Towards Multi-aspect Design Support Systems

Hideaki Takeda

February, 24, 1994

Contents

List of Figures

Chapter 1

Introduction

Recently CAD systems are largely used in industries, and there is no doubt that CAD systems contribute toward increasing design quality and decreasing designers' routine work like drafting. Although current CAD systems, in which geometric modeling is centered, have been developed rapidly and used widely, the next generation CAD systems, which can support designers from the beginning of design, remain obscure in their concept and realization.

In this research, we discuss how to support designers' various activities by computer. The next generation CAD systems should not remain a set of convenient tools like a drawing system, an analyzing system and so on, but a friendly partner that solves problems together with designers as well as provides its computational capability.

We emphasize two issues that we should solve. One is understanding designers' thinking, and the other is understanding variety of representation of objects. Although there are many systems, methods, and tools that can solve some identified problems, the relations among these problems are unknown. It is needed to realize CAD systems to cooperate designers in all design stages to connect such isolated problems.

In this report, we first discuss what is required for the next generation CAD systems in Chapter 2. We also show some research domains related to this, and identify research issues to solve. In those issues, we focus on integration of multiple aspects in this report. We introduce our logical design process model which is base of our discussion in Chapter 3. Then we discuss aspects and inter-aspect relationship in Chapter 4. We identify structure of aspects and relationship among aspects. In Chapter 5, we discuss abduction as integration of aspects. Then we show how cooperation can be done with synthesis and analysis with multi-aspects.

Chapter 2

Intelligent CAD

2.1 Research on intelligent CAD

The concept of so-called "intelligent CAD" has risen from both AI and design research field. From AI field, "design" is an attractive research issue because design is one of the most complicated but yet not-solved human activities. In early works, they preferred the words"automated design" rather than "computer-aided-design", but soon they realized it was not so easy to "automate" design activities because design activities includes various thinking processes, and then they have attacked smaller problems which design includes. That is, AI research would not provide something which would replace human designers, but some useful systems which could solve problems that designers would want to solve in their design.

We can find two streams in design research field, that is, geometric CAD based research and design theory or methodology based research. Geometric CAD research is confronting a turning point. There are few research issues left in geometric CAD and they are trying to extend CAD to support not only drawing but also earlier stages of design, e.g., conceptual design. That is, they want to make CAD more intelligent.

On the other hand there exist some design theories that explain or formalize design with various ways. Researchers have developed their theories independently from CAD research. Therefore, they have seldom taken care of "computability", and it makes them difficult to apply their theories directly to developing CAD.

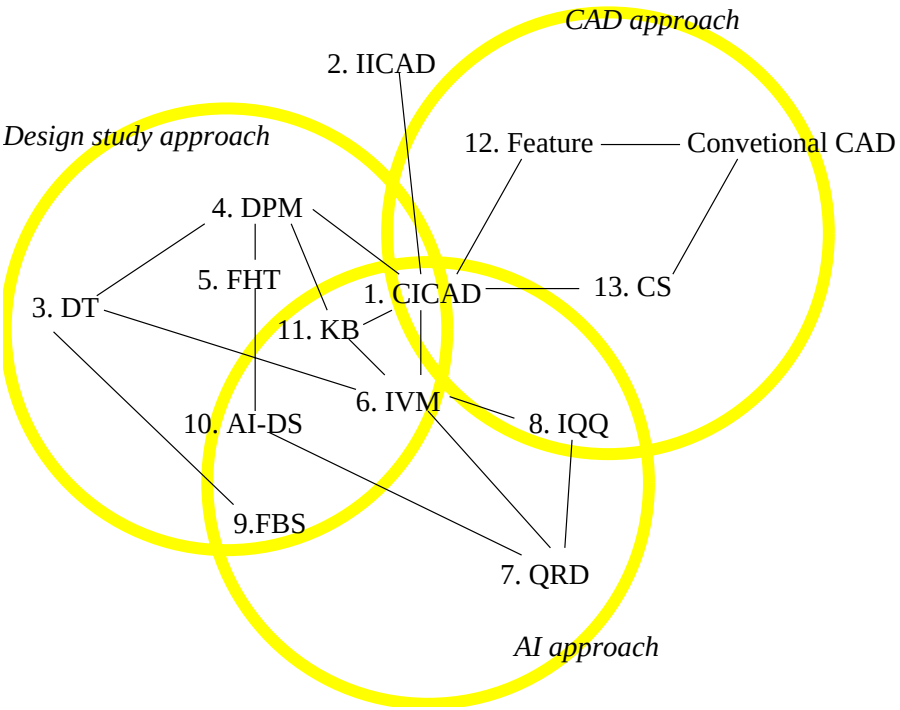
But now CAD researchers and "design theorists" agree that the common target is "intelligent CAD".

Now as a mixture of these trends, many works have done which concern this problems(1)(2)(3)(4)(5 and various problems are pointed out as problems in design.

2.2 Requirement for Intelligent CAD

Here I use the word "design" in a wide sense. That is, "design" does not mean merely "creating some artifacts", but including various activities such as analysis, optimization, searching database, documentation, etc, to accomplish creation of artifacts.

Here we want to discuss "COMPUTER-AIDED DESIGN" in the above sense, not convention meaning of "CAD".



1. CICAD: Concept of Intelligent CAD
2. IICAD: Implementation of Intelligent CAD
3. DT: Design Theory
4. DPM: Design Process Modeling
5. FHT: Formalization of Human Thinking in design
6. IVM: Integration of Various Models
7. QRD: Qualitative Reasoning in Design
8. IQQ: Integration of Qualitative and Quantitative reasoning
9. FBS: Function, Behavior, and Structure
10. AI-DS: AI Design Systems
11. KB: Knowledge Base
12. Feature: Feature, feature-based systems
13. CS: Constraint Solving

Figure 2.1: Research issues for intelligent CAD

There are mainly three ways to use computers in design now[1], that is, current CAD (geometric CAD) systems, CAE (computer-aided engineering), and expert systems (knowledge-based systems) in design.

In this meaning, current CAD (geometric CAD) systems deal apparently very small part of design. It is valid only in drawing. Other approach to use computer in design is so-called CAE (computer-aided engineering). In this case, it emphasizes analyzing phase in design, ignoring other activities. The last approach is utilization of expert systems technology in design. There are some successful results in routine design, but its application is restricted to clearly predefined problems.

So far, we can make requirements for a future CAD system. A future CAD system must be a "design-centered system". that is, the center of the system is neither drawing, nor analysis nor a certain problem solving, but designing itself. It should support from the beginning of design to the end of design with various aspects of designing. Ant it also should be a highly interactive system. Attempts of expert systems show us that design activities include many processes incapable to computers. Design is so flexible and limitless as a whole that we never get perfect description of design. It means the system must be always open to designers.

More detailed requirements for a future CAD system can be described as follows(7);

- The system should be integrated;
  - integration of subsystems,
  - integration of design models based on an integrated model,
  - integration of design processes which means computerization of even very early stages, and as its results,
  - integration of design knowledge.
- The system should be intelligent;
  - intelligent problem solving,
  - intelligent support of designer,
  - intelligent interface.
- The system should be interactive.

To achieve these requirements is not so easy. Of course one reason is a purely implementational one, that is, such a system will be a big and complicated and need many state-of-art computer techniques like knowledge engineering, geometric modeling, user interface management etc.

But a more fundamental reason is that we cannot capture the whole problem, i.e., design in a formal manner yet.

Apparently we need to know more about design and design objects, and also we need to know more about knowledge.

[1]Furthermore we may include database utilization in design.

The above requirements imply that the behavior of intelligent CAD should be in the same level of designers' behavior. Suppose if the system return a bare figure like "3.0" as a result of design or analysis. It forces a user n(designer) to interpret its meaning by her/himself. It cannot be intelligent nor interactive for the user, and also it cannot be integrated as a system because it cannot deal the figure any more. But if it knows that the figure is the length of a certain shaft and that the user is determining the shape of the shaft, it can response him more understandably and deal the figure in the system more properly.

It turns out that intelligent CAD should have knowledge about design and design objects. Intelligent CAD should be built as a knowledge-based or "knowledge-navigated" system.

Here we can divide the problem into three, which are different in perspective. That is(7);

Theory of design If we do not know what design is, we are obliged to use ad hoc approaches to build a CAD system. It might be powerful in a particular field, but not in general. If we want to build a CAD system applicable to wider fields, we should know the nature of design to design a fundamental architecture of the CAD system.

Theory of design is necessary;

- to clarify what design is,
- to formalize design process, and
- to formalize design knowledge.

Theory of design objects Although representation of objects is perhaps most developed area in CAD studies, yet there exist issues to be further studied; for example,

- how to represent ambiguous or rough descriptions
- how to integrate various models

Theory of knowledge Above two theories are expected to reveal the nature of design processes and design objects and tell us what is knowledge of design processes and design objects. But another problem is how to deal these types of knowledge in the system.

2.3 The Foundations of Intelligent CAD

2.3.1 Design Theory

Design theory is not a new concept. Some researchers in Europe, especially in Germany have developed some "design theories" (8) (9) (10). But these theories do not seriously concern the nature of design but how to improve actual design. And they are in many cases dependent of researchers' experience. Therefore they seem to be useful for designers, but understandable only for designers.

On the other hand, Yoshikawa (the Univ. of Tokyo) have proposed another "design theory" which is called as "general design theory"(11) (12) (13). It is based on topology in mathematics and roughly speaking, it starts with setting three axioms and defining design as a mapping from attribute space to function space. It derives some important concept such as "metamodel" and "evolutionary design process model" which lead more practical discussion.

Although Yoshikawa's theory is not completed and theoretically there are much room to develop it, it provides a good framework and some important concept needed to construct intelligent CAD.

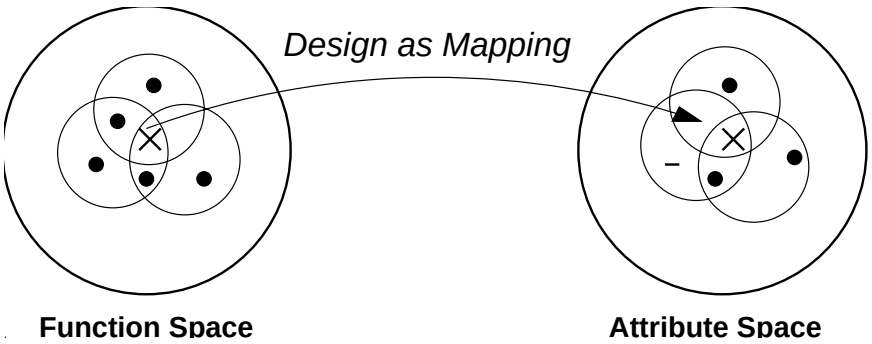


Figure 2.2: Design process in the ideal knowledge in GDT

2.3.2 Theory of Design Processes

We can identify two subproblems in the field where design theory is expected to cover. One is the problems about design processes and the other is one about design objects. While the latter is closely related to physical problems, the former is related to cognitive problems.

Takeda et al. proposed a logical framework for design processes (14)(15)(16). In it they defined design and a design process in terms of logic and explained how a design process is formed under given knowledge. This contributes to construct a general structure of intelligent CAD systems, for example, it clarifies what kind of inferences should be prepared (a strategy of integration of inference systems) and when they are used in design processes (a strategy of integration of inference execution).

2.3.3 Theory of Design Objects

Various models (models of artifacts) are used for design and analysis. Human designers can model a single object from various points of view. That is, they can get some different models from it and use them. But the important point is that they still regard these models as representation of the same object. So they can transfer new information they get in a model to another. It is not easy for computers to deal such "multi-viewed" object description. We have to re-think about what "modeling" means.

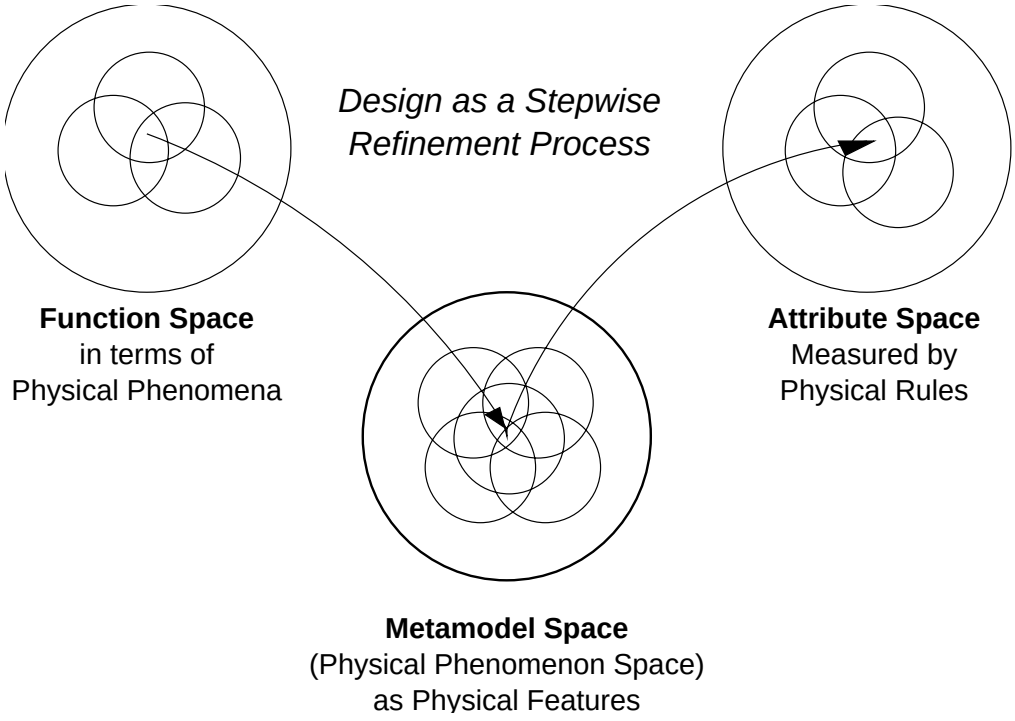


Figure 2.3: Design process in the real knowledge in GDT

"Metamodel" derived from General Design theory is one candidate to solve this problem (17). Every modeling is dependent on some physical laws (or physical views). To say more precisely, every model has a background theory to describe objects, and such background theory is formalization of physical phenomena (i.e., physical law).

Since metamodel is described in physical phenomenon space, we can get specified descriptions (a model) from metamodel descriptions by specifying physical phenomenon without losing information about relations between the model and the metamodel.

Chapter 3

Logical Design Process Modeling

3.1 The Logical Framework for Design

In order to describe design processes in the logical framework, we should clarify what we should represent in logic. Although many factors are complexly related to design, we use three factors which are prerequisite to describe design processes. These factors are required specifications, design solutions (design objects), and knowledge. And we interpret design as logical inference among them.

