The American Society of Mechanical Engineers

Reprinted From
DE–Vol. 42, Design Theory and Methodology -DTM '92-
Editors: D. L. Taylor, and L. A. Stauffer
Book No. H00768–1992

A LOGICAL AND COMPUTABLE FRAMEWORK FOR REASONING IN DESIGN

Hideaki Takeda
Division of Product Engineering
Norwegian Institute of Technology
University of Trondheim
Trondheim, Norway

Tetsuo Tomiyama and Hiroyuki Yoshikawa
Department of Precision Machinery Engineering
Faculty of Engineering
University of Tokyo
Tokyo, Japan

ABSTRACT

In this paper we propose a logical design process model that has a good capability as the framework of intelligent CAD systems. This model adopts abduction, deduction, circumscription, and meta-level inference for reasoning, and partial semantics and possible worlds semantics for representation. It can represent the important features of design processes, e.g., coupling of synthesis and analysis is represented by a combination of abduction and deduction, the step-wise refinement by iteration of abduction and deduction and multiworlds, and inconsistency handling by circumscription. Since design process models should be not only well defined, but also computable and capable to explain human design processes, we interpret the cognitive model of design processes by the logical design process model, and also demonstrate its computability by implementing a prototype of the design simulator which is based on this model.

Key words: design theory, artificial intelligence, design process modeling.

1 Introduction

The concept of intelligent CAD systems has drawn the attention of researchers in both the engineering and the AI fields. Although there have been some proposals of design systems that used AI techniques (for example, Dyer et al.[1], Murthy and Addanki[2], Cagan and Agogino[3]) and they are successful in solving some intellectual activities in design, there exists a wide gap between these systems and the concept of intelligent CAD systems, i.e, it is difficult to imagine an intelligent CAD system as an extension or combination of these systems.

One of the most important features that intelligent CAD systems should have is *integration* of subsystems, design models, and design processes[4]. Such integration will be accomplished only by a system that would be constructed with proper design object and design process models. Although there are many studies on modeling of design objects(e.g., geometric modeling), only a few studies have been done on modeling of design processes. This paper, therefore, discusses modeling of design processes for developing intelligent CAD systems.

There are some requirements for design process models which would contribute to intelligent CAD systems. First, the design process model should be formally described, i.e., it should be defined by some mathematical method. Second, since intelligent CAD systems should be interactive with designers, the design process model is expected to be understandable to designers. Furthermore it should be *computable* to have capability to be realized in computer systems[5].

Here we adopt logic as the method to describe design processes, and propose a logical design process model. In Section 2, we discuss the requirements for logical representation of design processes. In Section 3 we review logic as knowledge representation, and determine the logical framework to represent design processes. In Section 4, we propose a logical inference model for design processes that consists of abduction, deduction, circumscription, and meta-level inference. In Section 5, the logical design process model is compared with the cognitive design process model, and discussed as to how the logical model can interpret the cognitive model. In Section 6, we show a prototype system called *design simulator* which is based on the logical design process model in order to examine its computability. In Section 7 we discuss how the model satisfies the requirements listed in Section 2 and compare it with related work. Section 8 concludes the paper.

2 Requirements for Logical Representation of Design

Since the purpose of the paper is to describe design processes in the logical framework, we should clarify what we 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.

Then we can list the following requirements of design processes that we should represent with the logical model. These requirements come from observation and cognitive analysis of design processes[6][7].

1. Bidirectional processes. Design is not a one-way process, because synthesis and analysis are mutually performed in design processes. Except in some types of design such as routine design, the designer can not obtain the design solution straightforward from the given specifications only by designing in the narrow sense, i.e., composing descriptions of the solution, but designers should synthesize objects as well as analyze them in order to examine what their synthesis would bring on. It seems that two types of reasoning are used in design which have opposite directions in reasoning to each other.

2. Step-wise and iterative processes. We can not obtain design solutions at a stretch from the given specifications. The designer details design objects gradually and finally reaches desired descriptions of the solution[7]. Design is accomplished through such *step-wise refinement processes.* Thus, there are many intermediate states for design solutions, and in each state some progress should be performed. It means that reasoning is performed iteratively in step-wise refinement processes.

3. Feasible solutions. In design, there seldom exists any single or definite design solution that fulfills all the required specifications, but some feasible solutions. Each of them is a possible solution, but they may not satisfy all the specifications. Thus, solutions of the design should be feasible and, furthermore, multiple for a single set of specifications.

