Proofs of Nonconsequence as Abstract Design in Hyperproof

Ichiro Nagasaka
Faculty of Letter,
Kobe University
1-1 Rokkodai, Nada,
Kobe 657-8501, Japan
nagasaka@kobe-u.ac.jp

Yuzuru Kakuda
Dept. of Computer & Systems Eng.
Kobe University
1-1 Rokkodai, Nada,
Kobe 657-8501, Japan
kakuda@kobe-u.ac.jp

Abstract
Design process is a series of activities in which designers try to find or invent entities that satisfy specifications, the specification usually they take as given. The process could be seen as a kind of proof of nonconsequence since the entities must satisfy the specification since they usually invent entities and check if the entities satisfy given specifications, rather than they deduce the entities from the specifications. In this paper, we argue that this similarities between the proof of nonconsequence and the design process are essential, and they makes it possible to formulate the design process in an abstract way. Addition to the above argument, we formulate the logical relation between heterogeneous specifications as *heterogeneous logic* based on a mathematical theory of design called *Abstract Design Theory* (ADT), and discuss about such logic in a reasoning system called *Hyperproof.*
Keywords: Abstract design theory, Hyperproof, proof of nonconsequence, heterogeneous logic, information path.

1 Introduction

When mathematicians prove a theorem, they are showing that a particular claim follows from certain accepted information, the information they take as *given.* This kind of proof is what we call a proof of *consequence*, a proof that a particular piece of information must be true if the given information is correct. A very different, but equally important kind of proof is a proof of *nonconsequence*, where mathematicians show that it would be possible for a claim in given information not to be true, even if the other information *is* true. To show this, it is enough to give a single sentence, i.e. counterexample, that is not a consistent with the claim and consistent with the other information. However, it is not always easy to find such sentence since it does not follow from the claim in question. Mathematicians have to invent such sentence somehow or other in a non-deductive way.

Design process is a series of activities in which designers try to find or invent entities that satisfy specifications, the specification usually they take as also *given.*

International Journal of Computing Anticipatory Systems, Volume 11, 2002
Edited by D. M. Dubois, CHAOS, Liège, Belgium, ISSN 1373-5411 ISBN 2-9600262-5-X

The process could be seen as a kind of proof of consequence since the entities must satisfy the specification. However, if we closely look at the design process, especially in the case of creative design, they usually invent entities and check if the entities satisfy given specifications, rather than they deduce the entities from the specifications. Here, we could find important common activities between the process of design and proofs of nonconsequence, that is, they try to *invent* entities or sentences that satisfy their goals, i.e. the entities for designers and the counterexamples for mathematicians, in non-deductive way. In this paper, we argue that this common activities are essential, and they makes it possible to formulate the design process in an abstract way. Following these remarks, we investigate these common activities based on a mathematical theory of design called Abstract Design Theory (ADT)[1].

Addition to the above argument, we focus on the heterogeneous nature of specification. The specification usually have a multiple form of representation, such as drawings and sentential specifications, and they are often closely interrelated to give information that should be satisfied by the entities. To be able to find the entities, there must be a logical relation between these specifications with different representation. Thus, in this paper, we formulate the logical relation between heterogeneous specifications as *heterogeneous logic* by using a reasoning system called *Hyperproof*[4]. Hyperproof is a system for constructing proofs where the information is given in two different forms: graphical and sentential. With these information, the system allows users to solve simple reasoning problem. In Hyperproof, to construct proofs of nonconsequence, users are asked to invent a situation, — graphical information — that follows from given information but it neglects claim in question. We will focus on this process of proofs in Hyperproof, explain it in the framework of ADT, and finally, show that the creative design process is essentially the same as the process of proofs of nonconsequence.

2 Abstract Design Theory

2.1 Historical Background

In 1981, Yoshikawa proposed an axiomatic theory of design called *General Design Theory* (GDT)[5], where possibility of design is discussed in terms of topological spaces defined on a set of abstract concepts called a set of entity concepts. In the theory, a concept of function and attribute are defined as subsets of the set of the entity concepts, and the possibility of the design is discussed in terms of the continuity of the identity map from a set of the attribute concepts to a set of the function concepts.

On the other hand, Barwise and Seligman proposed a theory of information flow called *Channel Theory* in 1997[6]. As briefly introduced in later section, it is a mathematical theory which is intended to formulate flow of information in distributed systems with the notions of information channel and local logic.

Inspired by the idea of the GDT, the ADT was proposed to formulate design in more mathematically aciculate way on the basis of Channel Theory. While design was formulated in terms of the map from the attribute concepts to function concepts in the GDT, in the ADT, design is formulated in terms of an existence of information channel between the abstract concepts and the physical world. In other words, when design is possible, there is an information channel between our conceptual space and the world around us. Mathematically, this can be viewed as an application of Channel Theory to a theory of design. The development of ADT is still in progress and a further work has been done especially in the notion of information flow(cf. [2]).

