge engineering, the problem of knowledge representation is essential. In order to solve it, we must clarify our total viewpoint for design [2]; i.e., we must have a theory as the guiding principle to frame the structure of the design knowledge, to arrange the knowledge properly, and to utilize the knowledge efficiently as symbols.

GENERAL DESIGN THEORY based on axiomatic set theory was developed for this purpose. We first assume three axioms which outline our design knowledge, then we deduce theorems about designing itself and design processes. We can avoid the problem of observational accuracy and provide a method to verify the theory in such a way. But, unfortunately, theorems obtained in the former report [4] are insufficient to deal with real design processes, because we only paid attention to the *formal world* which really does not exist. To increase the power of the theory, we must give full consideration to the aspect of the *physical world.*

In the present study we attempt to extend GENERAL DESIGN THEORY and show its effectiveness in dealing with real design processes. To begin with, in the next chapter, briefly we review the reported part of GENERAL DESIGN THEORY. In Chapter 3, we argue its extension by adding a hypothesis about the physical world. Finally, in Chapter 4, we use the results of GENERAL DESIGN THEORY to obtain a guiding principle for building future CAD systems. We briefly analyze a problem about data description methods of future CAD systems.

2. REVIEW ON GENERAL DESIGN THEORY

Most of the contents of this chapter were already reported in the previous paper [4]. For the reader's convenience, we quote the main discussions.

2.1. Methodology of general design theory

GENERAL DESIGN THEORY is a theory which deduces theorems from three axioms based on axiomatic set theory. After the derivation, we examine and evaluate the compatibility or consistency of the theorems with realities. Only this test can certify the validity of the axioms.

GENERAL DESIGN THEORY aims at the following three points;

(a) clarifying the human ability to design in a scientific way;
(b) producing practically useful knowledge about the design methodology;
(c) framing design knowledge in a certain formality suitable for its implementation to a computer.

Before we discuss the theory itself, we must clarify our philosophical standpoint. From now on, we consider three worlds shown in FIGURE 1; i.e.,

R: the real world where all the concrete entities that we know exist;
C: the conceptual world where we think about entities of the real world, or more precisely this is the world that we have in our minds; an individual may have his/her own conceptual world;
L: the logical world; this is the world of symbols, logics, mathematics, philosophy, etc.

Having this world view, we obtain physics, epistemology, logic theory, engineering, technology, etc., as mappings among these three worlds. For example, design theories deal with a mapping from the conceptual world to the real world via the logical world. Here, designing is an activity to create an entity in the real world, through the logical world where drawings are existing, from the first idea about the design object born in the conceptual world. Note that written specifications normally exist in the logical world. Physics is a mapping from the real world to the logical world via the conceptual world. The first half of this mapping would be called *observation,* and it is governed by *epistemology* which has been one of the major concerns of philosophers. The latter half would be called *mathematics* or *logic operations* which helps physicists to describe physical phenomena in an objective and scientific way. Engineering or technology is a mapping from the logical world to the real world, which transforms logical descriptions to a concrete entity. Here, logical descriptions usually

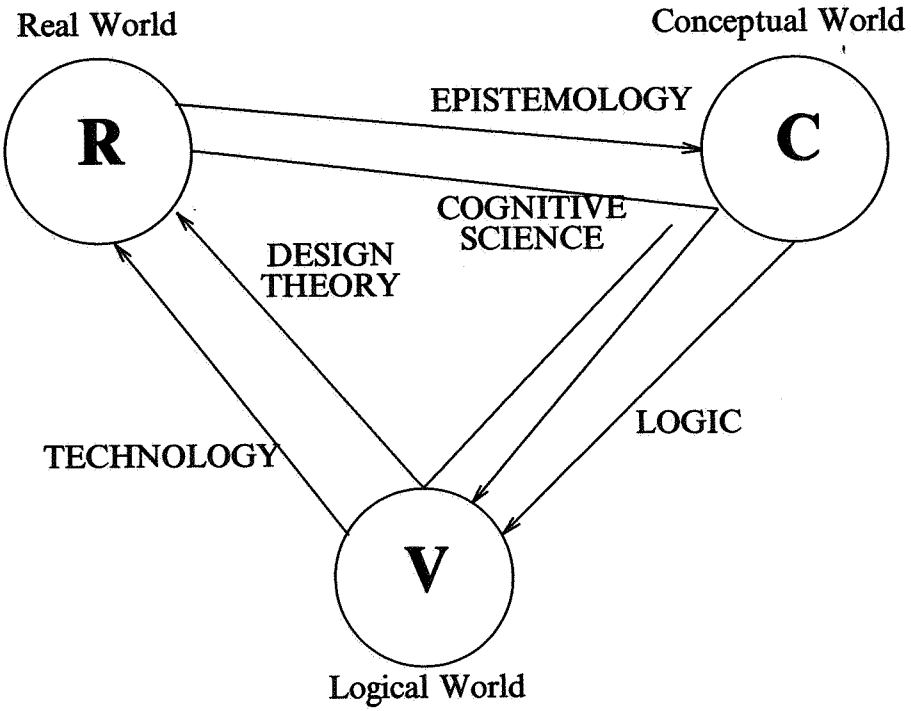


FIGURE 1. **World View of General Design Theory**

correspond to drawings and an entity corresponds to a product of manufacturing.

In this context, a design theory cannot be argued without mentioning its engineering aspect that designing is a mapping from the logical world to the real world. In fact, the former report [4] was lacking this aspect and discussing only the mapping from the conceptual world to the logical world. Therefore, the results were rather ideal and did not seem practically useful. In Chapter 3, we will discuss the engineering aspects of designing. Sections in this chapter only deal with rather ideal mappings.

2.2. Formulation of the problem

The necessary condition for designing is an ability to form a concept about nonexistent things as a result of logical operations of knowledge about existent things. In other words, a conceptive ability is essential for designing. From this point of view, we argue designing, design process, and design knowledge.

FIGURE 2. **Design Process**

A design process is commonly illustrated as in FIGURE 2, where its input is design specifications which might include both requirements and constraints for the design object, and the output is a design solution. Normally, design specifications are supposed to be written functionally in a language including symbols, numbers, existing machines, etc. Design solutions are to be described attributively

in drawings, in models, and sometimes in a language. Both are described in the logical world **L**. Based on this idea, let us begin the formulation of the problem with defining basic terms.

DEFINITION 1: The **entity set** is a set which includes all entities in it as elements. By all entities, we mean entities which existed in the past, are existing presently, and will exist in the future. This set is denoted by S'.

DEFINITION 2: An **attribute** of an entity is a physical, chemical, mechanical, geometrical, or other property which can be observed by scientific means.

DEFINITION 3: When an entity is exposed to a circumstance, peculiar behavior manifests correspondent to the circumstance. This behavior is called a **visible function.** Another behavior is observed under a different circumstance. The total of these behaviors are called a **latent function.** Both are called **function** inclusively. Sometimes, we use a function for a visible function when no confusion occurs. The **field** is the circumstance which is effective for a function to manifest.

DEFINITION 4: A **concept of entity** is a concept which one has formed according to actual experiences with an entity. This concept is different from an abstract concept, i.e., a concept of attribute or function, which is abstracted from the entity.

DEFINITION 5: An **abstract concept** is derived from classification of the concepts of an entity according to its meaning or value. As a result of classification, we get classes each of which includes entities carrying common meaning or value and corresponds to a peculiar abstract concept.

DEFINITION 6: A **concept of attribute** is one of the abstract concepts. This concept is effective for one to recognize entities. This concept has no direct relationship with value. The classification according to this concept is used, when one intends to classify entities independently of value, that is, scientifically.

DEFINITION 7: A **concept of morphology** is a sub-class of the concept of attribute. When attention is paid intensively to the morphology, this concept forms.

DEFINITION 8: A **concept of function** is a sub-class of the abstract concepts. When attention is paid to the functional value, it forms. In other words, when an entity has a peculiar latent function and is able to turn it into visible one in an existent field, then the entity is classified into a class which was made under the name of the peculiar function.

From now on, we use the following notations, such as

$\mathbb{T}$: the abstract concept,
$\mathbb{T}_0$: the concept of attribute,
$\mathbb{T}_1$: the concept of function,
$\mathbb{T}_2$: the concept of morphology.

All these concepts belong to the conceptual world **C**. Once they are written in a language, a syntactic description belongs to the logical world **L**, while the concept itself remains to be a semantic existence in the conceptual world.

2.3. Axioms

Here we are given three axioms.

AXIOM 1: (Axiom of recognition) Any entity can be recognized or described by attributes and/or other abstract concepts.

This axiom guarantees the observability of entities, but we have not described how to recognize or observe an attribute itself in DEFINITION 6. In order to solve this problem, first we must clarify what an *entity* is. Though DEFINITION 1 defines the entity set, we did not define the term *entity.* As easily

imagined, questions of this type are philosophically tough to answer, but we try it in Chapter 3.

In fact this axiom is seemingly trivial, but it yields a further philosophical big problem. It is that the description method of an entity concept must be **extensional** or **denotative,** but not **intensional** nor **connotative.** The entire axiomatic system of GENERAL DESIGN THEORY was built extensionally or denotatively, indeed. This will be discussed minutely in Section 4.2, showing its advantages and disadvantages for CAD applications.

> AXIOM 2 : (Axiom of correspondence) The entity set S' and the set of entity concept (ideal) S have one-to-one correspondence.

This axiom indicates that it is enough to think about only the set of entity concept S instead of the entity set S', and that between the real world **R** and the logical world **L** in FIGURE 1 there is a perfect mapping. But by definition, S may include even entities which will exist in the future, and this ideal knowledge is far from our real knowledge. In this context, this axiom guarantees the existence of a superman who knows everything; or in other words, it just shows an ideal and ultimate state of our knowledge, and that we have only imperfect design knowledge. This forces us to check the feasibility or compatibility of the knowledge with the realities besides the completeness, soundness, and inconsistency checks. This is assured also by the discussion in Section 2.1 where we introduced physical aspect to our world view.

> AXIOM 3 : (Axiom of operation) The set of abstract concept is a topology of the set of entity concept.

This axiom signifies that it is possible to operate abstract concepts logically, as if they were just ordinary mathematical sets. Accordingly, we get set operations, such as intersection, union, negation, etc. As discussed in the previous section, this operatability is essential to design.