4. Two types of knowledge. There are two types of knowledge in design — knowledge about how to design (e.g., knowledge about design procedures and design rules), and knowledge about properties and behaviors of objects. The difference lies in purpose of their representation. The purpose of the former is to describe knowledge as designers' action or behavior, while the latter to describe knowledge as what objects are or should be. It is a need that both types of knowledge should be handled.

5. Incompleteness of knowledge description. Since most of knowledge used in design is vague, it is not easy to describe or define it perfectly. We cannot assume that every piece of knowledge has a perfect description. Nevertheless a designer can deal with such *incompleteness of knowledge description* unconsciously and consciously.

6. Uncertainness of available knowledge. Although it is neither natural nor practical to assume that all knowledge is always ready to use, the range of knowledge that can be used in a design is not clear. Except routine design, it is not known what kind of knowledge should be used to accomplish design at the beginning of design. It is determined during design processes.

7. Uncertainness of specifications. In many cases in design, required specifications are not fully determined at the beginning of design. Rather incomplete or vague specifications are gradually added to or detailed during the deign processes.

8. Inconsistency in design. A designer is often confronted with inconsistency in the design processes. But different from inconsistency in logic, it does not mean that the effect of inconsistency is negative in every sense. Resolution or avoidance of inconsistency often makes a new progress of design.

3 The Logical Foundation for Design

In this section, we discuss what framework is appropriate to represent design processes in logic.

3.1 The Logical Foundation for Knowledge Representation

"Deduction" is an inference in which consequences (logical theorems) are derived from given premises (logical axioms) intuitionally. For example, we can say that "q can be deduced from p and $p \rightarrow q$" [1]. It can be represented symbolically as

$$p,\ p \rightarrow q \vdash q.$$

In general, we can represent the relation that formula can be derived from a given set of formula as follows;

$$A \vdash B$$

where A represents a given set of logical formulae (axioms), and B represents a set of logical formulae each of which can be derived from A. In knowledge representation, axioms can be divided into two, i.e., one corresponds to "knowledge" which are common to all situations and the other to "facts" which vary in situations. That is;

$$F \cup K \vdash G \qquad (1)$$

where F, K, and G are sets of logical formulae, and represent *facts*, *knowledge*, and *derived facts* respectively.

On the other hand, C. S. Peirce proposed another kind of inference called *abduction* in which axioms are found which can derive given consequences. Peirce introduced it as an *ampliative* reasoning, while he regarded deduction as *explicative* or *analytical* reasoning. For example, when we have a logical formula q to "explain", and $p \rightarrow q$ is included in axioms, p is a possible solution for abduction. That is, if we assume q and $q \rightarrow p$ as axioms, p is derivable from them. It does not mean that p is a determined solution, but only that p is a candidate which may be the solution, because this process is tracing back from consequents to antecedents, i.e., it commits an error of *the fallacy of affirming the consequent.*

In knowledge representation, we can use the same formula (1) to formalize use of abduction, but should read it differently. That is, F, K and G represent *hypotheses* which are used to derive G, *knowledge*, and the given facts we want to derive respectively. This is the framework of hypothesis reasoning (for example, Poole[8]), and some methods to compute abduction have been proposed.

These two approaches to use logic as knowledge representation are in contrast with each other, and we call the first one as the *deductive framework*, the second one as the *abductive framework.*

3.2 The Logical Framework for Design

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 formula 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[9], and Dietterich and Ullman[10] took this approach, and we also took it in [6].

[1]In this paper we use $\wedge$, $\vee$, $\rightarrow$, and $\neg$ for "and," "or," "implication," and "negation" in logic respectively. $\cup$, $\subseteq$ and $\in$ stand for "union," "subset" and "membership" in set theory respectively.

However, this "design as deduction" approach can not solve most of the problems listed in Section 2.

There are two major problems for this approach, one is about representation of design knowledge (Requirement 4), and the other is about feasible solutions(Requirement 3).

Knowledge in this approach should be knowledge about design procedures or design rules (knowledge about how to design). Since we adopt the "design is deduction" paradigm here, every formula used in deduction should describe something to proceed design. A typical example of this kind of knowledge is, "if there is a specification S_1, then use a design object D_1 as a candidate."