It may seem natural to take the *deductive framework* to describe design processes in logic. In this approach, we can formalize design as follows;

$S \cup K \vdash Ds$

where S, K, and Ds are sets of formulae that denote required specifications, knowledge used in design, and design solutions, respectively. Here solutions are derived from specifications and knowledge as the results of deduction. In short, this approach adopts the "design is deduction" paradigm.

Many works which explain design or design processes in logic are based on this framework in principle. For example, Treur(18), and Dietterich and Ullman(19) took this approach, and we also took it in Ref. (14).

This "design as deduction" approach may be suitable for routine design, but it cannot offer a sufficient framework for other more flexible and complicated design. Although solutions and knowledge are always incomplete in design, it requires solid and absolute knowledge and solutions.

Then we can use the second framework — the *abductive framework*. In this case, specifications can be derived from design solutions and knowledge.

$Ds \cup K \vdash S.$

Here design is abduction with knowledge and specifications.

Coyne(20) and RESIDUE system(21) stand for this approach for design formalization.

Knowledge represented in this framework is knowledge about objects themselves, i.e., knowledge about object properties and behaviors, because formulae in this framework should be prepared to deduce properties and behaviors of objects from descriptions of objects themselves. It is more desirable than knowledge representation in the deductive framework where knowledge is about how to design.

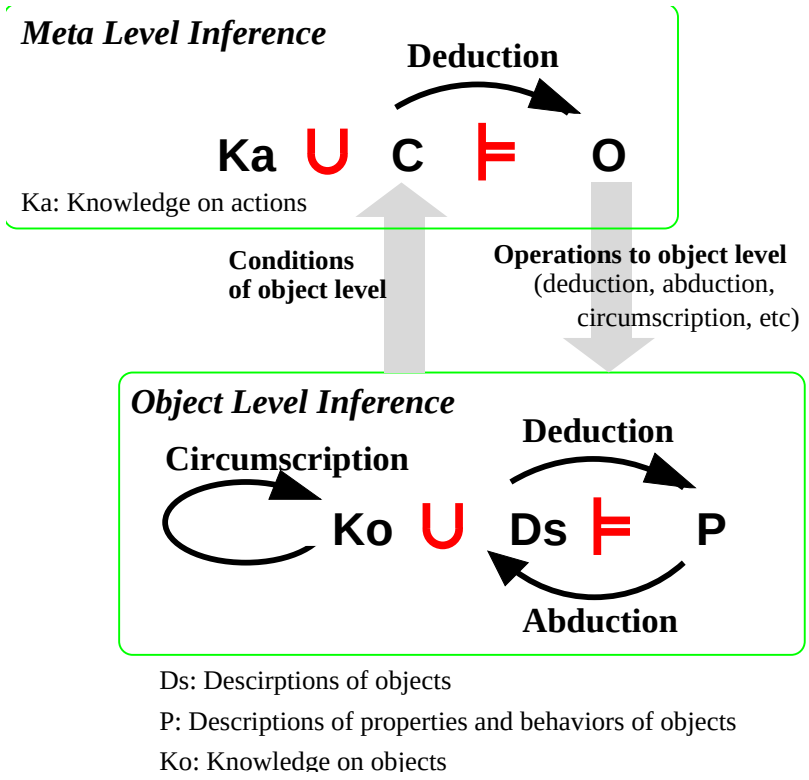


Figure 3.1: The logical design process model

Furthermore solutions the abductive inference can generate are, by definition, not definite solutions but feasible solutions.

Therefore, we adopt the abductive framework as the framework of the logical formalization of design.

3.2 The Logical Inference Model for Design Processes

The inference model we propose is illustrated in Figure 3.1. We define the design process model as a logical inference model.

Here there are two levels in the model, one is the object level and the other is the action level. The object level contains descriptions of design objects (design solution) Ds, knowledge about objects Ko, and descriptions of object properties and behaviors P. P can include required specifications.

The basic design process is interpreted by iteration of abduction and deduction that evolve design objects and their properties and behaviors, and circumscription is invoked to resolve inconsistency.

The action level contains knowledge about actions (knowledge about how to design) Ka, and the meta-level inference is performed to proceed design by specifying inferences in the object level and operating directly the contents of Ds, Ko, and P.

Changing of design objects (Ds) is managed by the multi-world mechanism based on a type of modal logic. Every state of design objects in design processes corresponds to a possible world in modal logic so as to manage multiple solutions and operations to design

processes themselves.

3.2.1 Iteration of Abduction and Deduction as the Basic Process

We regard a design process as an evolutionary process, that is, the design objects are refined in step-wise manner. We call each state of step-wise refinement as a *design state*. In each state, the following three types of descriptions hold; The first one is descriptions of the current design solution which is denoted by Ds. It consists of identifiers of design objects which are components of the current design solution, and properties and relations which are *necessary* to identify the objects. The second one is descriptions of properties and behaviors of the current design solution, P. It consists of all kinds of properties and behaviors that the current design solution has. Required specifications are included in P. The third one is knowledge that is available at the current state, Ko. These descriptions are kept consistent to satisfy the following formula;

$Ds \cup Ko \vdash P$.

Given design knowledge Ko and the required properties P as the specifications, the designer tries to find a candidate by abduction, hence, the current descriptions of the design objects are formed. Then deduction is performed to obtain all the properties of the current solution with respect to the current available knowledge. It is performed (i) to see what properties the solution has and (ii) to see whether the solution does not contradict with the given specifications and knowledge. Then again abduction is performed to evolve the solution more — new descriptions for the next state are formed. If the solution does not satisfy the specifications or can not evolve any more, the designer either tries an alternative solution or modifies the design knowledge and the specifications.

This iteration of abduction and deduction continues until the descriptions of the objects become fully detailed ones that are suitable to hand the next process (e.g., manufacturing) or reach a situation where no more evolutions are possible.

3.2.2 Circumscription for Resolution of Inconsistency

Inconsistency in design has not only negative effects in design but also positive ones.

Most cases of inconsistency in design does not mean that knowledge has wrong information essentially, but that knowledge is used in a wrong manner. Knowledge is used beyond situations where it is expected to be used. But it is not impossible to describe all applicable situations in advance, because it is the nature of knowledge in design that boundary of applicability is vague.

Here we assume that inconsistency comes from such incompleteness of the knowledge description. Then resolution of inconsistency is to find implicit descriptions of knowledge which restrict applicability of knowledge. One solution to accomplish this process is *circumscription*(22)(23).

Circumscription is a type of commonsense reasoning and has been developed to deal with *exceptions*. In circumscription, exceptions for given contexts can be determined by

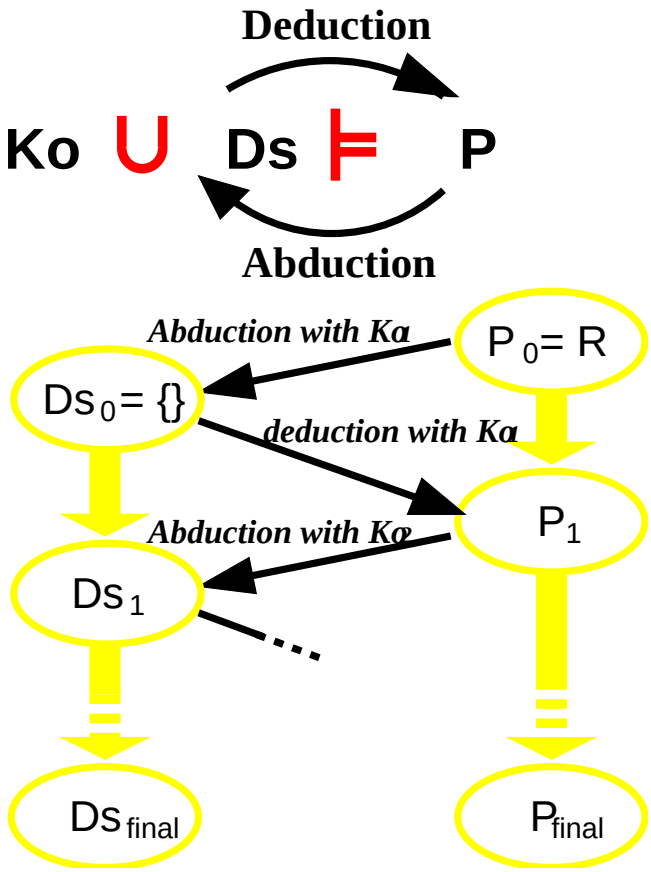


Figure 3.2: Iteration of abduction and deduction

minimizing logical extensions of the predicates which represent *abnormality* with keeping the whole context consistent.

Here abnormality is the implicit description of each piece of knowledge.

When abduction is performed with the modified knowledge by circumscription, we can obtain different results from before. Thus use of circumscription not only solves inconsistency but also helps design to proceed more by modifying knowledge.

3.2.3 Meta Level Inference for Actions

We defined the basic design process as iteration of abduction and deduction on descriptions of objects, knowledge about objects, and descriptions of object properties and behaviors. We also introduced circumscription to resolve inconsistency.

Although they explain what the designer can do with given knowledge and specifications, they can not deal with change of knowledge or specifications because such actions require change of axioms or theorems of the logical system and therefore it is out of a logic system.

In order to solve this problem, we introduce a meta-level inference architecture. Meta-level inference architectures for reasoning are suggested by many researchers. For example, Weyhrauch(24) proposed FOL which is a meta-level reasoning system based on first-order logic. Usually the relation between axioms and theorems in the object-level logical system corresponds to the atomic formula in the meta-level logical system. In our approach, what the meta-level system treats as its atomic formula is the relation among descriptions of objects, available knowledge about objects, and descriptions of object properties and behaviors in the object-level system. We can represent this as follows;

$$Ds \cup Ko \vdash_{L_O} P \Leftrightarrow \vdash_{L_M} design(Ds, Ko, P).$$

where $\vdash_{L_O}$ and $\vdash_{L_M}$ denote derivativeness in the object-level system and in the meta-level system respectively.

The current condition of the three elements in the object level systems is constantly reported to the meta-level system, and the results of inference in the meta-level system are reflected to the object-level system. The reflection is the change of the condition of the object level system, i.e., either specification of the next inference or modification of the contents of the three elements in the object-level system.

Knowledge about how to design can be described as formula in the meta-level system. For example, a rule "if you are designing a certain object g, you should use knowledge base K_g" can be described as follows;

$$design(Ds, K, P) \wedge g \in Ds \rightarrow design'(Ds, K \cup K_g, P)$$

We can thus describe knowledge like design rules and design procedures in this level.

3.2.4 Multi-worlds for Representation of Changing

Each element of the object level system (descriptions of objects, available knowledge about objects, and descriptions of object properties and behaviors) is changed dynamically by

either the object-level inference or the meta-level inference. We introduce the multi-world mechanism based on modal logic to manage this changing.

Since a designer accumulates her or his decisions as the design solution in step-wise refinement processes, it is crucial to distinguish what is already determined from what is not determined yet. It is suitable to represent such situations by partial semantics. Therefore we can use *data logic* to represent them. Data logic(25)(26) is intuitively a version of modal logic based on partial semantics. There are three truth values, i.e., t, f, and u. The third value can be interpreted as *undecided* . Among these values, a partial-order $\sqsupseteq$ is defined where $t \sqsupseteq u$ and $f \sqsupseteq u$ are hold. In this logic, we can access the other possible worlds from a certain possible world, if the value of every proportion in the world is not lower than that in the original world with respect to partial-order $\sqsupseteq$. This means that the *next* world is more determined one than the current world. The truth value u is thus expected to fall into either t or f at last.

Since changing of descriptions of object properties and behaviors (logically it means a set of derivable formulae) is monotonic, we can use possible worlds and accessibility to represent design states and their relations.

In this formalization, if a designer obtains two new different object descriptions from a single solution, two possible worlds are created as descendants of the current possible world. Revision and retraction of the design solution means backtracking to the desirable world (the latest world which does not contradict with the new object descriptions) and creating a new world as its descendant.

Chapter 4

Aspects and Inter-aspect Relationship

Designers have some different kinds of aspects when they recognize artifacts. Some aspects have been developed in traditional engineering fields and have firm theories like kinematics and electric circuits. Other aspects are more vague and have not established firm theories like cost estimation, manufacturability. Some aspects are numerical, others are symbolic or linguistic.

It is nature of design to take some different aspects into consideration. Even if purpose of design (function) can be described in an aspect, artifacts in real world should receive various kind of effects which are included not only in the original aspect but in many different aspects.

Traditional design studies dismiss the importance of aspects and emphasize uniqueness of representation of artifacts. On the other hand, various kind of analysis methods have been developed in the engineering field. But they emphasize completeness of their methods and representation of artifacts. They ignore aspects behind their analysis methods, which are important to use these analysis methods in design.

We discuss multiple aspects in design in two way, i.e., syntactically and semantically. As the former approach, We regard an aspect theory as a logical theory, and we discuss how to represent various aspects in a single framework, and how to cooperate aspects, in particular to combine design and analysis methods. As the latter approach, we regard an aspect theory as *F-B-S*(Function–Behavior-Structure) diagram, and discuss how design is represented in it.

4.1 Aspect in the Logical Framework

Firstly, we discuss syntax of aspect, i.e., give definition of an aspect in logical framework. Secondly we discuss semantics of aspect, i.e., we define aspect as a tuple of function, behavior, and structure.

4.1.1 Aspect Theory as Virtual Logical Theory

Aspects in the engineering field are so various in representation scheme and in reasoning style that it is impossible to provide a single representation scheme with a single reasoning style that covers all the aspects.

Instead of a representation scheme covering for all the aspects, we assume a representation scheme that can be accessed from all the aspects. We employ logic for the shared scheme. Every vocabulary in an aspect is defined in the logical framework. But it is impossible in general to represent whole of an aspect theory as a logical theory, because reasoning in some aspects is beyond logical reasoning. In such cases, we describe every execution of inference as a formula, i.e, condition as premise and results as conclusion. Since such logical formulae would cover all the situations ultimately, we can say we could represent an aspect theory as a virtual logical theory.

Generated formulae can be so enormous and every formula contains so detailed information that it seems difficult to deal with these formulae. But case based reasoning provides the way how to deal with huge case bases, and inductive reasoning can help to make them concise.

For example, an analysis system can calculate the maximum displacement of the beam with given force, we can write a formula as follows;

$$\begin{aligned} beem(X) \quad &\wedge \quad vertical\_force(F) \wedge contact\_with\_the\_end(X, F) \\ &\wedge \quad beem\_bendig\_calculation(X, F, D) \rightarrow maximum\_displacement(X, D) \end{aligned}$$

In antecedent there are conditions to determine whether this system is applicable and a predicate which is interface to the analysis system. In this example, $beem(X) \wedge vertical\_force(F) \wedge contact\_with\_the\_end(X, F)$ are conditions for applying the aspect system, and $beem\_bendig\_calculation(X, F, D)$ is the interface term. It passes values of X and F to the analysis, and returns a value of D as a result.