It sometimes matters whether a topology is defined to be an open set or a closed set. In pure mathematics *open* or *closed* is a relative problem and these two are essentially identical; however, this influences the interpretation of concepts. Suppose a concept, 'the color is white,' is an open topology, then its logical negation, 'the color is not white,' becomes a closed topology. But, this is obviously a confusing situation that there is no uniformity in interpretation of concepts as topology. To avoid it, we consider the negation of 'the color is white' is 'it does not happen that the color is white' but not 'the color is not white,' and we discriminate these two negative concepts. Here, the concept, 'it does not happen that the color is white,' is closed; but both 'the color is white' and 'the color is not white' are open concepts. Under this interpretation, it becomes difficult to recognize that the semantic negation of 'the color is white' is 'the color is not white.' In order to know it, we must have special semantic correspondences between them, and also we must construct a universal set S to follow so-called *the open world assumption.* Mathematical operations become a little bit difficult, because we must think about semantic negation.

To sum up, we have decided to follow the open world assumption, and logical operations are not the same as those of natural deduction. Instead, we must use intuitionistic logic or constructive mathematics where law of the excluded middle does not hold; or at least we need to use three valued logic with *unknown* truth value in addition to *true* and *false.*

These conclusions are valuable, when we think about where to extend GENERAL DESIGN THEORY and how to apply it to CAD systems.

2.4. Ideal knowledge

Let us derive theorems about the ideal knowledge which only a superman is allowed to have. In this and the next sections, we do not give proofs of the theorems just to save the space. Readers may refer to the former report [4] for them.

> DEFINITION 9 : Let the **ideal knowledge** be the knowledge which knows all of the elements of the entity set and that can describe each element by abstract concepts without ambiguity.

THEOREM 1 : The ideal knowledge is a HAUSDORFF space.

Next, we argue design specifications in the ideal knowledge.

DEFINITION 10 : The **design specification** T_s designates the functions of the designed machine (system) using abstract concepts. Thus, the specification is $T_s \in \mathbb{T}$ and $s \in T_s$ is the design solution. Conversely, suppose s' is given, then the specifications of s' are any of the neighborhood system of s'. Practically, a specification is described in terms of functions, hence $T_s \in \mathbb{T}_1$ which is called a functional specification.

THEOREM 2 : A set of specifications can be described by an intersection of proper abstract concepts.

THEOREM 3 : A set of design specifications, unless contradictory, is a filter.

THEOREM 4 : From the preceding definitions, the following properties are easily deduced.

i) $\mathbb{T} \supset \mathbb{T}_0 \supset \mathbb{T}_2$,

ii) $\mathbb{T} \supset \mathbb{T}_1$.

Then we can discuss the design solution.

DEFINITION 11 : The **design solution** is an entity concept, s, that is included in the correspondent specifications T_s and carries all the necessary information for manufacturing.

THEOREM 5 : The entity concept in the ideal knowledge is a design solution.

THEOREM 6 : The entity concept in the attribute space $(S, \mathbb{T}_0)$ is a design solution.

THEOREM 7 : A design solution is represented by the intersection of properly selected elements of the set of attribute concept.

From THEOREM 2 and 7, both the design specifications and the design solution are to be given in the abstract concept space. This leads us to a discussion about design processes.

DEFINITION 12 : **Designing** is to designate a domain on the attribute concept space $(S, \mathbb{T}_0)$ corresponding to the domain specified by the specifications on the abstract concept space $(S, \mathbb{T})$.

Suppose the knowledge is not always ideal and the specifications are described in terms of function concepts, then the existence of a known mapping between the function space and the attribute space becomes the necessary and sufficient condition to be able to design. Thus, we have the followings.

DEFINITION 13 : **Designing** is a mapping of a point in the function space onto a point in the attribute space.

THEOREM 8 : If the abstract concept space on which the specifications are described is limited to the attribute space as the subspace of the former, then designing is completed when the specifications are described.

THEOREM 9 : In the ideal knowledge, the solution is obtained immediately after the specifications are described.

THEOREM 9 merely describes an ultimately ideal design process where designing will be completed in a certain finite period of time at an infinite design speed. Consequently, a superman can guess the design solution immediately after he got the specifications. This does not necessarily mean that a design process is a transformation or mapping process shown in FIGURE 2, because here the designing is finished by just an association between attributes and functions. However, real design processes are definitely more complicated than that sort of associations, and there must be a more substantial

designing process where designers think about the design solutions, calculate their performance, and draw figures. This point is argued in the next section. Here, we just add three theorems about *designability* in the ideal knowledge.

THEOREM 10: If designing is possible, the identity mapping from the attribute space $(S, \mathbb{T}_0)$ to the function space $(S, \mathbb{T}_1)$ is continuous.

THEOREM 11: If two design solutions can be discriminated functionally, then $\mathbb{T}_0 \subset \mathbb{T}_1$.

From THEOREM 10, we can derive the next theorem about design specifications.

THEOREM 12: If designing is possible, the design specifications are described by an intersection of elements of the set of attribute concept.

This theorem suggests a possibility to describe a function by attributes. Generally speaking, a function might be expressed in the following form;

to X Y,

where X is a verb and Y is its objective. THEOREM 12 states that there exists a way to express functions other than this *to X Y* form. Let us call it **an attributive expression of a function.**

DEFINITION 14: A **function root** is an element of the attributive expression of a function.

If any of the function roots of a design solution, s, were changed, the function of s would change accordingly. For example, the fact that

the base circle of a gear is a true circle

is a function root of

smooth rotational power transmission.

If we obtained a collection of such relationships between function roots and functions, we might be able to transform specifications to attributes very easily. This transformation is possible only in the ideal knowledge, but pursuing this idea is important for the construction of knowledge based CAD systems.

2.5. Real knowledge

In this section, we intend to discuss real design processes where designers work and where unfortunately no ideality is guaranteed at all.

The real knowledge is peculiar for its finiteness:

F1) Finiteness of the amount of our knowledge, i.e., our storage capacity is limited.

F2) Finiteness of the operational speed.

Due to these two, some imperfections may occur as follows.

I1) Imperfections concerned with the entity concept; it is possible that some abstract concepts include no entity concept *(vacancy).*

I2) Imperfections concerned with categorization of entities by abstract concepts; the topology is not perfect, hence insufficient separation of entities *(dislocation).* Some entity can be isolated without categorization *(impurity).*

I3) Operational imperfections; due to erroneous set-theoretical operations, it is possible that the filter condition, THEOREM 3, is violated in the detailing process of the specifications; or else, it takes an infinite time for convergence.

I4) Imperfections concerned with the mapping; it is impossible to expect a known correspondence between different topologies.

I5) Imperfections concerned with the analysis; in a design process analytic operations are indispensable as shown in THEOREM 9. The solution cannot be sometimes described even when we succeed to make the specifications converged into a known entity concept.

Designers who cannot avoid these imperfections still skillfully succeed to design excellently. In order to understand this phenomenon, it is essential to find out the conditions to be able to design in spite of these imperfections.

DEFINITION 15: The **real knowledge** is the knowledge obtained by introducing imperfections to the ideal knowledge.

THEOREM 13: The design process appeared in THEOREM 9; convergence to an entity in the abstract concept space; discovery of the necessary filter in attribute space; does not operate in the real knowledge.

This leads to the following theorem.

THEOREM 14: In the real knowledge the design is possible, if and only if there is any rule of direct correspondence between the topologies of abstract concept and of attribute concept without intervention of the entity concept.

THEOREM 14 says that, in the real knowledge, designing is impossible like in the ideal knowledge where designing is an association to an entity concept from the specifications. Nonetheless, we must find out some realistic correspondences between topologies of the function concept and of the attribute concept. For instance, if both the function and attribute spaces are EUCLIDEAN, the correspondence can be given by a set of mathematical equations. Something equivalent to these equations are indispensable for real designers.

THEOREM 14 also indicates that designing is an activity to find imperfection of our knowledge about entity concepts. Because we cannot find out an entity concept directly from the specifications, we have to *create* a new entity. This is why we design.

3. EXTENTED GENERAL DESIGN THEORY

In the previous chapter, we have discussed the real knowledge as the negation of the ideal knowledge. THEOREM 14 simply tells that designing in general is impossible, and we cannot continue the discussion to get further useful conclusions. This is basically because we have introduced finiteness and imperfectness that cannot be analyzed mathematically enough to our knowledge in DEFINITION 15; although we started from the three axioms and have been discussing in a deductive way, this definition negated the logical basis of the whole discussion, i.e., the axioms.

However, we can go on discussing by introducing another realistic assumption that explains real design processes and, at the same time, preserves the idealities of our knowledge to continue mathematical derivation. If we called the definition of the real knowledge in DEFINITION 15 *negative definition,* we should call this knowledge *positive definition* in the sense that it does accept the three axioms.

3.1. A hypothesis

We need to have some realistic constraints in order to consider the realities of the real world where we live and which is governed by so-called physical laws or rules. To do so, let us begin with defining physical laws and attributes using physical quantity and field as nondefined terms.

DEFINITION 16: A **physical law** is a description about the relationship between physical quantities of the entity and the field.

This definition suggests that we may write a physical law P_j using related attributes a_i $(1 \leqslant i \leqslant N)$, for instance, in an equation,

$$P_j(a_1, a_2, \ldots, a_N) = 0$$

Note that this is not the definition of a physical law but an example expression.

DEFINITION 17: An **attribute** is a physical quantity which is identifiable using a set of a finite number of physical laws.

This definition of attributes does not contradict DEFINITION 2, because in DEFINITION 2 an attribute is a physical, chemical, mechanical, or geometrical property of an entity, and because those

properties are, in fact, supposed to be identified in a certain physical phenomenon by experiments that are theoretically derived from physical laws.

DEFINITION 18: A **concept of physical law** is one of the abstract concepts. This concept is formed, when one looks at a physical phenomenon as manifestation of physical laws.

From this definition and AXIOM 3, we have the topology of the concept of physical law $\mathbb{T}_p'$. Since the set of entity concept S contains by definition all the entities that we can imagine, it may include things unrealistic such as a perpetual mechanism. Thus, we have

$$S \notin \mathbb{T}_p',$$

which means $\mathbb{T}_p'$ is not a topology of the set of entity concept. But if we think about $\tilde{S}$, a subset of S, within which things do not contradict the known physical laws, we get

$$\tilde{S} \in \mathbb{T}_p'.$$

From now, we use $\tilde{S}$ instead of S, the set of entity concept, and in $\tilde{S}$ we think that the three axioms hold. Therefore, some of the characteristics of the ideal knowledge are still kept as well as in $\tilde{S}$. Let us call this $\tilde{S}$ **the set of feasible entity concept.** (See FIGURE 3.) The topology of the concept of physical law is, therefore, defined precisely by using this $\tilde{S}$ as follows;

$$\mathbb{T}_p = \{t \mid t = u \cap T, T \in \mathbb{T}_p', u \in \mathcal{P}(\tilde{S})\},$$

where $\mathcal{P}(\tilde{S})$ is the power set of $\tilde{S}$.