Although it may be useful for routine design because we may collect such kinds of knowledge, it is not appropriate to more flexible and creative design in which knowledge about object properties and behaviors plays an important role. We believe that knowledge about object properties and behaviors is more primary than knowledge about how to design. Designers can manage to design new objects when they have knowledge only about their properties and behaviors but no knowledge about how to design them, while it is difficult for them to design new objects only with knowledge about how to design.

In the deduction framework, although it is possible to obtain multiple solutions, they can not be feasible solutions because they should be satisfied with the given axioms, i.e., the required specifications and knowledge.

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 again design is abduction with knowledge and specifications.

Coyne[11] and RESIDUE system[12] stand for this approach for design formalization.

Then the problems for the deductive model can be solved.

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*, because, as we mentioned above, knowledge about objects is prior to knowledge about how to design.

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 logical formalizations of design.

Although this framework can already interpret some requirements of design process models, many are left to be solved. In the next section, we propose the inference model on this framework that can solve the rest of the requirements.

4 The Logical Inference Model for Design Processes

The inference model we propose is illustrated in Figure 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 includes required specifications.

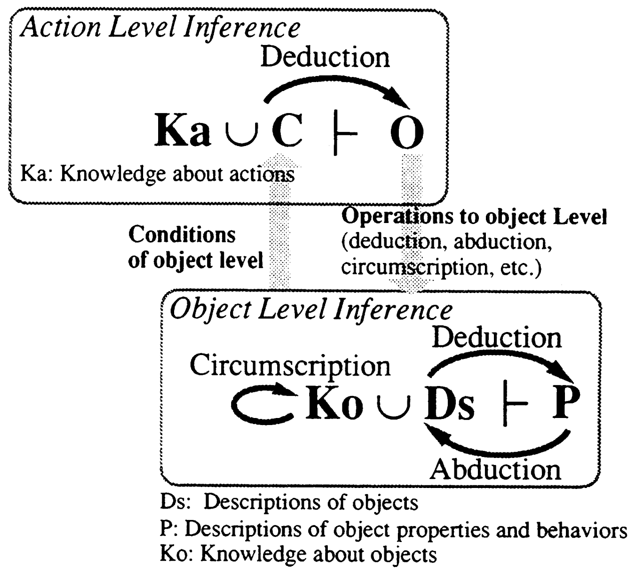

Action Level Inference
Deduction
Ka ∪ C ⊢ O
Ka: Knowledge about actions
Conditions of object level
Operations to object Level (deduction, abduction, circumscription, etc.)
Object Level Inference
Deduction
Circumscription
Ko ∪ Ds ⊢ P
Abduction
Ds: Descriptions of objects
P: Descriptions of object properties and behaviors
Ko: Knowledge about objects

Figure 1: The logical design process model

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.

4.1 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. 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. In the following discussion it is denoted by Ds. 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 indicates 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.

4.2 Circumscription for Resolution of Inconsistency

As mentioned in Section 2, inconsistency 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 it is expected to be used in). 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[2].

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. This process can be accomplished by *circumscription.*

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 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.

For example, consider the following two formula.

$$Rule1: \ spring(x) \rightarrow is\_in\_proportion(x)$$

$$Rule2: \ spring(x) \wedge overload(x) \rightarrow \neg is\_in\_proportion(x).$$

These formulae are inconsistent with each other. Then we can rewrite these formulae as follows;

$$Rule1': \ spring(x) \wedge \neg ab_1(x) \rightarrow is\_in\_proportion(x)$$

$$Rule2': \ spring(x) \wedge overload(x) \wedge \neg ab_2(x) \rightarrow \neg is\_in\_proportion(x).$$

where $ab_i(x)$ represents the implicit description of each formula. One of the results of circumscription of ab_i is

$$ab_1(x) = spring(x) \wedge \neg overload(x), \ ab_2(x) = false.$$

After substitution, we can obtain a modified formula of *Rule*1 as follows (*Rule*2 is unchanged);

$$Rule1'': \ spring(x) \wedge \neg overload(x) \rightarrow is\_in\_proportion(x).$$

It is no more inconsistent with *Rule*1'' and *Rule*2 even if $spring(x)$ and $overload(x)$ are true.

In this context, $\neg overload(x)$ is a newly revealed condition of *Rule 1*, and thus this formula is detailed during this process[3].

By doing abduction with this modified knowledge, 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.