A virtual logical theory is a set of formulae which are combination of conditions to use analysis systems, and interfaces to them.

From point of view of logical inference, it behaves like ordinal logical theories, and from point of view of application systems it acts as interface between users and other systems.

4.1.2 Structure in Theory

In logical design process modeling, we assumed a single theory K_o as designers' knowledge. As we mentioned, it is not a good assumption to deal with multiple aspects. So we redefine theory in logical process modeling.

When designers design even a single object, they usually use various kinds of knowledge which come from some different aspects. They can manage to combine and use such kinds of knowledge which sometimes seem to be inconsistent to each other. It is important for design to deal with such variety of knowledge.

Instead of assuming a single background theory, we here assume a set of background theories, i.e., the background theory is divided into separate *aspect theories* each of which has its own perspective of description. The perspective of an aspect theory is how to represent phenomena or concepts as propositions in laws or rules. It is definition of vocabulary for the aspect.

Furthermore we assume clusters of knowledge in an aspect theory. We often use some part of an aspect theory instead of the whole aspect theory. A cluster of knowledge is a unit to handle aspect theories. An aspect theory consists of a set of clusters.

We can define aspect theories as follows;

Definition 1 *Aspect.*
An aspect A_i consists of vocabulary V_i, an aspect theory K_i, and clustering of knowledge $KC_i = \{kc_i^j\}$. The aspect theory is a set of formulae written with the vocabulary, and divided into knowledge clusters, i.e., $KC_i = \bigcup_{j \in \Lambda_C} kc_i^j$ where Λ is a set of identifiers for knowledge clusters.

We need knowledge to connect different aspect theories in order to use them together. We call it an *inter-aspect* theory. Since different aspect theories may represent the same phenomena or concepts differently, the inter-aspect theory holds relations among such representations. Then we can define the background theory.

Definition 2 *Background theory.*
The background theory K_0 is union of aspect theories K_i and the inter-aspect theory K_I, i.e.,

$$K_U = \bigcup_{i \in \Lambda} K_i \cup K_I,$$

where Λ is a set of aspect identifiers.

Then an explanatory theory can be defined as collection of clusters of knowledge to explain the given observation (see Figure 4.1).

Definition 3 *Explanatory theory.*
An explanatory theory K for the background theory K_0 is union of knowledge clusters taken from aspect theories in the background theory.

4.2 Function-Behavior-Structure

The ultimate purpose of design is to create artifacts to achieve given functions, i.e., to obtain structural descriptions from functional descriptions. But usually it is not directly done but through intermediate descriptions. Behavior is often used as intermediate descriptions between function and structure. In this section, we model an aspect theory by analyzing meaning of its contents such as function, behavior, and structure.

4.2.1 Function-Behavior-Structure Diagram

Each aspect has definitions of elements which are dealt with in it. And there are a set of attributes of elements and relations among elements, which can be measured and operated in it.

Then we can represent *structure* of objects by attributes and relations. Some of attributes and relations of objects are not constant, and thus we call structure at a moment

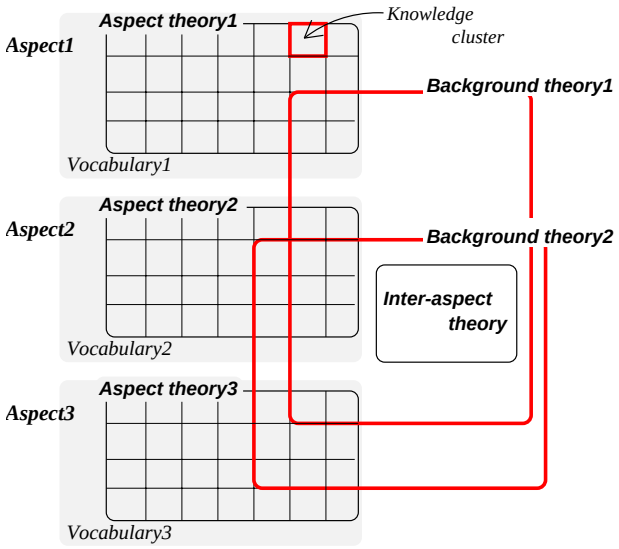


Figure 4.1: Explanatory theory

state of structure. Changing of structure is recognized as *behavior* of objects. It does not mean that all sorts of changing in structure are recognized as behavior. But a given set of behaviors in the aspect enables to recognize some sorts of changing. Some behaviors concerns changing of only attributes, others concern changing of attributes and relations too.

Relationship between structure and behavior is determined by phenomena which the aspect concerns, In particular physical phenomenon are explicitly written as physical laws. Some of phenomena in other fields are also written as laws or rules, for example, economical laws.

Although selecting a set of behavior is arbitrary, definitions of structure and behavior can be strict and definite. On the contrary definition of function is more vulnerable, because it is deeply related to recognition of value by human, that is, related to some subjective affairs. Here we define *function* as interpretation of behavior. Practically it is defined as a set or a network of behaviors. Subjectivity is implicitly expressed in selection of behaviors.

In logical definition, vocabulary of aspects are divided into three, i.e., vocabulary for function, behavior, and structures. Aspect theory consists of descriptions of definitions of elements of function, behavior, and structures, and descriptions of function-behavior, behavior-structure relationship.

4.2.2 Design with FBS

In this section we will explain how design is achieved with FBS diagram in terms of logic.

We represent an aspect FBS system A_x as a proposition $FBS_{A_x}(f, b, s)$ and logical formulae K_{A_x} as definition of $FBS_{A_x}(f, b, s)$. Here lower case term $f, b, s...$ represent a set of variable terms, and upper case term $F, B, S...$ represent a set of constant terms.

If you specify a tuple $\langle F, B, S \rangle$ each of which are constants, the logic system can

determine either

$$K_{A_x} \vdash FBS_{A_x}(F, B, S)$$

or

$$K_{A_x} \nvdash FBS_{A_x}(F, B, S).$$

$FBS_{A_x}(F, B, S)$ stands for complete description of an object, while $FBS_{A_x}(f, b, s)$ indicates all objects that aspect A_x can represent.

If you want to get descriptions of function and behavior from descriptions of structure like diagnosis and analysis, it is represented as follows;

$$K_{A_x} \vdash FBS_{A_x}(f, b, S)$$

As a result of proof, you will get values for variable f and b. In FBS diagram, since behaviour is defined by structure and function is defined by behavior, this process is not difficult.

In pure conceptual design, required specifications are given as function. It seems reasonable to assume

$$K_{A_x} \vdash FBS_{A_x}(F, b, s).$$

But it must be

$$K_{A_x} \vdash FBS_{A_x}([F|f], b, s).$$

Here $[A|b]$ represents a part of this term is matched to constant A and the other matched to variable b. In this formula, constant F stands for required specifications and variable f for derived function or *unexpected functions*.

In general, specifications do not consist of only functions, but mixture of function, behavior, and structure, that is,

$$K_{A_x} \vdash FBS_{A_x}([F|f], [B|b], [S|s]).$$

Here F, B, and S stands for required function, behavior, and structure respectively, and f, b, and s stands for derived function, behavior, and structure respectively.

4.3 Inter-aspect Relationship

We have been putting models and knowledge from various backgrounds in computer in order to support design. But there are no unified methods to integrate them. Since design is not archived with a single model, but various perspectives should be taken into account, integration of various models and knowledge is crucial to realize future CAD systems.

Furthermore future CAD systems should have not only ability of exchanging information in various models to each other, but also ability of guiding use of various models. That is, static and dynamic integration of models are required.

Methodology to exchange information among models tends to deal with objects and aspects ontologically, while integrated use of models in design tends to deal with objects and aspects teleologically. If every relation among models be clear enough, there would be no reason to be teleological. Since we could not expect such a situation in design, designers assume relations under their purpose. Then such relations are very vulnerable

and should be examined by experiment and manufacturing. But if it turns out that they are true and useful relations, then they can be included into ontological knowledge.

The easiest way to use different aspects in the same time is to provide translation between different two aspects. For example, translation between B-reps and CSG representation in geometric modeling and translation between geometric modeling aspect and FEM modeling aspect are well investigated. It may works well among a few aspects, but one-to-one correspondence would be exponential if the number of aspects is increased.

As an approach to model integration, Tomiyama et al.(27) proposed the concept of metamodel for a new modeling framework for design objects. The metamodel is used as (1) as a central modeling mechanism to integrate models, (2) as a mechanism for modeling physical phenomena, and (3) as a tool for describing evolving design objects. Each model in CAD systems is connected only through metamodel where physical phenomena as concepts are used to describe objects. They also proposed a metamodel system based on qualitative physics(28). Here qualitative physics plays an inter-aspect theory among models. It represents physical phenomenon as a model instead of representing every detail of aspect models. It defines conceptual relations between metamodel and aspect models. If two aspects represent one physical phenomena by their own ways, they share the same concept in metamodel. That is, two terms in the two aspects are related via metamodel. The benefit of this approach is that physical phenomenon as mediator are so general that it is expected to be easy to connect new aspects. The disadvantage is that defined relations are so conceptual that it does not imply precise relations like numerical relations.

We showed the way of integration of synthetical method and analytical method by virtual logical theory. We will show an example for it in Section 6.2.3. We do not use qualitative physics but just symbolic representation for object as metamodel.

The metamodel based on qualitative physics can provide basic and common connections among aspect models, but it is not appropriate to represent and describe integration generated in design processes. Integration of aspects differs in every design, and furthermore it is also the goal of design because objects should be represented as integration of aspects. We discuss abduction as a method for dynamic integration of aspects in Chapter 5.

Thus corporation of two types of integration of knowledge could make CAD systems more flexible and more designer-oriented.

Chapter 5

Abduction for Design

5.1 Characters of Abduction in Design

We have presented our model of design processes that consists of abduction, deduction, circumscription, meta-level inference, and multi-world mechanism. Abduction is crucial part of this model, because it should represent synthesis in design. Abduction generates object descriptions as a hypothesis, while other types of reasoning assist this process. Deduction examines validity of the object descriptions proposed by abduction, circumscription maintains knowledge used in abduction and deduction by resolving inconsistency, meta-level inference provides knowledge for abduction, and multi-world mechanism represents evolution of the object descriptions.

Although we have shown the function of abduction in design, we have not discussed mechanism how abduction should be performed. We discuss nature of abduction in design in this section, and then discuss the mechanism to involve such nature of abduction as an inference in the next section.

5.1.1 Abduction in Computer Science

C.S. Peirce introduced abduction as the third kind of reasoning in logic in addition to deduction and induction.

One of important characters of abduction he argued is that direction of inference in abduction is opposite to that in deduction. For example, he demonstrated abduction as follows(29);

> The surprising fact C is observed,
> But if A were true, C would be a matter of course;
> Hence, there is reason to suspect that A is true.

Many logical formalizations for abductive reasoning have been proposed recently (for example, (30) (31) (32)
(21)), but their definitions for abduction are basically similar, i.e., abduction for an observation O with a theory T is to find a hypothesis A which consists of (ground instances of) possible hypotheses and satisfies both $A \cup T \vdash O$ and $A \cup T$ is consistent. This definition is logically sound and suitable to represent the character of abduction mentioned above.

Unfortunately, this definition of abduction fails to capture another important character of abduction. Abduction is *ampliative* reasoning, while deduction is merely *explicative* reasoning. In ampliative inference the conclusion introduces new ideas into our store of knowledge, but it it does not follow from the premises with necessity(33). In explicative inference the conclusion explicates what is stated in the premises and follows form the premises necessarily.

Hypotheses generated by the above definition are *definitely* all what can deduce the given observation with the given theory, and ampliativity is realized just by enumeration of multiple hypotheses.

This *clear and definite* abduction is unattractive in design because of complexity and quantity of object structures and knowledge. Since it translates ampliative ability of abduction into enumeration of multiple hypotheses, it would generate an enormous number of hypotheses. We need the other way to interpret ampliative ability of abduction.

The problem lies in the following two issues. One issue is that they put abduction into a traditional problem solving scheme. It should include not only problem solving but also problem formation to some extent. Although abduction may generate hypotheses by using reasoning like *reversed deduction*, it does not imply that the whole process of abduction is such reasoning. The other issue is lack of structures in hypotheses and the background theory. They assume simple and uniform structures that hide crucial problems in abduction like composition of hypotheses.

In the following discussion, a problem given to abduction to solve is called an *observation*. It represents facts in the target world and it is what we should find explanation for. Knowledge which is used to find explanation is call a *background theory*. A *hypothesis* is an idea conjectured by abduction.

Then we will discuss what kind of characters are needed for abduction in design.

5.1.2 Explanation = Hypothesis + Theory

When a set of facts is given as an observation, abduction is to make hypotheses that explain the given facts with a background theory. It is important to show not only proposed hypotheses but also how the background theory is used to explain the facts. One of the requirements that the proposed hypotheses should satisfy is to show that they can deduce the given facts. In this deduction, at least the background theory that is used in abduction should be included in the axiom.

But there are no reasons that the background theory used in abduction is identical to the whole background theory that exists before abduction. It is natural to assume that the background theory used in abduction can be extracted from the whole background theory. We call this used background theory *an explanatory theory*. Then abduction is to make a hypothesis and an explanatory theory of which combination can explain the given facts.

The content of explanatory theories need not be created newly. To create new rules or laws is another abduction problem in a higher level. An explanatory theory consists of formulae that are selected from the existing background theory.

5.1.3 Integration of Explanations

As we mentioned, the first requirement for hypotheses is to derive the observation with the background theory. But it is not sufficient to restrict generation and selection of hypotheses, and some criteria are proposed as methods, for example, specificity(34) and creditability (35).

In this paper, we use *integration of explanations* for criteria for generating and selecting explanations, It means how parts of explanation are integrated together as an explanation, and this criteria is thus to ensure that an explanation is valid to explain observation as a whole. Integration of explanations has two meanings; One is coherence of explanations that is integration of the way how they explain the observation. It represents plausibility of hypotheses as explanation. We will discuss this problem in Section 5.2.2. The other is integration of hypotheses that is how parts of a hypothesis are integrated together. It represents plausibility of hypotheses themselves. It is related to the problem how abduction can yield new ideas. We clarify it in the next subsection.

5.1.4 Creation of New Ideas

Creation of new ideas in abduction can be realized as new combinations of propositions in hypotheses. Even if every proposition in a hypothesis is well-known, combination of these propositions can express a new idea if it is a new and meaningful combination. Therefore, it is crucial for abduction to provide meaningful combinations in a hypothesis.

In order to obtain meaningful combinations, members of a hypothesis should be related to each other. A set of irrelevant propositions does not carry any new ideas as a hypothesis. Integration of hypotheses to yield new ideas is therefore to conglomerate propositions which are able to carry new ideas.

In design, an important relationship among propositions is sharing of entities. Combination of propositions in a hypothesis is more important in design if they share entities, because it forms a description of an object.