Next, we consider the relationship between $\mathbb{T}_p$ and $\mathbb{T}_0$ on the set of feasible entity concept $\tilde{S}$. From now on, $\|$ indicates the end of a proof. Basically, we attach a proof to a theorem, but if the theorem can be easily proved its proof may be omitted.

THEOREM 15: On the set of feasible entity concept $\tilde{S}$, the topology of physical law concept $\mathbb{T}_p$ is a subbase of the topology of attribute concept $\mathbb{T}_0$.

PROOF: Firstly we show that $\mathbb{T}_p$ is a subset of $\mathbb{T}_0$. Because a physical law P_j is identified by a set of attributes, a_i $(1 \leqslant i \leqslant N)$, like

$$P_j(a_1, a_2, \cdots, a_N) = 0,$$

we can express $T_j^p \in \mathbb{T}_p$ corresponding to P_j using $T_i^0 \in \mathbb{T}_0$ corresponding to a_i as follows.

$$T_j^p = \bigcap_{i=0}^{N} T_i^0 \in \mathbb{T}_0.$$

Therefore, any T in $\mathbb{T}_p$ satisfies $T \in \mathbb{T}_0$. This means $\mathbb{T}_p$ is a subset of $\mathbb{T}_0$, and that $\mathbb{T}_p$ is a sub-topology of $\mathbb{T}_0$. Next, let us consider the following proposition;

$$O = \{U \in \mathcal{P}(\tilde{S}) \mid (\exists t \subset \mathbb{T}_p,\ |t| < \infty,\ t \neq \tilde{S},\ U = \bigcap t)\}.$$

Then a proposition,

$$\forall X [X \in \mathbb{T}_0 \Rightarrow \exists t (O,\ X = \bigcup t)],$$

is satisfied, if all the attributes are identifiable using physical laws. And it is satisfied by AXIOM 1 and DEFINITION 17. Thus, O is an open base of $(\tilde{S}, \mathbb{T}_0)$, and consequently $\mathbb{T}_p$ is a subbase of $\mathbb{T}_0$. $\|$

We see that an attribute is identifiable, recognizable, expressible, and describable using physical laws. Thus, we can employ physical laws to describe entities, because even attributes are recognized by them. This idea results in a hypothesis which defines the real design in a *positive* meaning.

HYPOTHESIS 1: The **real knowledge** is the set of feasible entity concept which is made compact by coverings selected from the topology of the known physical law concept.

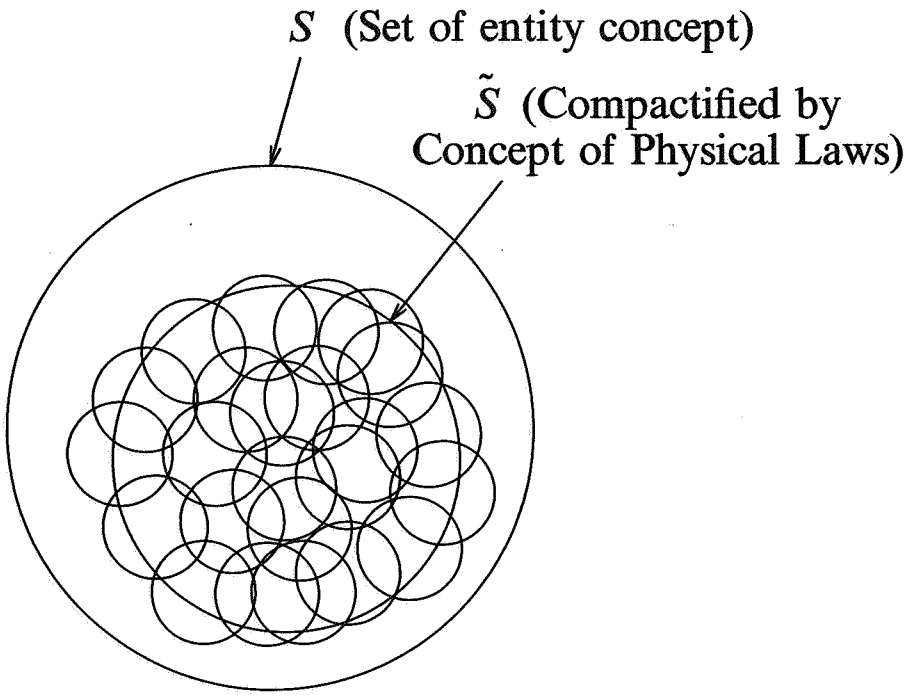


FIGURE 3. **The Set of Feasible Entity Concept**

This hypothesis does not contradict the real knowledge in DEFINITION 15, because in the set of entity concept S the three axioms do not hold, while in $\tilde{S}$ they hold. It also indicates that any entity in $\tilde{S}$ is explainable by not an infinite but a finite number of physical laws, and that this entity has physical constraints exactly of that number (FIGURE 3).

We can derive some properties of the real knowledge from HYPOTHESIS 1.

> THEOREM 16: The topology of attribute concept $\mathbb{T}_0$ on the set of feasible entity concept $\tilde{S}$, $(\tilde{S}, \mathbb{T}_0)$, is a compact HAUSDORFF space.
> PROOF: From THEOREM 1, $(\tilde{S}, \mathbb{T}_0)$ is a HAUSDORFF space. Because $\mathbb{T}_p$ is a subbase of $\mathbb{T}_0$ from THEOREM 15, $\mathbb{T}_p$ makes coverings for $(\tilde{S}, \mathbb{T}_0)$. Because HYPOTHESIS 1 says that $(\tilde{S}, \mathbb{T}_0)$ is compact, any open covering of $(\tilde{S}, \mathbb{T}_0)$ has finite subcoverings. Thus, $(\tilde{S}, \mathbb{T}_0)$ is compact, hence a compact HAUSDORFF space. ||

This theorem says that $(\tilde{S}, \mathbb{T}_p)$ in HYPOTHESIS 1 and $(\tilde{S}, \mathbb{T}_0)$ in this theorem are equivalent, because both of them are compact HAUSDORFF spaces. Thus, from now on, we consider the real knowledge to be $(\tilde{S}, \mathbb{T}_0)$ instead of $(\tilde{S}, \mathbb{T}_p)$. Next four theorems deal with the properties of this space.

> THEOREM 17: The real knowledge $(\tilde{S}, \mathbb{T}_0)$ is second countable.
> PROOF: A topological space (X, O) is second countable, if a countable set O' can be chosen as its open base. Here, $\mathbb{T}_p$ is an open base, since it is a subbase of $(\tilde{S}, \mathbb{T}_0)$. And $\mathbb{T}_p$ is a countable set, because $\mathbb{T}_p$ is made from a finite, or at least countable, number of known physical laws. Therefore, the real knowledge satisfies the second countability axiom. ||

> THEOREM 18: The real knowledge $(\tilde{S}, \mathbb{T}_0)$ is a closed subset of the (ideal) set of entity concept S.
> PROOF: From THEOREM 1 $(S, \mathbb{T}_0)$ is a HAUSDORFF space, and from THEOREM 17 $(\tilde{S}, \mathbb{T}_0)$ is compact. Since a compact subset C of a HAUSDORFF space X becomes a closed set of X, $(\tilde{S}, \mathbb{T}_0)$ is a closed subset of S. ||

THEOREM 19: If a continuous function

$$f: \tilde{S} \to \mathbb{R} \quad (\mathbb{R}: \text{the set of real numbers})$$

exists in the real knowledge $(\tilde{S}, \mathbb{T}_0)$, this function f has the maximum value and the minimum value.
PROOF: From THEOREM 18 and the theorem of maximum and minimum values, it is obvious. ||

THEOREM 20: The real knowledge $(\tilde{S}, \mathbb{T}_0)$ is a LINDELÖF space.
PROOF: A topological space is a LINDELÖF space, when any of its open coverings has at most a countable number of subcoverings. The real knowledge $(S, \mathbb{T}_0)$ is a HAUSDORFF space from THEOREM 16 and, consequently, it has a finite number of subcoverings. Therefore, it is a LINDELÖF space*. ||

THEOREM 17 tells the relationship between $\mathbb{T}_0$ and $\mathbb{T}_p$ and that a topology of $\mathbb{T}_0$ has countable neighborhood systems. THEOREM 18 and 19 state that in our world everything has the maximum and minimum limits for its attributes, because it should be more or less influenced and governed by physical laws. For instance, we can by no means produce a machine which is infinitely heavy; there is clearly the upper limit, i.e. the weight of the earth. We cannot produce a machine of which eigen frequency is equal to zero, either. However, an *infinite* can mathematically exist, since it shall not be influenced nor governed by physical laws; this fact does not contradict THEOREM 19.

3.2. Convergence of design solution

Let us argue the convergence problem of the design solution in the real knowledge.

Firstly we think about design specifications. Like in the ideal knowledge a specification is a filter, in the real knowledge it should satisfy some conditions to guarantee its convergence to a certain entity concept. In the ideal knowledge, there are three classes of specifications as follows, where T is a specification and each s_i is an entity as a solution.

(1) An ambiguous case; $T = \{s_1, s_2, \cdots\}$.
(2) An ideal and noncontradictory case; $T = \{s_1\}$.
(3) A contradictory case; $T = \{\varnothing\}$.

These three also apply to the case of the real knowledge. Let us call first two cases **feasible specifications** and the third one **impossible specification.** Note that the *impossible one* may include an impossible one, such as a perpetual mechanism, together with a nonfeasible one, such as an alloy which has absolutely no imperfection at all and is at present difficult to obtain. We call feasible design specifications just *specifications* from now on, and we define them more precisely as follows.

DEFINITION 19: Let $\mathbb{T}$ be a topology of abstract concept and Λ be a countable set. **Feasible design specifications,**

$$T = \bigcap_{\lambda \in \Lambda} T_\lambda \quad (T_\lambda \in \mathbb{T},\ T_\lambda \neq \varnothing),$$

is defined by the following conditions;

$$\bigcap T_\lambda \neq \varnothing.$$

Next let us consider the convergence problem of design specifications of DEFINITION 19.