4.3 Meta Level Inference for Actions

We defined the basic 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 meta-level inference architecture. Metal-level inference architectures for reasoning are suggested by many researchers. For example, Weyhrauch[14] 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 derivability 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 is reflected to the object-level system. The reflection is the change of the condition of the object level system 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.

4.4 Multiworlds 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 multiworld mechanisms 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[15][16] is intuitionally 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, partial-order $\sqsupseteq$ can be 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

[2]And the definitions of boundary will be infinite even if such definitions exist[13].

[3]Another problem is whether this modification is desirable, and it depends on priority among formulae with respect to *generality.*

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.

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 different object descriptions from a single solution, two possible worlds are created as descendant 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.

5 Interpretation of the Cognitive Model by the Logical Process Model

In the previous section, we explained the design process model based on the logical framework. Here we compare this model with the cognitive model which was obtained by observation of protocol data[6][7], and interpret the cognitive model with the logical design process model.

We proposed the *design cycle* as a cognitive design process model. We observed protocol data and picked up five basic processes, i.e., *awareness-of-problem, suggestion, development, evaluation, and conclusion* subprocesses. Utterance in the protocol data can be classified into these subprocesses. Since the sequence of these subprocesses in this order appeared repeatedly, we called it a design cycle. Basically a single design cycle solves one problem, but sometimes new problems that should be solved in other design cycles are arisen in the suggestion and evaluation subprocesses.

The suggestion subprocess is a process where the designer tries to find a feasible solution. This means that this subprocess is to obtain Ds from P and Ko and it can be regarded as an abduction process.

On the other hand, both the development and the evaluation subprocesses are regarded as deduction. In these subprocesses, the designer applies his/her knowledge to the solution and obtains what is known about it at the current state. That is, these two processes are to obtain P from Ds and Ko. The difference between those two subprocesses lies in what kind of knowledge is applied. The development subprocess uses knowledge to find out what properties the design object has, while the evaluation subprocess uses knowledge to obtain properties which are used for comparison with other solutions and some evaluation scheme.

While the designer is developing or evaluating the design solution, she or he sometimes encounters a difficulty about the solution and defines a new problem in order to solve the original problem. It is a jump from the development or evaluation subprocess to the awareness-of-problem subprocess. We interpret it as circumscription.

A difficulty is interpreted as inconsistency in logic. As mentioned in Section 4, circumscription solves inconsistency. And furthermore, as the result of circumscription, some pieces of knowledge are modified. It sometimes makes a new problem to be solved. For example, consider the example we mentioned Subsection 4.2 again. Suppose $\{spring(x), overload(x)\}$ is hold now. If $is\_in\_proportion(x)$ is included in the required specifications, this state does not satisfy the specifications be-

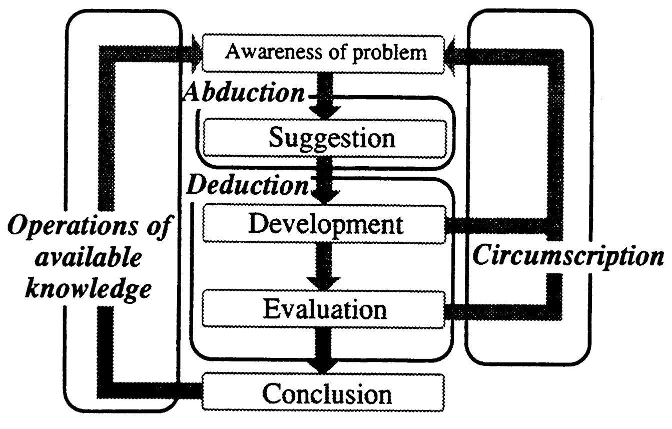


Figure 2: Inferences in the design cycle

cause the modified knowledge can not derive this property from $\{spring(x), overload(x)\}$ any more, whereas $is\_in\_proportion(x)$ is derivable from $\{spring(x), \neg overload(x)\}$. If you want to make $is\_in\_proportion(x)$ true, you have to make $overload(x)$ false. It can be accomplished by assuming it as the new object descriptions or finding another formula to support it. Thus modification of knowledge and successive execution of abduction and deduction can generate a new problem to be solved.

Except backing in to the awareness-of-problem subprocess from the development and evaluation subprocesses, a design cycle ends successfully and a new design cycle follows it. It is provided by changing of available knowledge and determination to the facts to be used in the next abduction. The former can be interpreted by the meta-level inference because it can operate available knowledge on the object level with meta-level knowledge. If knowledge about design procedures are fully provided, we can interpret the latter as the meta-level inference. But in most cases we can not deal it in this framework because it is highly complicated human activity to determine which is better to think next.