Integration of hypotheses in design is, so far, to form conglomerate of information around entities. More propositions share entities in a hypothesis, more integrated it is to be able to carry new ideas.

This problems is significant when an explanation needs new entities. Since each proposition in a hypothesis is proposed as explanation of some propositions in the observation, it can have its own new entities if the explanation requires entities which are different from entities in the observation.

If different propositions in the hypothesis can share such newly required entities, we can obtain a more integrated hypothesis. It means to identify entities which are introduced independently and therefore have different properties. We call identification of different entities *superposition*. It is a fusion of different concepts and such fusion can produce a new concept. In abduction this process is performed in order to obtain integrated hypotheses. We will discuss how to realize it in Section 5.2.1.

5.1.5 Abduction as Problem Formation

Abduction can work not only to solve problems but also to generate information to define problems.

Since design consists of a lot of ill-defined problems, it is difficult in design to define space for solution and domain of knowledge (heuristics) for each problem in advance. In such domains, abduction can serve to define space for solution and extent of knowledge to solve the problem preliminarily.

As we have shown in the design process model, abduction in design is used iteratively and develops hypotheses gradually. Each abduction stops its inference by using some criteria, and generates hypotheses and explanatory theories gathered from the background theory. Since each hypothesis is not so definite, it does not indicate that the hypothesis is a solution exactly, but that there would exist a solution around the hypothesis. Thus we can use it as preliminary definition of space for solution. An explanatory theory also indicates preliminary definition of knowledge. Because the explanatory theory is collected just to explain the observation, it is not all information we can solve the problem. But at least it is related and maybe needed to solve the problem. It serves, therefore, the first definition of extent of knowledge.

After we obtain preliminary definitions of space for solution and extent of knowledge, we can use more definite methods like deduction.

Furthermore characters of abduction we have explained can generate other information to define problems. We will discuss how these types of information can be used in a knowledge-based inference in Section 5.2.3.

5.2 Formalization of Abduction as a Knowledge-based Inference

We have discussed general characters of abduction in design in the previous section. In this section we focus on how to realize abduction in the current knowledge-based framework.

Here we provide a first-order language $\mathcal{L}$, and explanatory hypotheses, observations, and background theory are written in the first-order predicate language. We can define abduction as follows;

As we mentioned in Chapter 4, instead of assuming a single background theory, we here assume a set of aspect theories. Furthermore we assume clusters of knowledge in an aspect theory.

We defined aspect theories as follows;

Definition 4 *Aspect.*
An aspect A_i consists of vocabulary V_i, an aspect theory K_i, and clustering of knowledge $KC_i = \{kc_i^j\}$. The aspect theory is a set of formulae written with the vocabulary, and divided into knowledge clusters, i.e., $KC_i = \bigcup_{j \in \Lambda_C} kc_i^j$ where Λ is a set of identifiers for knowledge clusters.

Definition 4 *Explanation.*
An explanation of an observation O with a background theory K_0 is $\langle A, K \rangle$, a tuple of an explanatory hypothesis A and an explanatory theory K which satisfy the following conditions;

- $K \subseteq K_0$,
- $K \cup A$ *is consistent,*
- $K \not\models O$,
- $A \cup K \models O$, *and*
- *there are no* $E \subset A \cup K$ *that stratifies* $E \models O$.

We can say that a hypothesis A explains an observation O by an explanatory theory K. In this paper, we restrict both observations and hypotheses to ground formulae, i.e., no variables are appeared in them. Furthermore observations are given as a set of literals (atomic formulae or negation of atomic formulae).

This definition may seem identical to the definition in Section 5.1.1, but an explanation is not a hypothesis but combination of a hypothesis and an explanatory theory, and the whole background theory is not required to use in abduction. But we have not defined how an explanatory theory is taken from the background theory, and what an explanatory hypothesis consists of.

Under this definition, we then define structures of a background theory and hypotheses to realize integration of explanation.

5.2.1 Superposition in Hypotheses

As we discussed in Section 4.1.2, theory in design is not a single one, but consists of serveral aspect theories which is needed for the current design task.

Then we can also divide an explanatory hypothesis as follows;

$A = A_{TH} \cup A_I$

Here, A_{TH} is the derivative hypothesis that can be derived from the background theory and the observation. A_{TH} is a set of formulae with vocabulary $V_0 = \bigcup_{i \in \Lambda} V_i$. A_I is the connective hypothesis that integrates members of the derivative hypothesis (see Figure 5.1). A derivative hypothesis A_{TH} alone can satisfy derivativeness of the observation O, i.e.,

$A_{TH} \cup K_0 \models O$.

Since the hypothesis is generated from combination of different aspect theories, it may be merely a set of hypotheses each of which is generated from an aspect theory. To ensure integration of the hypothesis, we need the connective hypothesis which combines parts of the derivative hypothesis together. We realize this connective hypothesis as superposition of entities.

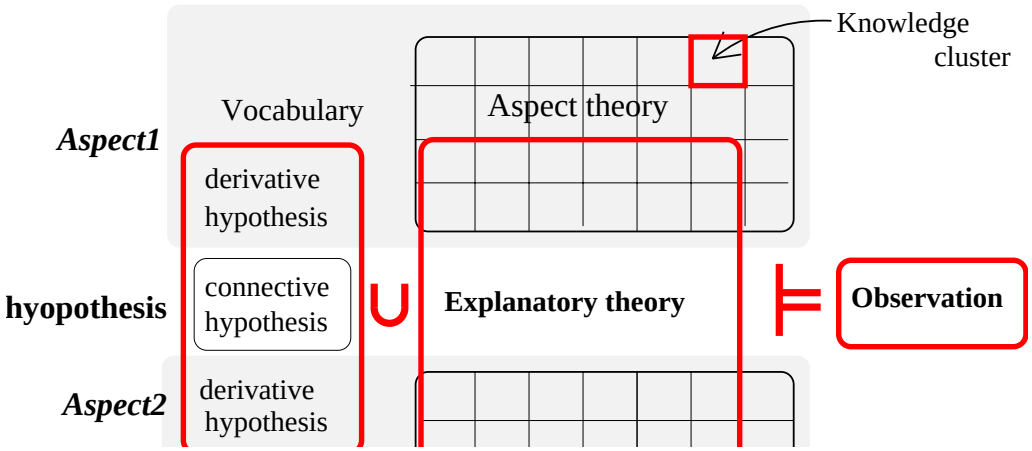


Figure 5.1: Hypothesis and explanatory theory

5.2.1.1 Instantiation of Entities

An observation is a description about entities, and a hypothesis is another description about entities appeared in the observation. But in synthesis one should consider not only entities presented in the observation, but also other entities needed to solve the given problem. We can call these entities instantiated entities.

Introduction of new entities should be careful because it changes the degree of integration of explanation. It is one of important criteria to create and evaluate hypotheses.

Suppose $\boldsymbol{a} = \langle a_1, ..., a_i \rangle$ a tuple of constants appeared in the observation, $\boldsymbol{i} = \langle i_1, ..., i_j \rangle$ a tuple of instantiated constants appeared in the hypothesis A, and $\boldsymbol{x} = \langle x_1, ..., x_i \rangle$ a tuple of variables.

Suppose that there are no constants in the explanatory theory.

We can get $A(\boldsymbol{x})$ by substituting each constant in A, $A(\boldsymbol{x})$ itself can explain the observation too, i.e.,

$$\forall \boldsymbol{x} A(\boldsymbol{x}) \cup K \models O.$$

Since we need hypotheses of ground formulae, we find a substitution θ to all variables in $A(\boldsymbol{x})$ so that $A(\boldsymbol{x})\theta = A(36)$.

We can also represent O as $O(\boldsymbol{y})\gamma_a$ where $\gamma_a = \{a_1/y_1, ..., a_i/y_i\}$ is a substitution. Then

$$\forall \boldsymbol{x} A(\boldsymbol{x}) \cup K \models \forall \boldsymbol{y} O(\boldsymbol{y}).$$

Since the observation is given as $O(\boldsymbol{y})\gamma_a$ not as $O(\boldsymbol{y})$, terms which satisfy every predicate in O should be restricted to constants used in the substitution γ_a. It means that $A(\boldsymbol{x})\theta \cup K \cup O$ should be minimal with respect to each predicate in O. Minimality with respect to a predicate is that the extension of the predicate (a set of tuples which satisfy the predicate) is minimal(37). The extension of a predicate in O for $A(\boldsymbol{x})\theta \cup K \cup O$ should be the same to the extension for O. This restriction can find a substitution θ_a for $A(\boldsymbol{x})$. Abductive procedures with the resolution principle can find this substitution. But still $A(\boldsymbol{x})\theta_a$ can have free variables. Then these free variables in $A(\boldsymbol{x})\theta_a$ are assigned either to instantiated constants or to constants in O. Here θ_s stands for a substitution from

variables to variables, θ_i for a substitution from variables to instantiated constants. Then $A = A(\boldsymbol{x})\theta_s\theta_i\theta_a$. θ_s represents identification between different terms, i.e., the way which entities in hypotheses should be identified.

For example, suppose

$$\{ \quad is\_alive(x) \wedge has(x,y) \wedge wing(y) \wedge is\_feather(y) \rightarrow bird(x),$$
$$has(x,y) \wedge wing(y) \wedge is\_big(y) \rightarrow fly(x)\}$$

as K and $\{bird(a),\ fly(a)\}$ as O. If there are no ideas to identify entities, both

$$A_1 = \{is\_alive(a), has(a,b), wing(b), is\_feather(b), has(a,c), wing(c), is\_big(c)\}$$

and

$$A_2 = \{is\_alive(a), has(a,b), wing(b), is\_feather(b), is\_big(b)\}$$

can be hypotheses. The former seems redundant, but both hypotheses are minimal because $A_1 \not\supset A_2$ and $A_1 \not\subset A_2$. The difference is the way how to introduce entities in hypotheses.

5.2.1.2 Minimality of Entities in Explanation

One of criteria to integrate hypotheses is minimality of entities. Domain circumscription (23) can be used to achieve minimality of entities in explanations. Domain circumscription finds models that have minimal domains to hold given formulae. In this case $A(\boldsymbol{x})\theta_a \wedge K \wedge O(\boldsymbol{a})$ is a formula to circumscribe. But using domain circumscription without any restrictions will make undesirable results. For the above example, we can get

$$\{is\_alive(a), has(a,a), wing(a), is\_feather(a), is\_big(a)\}$$

as a hypothesis with domain circumscription. This hypothesis seems unnatural, because we have knowledge about what kind of entities can be unified or not. In this case, entities which can satisfy $wing(x)$ and $bird(x)$ should be different, while entities which can satisfy $wing(x)$ can be unified to each other[1].

Superposition is identification between entities, but it is specified by two propositions which have entities to be identified.

Although it is impossible to describe all possible unifiable entity relations in knowledge[2], we can postulate at least consistency of aspect theories. Relations among predicates in an aspect are all what are written in the aspect theory. If two proposition have predicates in the same aspect, they are not allowed to identify unless these predicates are the same.

Suppose

$$K_1 = \{is\_alive(x) \wedge has(x,y) \wedge wing(y) \wedge is\_feather(y) \rightarrow bird(x)\}$$

$$K_2 = \{part(x,y) \wedge lift\_force\_device(y) \rightarrow fly(x)\}$$

[1]It is not a matter of course. If there are more than two entities which satisfy the same predicate, each of such predicates can be related to different entities.

[2]It is the frame problem to enumerate all combinations among predicates (38).

$K = K_1 \cup K_2$

$O = bird(a) \wedge fly(a)$

where K_1 and K_2 are aspect theories. We can get a hypothesis

$A = \{is\_alive(a), has(a,b), wing(b), is\_feather(b), part(a,c), lift\_force\_device(c)\}$.

If we assume superposition $\{has(x,y),\ part(x,y)\}$ and $\{wing(x),\ lift\_force\_device(x)\}$. Then the hypothesis is

$A' = \{is\_alive(a), has(a,b), part(a,b), wing(b), is\_feather(b), lift\_force\_device(b)\}$.

Notice such superposition is also a hypothesis, and validity of the superposition is examined by deduction and further abduction from the whole or part of the hypothesis A'. In particular, part of the hypothesis which includes identified entities is important in further abduction and deduction in order to realize how the superposition is feasible. In this example, it is $\{wing(b), is\_feather(b), lift\_force\_device(b)\}$.

5.2.2 Explanatory Coherence

Ng and Mooney(39) proposed *explanatory coherence* as the primary measure to evaluate the quality of an explanation. Explanatory coherence computes the degree of connectivity of a hypothesis as follows;

$$C = \frac{\sum_{1 \le i \le j \le l} N_{i,j}}{Nl(l-1)/2}$$

where l is the total number of the observation, N is the total number of nodes in the proof graph, and $N_{i,j}$ is the number of distinct nodes n_k in the proof graph such that there is a sequence of directed edges from n_k to n_i and also n_k to n_j where n_i and n_j are elements of the observation.

This quantity may be useful to compare some tightly connected hypotheses, but we need a more qualitative scale to evaluate coherence of explanations where connectivity is not so tight, and finding connectivity of explanation itself is one of purposes of abduction.

Here we introduce a *coherent segment of explanation* to evaluate explanations.

Definition 5 *When an explanation $\langle A, K\rangle$ for an observation O is given, a partial explanation $\langle A(O'), K(O')\rangle$ for the observation $O' \subset O$ is defined as follows;*

$A(O')$ and $K(O')$ are both minimal sets of formulae that satisfy $A(O') \subseteq A$, $K(O') \subseteq K$ and $A(O') \cup K(O') \models O'$.

In case of multiple partial explanations, we denote $A(O')[i]$ and $K(O')[i]$.

Definition 6 *Given an explanation $\langle A, K\rangle$ for O, $O_1 \subset O$ and $O_2 \subset O$ are directly connected to each other if and only if $\bigcup_{i,j}(A(O_1)[i] \cap A(O_2)[j]) \cup \bigcup_{i,j}(K(O_1)[i] \cap K(O_2)[j])$ is non empty. This set is called "direct connection of explanation between O_1 and O_2".*

Direct connection of an explanation corresponds there exists n_k for specified n_i and n_j in Ng and Mooney's definition.

Definition 7 *Given an explanation $\langle A, K \rangle$ for O, $O_1 \subset O$ and $O_2 \subset O$ are indirectly connected to each other if O_1 and O_2 are directly connected to each other or there is $O_3 \subset O$ that is indirectly connected to both O_1 and O_2.*

Definition 8 *Given an explanation $\langle A, K \rangle$ for O, if every element of $O' \subseteq O$ is indirectly connected to other element in O', and any element in $O - O'$ is not indirectly connected to element in O', $\langle A(O'), K(O'), O' \rangle$ is a coherent segment of explanation.*

If the number of coherent segments for an explanation is 1, the whole explanation is connected. If the number is more than 1, the explanation includes some explanations that are not related to each other. This coherent segments are calculated by tracing dependency of members of hypotheses.