THEOREM 21: Any feasible specification in the real knowledge has a cluster point.
PROOF: To begin with, let us show that from a specification

$$T = \bigcap_{\lambda \in \Lambda} T_\lambda,$$

* It is known that a second countable topological space is a LINDELÖF space, and from this we can also prove the theorem.

we can make a point sequence in $(\tilde{S}, \mathbb{T}_0)$,

$$\{x_N\}\ (n \in \mathbb{N},\ \mathbb{N}: \text{the set of natural numbers}).$$

Since Λ is a countable set, we can use $\mathbb{N}$ instead of Λ. Thus, the specification is rewritten as follows;

$$T = \bigcap_{i \in \mathbb{N}} T_i\ (T_i \in \mathbb{T}).$$

Next, we define T^n in the following way;

$T^1 = T_1$ ($\neq \varnothing$: from DEFINITION 19),

$T^n = T_{n-1} \bigcap T_n \neq \varnothing$;

and we choose an arbitrary element x_n from T^n. Then, here we obtained a point sequence $\{x_N\}$ $(n \in \mathbb{N})$. A LINDELÖF space L is compact, iff any point sequence of L has a cluster point. Here, $(\tilde{S}, \mathbb{T}_0)$ is a LINDELÖF space from THEOREM 20, and it is compact from THEOREM 16. Therefore, any $\{x_N\}$ $(n \in \mathbb{N})$ of $\tilde{S}$ has a cluster point. ||

THEOREM 22: In the real knowledge, it is possible to make a converging subsequence from any design specifications and to find out the design solution for the specifications.
PROOF: It is sufficient to show that there exists a subsequence of $\{x_N\}$ $(n \in \mathbb{N})$ made from the specifications, and that this subsequence converges to y that is the cluster point of $\{x_n\}$. From THEOREM 20, it is possible to define a fundamental countable systems of the neighborhood of y, $V(y)$. Suppose V_i $(i \in \mathbb{N})$ be an element of $V(y)$, then

$$U_n = \bigcap_{i=1}^{n} V_i\ (n \in \mathbb{N})$$

is a neighborhood of y satisfying

$$U_n \supset U_{n+1}.$$

We can make a set of

$$S_{n,m} = \{x \in U_m \mid \exists k\,(k \in \mathbb{N},\ n < k,\ x = x_k)\} \neq \varnothing\ (n, m \in \mathbb{N})$$

and its selection function ψ of which existence is guaranteed by the axiom of selection. Using this ψ, we have a sequence $\{y_k\}$ $(k \in \mathbb{N})$ such as,

$$y_1 = \psi(S_{1,1}) = x_1,$$
$$y_2 = \psi(S_{n_1,1}) = x_2,$$
$$\cdots$$
$$y_k = \psi(S_{n_{k-1},k}) = x_k,$$

This $\{y_k\}$ is a subsequence of $\{x_n\}$ which converges to y. ||

Now, from THEOREM 21 we are able to obtain the design solution for any specifications as a set of cluster points, i.e., there is a possibility of having more than two candidate entities. However, THEOREM 22 tells that we can get one specific design solution by some method which is symbolized to be ψ in the proof. And it also explains the fact that design solutions which depend on the designer are generally different, because each designer has a different selection function ψ.

Our next goal is to find out convergence policies, so that we can get the design solution not as a cluster point but as one point. To do it, it is sufficient that the design specifications make a directed sequence of points, because such a sequence $s_i \in \tilde{S}$ $(i \in \mathbb{N})$ converges due to its upward boundedness, i.e., compactness. To sum up, a convergence policy to find out the solution is to give some partial

order to entity concepts, such as distance, for example. This discussion is actually equivalent to defining a selection function mentioned above, and we can get the following theorem.

THEOREM 23: In the real knowledge the design specifications converge to one point, if it is possible to get a directed sequence of points from the specifications.

3.3. Metamodel

We discuss here a method to describe abstract concepts which is necessary to build models of entity concepts, for instance of machine, in a computer. Firstly we introduce a description method of entity concepts.

DEFINITION 20: A **metamodel** M_Λ is defined in the real knowledge $(\tilde{S}, \mathbb{T}_0)$ such that

$$M_\Lambda = \bigcap_{\lambda \in \Lambda} M_\lambda \ (M_\lambda \in \mathbb{T}_0),$$

where Λ is a finite set.

DEFINITION 21: The **metamodel set,** $\mathbb{M}$, is a subset of the set of all the metamodels and has the property of finite intersection.

Meta usually qualifies a thing superior to others. We use *metamodel* in this sense, i.e., *model of the models,* just the same as *meta* in *metaknowledge* which means knowledge to describe knowledge itself or the usage of knowledge. This concept of *model of the models* is really needed in CAD systems, because the integration of the models is one of the most urgent tasks in CAD studies [2]. Naturally, models mean those of design objects.

Another meaning of introducing metamodels is a possibility to describe an entity with only a finite number of attributes, although originally we have assumed an infinite number to do so. Thus, a metamodel M is the representative of an entity s such that

$$s \in M,$$

and it is described by a finite number of attributes.

The property of finite intersection of the metamodel set means that we are to think about only models of feasible entities. In fact, it suggests

$$\bigcap M \neq \varnothing \ (M \neq \varnothing, \ M \in \mathbb{M}),$$

which means there always exists an entity concept for a metamodel.

Several theorems about the properties of the concept of metamodel and the conditions to be able to design can be easily proved.

THEOREM 24: The metamodel set $\mathbb{M}$ is a topology of the real knowledge.

THEOREM 25: The metamodel set $\mathbb{M}$ is a topology weaker than the topology of attribute in the real knowledge.

THEOREM 26: In the real knowledge the necessary condition to be able to design is satisfied, when the topology of metamodel $(S, \mathbb{M})$ is stronger than that of function $(\tilde{S}, \mathbb{T}_1)$.

PROOF: From THEOREM 25, we have

$$\mathbb{T}_0 \supset \mathbb{M}.$$

If

$$\mathbb{M} \supset \mathbb{T}_1,$$

we have

$$\mathbb{T}_0 \supset \mathbb{M} \supset \mathbb{T}_1,$$

implying the following condition necessary to be able to design,

$$\mathbb{T}_0 \supset \mathbb{T}_1,$$

is satisfied. ||

THEOREM 27: If designing is possible, the identity mapping from the attribute space $(\tilde{S}, \mathbb{T}_0)$ to the metamodel space $(\tilde{S}, \mathbb{M})$ is continuous.

THEOREM 28: If the identity mapping from the metamodel space to the function space is continuous, the identity mapping from the attribute space to the function space is continuous.

The idea of metamodel can be applied to the theory of design process model and the convergence policy in the real knowledge. In the ideal knowledge, as mentioned in DEFINITION 13, designing is a mapping from the function space to the attribute space, while it is rather difficult to get such a mapping in the real knowledge. We then consider an intermediate space to decrease the difficulty to find it. The metamodel space is expected to be such an intermediate space (FIGURE 4). Based on this idea, the design specifications are described in the function space, and the designer rewrites them in the metamodel space. Because the metamodel itself has only a finite number of attributes, he may increase the number of attributes and detail the design object. We call this type of design process model *evolution model.*

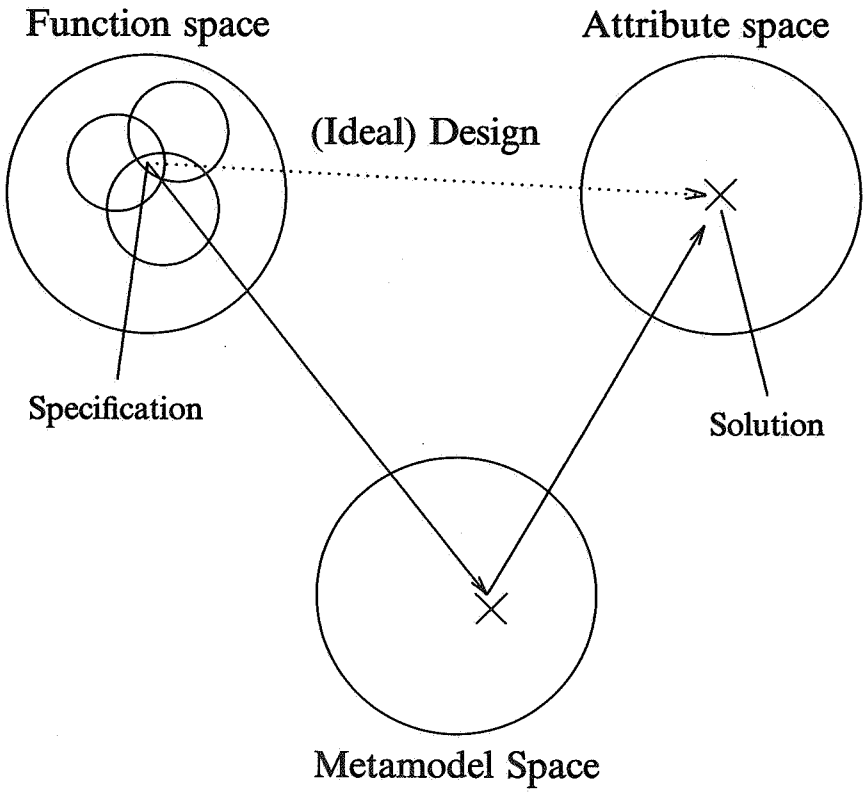


FIGURE 4. **Metamodel and Design Process**

In the evolution model, first there do not exist so many contents in an entity concept (FIGURE 5). We get just a rough description of an entity concept corresponding to the specifications, and then we detail its attributive descriptions. As the design proceeds, the amount of information of the entity concept will be increased by detailing the attributes. And finally, we get the design solution. In each

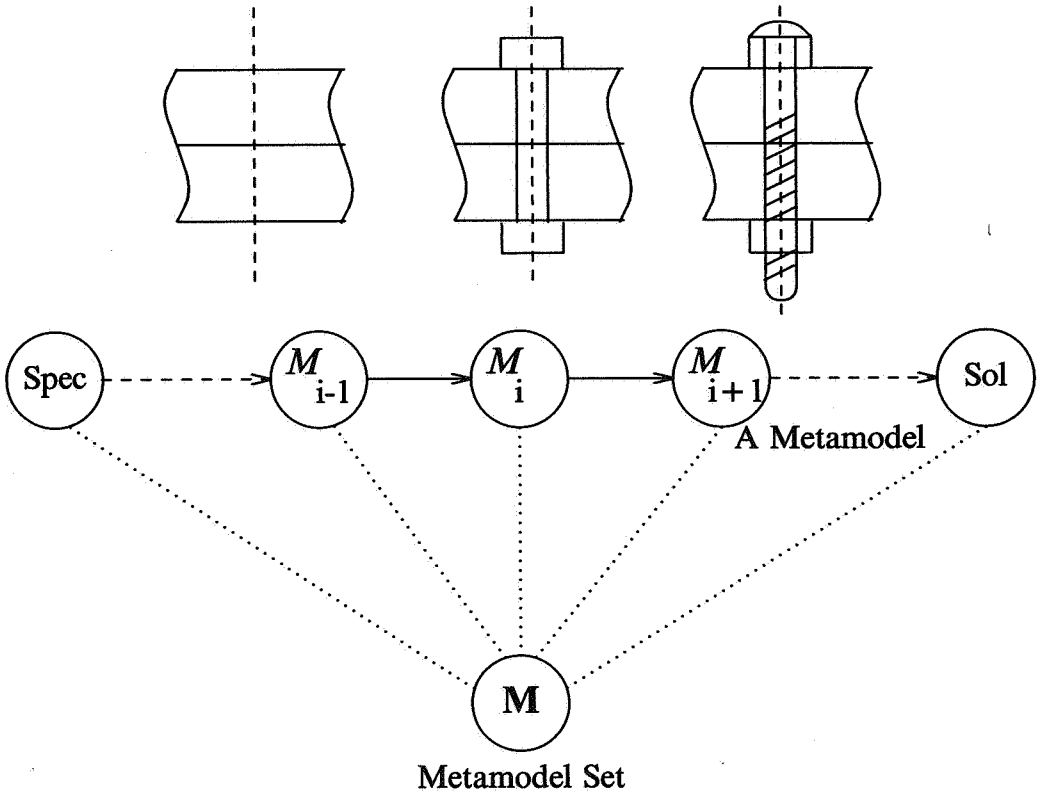


FIGURE 5. **Evolution Model**

detailing process there exists an intermediate solution which does not have all the necessary attributes being an entity. On the contrary, in the ideal knowledge the specifications are first detailed, and then we get the solution as an entity concept. The attributive information, necessary for manufacturing the solution, will be obtained by analyzing its neighborhood.