Thus we can interpret most parts of the design cycle with the logical design process model (see Figure 2). It means the cognitive model can be re-constructed as the logical model. It is important because, as mentioned Section 1, our purpose is to propose a logical model which is not only well defined but also well related to human design processes.

6 The Design Simulator

We implemented a prototype of the *design simulator* that realizes the inferences discussed in Section 4. We call it the 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 The Architecture

The design simulator consists of three main parts; i.e., the action level inference system, the object level inference system, and the multiworld management system (see Figure 3). The object level inference furthermore consists of workspace Ds, P, and Ko, and three inference subsystems, i.e., deduction, abduction and circumscription subsystems.

The Action-level. The action-level inference system works as

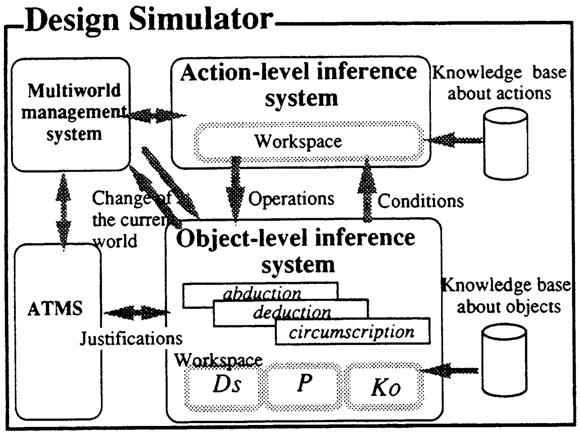

Design Simulator
Multiworld management system
Action-level inference system
Knowledge base about actions
Workspace
Change of the current world
Operations
Conditions
Object-level inference system
abduction
deduction
circumscription
Knowledge base about objects
ATMS
Justifications
Workspace
Ds
P
Ko

Figure 3: The architecture of the design simulator

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 reasoning according to the condition of the object level. Results of this deduction is a single or a sequence of operations on the object level which are described in Section 4.

The Object-level. In the object level, abduction, deduction, and circumscription are performed that change the current state of Ds, P, and Ko. Workspace Ds contains the current descriptions of objects, P the current descriptions of object properties and behaviors, and Ko the current available knowledge.

The form of the content of Ko is Horn clause, while those of P and Ds are literals, that is, atomic formula or its negation. The 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[12][8]. We can infer whether a certain formula is derivable from a given set of formulae by using it. As the result, we can obtain a set of formulae selected from the given set of formulae which can derive the queried formula. 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). To eliminate these solutions, we collect only maximal solutions with respect to the relation *deeper*. 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[17] that is an algorithm for computing circumscription[18] on clausal forms. Their basic idea is to use the technique of program transformation such as *unfolding* when eliminating variable predicates and abnormal predicates.

Multiworld Management System. The multiworld management system keeps design states and their relations as data dependency cooperated with ATMS system[19]. The idea of multiworlds with ATMS is shown by Morris and Nado[20]. Here we can operate the multiworlds by logical formulae with modal operators $\Box$ (necessity) and $\Diamond$ (possibility).

(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 should use simpler mechanism.
...

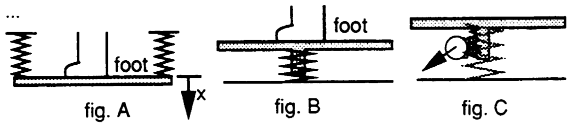


Figure 4: The examples of protocol data

6.2 Examples

We demonstrate how this system works with a set of formulae which we extracted from the protocol data[7]. Figure 4 shows the original protocol data. This is the first part of the design of which the task is "to design a scale". Figure 5 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 5(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 5(b)). This state corresponds to protocol 8 in Figure 4. Then, the formula which represent protocol 11 and 12 are introduced into Ko. This makes an inconsistency situation (World 6, Figure 5(c)). It is because protocol 11 and 12 are contradictory sentences to protocol 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 5(d)). Finally we obtain Ds and P shown as Figure 5(d). Figure 6 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.

7 Discussion and Related Work

In Chapter 2, we discussed what should be represented in the design process model, and we proposed the logical design process model to satisfy these requirements.