5.2.3 Information for Next Inferences

Abduction can generate useful results for next inferences.

As discussed, an explanation consists of not only a hypothesis but also an explanatory theory. Furthermore we can obtain superposition of propositions to identify entities, and coherent segments of the explanation.

These types of information are used by two different ways, i.e., for successive inference steps, and for inferences to solve similar problems.

Explanatory theories and explanatory coherence are useful for successive inferences. An explanatory theory is a primary theory for deduction that is used to confirm the hypothesis proposed by abduction, i.e., deduction should use at least this theory. And it is also a reference theory for the next abduction. Clusters of knowledge in an explanatory theory are most plausible candidate theories for the next abduction.

Explanatory coherence helps us to determine the extent of next problems. If there are some coherence segments, it seems reasonable to divide the current problem. Even if there is only one coherent segment, we can form sub problems by direct or indirect connections among members in the observation and the hypothesis.

Explanatory theories and superposition of propositions are useful to solve similar problems. The sets of clusters that are used in abduction are examples of adherence among knowledge. And superposition of propositions is also examples of connections among predicates in different aspect theories. By gathering such information, we can develop the inter-aspect theory, or add more descriptions in meta-level knowledge (see Section 3.2.3).

Chapter 6

Design Simulation

6.1 Design Simulator

We implemented a prototype of the *design simulator* that realizes the inferences discussed in Section 3. We call it design simulator because it is designed to track the design processes performed by designers. The purpose of this system is to show the proposed model is computable as well as suitable to represent design processes.

This system is implemented in Allegro Common Lisp, CLX (Common Lisp X interface), and X11 on Sparcstations.

6.1.1 The Architecture

The design simulator consists of three main parts; i.e., the action level inference system, the object level inference system, and the multi-world management system (see Figure 6.1). The object level inference furthermore consists of workspace Ds, P, and Ko, and three inference subsystems, i.e., deduction, abduction and circumscription subsystems.

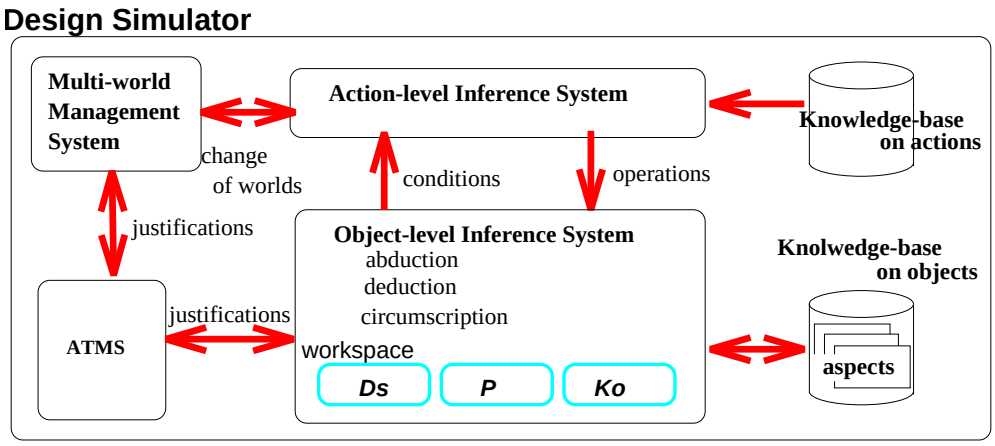


Figure 6.1: The architecture of the design simulator

The system invokes the action level inference and the object level inference mutually.

Every time it executes the action level inference and gets the results, it asks the user to choose one among the results which are a list of operations to the object level. It executes then the object level inference according to the user's choice. Again it goes back to the action level and repeats this process until there are nothing to infer. It also invokes the multi-world mechanism when a new design state is required (see Figure 6.2).

6.1.2 The Action-level

The action-level inference system works as the meta-level to the object-level inference system. The inference on this level is currently performed by a rule-based deductive system. Knowledge used on this level is about how to design, for example, knowledge about selecting a knowledge base and scheduling inferences according to the condition of the object level. Results of inference in this level is a single or a sequence of operations on the object level.

6.1.3 The Object-level

Abduction, deduction, and circumscription are provided so as to change the current state of Ds, P, and Ko in the object level. Workspace Ds contains the current descriptions of objects, P the current descriptions of object properties and behaviors, and Ko the current available knowledge.

Ko is a set of Horn clause formulae, while P and Ds are a set of literals, i.e.,, atomic formula or its negation. Inference on the object level modifies the contents of Ds, P, or Ko and sometimes causes contradictions, which are reported to the action-level as a condition of the object level.

The Abductive Inference There are some studies in which the abductive inference with Horn clauses is realized by using the resolution principle(21)(31). The basic idea is that we can obtain a subset of Horn clause database that is used to infer whether the given formula is derivable from the database by the resolution principle. Suppose G is a single or a set of atomic formula, $\mathcal{A}$ is a set of formulae which represent possible hypotheses, and $\mathcal{K}$ is a set of formulae which represent knowledge. We can find $A \subseteq \mathcal{A}$ and $K \subseteq \mathcal{K}$ which satisfy $A \cup K \vdash G$. This A is the result of abduction.

This algorithm may generate many solutions which include trivial ones (e.g., G itself). We choose only maximal solutions with respect to the relation *deeper* in order to eliminate these solutions. Here a set of formulae A_1 is *deeper* than A_2 iff $A_1 \cup K \vdash A_2$ and $A_1 \not\supseteq A_2$.

Circumscription The circumscription system is implemented using the algorithm proposed by Nakagawa and Mori(40) that is an algorithm for computing circumscription(22) on clausal forms. Their basic idea is to use the technique of program transformation such as *unfolding* when eliminating variable predicates and abnormal predicates.

6.1.4 Multi-World Management System

The multi-world management system keeps design states and their relations as data dependency cooperated with ATMS system(41). The idea of multi-worlds with ATMS is shown by Morris and Nado(42). Here we can operate the multi-worlds by logical formulae with modal operators 2 (necessity) and 3 (possibility).

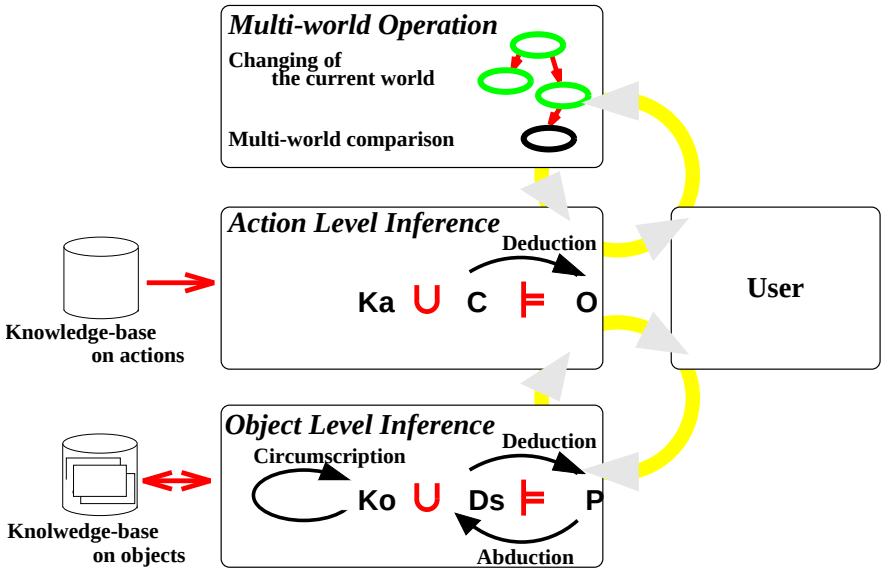


Figure 6.2: Inference in the design simulator

6.2 Examples

6.2.1 Example 1 — Tracking of Design Processes —

We demonstrate how this system works with a set of formulae which we extracted from the protocol data(43). Figure 6.3 shows the original protocol data. We interpret this design session as an inference by knowledge. We pick up pieces of protocol (verbal protocol and figures), and represent them as logical formulae. Inference procedures in the system has been described in Ref. (44).

This is the first part of the design of which the task is "to design a scale". Figure 6.4 shows how Ds (descriptions of objects) and P (descriptions of object properties and behaviors) change during the inference. In this figure, a bold formula denotes a newly added one, and an italic formula indicates a contradiction.

World 1 (Figure 6.4(a)) is the beginning of a session where there are only the required specifications in them. Then the action level inference prepares available knowledge. After abduction and deduction are executed three times we can obtain World 5 (Figure 6.4(b)). This state corresponds to protocol 8 in Figure 6.3. Then, the formula which represent protocol 11 and 12 are introduced into Ko. This makes an inconsistency situation (World 6, Figure 6.4(c)). It is because protocol 11 and 12 are contradictory sentences to protocol

(1) What mechanism does a standard scale use?
(2) It measures the weight like this (Figure A).
(3) You use the spring to pull, but you can use it oppositely.
(4) If we use it to push, it is like this (Figure B).
(5) If we can use a rack and pinion (Figure C), we can measure the weight because the displacement is in proportion to the weight.
(6) Do you know any other way to support the weight?
(7) No, only spring.
(8) Anyway, we think the indicator first.
(9) As a conclusion, what we want is something to measure the displacement (x in Figure A).
(10) It is better to make it easy to see.
(11) Since it translates 5mm of the displacement to 100kg weight, the displacement per 1kg is 0.05mm.
(12) It is impossible to realize it with this (Figure C).
(13) If we don't mind the accuracy, it is possible by using many gears.
(14) Indicators in scales we can buy are upturned.
(15) Then, we have to use helical gears, but scales we can buy must use simpler mechanism.
...

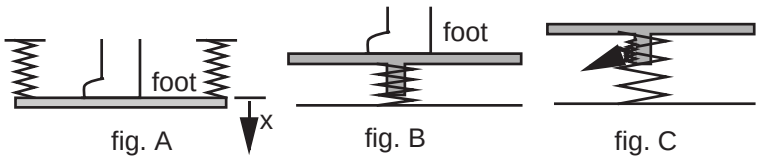


Figure 6.3: The examples of protocol data

5. Circumscription modifies the formula which represents protocol 5, and it makes us abandon the current solution. The we obtain another solution (World 9, Figure 6.4(d)). Finally we obtain Ds and P shown as Figure 6.4(d). Figure 6.5 shows all the worlds generated during this session. The path from the world 19 to world 21 does not appeared in the protocol data, but the system can generate such possible design processes. The difference between World 18 and World 21 is whether the spring is used for compression or expansion.

6.2.2 Example 2 — Design with Multi-aspects —

In this example, we assume multi-aspects in designers' knowledge. The example is taken from the same series of design experiments to Example 1, but the experiment done by different subjects.

In this session, we assume five aspects, i.e., *scale aspect, exterior-design aspect, support-motion aspect, translate-motion aspect*, and *manufacturing aspect.* These aspects have rules that are representation of designers' knowledge in this design session.

The specification is to design a scale. It means to design an object that can support and measure given weight (see Figure 6.7(a)).

Then designers suggest a structure of typical scales(see Figure 6.7(b)). Revered lines are newly added members of the hypothesis.

Since "`(support sc1 w)(measure sc1 w)(weight w)(quantity 100kg w)(move d w sc1) (displacement d)(quantity 5mm d)`" have not been abduced yet and other