The evolution model gives a good account of real design processes [2]. The first step where we set up the first metamodel corresponds to a conceptual design stage; the detailing steps correspond to basic and detail design stages. Checking a metamodel can be called evaluation. In fact, if the result of an evaluation is not sufficient, the metamodel would be changed and tested again. This kind of activities is often found in real design processes.

About this evolution model we can prove the following theorem.

THEOREM 29 : If we evolve a metamodel, we get an entity concept as the limit of evolution.
PROOF: A metamodel,

$$M_\Lambda = \bigcap_{\lambda\in\Lambda} M_\lambda,$$

can be rewritten such that

$$M_n = \bigcap_{i=1}^{n} M_i,\ (M_i \in \mathbb{M},\ i \in \mathbb{N},\ M_i \neq \varnothing),$$

since Λ is a finite set. What we must prove is that the evolution of this metamodel is a convergence process such that

$$\lim_{n\to\infty} M_n = \bigcap_{i=1}^{\infty} M_i = \{s\},\ s \in \tilde{S},\ s \neq \varnothing,$$

where s is an entity concept as the design solution. Now define

$$M^1 = M_1,$$
$$M^2 = M_1 \bigcap M_2 = M^1 \bigcap M_2,$$
$$\cdots$$
$$M^n = M^{n-1} \bigcap M_n \ (\neq \varnothing\text{: The property of finite intersection}),$$

and suppose s_n be an element of M^n. The point sequence, $\{s_n\}$, is a directed sequence due to the inclusion relation,

$$M^{n-1} \supset M^n.$$

Therefore, from THEOREM 23, this sequence converges to

$$s = \lim_{n\to\infty} \{s_n\},$$

where s is an entity concept. ||

When we have a metamodel expression M for a design specification T, according to this theorem it is possible to get an entity,

$$s = \lim_{n\to\infty} \{s_n\}.$$

But this sequence converges in the metamodel space $\mathbb{M}$, not in the abstract concept space $\mathbb{T}$. This means that s is not the design solution but an approximation.

3.4. Function and physical phenomenon

We now know that mappings from the function space to the attribute space are basically hard to obtain. The concept of metamodel was supposed to reduce these difficulties, *prima facie.* Nevertheless, since it is attribute-oriented, there still remain difficulties to get mappings from the function space to the attribute space. Thus, we need to study how to present functions by attributes or how to transform functions to attributes.

In DEFINITION 3, a function was defined to be a behavior of an entity in a certain circumstance. This behavior is apparently governed and controlled by physical laws from a point of view of simple mechanical determinism. As we assumed in HYPOTHESIS 1 and DEFINITION 18, any entity in the real knowledge should be explicable by a finite number of physical laws. This idea leads us to the definition of *function* using *physical phenomenon* as a nondefined word.

> DEFINITION 22: A **function** of an entity is a physical phenomenon caused by the physical laws governing the circumstance where the entity was put in.

This definition does not refer to our sense of value. Of course, it would be possible to define another 'function' taking account of our values. But, in this study, we examine function only from a viewpoint of physical phenomenon.

If a function could be defined as in DEFINITION 22, then we can use a special type of metamodels.

> DEFINITION 23: A **function element** is a metamodel,
>
> $$M_\Lambda = \bigcap_{\lambda\in\Lambda} M_\lambda \ (M_\lambda \in \mathbb{T}_0,\ \Lambda\text{: a finite set}),$$
>
> such that
>
> $$\forall M_\lambda \in \mathbb{T}_p \ (\lambda \in \Lambda).$$

A function element is an entity concept that materializes some physical phenomena caused by pertaining physical laws. A simple example is a conductor in a magnetic field shown in FIGURE 6.

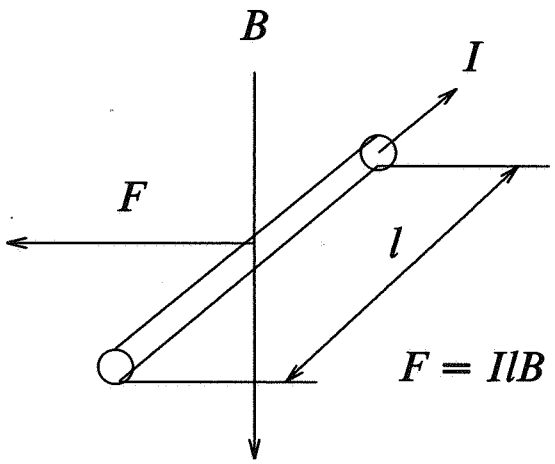


FIGURE 6. **An Example of Function Elements**

Here, the conductor is the entity and the function element of a physical phenomenon, *generation of force*; the electromagnetic field is the circumstance; the current in the conductor is an effect from the electric field to the conductor and *vice*versa. The force to the conductor and its correspondent motion are interactions between the conductor and the electromagnetic field. Physical laws are OHM's law, FLEMING's rule, FARADAY's law of induction, etc.

In the ideal knowledge, theoretically we need an infinite number of attributes to describe an entity precisely, indeed. But this example tells us what we need is just a finite number of attributes, such as the length of the conductor, the impedance, etc. This supports the idea of metamodel which merely requires a finite number of attributes.

If we made a collection of these function elements, we would be able to get the design solution easily because the specifications were described by physical laws. About this expectation, we have the following theorem.

THEOREM 30: If we choose function elements as the metamodel, we can describe the design specifications by the topology of metamodel, and there exists the design solution which is an element of this metamodel.

PROOF: The design specification is expressed by

$$T = \bigcap_{i \in I} T_i^p \ (T_i^p \in \mathbb{T}_p,\ I\text{: a countable set}).$$

Now, I is a countable set and might be infinite; but, since $\mathbb{T}_p$ is in fact a finite set, we can fix $T \neq \varnothing$. From HYPOTHESIS 1, $(\tilde{S}, \mathbb{T}_p)$ is compact, i.e.,

$$T = \bigcup_{\lambda \in \Lambda} T_\lambda^p \ (T_\lambda^p \in \mathbb{T}_p,\ \Lambda\text{: a finite set}).$$

Because we have constructed a metamodel from function elements, a metamodel,

$$T = \bigcap_{\lambda \in \Lambda} T_\lambda^p \neq \varnothing,$$

which can be actually derived from the finite intersection property of the metamodel space, is a subset of the specifications (FIGURE 7), i.e.,

$$M_\Lambda \subset T,$$

and all the members of this metamodel satisfy the specifications,

$$\forall s(s \in M_\lambda)\ s \in T.$$

Hence the theorem is proved. ||

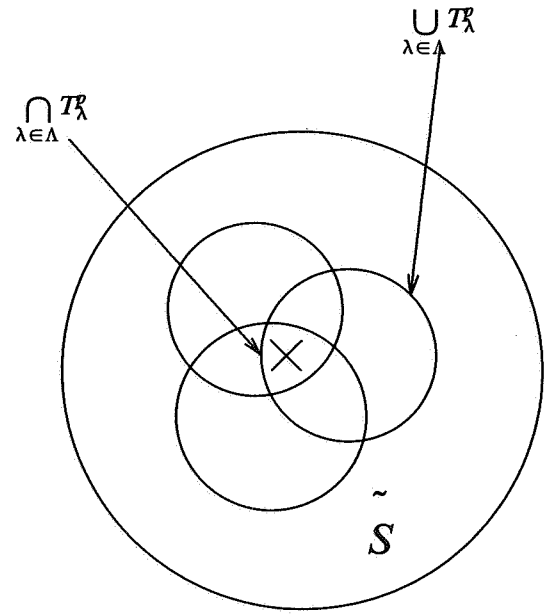


FIGURE 7. **Design Using Function Elements**

This theorem guarantees that, if the physical laws could be used to describe the specifications and to designate the physical phenomena needed, there should exist a metamodel derived from the specifications and the design solution obtained by detailing the metamodel. In this case, the solution is not an approximation but an accurate one, and we surely get it, even if we are not able to make a directed sequence as in THEOREM 23.

3.5. Metrization of the attribute space

We have considered *attributes* in general in DEFINITION 2 and 17. The word *attribute* should have the same meaning or effect as we expect in its everyday usage. Accordingly, an attribute should have a set of the name (or item) and the value. But, up to now, we have treated the attribute space to be a topological space. This leads to metrization of the attribute space; we must make the attribute space a metric space, such as a EUCLIDEAN space, for instance. From this metrization, furthermore, we can use numerical calculation to detail metamodels. This conclusion of GENERAL DESIGN THEORY is quite natural and acceptable, because our world is definitely a distance space.

Let us begin with four lemmas about metrization of topological spaces.

LEMMA 1: A compact HAUSDORFF space is a T_4 space.

LEMMA 2: A T_4 space is a normal and T_1 space.

LEMMA 3: (URYSOHN) A topological space N is normal, iff it has the property that for every two mutually disjoint, closed subsets A, B of N, there exists a continuous function $f: \mathbb{N} \to [0,1]$,

1) such that $f(x) = 0$ for all $x \in A$ and $f(x) = 1$ for all $x \in B$, and

2) such that for any $x \in N$, $0 \leqslant f(x) \leqslant 1$.

LEMMA 4: A normal space is metrizable iff it is second countable.

Now, we can prove the following three theorems.