First, combination of synthesis and analysis is represented by coupling of abduction and deduction, i.e., we realized the bidirectional processes (Requirement 1).

And reasoning in the step-wise refinement is realized by iteration of abduction and deduction, and the state changing is formalized by worlds and their relations under multiworld possi-

Ds	P	
displacement(5mm)	displacement(5mm)	
weight(100kg)	weight(100kg)	
translate(sc1 100kg 5mm)	translate(sc1 100kg 5mm)	
scale(sc1)	scale(sc1)	

(a) World 1

Ds	P	
displacement(5mm)	displacement(5mm)	scale(sc1)
weight(100kg)	weight(100kg)	**support(sc1 100kg)**
translate(sc1 100kg 5mm)	translate(sc1 100kg 5mm)	
indicator(i8)	**indicator(i8)**	**can-measure(sc1 100kg)**
has(sc1 i8)	**has(sc1 i8)**	
spring(sp9)	**spring(sp9)**	
has(sc1 sp9)	**has(sc1 sp9)**	
push(sc1 sp9)	**push(sc1 sp9)**	

(b) World 5

Ds	P	
displacement(5mm)	displacement(5mm)	scale(sc1)
weight(100kg)	weight(100kg)	support(sc1 100kg)
indicator(i8)	indicator(i8)	can-measure(sc1 100kg)
has(sc1 i8)	has(sc1 i8)	***translate(sc1 100kg 5mm)***
spring(sp9)	spring(sp9)	***¬translate(sc1 100kg 5mm)***
has(sc1 sp9)	has(sc1 sp9)	
push(sc1 sp9)	push(sc1 sp9)	
is-in-prop(sc1 100kg 5mm)	is-in-prop(sc1 100kg 5mm)	

(c) World 6

Ds	P	
displacement(5mm)	displacement(5mm)	scale(sc1)
weight(100kg)	weight(100kg)	support(sc1 100kg)
indicator(i8)	indicator(i8)	can-measure(sc1 100kg)
has(sc1 i8)	has(sc1 i8)	translate(sc1 100kg 5mm)
spring(sp9)	spring(sp9)	¬ **is-upward(i8)**
has(sc1 sp9)	has(sc1 sp9)	
push(sc1 sp9)	push(sc1 sp9)	
is-in-prop(sc1 100kg 5mm)	is-in-prop(sc1 100kg 5mm)	
many-gears(mg12)	**many-gears(mg12)**	
has(i8 mg12)	**has(i8 mg12)**	

(d) World 9

Ds	P	
displacement(5mm)	displacement(5mm)	scale(sc1)
weight(100kg)	weight(100kg)	support(sc1 100kg)
indicator(i13)	indicator(i13)	can-measure(sc1 100kg)
has(sc1 i13)	has(sc1 i13)	translate(sc1 100kg 5mm)
spring(sp18)	spring(sp18)	is-upward(i13)
has(sc1 sp18)	has(sc1 sp18)	¬ is-easy-mechanism(i13)
push(sc1 sp18)	push(sc1 sp18)	is-easy-to-see(i13)
many-gears(i17)	many-gears(i17)	is-in-prop(sc1 100kg 5mm)
has(i13 i17)	has(i13 i17)	
helical-gear(hg16)	helical-gear(hg16)	
has(i13 hg16)	has(i13 hg16)	
rack&pinion(rp20)	rack&pinion(rp20)	
has(sc1 rp20)	has(sc1 rp20)	

(e) World18

Figure 5: Changing of Ds and P

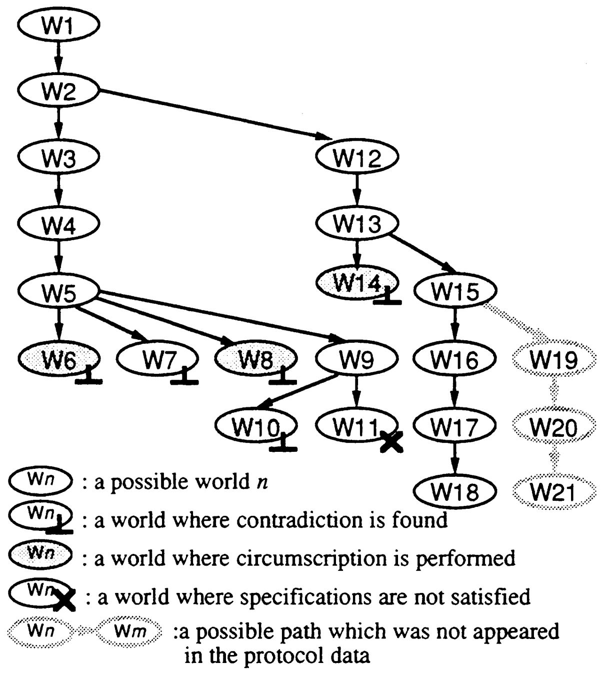
ble semantics (model logic) and partial semantics (Requirement 2).