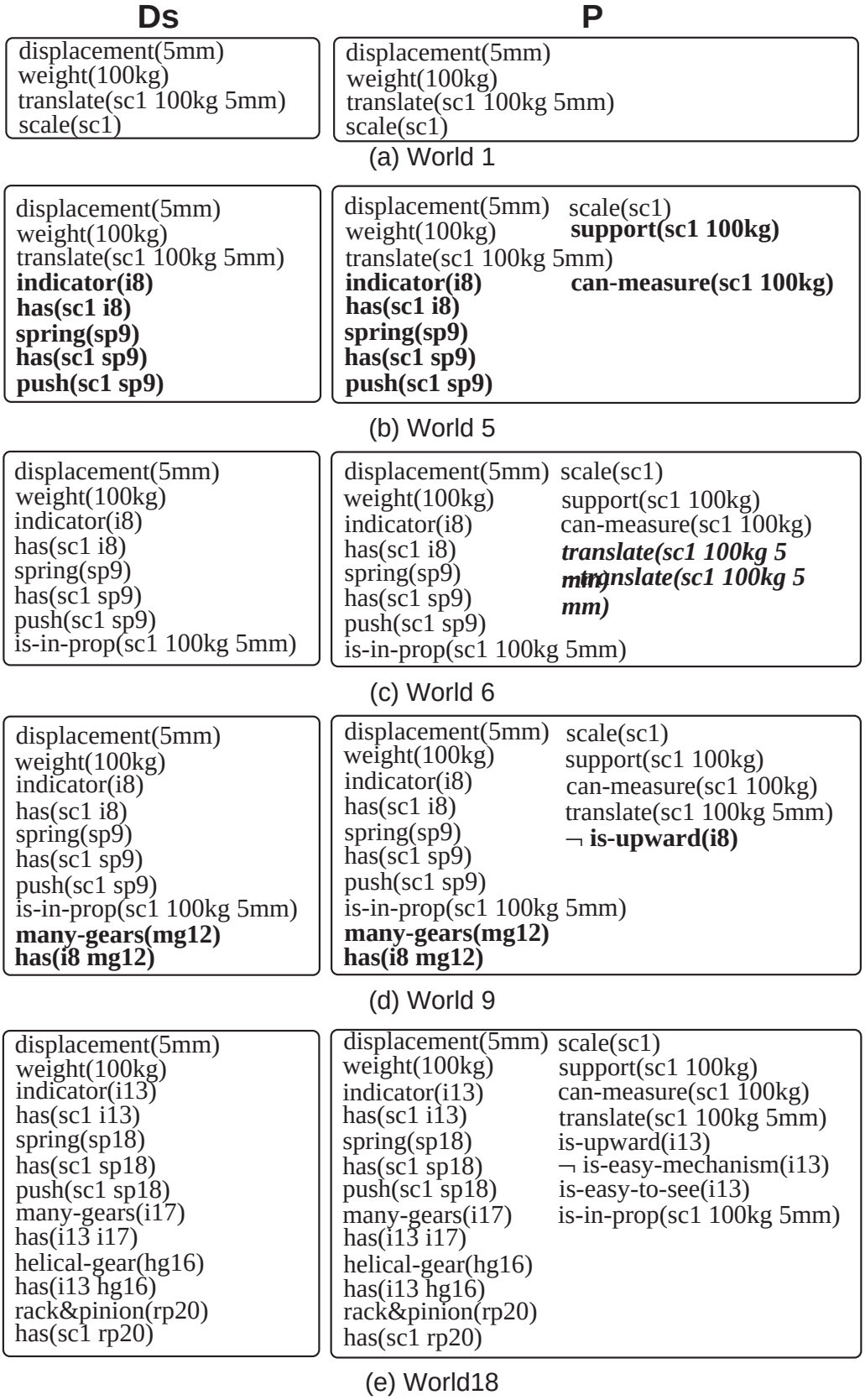


Figure 6.4: Changing of Ds and P

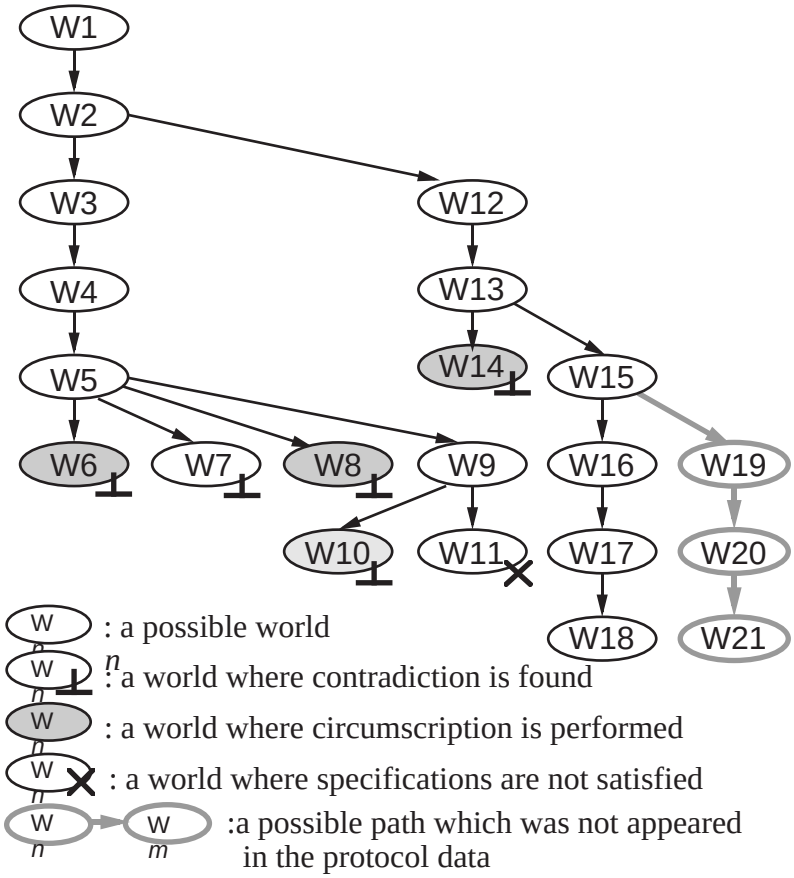


Figure 6.5: The generated worlds

members of the hypothesis are connected to each other at this moment, there are eight coherent segments. Then they abduce "`(support sc1 w)`"and get a hypothesis using the following rule;

```
(support *s *w) <-
          (upper-frame *u)(has *s *u)(base-frame *b)(has *s *b)
          (slide-guide *sl)(connected *sl *u)(connected *sl *b)
          (pickup *pk)(has *pk *sc)
          (stopper *st)(connected *st *u)(connected *st *b))
```

This rule introduce "`(stopper st14)`" and "`(slide-guide sl11)`', and also make these propositions connect to the segment which includes "`(upper-frame u3)`" and "`(base-frame b1)`". Thus the number of coherent segments is decreased. Using the exterior-design aspect, they get a new hypothesis as design descriptions shown in Figure 6.7(c).

Furthermore they decide to connect the plastic cover and the upper base by screws. "`(connected dp15 ub17)`" is abduced to "`(screw z18)(fixed z18 ub17)(fixed z18 dp15)`" by a rule in manufacturing aspect, Then they notice these screws can be used as the stopper of the vertical movement. They identify "`(screw z18)`" and "`(stopper st14)`". Figure 6.6 is a snapshot of the design simulator when superposition of propositions is asked to users. Using some other rules, they can get a hypothesis shown in Figure 6.7(d). In this hypothesis, revered lines are descriptions of the entity which should play two roles in this design. One problem they should solve next is to develop and examine descriptions of this entity.

6.2.3 Example 3 — Connection to Analysis —

In the session of scale design, the designers adopted a spring system to translate weight into displacement and wire-spring-pulley mechanism to indicate displacement.

When we interpret aspect of mechanical dynamics, we can extract models of mechanical dynamics from descriptions of the scale. Then we can simulate behavior of the scale.

We consider the spring system to translate weight into displacement, we just use three element in the aspect, i.e., mass, spring, and dumper. `upper-frame` and a `human-body` are material particles, `springs` are a spring and a dumper. The properties of material particle, spring, and dumper are mass, spring rate, and dumping rate respectively.

We can extract a system structure and some properties of elements in the model using knowledge base about mechanical dynamics parts from the current descriptions of design objects.

In order to simulate motions, we should fill other properties of elements and add initial and boundary conditions which are not specified in the previous design session.

In this example, there are no descriptions for dumping rate and no clear initial conditions. Firstly we assume 1,000 [Ns/m] for the dumping rate here. We assume initial conditions as follows;

- The system is an equilibrium state before putting mass M

And we assume that the lower-frame is fixed and a mass of M is put on the upper-frame without force. It means that force from outside is constant M, and velocity is 0 at $t = 0$,

The initial situation is as follows;

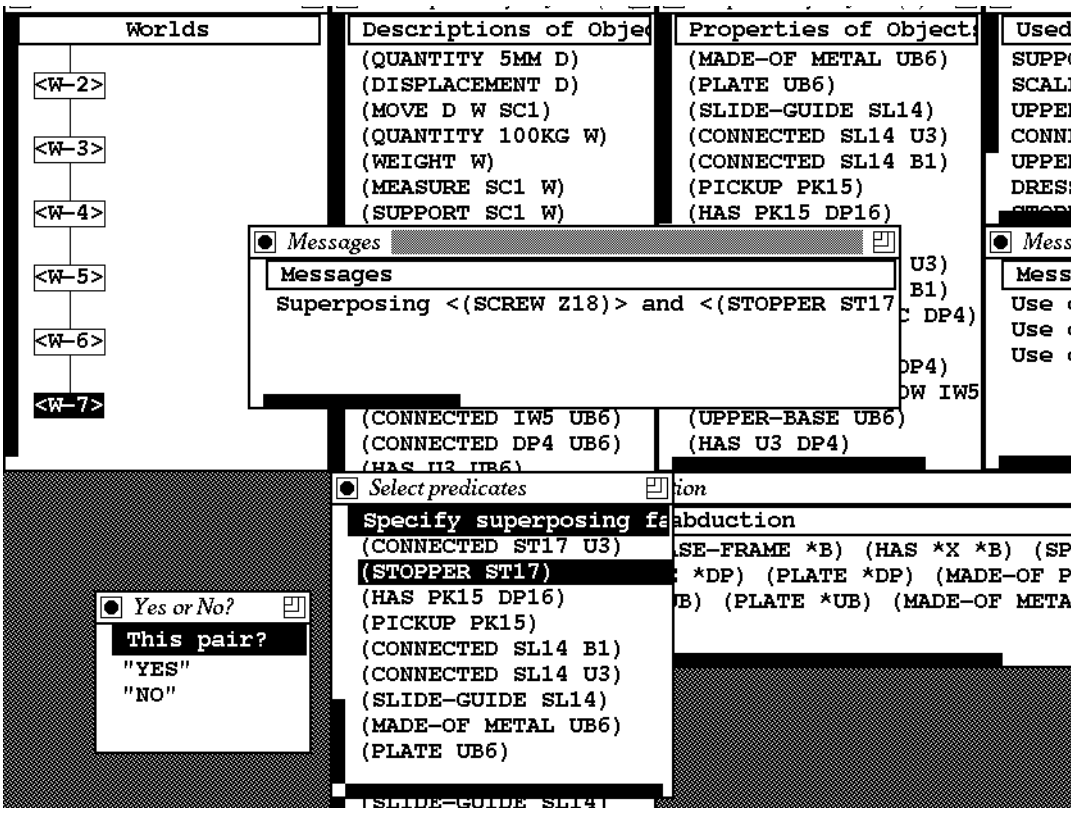


Figure 6.6: A snapshot of the design simulator

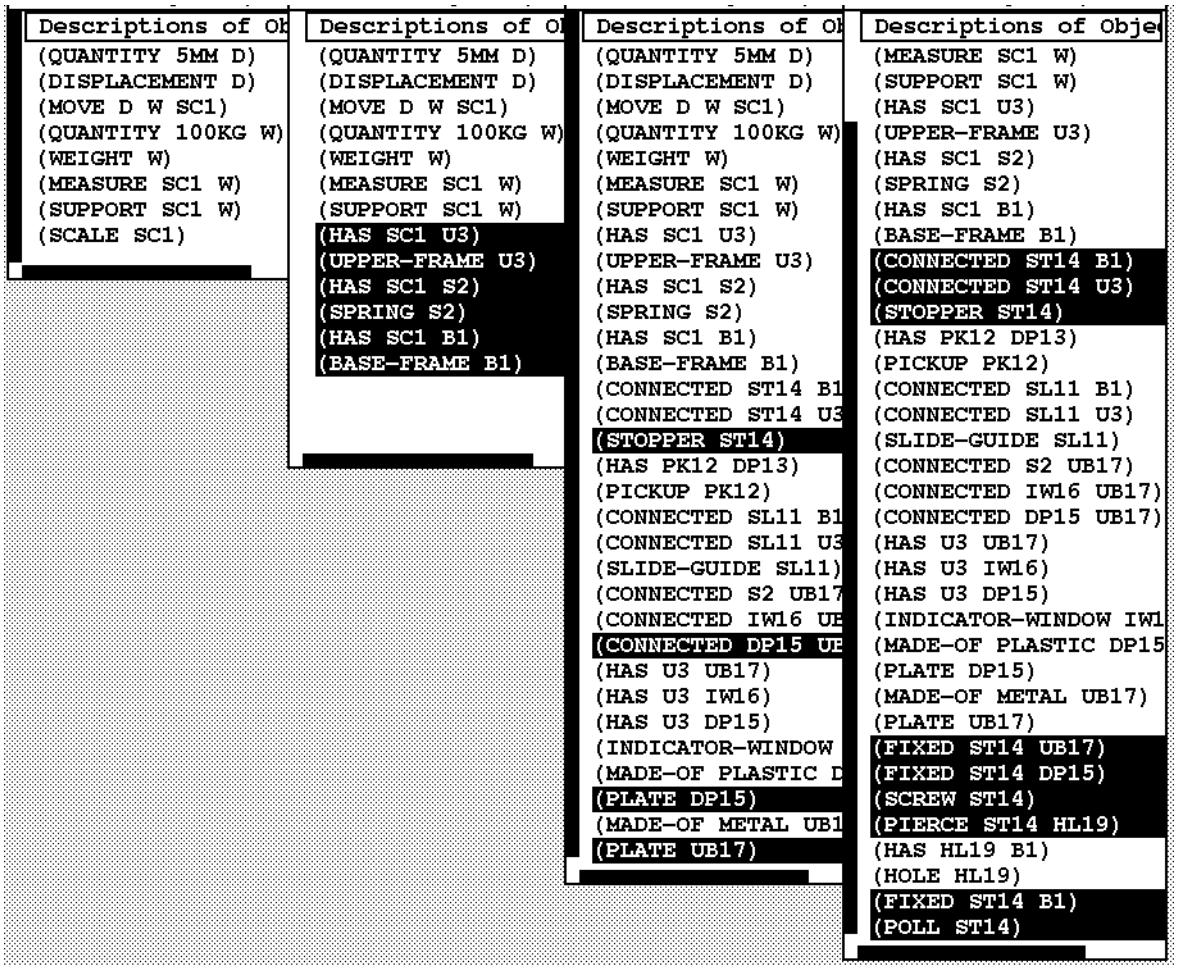


Figure 6.7: Changing of descriptions of design objects

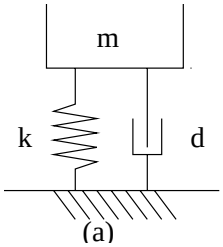


Figure 6.8: A simple mass-dumper-spring system

```
(upper-frame uf) (lower-frame lf) (spring sp) (body bd) (fixed uf sp)
(fixed lf sp) (spring-rate sp 196000) (damping-rate sp 5000)
(weight bd 100) (axis sp z) (ground lf) (movable uf z)
```

Weight of the upper-frame is calculated from its shape, dimension, and density. Shape and dimension are dealt with geometric aspect in which sizes and shapes are explicitly dealt with. It is the results of the current design, and can be supplied by the design system. Furthermore we added the following proposition to determine the initial situation.

```
(fixed bd uf)
(force bd 981 z)
```

The first proposition means that the human body is fixed to the upper frame, and the second that it receives z-axis force of 981 [N].

Then we introduce knowledge about mechanical-dynamic aspect. In this aspect *unit* is defined, which shows what is a member of the aspects, and relations like *fixed* between units, properties like *mass-of, spring-rate, dumping-rate* for units. For example, *mass-unit* consists of one solid object or some fixed solid objects and has *mass, axis* and *force* as property

In this aspect, we have a simple model which consists of a mass, a dumper, a spring, and ground shown (figure 6.2.3).

In this case, the human body and the upper frame form the mass in the simple model, and the lower frame is eliminated by regarding as a part of ground.

Then we can get the following results.

```
(SIMPLE-MD-SYSTEM ((UF BD) (SP) Z))
(K-OF-MD-SYSTEM ((UF BD) (SP) Z) 196000)
(D-OF-MD-SYSTEM ((UF BD) (SP) Z) 5000)
(M-OF-MD-SYSTEM ((UF BD) (SP) Z) 101.25)
(F-OF-MD-SYSTEM ((UF BD) (SP) Z) 981)
```

The first proposition means that a simple mass-spring-dumper system can consist of the upper frame and the human body as a mass unit, spring as a spring and dumper unit.

In the initial situation, z coordinate of the upper frame is 0, and its derivation is also 0.