THEOREM 31: The real knowledge is a normal space.
PROOF: From THEOREM 18 the real knowledge is a compact HAUSDORFF space, and a compact HAUSDORFF space is normal from LEMMA 1 and 2. Consequently, the real knowledge is normal. ||

THEOREM 32: In the real knowledge there exits a distance between two different entities.
PROOF: From THEOREM 31 and LEMMA 3 there exists a continuous function

$$f: \tilde{S} \rightarrow [0,1]$$

for every two mutually disjoint, closed subsets A, B of the real knowledge, such that $f(x) = 0$ for all $x \in A$ and $f(x) = 1$ for all $x \in B$. This fact can be easily applied to the relationship between two different entity concepts, s_1 and s_2, and two different real numbers, a and b. Thus, a function,

$$d(s_1, s_2) = |f(s_1) - f(s_2)| = |a - b|,$$

satisfies the axiom of the metric space. This means that there exists a distance between two different entity concepts. ||

THEOREM 33: In the real knowledge an attribute has a value.
PROOF: Following the proof of the previous theorem, the real knowledge is metrizable. Here, it is possible to make the real knowledge a metric space, which means there is a distance between two different entity concepts. Using this distance, the attribute space can be formed. This states that an attribute is given metric, i.e., it has a value. ||

According to THEOREM 32 and 33, we get followings.

(1) Given a certain metric, different entities can be measured differently. We may normally employ attributes as the metric as far as attributes are second countable. This requires attributes should have a countable number of open basis. All the attributes can not be always measured.
(2) When we have two candidate solutions, A and B, it is possible to judge whether A is nearer to the specifications than B or not.
(3) It is also possible to measure the convergence speed of the design solution.
(4) Suppose an entity A has an index 0 and B has 1 in a certain context. From LEMMA 3, we now know there exists an entity which has an intermediate value such as 0.5. This corresponds to the fact that there exists a mule between father donkey and mother horse. Or, it must be possible to convert a lathe to a milling machine continuously. This is the principle of design by modification.

3.6. Finiteness in the theory

In the present chapter, the discussion started from HYPOTHESIS 1 which preserved the completeness and infinity of the ideal knowledge and which was restricted by compactness. Thus, the properties of the real knowledge are still somewhat ideal, although apparently our actual knowledge is limited and imperfect. In this section, we examine HYPOTHESIS 1 from a point of view that how much finiteness and imperfectness are taken into consideration.

Followings are the finiteness and imperfectness of the real knowledge in the negative meaning pointed out in Section 2.5.

F1) Finiteness of our storage capacity.
F2) Finiteness of the operational speed.
I1) Imperfections concerned with the entity concept.
I2) Imperfections concerned with categorization of entities by abstract concepts.

I3) Operational imperfections.
I4) Mapping imperfections.
I5) Analytical imperfections.

Essentially, the real knowledge of HYPOTHESIS 1 is introducing a sort of finiteness, i.e., boundedness, resulting from its compactness. Because, basically, $\tilde{S}$ does not contain what we do not know, F1 and I1 are already taken into consideration.

Imperfections, I2 and I5, are related to finiteness in describing entities. Thus, we can replace I2 and I5 with another finiteness.

F3) Finiteness in describing entities (FIGURE 8).

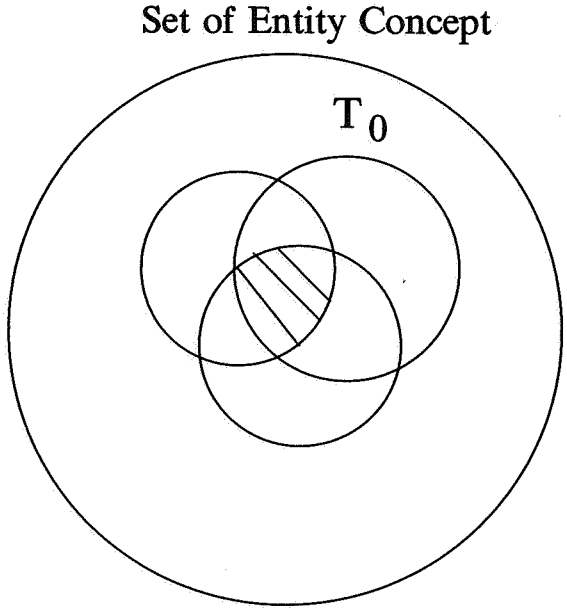


FIGURE 8. **Finiteness in Describing Entities**

Imperfection I3 and I4 would not be observed as far as the specifications are feasible and described by physical phenomena or laws as in DEFINITION 19, for the existence of the design solution is guaranteed by THEOREM 30. This is another finiteness we have instead of I3 and I4.

F4) Finiteness in describing the specifications.

Therefore, in the real knowledge we must pay attention to three points of finiteness, i.e., F2, F3, and F4. Let us call them *finiteness in the real knowledge.*

Because of them, obviously in real design processes, design specifications do not designate a specific point in the set of entity concept. They only designate an area and, thus, they are ambiguous. However, with even ambiguous specifications, real designers can design. This is perhaps because he does not have to describe the design solution rigorously. In this case the design solution does not need to be described minutely, either; it can also be an area (FIGURE 9). Additionally, designers can sometimes change the specifications. This is another reason why we can design and, sometimes, why we cannot design.

We can summarize the abovementioned issues as follows.

> DEFINITION 24: When the design solution is materialized, although it is exposed to the specified field, it might have behaviors different from the specifications. These behaviors are called **unexpected functions.**
>
> THEOREM 34: In the real knowledge the design solution has unexpected functions.

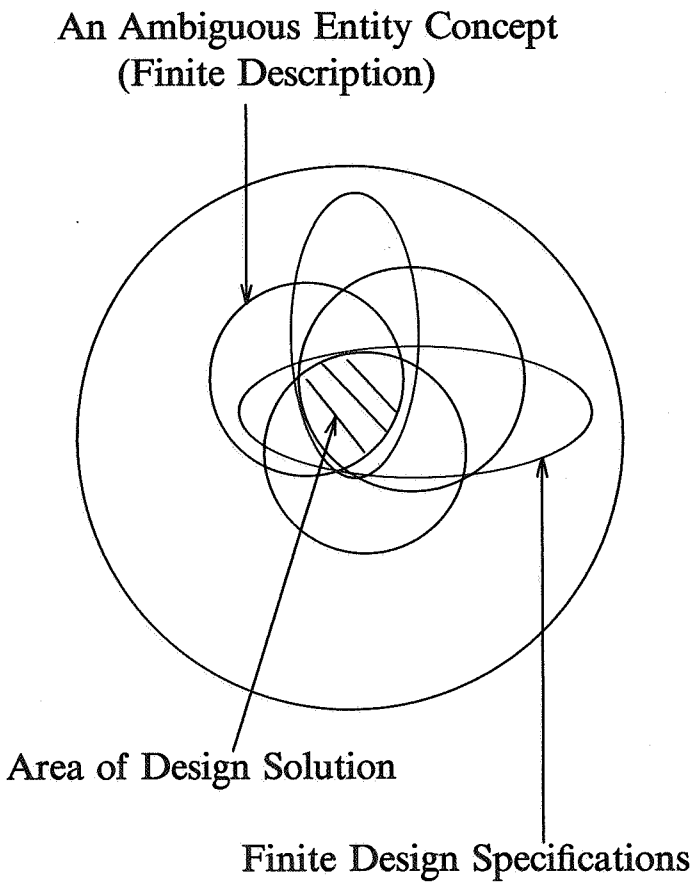


FIGURE 9. **Real Design**

4. SOME COMMENTS ON DATA DESCRIPTION METHOD FOR CAD SYSTEMS

In this chapter we remark on data description methods of future CAD systems, as an application of GENERAL DESIGN THEORY. It is expected that CAD systems in the near future should be materialized by means of knowledge engineering [3]. What makes a system intelligent is its ability to operate the meta-level knowledge, i.e., the framework of the knowledge. To do it, the integration of the data descriptions is essential. From this viewpoint, the integrated data description schema would be the most important factor among the elements of knowledge based CAD systems [2].

Now we have two questions. The first one is,

Do we really have a uniform method to describe an entity?

And the second one is like this;

Is that method feasible?

To answer the first question, the next section deals with an example of metrization of the attribute space. We simply expect such distance brought into the attribute space to be an integrated and objective way for representation. Secondly, in Section 4.2, we discuss an important issue derived from the basic standpoint of GENERAL DESIGN THEORY. It is this discussion, of course, that deals with the implementation problem.

But, unfortunately in the present paper, we can only show the necessary conditions to achieve our goal. Nothing has been obtained for the sufficient conditions up to now.

5. Example of measurable attributes

Theorem 32 and 33 state that attributes generally have values or that the attribute space is a distance space. This conclusion might look too obvious or trivial. However, what we have now as attributes are quite arbitrary and are in disorder; otherwise, they must have a kind of correspondence with the natural numbers and must be used practically to put entities in order.

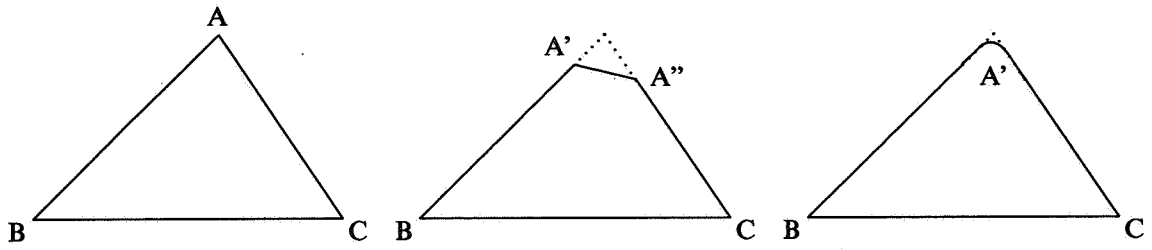


Figure 10. **Similar Figures**

Figure 10 shows three similar figures that possibly appear in mechanical parts. Figure 10 (a) shows a triangle; in (b) the corner A is chamfered and the figure is a quadrangle; in 10 (c) corner A is rounded and the figure is mathematically complicated, though it is quite usual as a mechanical part. If we look at these three from a viewpoint of mechanical engineers, strictly speaking (a) is nonexistent, while both (b) and (c) are feasible.

Thus, these three figures are geometrically different from each other. But, from a viewpoint of mechanical engineering they must be dealt with similarly because these are originally representing the same part. In conventional CAD systems these figures may be expressed by completely different data structures, if we stick to calling them polygons. This produces a rather confusing situation, because we want to use a data structure common to them. To sum up, if we gave distance to polygons by the number of corners, it would not work well.