Adoption of abduction as the basic process allows us to represent feasible solutions (Requirement 3).

The two level inference not only can represent two types of design knowledge, but also can use them properly. The object-level inference and the meta-level inference work complementarily so that lack of object-level knowledge can be covered by meta-level knowledge, and coarseness of meta-level knowledge can be interpolated by object-level knowledge (Requirement 4).

Execution of circumscription for resolving inconsistency assures us that we should not care about incompleteness of knowledge descriptions unless we meet inconsistency of knowledge, and furthermore incompleteness of each piece of knowledge can be reduced by finding new additional descriptions for it (Requirement 5).

As mentioned above, circumscription is used to solve inconsistency by modifying knowledge. Modification of knowledge can cause revision of the current solution so that we can obtain a new solution after resolution of inconsistency. Thus resolution of inconsistency can be a trigger to proceed design (Requirement 8).



W1
W2
W3
W4
W5
W6
W7
W8
W9
W10
W11
W12
W13
W14
W15
W16
W17
W18
W19
W20
W21
Wn : a possible world n
Wn : a world where contradiction is found
Wn : a world where circumscription is performed
Wn : a world where specifications are not satisfied
Wn Wm :a possible path which was not appeared in the protocol data

Figure 6: The generated worlds

Uncertainness of available knowledge and specifications can be treated with the metal-level inference. But it is not the complete solution for this problem. We provided the framework to deal them but did not provide any explanation why and when these uncertainness should be reduced (Requirement 6 and 7).

Although there are still some problems to be solved, the model we proposed in this paper has enough capability as the design process model. We are now proposing a framework of intelligent CAD systems using this model[21].

There are some related studies. Dietteirch et al. proposed FORLOG as a logical representation of design[10]. Although we agree with them in some points (for example, assertion-based approach but not term-based one), it has various drawbacks because it is based on the *deductive framework* in our term, in other words "design is deduction" paradigm, for example, it can deal only knowledge about how to design.

On the other hand, RESIDUE[12] which is based on the *abductive framework* can only deal knowledge about objects. It is not sufficient except in the domain where complete descriptions of objects can be obtained such as electric circuit design which they used as examples.

Kannapan and Marshek proposed a use of logic as realization of mechanical design[22]. They allow four types of operations for logical formula, but their meaning and relation is not clear as logic and they are used only for a representation method for design methodology.

Treur proposed the use of partial semantics in the logical framework for design[9].

8 Conclusions

In this paper we proposed a design process model based on a logical framework. This model adopts abduction, deduction, circumscription, and meta-level inference for reasoning, and partial semantics and possible worlds semantics for representation.

As mentioned in Section 1, the design process model should be not only well defined, but also computable and capable to explain human design processes. We interpreted the cognitive model of design processes by the logical design process model, and showed that most of the cognitive model can be realized in the logical process model. And we also demonstrated its computability by implementing a prototype of the design simulator based on this model.

Thus the logical design process model has a good capability as the framework of intelligent CAD systems.

Acknowledgments

We would like to thank Professor Øybind Bjørke at the Norwegian Institute of Technology for his helpful advice and support. We are also grateful to Kazuto Hayashi, Juro Baba, and Masaharu Yoshioka at the University of Tokyo for discussions and implementations of the design simulator.

References

[1] 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.

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

[3] 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.

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

[5] J.R. Dixon. On research methodology towards a scientific theory of engineering design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, Vol. 1, No. 3, pp. 145–157, 1987.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

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

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

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

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

[15] 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.

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

[17] 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).

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

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

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

[21] Deyi Xue, Hideaki Takeda, Takashi Kiriyama, Tetsuo Tomiyama, and Hiroyuki 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.

[22] Srikanth M. Kannapan and Kurt M. Marshek. Design synthetic reasoning: A methodology for mechanical design. *Research in Engineering Design*, Vol. 2, pp. 221–238, 1991.