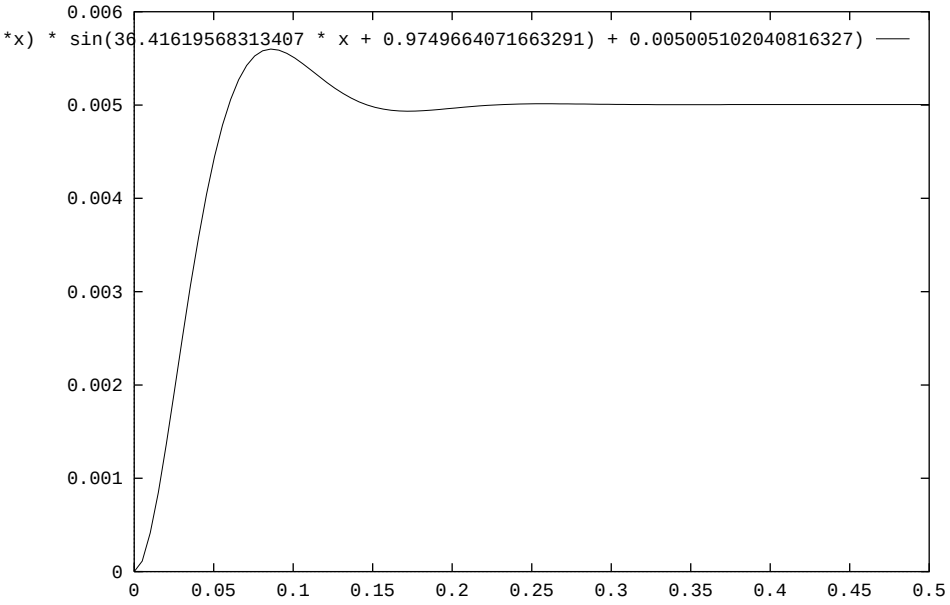


Figure 6.9: Behavior of displacement

```
(cord uf z 0)
(derv uf z 0)
```

The formula corresponding this model is a simple as follows;

$mx'' + dx' + kx = F$

Since solution of this formula is well known, we can solve it for x with the initial condition (see 7).

Using these formulae, we can get the solution as follows;

```
(BEHAVIOR-OF-DISPLACEMENT-OF-MD-SYSTEM
  ((UF BD) (SP) Z)
  (+ (* -0.006047124104574503 (EXP (* -1 24.691358024691358 X))
     (SIN (+ (* 36.41619568313407 X) 0.9749664071663291))) 981/196000))
```

$$x = -0.00605 \exp(-24.7t) \sin(36.4 \times t + 0.975) + 981/196000$$

Then we can trace the behavior of this value. Figure 6.9 shows behavior of displacement.

The behavior of acceleration is shown as Figure 6.10.

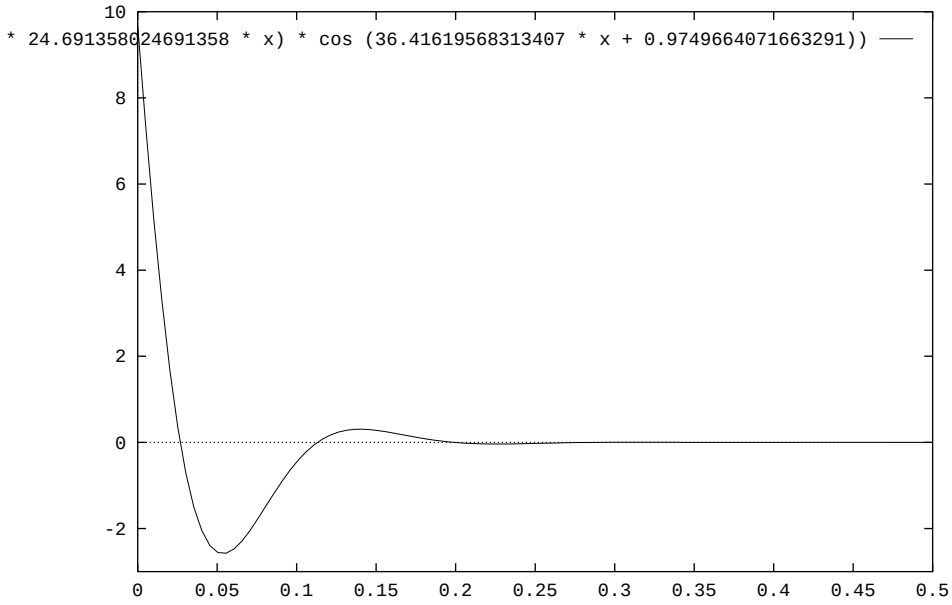


Figure 6.10: Behavior of acceleration

```
(BEHAVIOR-OF-ACCELERATION-OF-MD-SYSTEM
  ((UF BD) (SP) Z)
  (- (* -0.006047124104574503
        (- (SQR 24.691358024691358) (SQR 36.41619568313407))
        (EXP (* -1 24.691358024691358 X))
        (SIN (+ (*  36.41619568313407 X) 0.9749664071663291)))
     (* 2 -0.006047124104574503  24.691358024691358 36.41619568313407
        (EXP (* -1 24.691358024691358 X))
        (COS (+ (* 36.41619568313407 X) 0.9749664071663291)))))
```

$$
\begin{aligned}
x \;=\; & -0.00605 \times (24.7^2 - 36.4^2) \times \exp(-24.7t)\sin(36.4t + 0.975) \\
& -2 \times (-0.00605) \times 24.7 \times 36.4 \times \exp(-24.7t)\cos(36.4t + 0.975)
\end{aligned}
$$

From this behaviors, we can pick up some important features.

- The maximum of positive displacement from the equilibrium position is about 0.6 mm. It means that clearance between the upper frame and the lower frame should be more than 0.6 mm (usually it should be multiplied by safety factor).
- The minimum of acceleration is about -2.58. It tells us how tension of measurement wire should have.

These features, then, are used to modify the current design object.

We can analyze the behavior of the wire which is connected to pulleys, gears, and a spring.

Chapter 7

Summary

In this report, we discussed what is a design process and how we can support design by computer. Firstly, we defined design in a logical framework and modeled a design process as a logical inference. We adopted not only classical logical inference but also non-classical logical inferences such as abduction, circumscription, and meta-level inference. Thus a design process is represented formally by the logical framework.

Then we discussed aspects in design. Designers use various aspects even in a single design. Variety of aspects is essential, that is, to involve various aspects is necessary for design because any artifacts in the real world can not be represented by a single aspect. We showed two approach to represent aspects. One is an aspect as virtual logical theory, and the other is an aspect as F-B-S(Function-Behavior-Structure) diagram. Furthermore we discussed how different aspects are related in design. One is ontological relationship, and the other is teleological relationship. The former is relationship already established between aspects, and it is described as a set of relations between attributes in aspects. The latter is relationship which we wish to establish in design. We described it by abduction.

Abduction is reasoning to find feasible hypotheses from given theory. In this report, we characterized abduction as integration of aspect theories. Abduction with mixture of some aspect theories proposes hypotheses as results. But most parts of a hypothesis describe objects from a single aspect because each aspect theory is isolated in the theory. By pasting such single-aspect descriptions on an object, we can create a real object which has multiple aspects. This is another way to achieve integration of aspects. In other words, designing itself is integrating of aspects.

Finally, we built a system called *design simulator* to examine how our discussion can be achieved in a computer system and what our theory can solve. Three examples are shown.

In this report, we emphasized theoretical approach to build future CAD systems. Furthermore we concentrated to design processes but design objects. Thus, our research alone is not enough to realize future CAD systems, but theory of objects and practical knowledge of design are needed to cooperated with it. Since theory of design processes is a backbone to integrate such theory and knowledge, we hope that this research will accelerate realization of new-generation CAD systems.

Acknowledgement

The author would like to thank Prof. Øyvind Bjørke for his useful advice and support. The author would also like to thank Prof. Kesheng Wang for his kind advice and useful discussion with him, and Dr.ing students in Production Engineering for their kind advice in research and also in living in Norway.

The author are also grateful to Prof. Hiroyuki Yoshikawa and Prof. Tetsuo Tomiyama of the University of Tokyo. The help provided by all former colleagues is gratefully acknowledged.

Bibliography

[1] P.J.W. ten Hagen and T. Tomiyama, editors. *Intelligent CAD systems I: Theoretical and Methodological Aspects.* Springer-Verlag, Berlin, 1987.

[2] H. Yoshikawa and D. Gossard, editors. *Intelligent CAD, I.* North-Holland, Amsterdam, 1989.

[3] V. Akman, P.J.W. ten Hagen, and T. Tomiyama, editors. *Intelligent CAD systems II: Implementation Issues.* Springer-Verlag, Berlin, 1989.

[4] H. Yoshikawa and T. Holden, editors. *Intelligent CAD, II.* North-Holland, Amsterdam, 1990.

[5] P.J.W. ten Hagen and P.J. Veerkamp, editors. *Intelligent CAD systems III: Practical Experience and Evaluation.* Springer-Verlag, Berlin, 1991.

[6] H Yoshikawa and F. Arbab, editors. *Intelligent CAD, III.* North-Holland, Amsterdam, 1991.

[7] T. Tomiyama. Intelligent CAD systems. In *Eurographics '90 Tutorial Note 2.* Eurographics Technical Report Series, 1990.

[8] VDI. VDI guidelines 2221: Systematic approach to the design of technical systems and products, 1987.

[9] K. Roth. *Konstruieren mit Konstruktionskatalogen.* Springer-Verlag, Berlin, 1982.

[10] V. Hubka and W.E. Eder. *Theory of Technical Systems.* Springer-Verlag, Berlin, 1988.

[11] H. Yoshikawa. General design theory and a CAD system. In T. Sata and E.A. Warman, editors, *Man-Machine Communication in CAD/CAM, Proceedings of the IFIP Working Group 5.2 Working Conference 1980 (Tokyo)*, pp. 35–58. North-Holland, Amsterdam, 1981.

[12] H. Yoshikawa. CAD framework guided by general design theory. In K. Bϕ and E.M. Lillehagen, editors, *CAD Systems Framework, Proceedings of IFIP Working Group 5.2*, pp. 241–253. North-Holland, Amsterdam, 1983.

[13] T. Tomiyama and H. Yoshikawa. Extended general design theory. In H. Yoshikawa and E.A. Warman, editors, *Design Theory for CAD, Proceedings of the IFIP Working Group 5.2 Working Conference 1985 (Tokyo)*, pp. 95–130. North-Holland, Amsterdam, 1987.

[14] H. Takeda, T. Tomiyama, and H. Yoshikawa. A logical formalization of design processes for intelligent CAD systems. In H. Yoshikawa and T. Holden, editors, *Intelligent CAD, II*, pp. 325–336. North-Holland, Amsterdam, 1990.

[15] H. Takeda, P. Veerkamp, T. Tomiyama, and H. Yoshikawa. Modeling design processes. *AI Magazine*, Vol. 11, No. 4, pp. 37–48, 1990.

[16] H. Takeda, T. Tomiyama, H. Yoshikawa, and P.J. Veerkamp. Modeling design processes. Technical Report CS-R9059, Centre for Mathematics and Computer Science (CWI), Amsterdam, The Netherlands, October 1990.

[17] T. Tomiyama, T. Kiriyama, H. Takeda, and D. Xue. Metamodel: A key to intelligent CAD systems. *Research in Engineering Design*, Vol. 1, pp. 19–34, 1989.

[18] J. Treur. A logical framework for design processes. In P.J.W. ten Hagen and P.J. Veerkamp, editors, *Intelligent CAD Systems III — Practical Experience and Evaluation*. Springer-Verlag, Berlin, 1991.

[19] T.H. Dietterich and D.G. Ullman. FORLOG: A logic-based architecture for design. Rep. no. 86-30-8, Computer Science Department, Oregon State University, 1987.

[20] R. Coyne. *Logic Models of Design*. Pitman Publishing, London, 1988.

[21] J.J. Finger and M.R. Genesereth. RESIDUE: A deductive approach to design synthesis. Technical report stan-cs-85-1035, Stanford University, 1985.

[22] V. Lifschitz. Computing circumscription. In *Proceedings IJCAI-85*, pp. 121–127, Los Angles, CA, 1985.

[23] J. McCarthy. Circumscription — a form of non-monotonic reasoning. *Artificial Intelligence*, Vol. 13, pp. 27–39, 1980.

[24] R.W. Weyhrauch. Prolegomena to a theory of mechanized formal reasoning. *Artificial Intelligence*, Vol. 13, pp. 133–170, 1980.

[25] F. Landman. *Towards a Theory of Information: the Status of Partial Objects in Semantics*. Foris, Dordrecht, 1986.

[26] F. Veltman. Data semantics. In J.A.G.Groenendijk, T.M.V. Janssen, and M.B.J.Stokhof, editors, *Formal methods in the study of language*, pp. 541–565. Mathematisch Centrum, Amsterdam, 1981.

[27] T. Tomiyama, T. Kiriyama, H. Takeda, and D. Xue. Metamodel: A key to intelligent CAD systems. *Research in Engineering Design*, Vol. 1, pp. 19–34, 1989.

[28] D. Xue, H. Takeda, T. Kiriyama, T. Tomiyama, and H. Yoshikawa. An intelligent integrated interactive CAD — a preliminary report. In *Proceedings of the IFIP 5.2 Working Conference on Intelligent Computer Aided Design*, Ohio, USA, 1991.

[29] C.S. Peirce. *Collected Papers of Charles Sanders Peirce*, volume 5. Harvard University Press, Cambridge, MA, 1935.

[30] H.J. Levesque. A knowledge-level account of abduction. In *Proceedings IJCAI-89*, pp. 1061–1067, Detroit, 1989.

[31] D. Poole. A logical framework for default reasoning. *Artificial Intelligence*, Vol. 36, pp. 27–47, 1988.

[32] P.T. Cox and T. Pietrzykowski. Causes for events: their computation and applications. In *Lecture Notes in Computer Science 230*, pp. 608–621. Springer-Verlag, Berlin, 1986.

[33] K.T. Fann. *Peirce's Theory of Abduction.* Martinus Nijhoff, The Hague, The Netherlands, 1970.

[34] Harry E. Pople, Jr. On the mechanization of abductive logic. In *Proceedings IJCAI-73*, pp. 147–152, Stanford, CA, 1973.

[35] J.R. Hobbs, M. Stickel, P. Martin, and D. Edwards. Interpretation as abduction. In *Proceedings of 26th Annual Meeting of the Association for Computational Linguistics*, pp. 95–103, Buffalo, NY, 1988.

[36] J.W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, Berlin, 1984.

[37] M. Davis. Notes on the mathematics of non-monotonic reasoning. *Aritifical Intelligence*, Vol. 13,, 1980.

[38] J. McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, Vol. 4, pp. 463–502, 1969.

[39] H.T. Ng and R.J. Mooney. On the role of coherence in abductive explanation. In *Proceedings AAAI-90*, pp. 337–342, 1990.

[40] H. Nakagawa and T. Mori. Computable circumscription in logic programming. *Transactions of Information Processing Society of Japan*, Vol. 28, No. 4, pp. 330–338, 1987. (In Japanese).

[41] J. de Kleer. An assumption-based tms. *Artificial Intelligence*, Vol. 28, pp. 127–162, 1986.

[42] P. Morris and R. Nado. Representing actions with an assumption-based truth maintenance system. In *Proceedings AAAI-86*, pp. 13–17. Philadelphia, 1986.

[43] H. Takeda, S. Hamada, T. Tomiyama, and H. Yoshikawa. A cognitive approach of the analysis of design processes. In *Design Theory and Methodology – DTM '90 –*, pp. 153–160. The American Society of Mechanical Engineers (ASME), 1990.