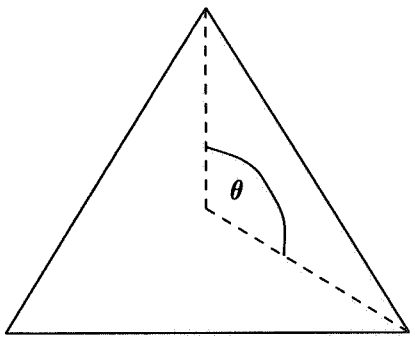


Figure 11. **Definition of a Regular Polygon**

Therefore, we should find out some attributes in order to describe polygons in a reasonable way. One example is shown in Figure 11. A regular polygon with n corners can be defined by θ which is central angle from one vertex to another, such that

$$n = 2\pi / \theta,$$

where n is not necessarily an integer. In Figure 12, three examples of polygons for non-integer number of vertices are shown, such as

$$n = 2.5 \ (\theta = 144°).$$

If the definition of polygons were the formula shown above, we could measure the distance between a regular triangle and a regular quadrangle. Of course, this distance might not meet with our intuitive distance for regular polygons. But, at least we could know the direction to get continuous distance for polygons which might be used in describing the resemblance of the three figures in FIGURE 10.

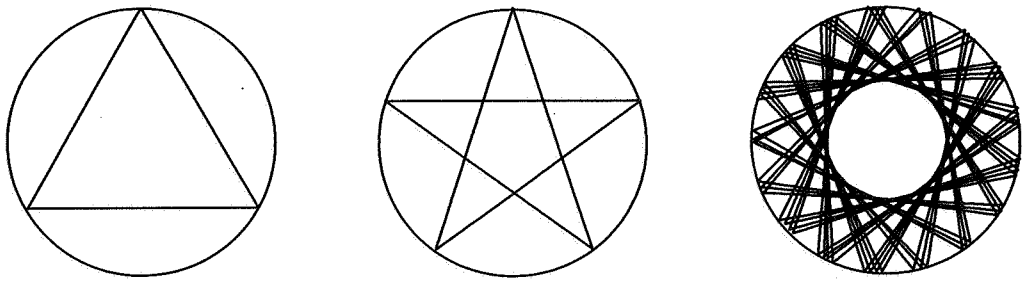


FIGURE 12. **Regular Polygons**

How do we interpret the case of $n = 2.83$, in FIGURE 12 (c), where the number of edges is infinite? This polygon would fill out the whole area of the figure with its edges, but in fact the number of edges would become infinite. This situation looks contradictory to THEOREM 19. However, even if we used all the ink we have on the earth, we could not paint a mathematically infinite number of edges; mathematical infinity is not a real existence but an imaginary product, and it is not denied by THEOREM 19.

This is the way we bring metric into the attribute space. In this direction we will find out a way to build the integrated data description method for attributes. However, we must always remind that what we actually need in machine design is metrization of mechanical entities and that it is not obtained from either THEOREM 35 nor 36. They simply refer to the possibility of making the attribute space a distance space. Unfortunately, a reasonable way for metrization of all the attributes is not obtained until now indeed, but analyzing aspects of physical phenomena of entities seems promising.

5.1. Intensional and extensional description

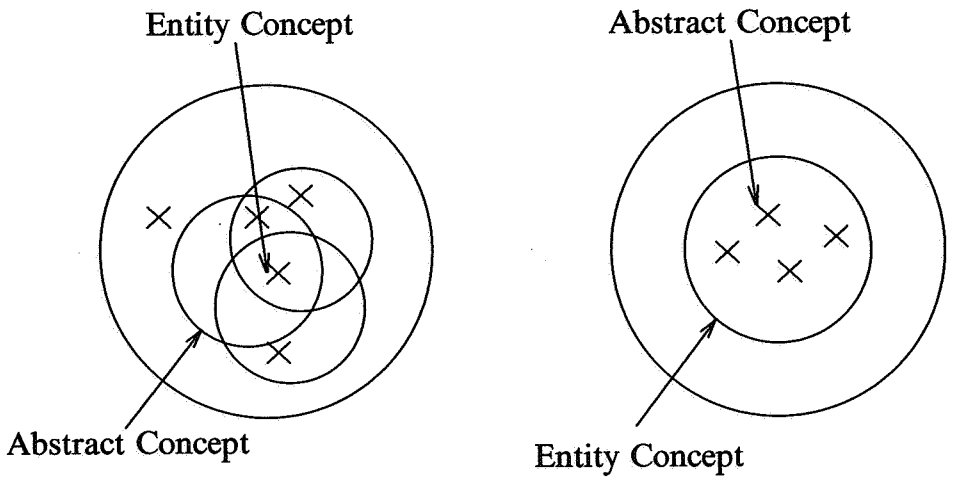
In Section 2.3, we mentioned that the description method of entities must be **extensional** or **denotative** but not **intensional** nor **connotative.** Here in this section, we minutely examine this issue which seems important in materializing an integrated data description schema for intelligent and integrated CAD systems. This description method is considered to be the necessary condition to bring distance into the attributes in a way explained in the previous section.

5.1.1. Extensional description. In AXIOM 1 and 3, the set of attribute concept was a topology of the set of the entity concept. Naturally, it is also possible to put it conversely. FIGURE 13 (a) and (b) show extensional and intensional ways of describing entity concept, respectively.

In FIGURE 13 (a), the subject of the situation is an entity, and the predicate is its attribute; and the total of this figure implies *extensional facts* about the entity that it has such and such attributes. This statement can be accepted also as an extensional definition of attributes. The extensional definitions are *fact oriented,* in this sense, and we represent an entity concept only by showing the properties of the object or its relationships to other entities, and thus the entity concept itself simply has symbolic meanings. When a set notation is used, the situation of FIGURE 13 (a) will be represented by

$$A = \{s_1, s_2, \cdots\},$$

where A is an attribute and s_i is an entity. This is, needless to say, the extensional definition of a set



(a) Extensional Description **(b) Intensional Description**

FIGURE 13. **Entity Concept and Abstract Concept**

A. Here, each s_i does not have a meaning more than just being a symbol.

5.1.2. Intensional description. On the contrary, in FIGURE 13 (b), the predicate is the entity to which the attribute belongs, while the subject of the situation is an attribute. This implies a situation that an abstract concept is found in such and such entities, and it is an intensional *object oriented* statement that describes an entity itself. Consequently, using a set notation, the situation in FIGURE 13 (b) could be represented by

$$S = \{a_1, a_2, \cdots\},$$

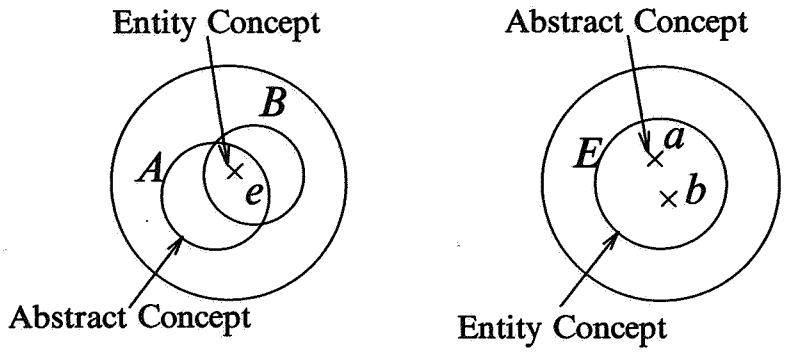
where S is an entity and a_i is an attribute. Normally a set of constraints, Σ, must be added to this representation, such that this entity exists meaningfully in the real world. Therefore, we have a different notation, usually, as a Cartesian product set,

$$S = \{(a_1, a_2, \cdots) | \ \Sigma(a_1, a_2, \cdots)\}.$$

This is the intensional definition of a set A. In this notation, we represent an entity concept by showing its structure which *a priori* exists, and each attribute, a_i, has no meaning more than being a symbol describing the structure.

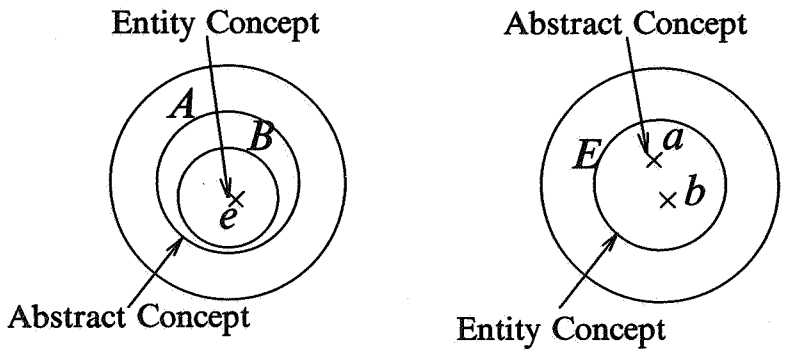
In FIGURE 14, the two representation methods are compared. FIGURE 14 (a) shows that two different abstract concepts are denoting an entity. This situation can be identically described in the intensional definition of the concept (FIGURE 14 (b)). But, as in FIGURE 15 (a), if two *similar* or *hierarchical* abstract concepts are denoting an entity, the similarity or the hierarchy cannot be expressed so well in the intensional definition because even a subset of a certain set would be recognized differently (FIGURE 15 (b)). Basically, the intensional definition method is not good to describe *relationships* precisely.

The whole story tells that in case of the intensional definition method of concepts slight differences in the meanings would be lost or quite similar concepts would be recognized differently. This would happen to CAD systems, especially, if we changed the data descriptions and one of them were a broader concept than the other. Therefore, the data description schema should be *denotative* or *extensional.*



(a) Extensional Definition **(b) Intensional Definition**

FIGURE 14. **Extensional and Intensional Representations**



(a) Extensional Definition **(b) Intensional Definition**

FIGURE 15. **Disadvantage of Intensional Representation**

5.1.3. Comparison between extensional and intensional description. Let us examine this problem with a more concrete example. FIGURE 16 shows a cube, and the following facts denoted by predicates are its extensional representations, because in an extensional representation system, the subject is an entity and the predicates are its attributes.

vertex(1).

. . .

vertex(8).

line(*1*).

. . .

line(*12*).

surface(1).

. . .

surface(6).

consists_of(1, *9*).

consists_of(5, *9*).

...

consists_of(*1*, **1**).

consists_of(*2*, **1**).

...

consists_of(**1**, *cube*).

...

consists_of(**6**, *cube*).