[44] H. Takeda, T. Tomiyama, H. Yoshikawa, and P.J. Veerkamp. Modeling design processes. Technical Report CS-R9059, Centre for Mathematics and Computer Science (CWI), Amsterdam, The Netherlands, October 1990.

[45] B. Veth. An integrated data description language for coding design knowledge. In P.J.W. ten
0Hagen and T. Tomiyama, editors, *Intelligent CAD Systems I*
- *Theoretical and Methodological Aspects*, pp. 295–313. Springer-Verlag, Berlin, 1987.

[46] V. Akman, P.J.W. ten Hagen, and T. Tomiyama. A fundamental and theoretical framework for an intelligent CAD system. *computer-aided design*, Vol. 22, No. 6, pp. 352–367, 1990.

[47] V. Akman, P.J.W. ten Hagen, J. Rogier, and P.J. Veerkamp. Knowledge engineering in design. *Knowledge-Based Systems*, Vol. 1, No. 2, pp. 67–77, 1988.

[48] Tim Smithers, Alistair Conkie, Jim Doheny, Brian Logan, Karl Millington, and Ming Xi Tang. Design as intelligent behaviour: an ai in design research programme. *Artificial Intelligence in Engineering*, Vol. 5, No. 2, pp. 78–109, 1990.

[49] K.D. Forbus. Intelligent computer-aided engineering. *AI Magazine*, Vol. 9, No. 3, pp. 23–36, 1988.

[50] N.P. Suh, A.C. Bell, and D.C. Gossard. On an axiomatic approach to manufacturing and manufacturing system. *Journal of Engineering for Industry*, Vol. 100, pp. 127–130, 1978.

[51] N.H. Suh. *The Principles of Design*. Oxford University Press, New York, Oxford, 1990.

[52] D.G. Ullman, T.G. Dietterich, and L. A. Stauffer. A model of the mechanical design process based on empirical data: a summary. *Artificial Intelligence in Engineering, Design, and Manufacturing*, Vol. 2, No. 1, pp. 33–52, 1988.

[53] D.G. Ullman, T.G. Dietterich, and L.A. Stauffer. A model of the mechanical design process based on empirical data: a summary. In J.S. Gero, editor, *Artificial Intelligence in Engineering Design*, pp. 193–215. Elsevier, Amsterdam, 1988.

[54] D.G. Ullman, L.A. Stauffer, and T.G. Dietterich. Toward expert cad. *Computers in Mechanical Engineering*, Vol. 6, No. 3, pp. 56–70, 1987.

[55] V. Goel and P. Pirolli. Motivating the notion of generic design within information-processing theory: the design problem space. *AI magazine*, Vol. 10, No. 1, pp. 18–47, 1989.

[56] P. Hayes. Naive physics manifesto I: Ontology for liquids. In J. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pp. 71–107. Ablex, Norwood, NJ, 1985.

[57] K.D. Forbus. Qualitative process theory. *Artificial Intelligence*, Vol. 24, pp. 85–168, 1984.

[58] J. de Kleer and J.S. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, Vol. 24, No. 3, pp. 7–83, 1984.

[59] Jon Sticklen, Ahmed Kamel, and W. E. Bond. Integreting quantitative and qualitative computations in a functional framework. *Engeering Applications of Articial Intelligence*, Vol. 4, No. 1, pp. 1–10, 1991.

[60] B. J. Kuipers and Daniel Berleant. Using incomplete quantitative knowledge in qualitative reasoning. In *Proceedings AAAI-88*, pp. 324–329, 1988.

[61] Y. Umeda, H. Takeda, T. Tomiyama, and Y. Yoshikawa. Function, behaviour, and structure. In J.S. Gero, editor, *Applications of Artificial Intelligence in Engineering V*, volume 1, pp. 177–194, Berlin, 1990. Springer-Verlag.

[62] John S. Gero. Design prototype: A knowledge represenation scheme for design. *AI magazine*, Vol. 11, No. 4, pp. 27–36, 1990.

[63] S. Murthy and S. Addanki. Prompt: An innovative design tool. In *Proceedings AAAI-87*, pp. 637–642, 1987.

[64] J. Cagan and .A.M. Agogino. Innovative design of mechanical structures from first principles. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 1, No. 3, pp. 169–189, 1987.

[65] M.G. Dyer, M. Flowers, and J. Hodges. Edison: An engineering design invention system operating naively. *Artificial Intelligence in Engineering*, Vol. 1, No. 1, pp. 36–44, 1986.

[66] K.J. MacCallum. Does intelligent CAD exist? *Artificial Intelligence in Engineering*, Vol. 5, No. 2, pp. 55–64, 1990.

[67] R.V. Guha and D.B. Lenat. Cyc: a midterm report. *AI Magazine*, Vol. 11, No. 3, pp. 32–59, 1990.

[68] D.B. Lenat and R.V. Guha. *Building Large Knowledge-Based Systems.* Addison-Wesley, Reading, MA, 1989.

[69] S. Finger and S.A Safier. Representing and recognizing features in mechanical designs. In J.R. Rinderle, editor, *Design Theory and Methodology – DTM '90 –*, pp. 19–26. ASME, 1990.

[70] J.R. Dixon, J.J. Cunningham, and M.K. Simmons. Research in designing with features. In H. Yoshikawa and D. Gossard, editors, *Intelligent CAD, I*, pp. 137–148. North-Holland, Amsterdam, 1989.

[71] H. Suzuki, H. Ando, and F. Kimura. Synthesizing product shapes with geometric design constraints. In H. Yoshikawa and T. Holden, editors, *Intelligent CAD, II*, pp. 309–324. North-Holland, Amsterdam, 1990.

Appendix A: Related Work to Intelligent CAD

Now there are many and various works about "AI in design"or "AI in engineering",and relationship between design research and AI research is not so simple nowadays. Here I focus on "Intelligent CAD", which has been my research target, and show some related works.

1. **Concept of intelligent CAD** As mentioned above, the concept of intelligent CAD comes from various backgrounds. So we need general discussion of the concept of intelligent CAD. For example,

   - What is needed as the function of intelligent CAD?
   - How should intelligent CAD work?
   - What is needed to realize intelligent CAD?
   - How would intelligent CAD look like?
   - ...

   Veth and his group (CWI, Holland) presented some discussions about intelligent CAD and showed its framework (45) (46) (47). They characterized design as knowledge processing and proposed a framework of "III CAD" in which some AI techniques are embodied.

   Closely related to above, Tomiyama et al. (The Univ. of Tokyo) also showed the concept of intelligent CAD systems (7).

   From AI field, Smithers et al. (48) showed their wide-ranged "AI in design research programme" and an architecture about AI-based design support system.

   Forbus(49) also proposed the concept "intelligent computer-aided engineering" in which qualitative reasoning play an important role.

   State-of-art: Many people seem to agree that intelligent CAD will not be achieved as simple extension of conventional CAD nor application of AI. We need a new concept for intelligent CAD, but it is still a matter in dispute.

2. **Implementation of intelligent CAD** There are few reports about implementation of intelligent CAD system. For example, Xue et al. (The Univ. of Tokyo) presented a prototype of intelligent CAD(28). It is an implementation of some theoretical results

(matamodel theory and logical formalization of design processes) shows that such theories can contribute realization of intelligent CAD. But it is not a system which can be used by actual design activities.

3. **Design theory** Design theory is not a new concept. Some researchers in Europe, especially in Germany have developed some "design theories" (8) (9) (10). But these theories do not seriously concern the nature of design but how to improve actual design. And they are deeply dependent of researchers' experience. Therefore they seem to be useful for designers, but understandable only for designers.

   On the other hand, Yoshikawa (the Univ. of Tokyo) have proposed another "design theory" which is called as "general design theory"(11) (12) (13). It is based on topology in mathematics and roughly speaking, it starts with setting three axioms and defining design as a mapping from attribute space to function space. It derives some important concept such as "metamodel" and "evolutionary design process model" which lead more practical discussion.

   Suh have proposed "axiomatic" design theory(50)(51). Here there are some axioms of design which drive design. But his theory is prescriptive one, that is, his axioms describe not the nature of design but how to obtain a good design, and they are given a priori.

   State-of-art: Although Yoshikawa's theory is not completed and theoretically there are much room to develop it, it provides a good framework and some important concept needed to construct intelligent CAD.

4. **Design process modeling** We can identify two subproblems in the field where design theory is expected to cover. One is the problems about design processes and the other is one about design objects. While the latter is closely related to physical problems, the former is related to cognitive problems.

   Takeda et al. proposed a logical framework for design processes (14) (15)(16). It explains how a design process is formed in terms of logic. He also shows more cognitive nature of design processes using protocol analysis(43).

   Ullman investigates design processes and proposes a design process model called "TEA model" in which ten operators are defined to operate design objects and other information (52) (53) (54).

   Goel and Pirolli views design processes as problem solving and showed tasks and flow of information in design(55).

   State-of-art: Different ideas come from Different fields. But it may be natural because design is so complicated that we can take various ways to view it. Discussions here is related to information processing to some extent so it is expected to be a bridge between design theory and system oriented issues (the concept of intelligent CAD and AI design systems).

5. **Formalization of human thinking in design** Basically it belongs to AI field, but even for AI researchers formalization of thinking in "design" is a challenging issue which can yield new idea in AI fields. There are many topics abut this issue;

- analogy
- learning
- case-based reasoning
- abduction
- hypothesis reasoning
- abstraction
- ...

State-of-art: It is a hot topic for AI field. Many people are interested in this issue and some propose new ideas. Each of them seem to be useful for some parts of design, but we will need some strategy to apply these ideas to CAD.

6. **Integration of various models** Following four issues can be categorized as "the design object issue". In design various models (models of artifacts) are used for design and analysis. Human designers can view a single object from various points of view. That is, they can get some different models from it and use them. But the important point is that they still regard these models as representation of a single object. So they can transfer new information they get in a model to another. It is not easy for computers to deal this "multi-view" concept. We have to re-think about what "modeling" means.

   "Metamodel" derived from General Design theory is one candidate (17). Every modeling is dependent on some physical laws (or physical view). Since metamodel is described in physical phenomenon space, we can get specified descriptions (a model) from matamodel descriptions by specifying physical phenomenon without losing information about relations between the model and the metamodel.

   In AI fields, this problem is conceived as "ontology" on knowledge representation(56). Their concern is how to connect different knowledge representations which have different kinds of ontology.

   There are many results how to connect two models; e.g., translation between FEM model and solid model.

7. **Qualitative reasoning in design** One of the important issues in design object representation is "qualitative" modeling and reasoning. Designers often think their design problems qualitatively, i.e, they represent and analyze their objects qualitatively. But researchers in design study have not been much interested in such qualitative thinking.

   In AI field, "qualitative reasoning" is becoming an important subject. Qualitative reasoning in AI have proposed as a solution of naive physics. It is not physics described with differentials but physics which people have before formalizing with differentials. (We can predict the orbit of ball which is thrown). Roughly speaking, in qualitative reasoning, variables are represented as some discrete values divided by "landmarks". For example, there are three values $+$, 0, $-$ if we use "0" as landmark. There are many different types of qualitative reasoning in AI (for example, (57),(58)).

Qualitative can play a crucial role in intelligent CAD(49). Various works have done to apply qualitative reasoning to design field. Some uses qualitative reasoning to design, but most uses it to analyze behavior of objects.

8. **Integration of qualitative and quantitative reasoning** The problem using qualitative reasoning is that only using qualitative reasoning is not sufficient for design. Cooperation with quantitative reasoning is needed.

   It is not so easy to accomplish this without ad hoc technique. There are some works (59) ((60)), but still there need more investigation.

9. **Function, behavior, and structure** The terms "function", "behavior" and "structure" are commonly used in very domain, but the meaning of these words are not clear. Although many people agree that design starts with "functional" descriptions as required specifications, it is hard to find the definition of function.

   There are few works concerning this issue directly (61). Gero(62) also shows function-behavior-structure model in the context of design.

   But many works are related to this issue.

   There are many descriptions about function in "traditional" design theories or methodologies. But these descriptions are not universal and not applicable to other domains.

   Some people try to define "function" in terms of qualitative reasoning (59) (55). Although behavior is clearly defined in qualitative reasoning, at the current state, their approach is not so successful.

   Many people realizes that function is another important keyword to connect engineering and AI field, and many works will be done from various viewpoints.

10. **AI design systems** There are some works about "innovative" design systems using AI technique. Following research are proposals of design systems using qualitative reasoning.

    PROMPT(63) showed a way how to change models if necessary. it provide multiple models and modification operators, and if it finds it is impossible to deal a problem in a model, change models by using an appropriate modification operators.

    In 1stPRINCIPLE(64), the system analyzes a set of equations which describe the model qualitatively and tries to find optimized values. if it does not match specifications, it divides a region of the object into two parts and gets a new set of equations, and then repeats the optimization process to find the solution.

    Furthermore Dyer et al. propose EDISON system for "design invention system"(65). Here objects are represented by "device topology" and behavior by processes on device topology. The system try to make a structure which satisfies given function by using stored object database.

11. **Knowledge base** So-called expert systems have made and used knowledge base ad hoc. It is the reason that early expert systems met with success and also the reason they are getting trouble to apply more practical or larger problems now. Knowledge base itself is an important problem. We need knowledge base independent

from application programs. There are many works how to organize such knowledge base. One direction is to investigate the nature of knowledge in a domain (for example, MacCallum(66)). Another direction is to make large scale knowledge base (for example, CYC project(67)(68)).

12. Feature, feature-based systems Features derivable from the geometry are called "form feature", for example hole is one of the form features.

Definition of features has a wide variety, because features are defined from various perspectives, e.g., designer, manufacturer, process planner ... Feature is expected to be a bridge between geometric representation and symbolic representation and many researchers are interested in it (for example, (69)(70)). The current main streams are "feature extraction" and "design with feature".

13. Constraint solving Another bridge between geometric representation and symbolic representation is constraint solving. As constraints are described as symbolic representation, it is expected to cooperate with knowledge systems or AI systems. It is also related geometric reasoning (for example, (71)).

Appendix B: An Analytical Solution for the Differential Equation

The formula corresponding this model is a simple as follows;

$nmx'' + dx' + kx = F$

It is easily translated into more general formula as follows;

$$\begin{aligned} x'' + 2bx' + \omega_0(x - x_0) &= 0 \\ \omega_0 &= \sqrt{k/m} \\ b &= d/2m \\ x_0 &= F/m\omega_0 \end{aligned}$$

The solution of its formula is as follows;

$$\begin{aligned} x &= ae^{-bt}\sin(\omega t + \epsilon) + x_0 \\ \omega &= \sqrt{\omega_0^2 - b^2} \end{aligned}$$

With initial conditions $x|_{t=0} = 0$ and $x'|_{t=0} = 0$,

$$\begin{aligned} \epsilon &= \tan^{-1}(\omega/b) \\ a &= -x_0/\sin\epsilon \end{aligned}$$