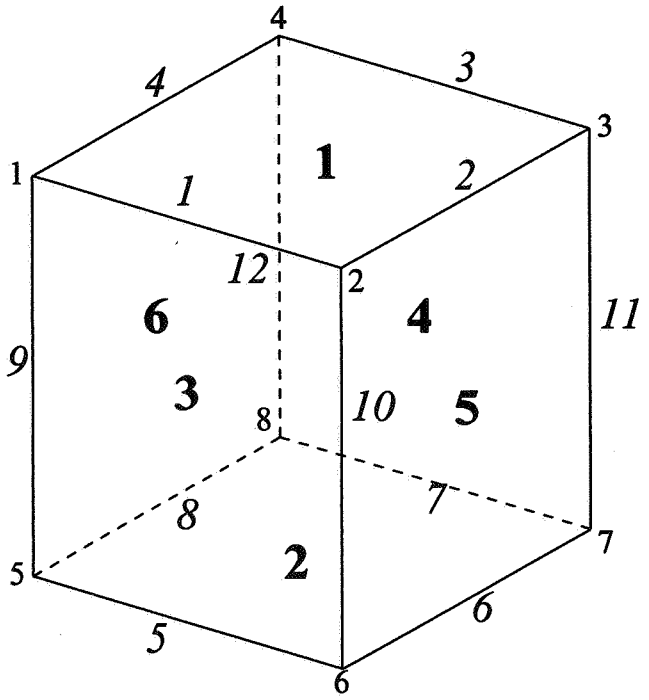


FIGURE 16. A **Cube**

On the other hand, the intensional representation of this cube would be

$$cube\{(\mathbf{1}, \mathbf{2}, \cdots, \mathbf{6}, \mathit{1}, \mathit{2}, \cdots, \mathit{12}, 1, 2, \cdots, 8) \mid \Sigma(\mathbf{1}, \mathbf{2}, \cdots, \mathbf{6}, \mathit{1}, \mathit{2}, \cdots, \mathit{12}, 1, 2, \cdots, 8)\},$$

where Σ implies the necessary conditions for this object to exist as a cube. Here, the entity is the predicate and the attributes are the subjects. Usually, this data structure can be realized in CAD systems as a set of data connected by pointers shown in FIGURE 17, or sometimes as a tuple of a relational database [5] (FIGURE 18).

The example leads us to the comparison of these two representations when they are used in CAD systems.

In an extensional representation system, the data would be described by a set of facts (e.g., by predicate logic formulae) independent from each other. Even the constraints will be expressed generally as a set of predicates. THEOREM 33 tells us that the value of an attribute, such as the

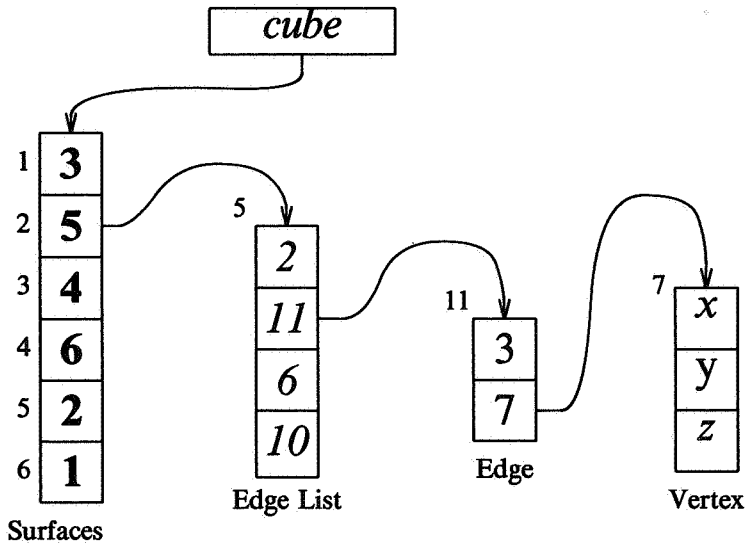


FIGURE 17. **Example of Data Structure of a Cube**

Surface	line	line	line	line
1	1	2	3	4
2	5	8	7	6
3	1	9	5	10
4	3	11	7	12
5	2	10	6	11
6	4	12	8	9

FIGURE 18. **Relation Type Data Structure of a Cube**

coordinates of a point P, shall be originally expressed as a topology. Because this is rather difficult to implement, normally we regard the predicate,

$$point(P),$$

as a function, and the value of this function becomes the coordinates, such as

$$[X, Y, Z] = point(P).$$

This means we can treat an entity, its attributes (i.e. predicates), and their values individually.

On the contrary, in an intensional representation system, the data would be totally described in a chunk of data strongly connected by pointers together with the constraints Σ. In this case, dependencies between the data become so strong that it is difficult to change or to modify the data schema. But, the meaning of a symbol can be easily decided by its relative position in the data structure.

Moreover, there is no separation of an entity, its attributes, and their values. For example, in a relational database system, we can separate relations and tuples like in FIGURE 18. But, this separation of the relation and the tuple can be so complete that there will be an inevitable mutual dependencies such as the order among the data in the relation.

To sum up, an extensional representation system has the following advantages and disadvantages.

EA1) It is easy to add new facts about entities, i.e., subjects.

EA2) It is also easy to modify facts and predicates.
EA3) Assertion of a proposition can be done by a simple search with pattern matching.
ED1) It is rather difficult to grasp the entire meaning of what the logical formulae as a whole are saying, because we need to interpret logically all of the descriptions and fairly large amount of computation would be required.
ED2) Predicates may loose their meanings. In other words, they can have meanings defined by each other. This is one of the typical disadvantages of formal logic and not particular to the extensional representation.

On the other hand, an intensional representation system has the following advantages and disadvantages.

IA1) Normally the meanings of each predicate can be easily understood, because the constraints, Σ, define them clearly.
IA2) What a piece of data says can be easily understood from its position in the data structure, i.e., by its address.
ID1) It is difficult to modify the data schema totally due to its strong mutual dependencies. But adding new facts is quite easy.
ID2) Modification of the structure of propositions requires changing the constraints Σ. This is also difficult.
ID3) Assertion of a proposition may contain considerably complicated calculation of the constraints Σ.

5.1.4. CAD applications. Usually in CAD systems, metamodels as the representation of design objects have the following properties [2].

(1) Dynamic changeability; metamodels are evolved corresponding to the progress of design.
(2) Diversity; there may be various metamodels probably as many as elemental design evolutionary steps.
(3) Bulkiness; data quantities would be large.
(4) No uniformity in interpretation of expressions; practically, expressions must be multi-purpose, i.e., they are interpreted in various ways corresponding to the lack of uniformity in design work.

These four points result in a conclusion that a CAD system must be equipped with a data schema flexible enough to be modified, added, and deleted easily. But, at the same time the whole database system must be *sufficiently practical.* For instance, it must be able to answer in a quick response time.

An extensional data description method is, generally speaking, suitable for CAD systems from the points (1), (2), and (4). In addition to them, if we considered an example of FIGURE 10 where we compared a triangle with a triangle-like quadrangle, the extensional description is much better than the intensional one. For example, the differences would be detected clearly by a set of predicates in the extensional representation method, while, in the intensional method, the differences are given by only partial data.

A more concrete example is the chamfering of the corner A of FIGURE 10 (b). In the intensional description, this chamfering causes a change of the data structure. The modified data structure cannot be easily judged as *different from the original* because of its demerit pointed in Section 4.2.1. On the contrary, in the extensional description, the chamfering is just an adding of a predicate

$$\textit{chamfered}(\mathrm{A}).$$

But, if we are going to use a computer, intensional description systems become much more convenient than extensional ones. This is just due to that we can grasp the meanings of descriptions at the moment we read them by using their positions, i.e., *addresses.* Moreover, the disadvantages of the intensional description will be inverted to advantages. It is definitely difficult to change predicate systems of the intensional representation. But as in business applications, once the conceptual schema is fixed, data retrieval is easily executed, because it is done by calculations of addresses which is faster

than pattern matching contained in extensional representation systems.

Concerning implementation of CAD systems, this problem is crucial. Basically and logically, these two description methods are equivalent. However, CAD applications theoretically may prefer extensional description systems to intensional ones, because of the dynamic changeability of the data schema, etc. But, if we think about real implementations, intensional one practically is better. Without intension, we can never compute anything, actually.

Thus, what we must do here is to combine or integrate both extensional and intensional description methods. But, this breaks out the uniformity of the information, since the transformation between the extension and intension is no more reversible. A famous example is *the morning star* and *the evening star.* The morning star has an extension, Venus. The evening star is an intension of Venus. Thus, an intension of an extension of an entity is not always identical to the entity. This suggests that we must be careful to losses or changes of information, when there is a transformation between the extension and intension.

As a matter of fact, up to now, we have not yet obtained the final solution to this problem. In this paper, we could just point out the importance and the necessary conditions of this problem which is the key to improve CAD systems of today. In other words, we need a new programming paradigm which can solve the presentation problem for the implementation CAD systems.

6. Conclusion

(1) A hypothesis concerning the physical aspects of designing was introduced to General Design Theory, and from it plenty of useful theorems were deduced.

(2) In the real knowledge, the design solution is obtained as a cluster point.

(3) In the real knowledge, the concept of metamodel which is defined by a set of a finite number of attributes is useful, and it leads to a design process model called evolution model. This design process model can give a reasonable explanation on real design processes.

(4) It is possible to give metric to the attribute space in the real knowledge.

(5) From General Design Theory, we can deduce several comments on the data structure of CAD systems. For example, an extensional description method is more suitable for CAD systems than intensional one theoretically, but practically we need to combine both of them.

(6) Future CAD systems would be realized by techniques of knowledge engineering, and as a guiding principle to build such systems design theories would become important. General Design Theory would serve as such a principle.

Acknowledgement

We would like to thank Ms. Masami Kumazawa of the University of Tokyo for her helpful advice, especially, about mathematical derivation. We are also grateful to Mr. Takaaki Yagyu of Univac Japan for his stimulating us to study the data model description problem.

References

1. V. Hubka (1977) *Theorie der Konstruktionsprozesse,* Springer, Berlin, Heidelberg, New York.
2. T. Tomiyama, H. Yoshikawa (1985) Requirements and principles for intelligent CAD systems, in [3], pp. 1.
3. J. S. Gero (ed.) (1985) *Knowledge Engineering in Computer-Aided Design,* Proceedings of IFIP WG5.2 Working Conference 1984 (Budapest), North-Holland, Amsterdam.
4. H. Yoshikawa (1981) General design theory and a CAD system, in T. Sata, E. Warman (eds.), *Man-Machine Communication in CAD/CAM,* Proceedings of IFIP WG5.2/5.3 Working Conference 1980 (Tokyo), North-Holland, Amsterdam, pp. 35.
5. R. A. Lorie (1982), Issues in database for design applications, in J. Encarnacao, F.-L. Krause (eds.), *File Structures and Data Bases for CAD,* Proceedings of IFIP WG5.2 Working Conference 1981 (Seeheim), North-Holland, Amsterdam, pp. 213.