2.2 Channel Theory

2.2.1 Classification

Since the notion of classification and infomorphisms are fundamental to the notion of information channel in Channel Theory, it is also true in the ADT. Most parts of following definitions are from [6].

Definition 2.1. A *classification* $\boldsymbol{A} = \langle \mathrm{tok}(\boldsymbol{A}), \mathrm{typ}(\boldsymbol{A}), \models_{\boldsymbol{A}} \rangle$ consists of a set $\mathrm{tok}(\boldsymbol{A})$ of objects to be classified, called *tokens* of $\boldsymbol{A}$, a set $\mathrm{typ}(\boldsymbol{A})$ of objects used to classify the tokens, called the *types* of $\boldsymbol{A}$, and a binary relation $\models_{\boldsymbol{A}}$ between $\mathrm{tok}(\boldsymbol{A})$ and $\mathrm{typ}(\boldsymbol{A})$.

A classification is depicted by means of a diagram as follows.

$$\begin{array}{c} \mathrm{typ}(\boldsymbol{A}) \\ \Big| \models_{\boldsymbol{A}} \\ \mathrm{tok}(\boldsymbol{A}) \end{array}$$

The binary relation $\models_{\boldsymbol{A}}$ tells that which tokens of $\boldsymbol{A}$ is classified as being of which types of $\boldsymbol{A}$. These tokens and types are not restricted to the mathematical objects. They can be any theoretical vocabularies such as terminologies used in mechanical engineering or cognitive science.

Next, when we have two classifications at hand, the information flow between them can be modeled by the notion called *infomorphism.*

Definition 2.2. If $\boldsymbol{A} = \langle \mathrm{tok}(\boldsymbol{A}), \mathrm{typ}(\boldsymbol{A}), \models_{\boldsymbol{A}} \rangle$ and $\boldsymbol{C} = \langle \mathrm{tok}(\boldsymbol{C}), \mathrm{typ}(\boldsymbol{C}), \models_{\boldsymbol{C}} \rangle$ are classifications then an *infomorphism* is a pair $f = \langle f^{\wedge}, f^{\vee} \rangle$ of functions satisfying the analogous biconditional:

$$f^{\vee}(c) \models_{\boldsymbol{A}} \alpha \Longleftrightarrow c \models_{\boldsymbol{C}} f^{\wedge}(\alpha)$$

for all tokens c of $\boldsymbol{C}$ and all types α of $\boldsymbol{A}$.

With two classification diagrams, infomorphisms can be depicted as follows. The notion of an infomorphism $f : \boldsymbol{A} \rightleftarrows \boldsymbol{C}$ gives a mathematical model of the *whole-part relationship*, i.e., a whole modeled by a classification $\boldsymbol{C}$ and that of a part modeled by a classification $\boldsymbol{A}$.

$$\begin{array}{ccc} \mathrm{typ}(\boldsymbol{A}) & \xrightarrow{f^{\wedge}} & \mathrm{typ}(\boldsymbol{B}) \\ \Big| \models_{\boldsymbol{A}} & & \Big| \models_{\boldsymbol{B}} \\ \mathrm{tok}(\boldsymbol{A}) & \xleftarrow{f^{\vee}} & \mathrm{tok}(\boldsymbol{B}) \end{array}$$

2.2.2 Theory

In mathematical logic, theory is considered to be a set of sentences with some kind of notion of entailment between theories and sentence. Here, this notion is generalized to work with more general setting, such as a certain scientific theory. In this section, a definition of the notion of theory and related topics those are needed to be defined for the ADT are introduced.

Given a set Σ, a *sequent* of Σ is a pair $\langle \Gamma, \Delta \rangle$ of subsets of Σ. A sequent $\langle \Gamma, \Delta \rangle$ is a *partition* of a set Σ' if $\Gamma \cup \Delta = \Sigma'$ and $\Gamma \cap \Delta = \emptyset$. We say that $\langle \Gamma', \Delta' \rangle$ is an *extension* of the sequent $\langle \Gamma, \Delta \rangle$ if $\Gamma \subseteq \Gamma'$ and $\Delta \subseteq \Delta'$ and write $\langle \Gamma, \Delta \rangle \leq \langle \Gamma', \Delta' \rangle$.

Definition 2.3. A *theory* is a pair $T = \langle \mathrm{typ}(T), \vdash_T \rangle$ of a set $\mathrm{typ}(T)$ and a binary relation $\vdash_T$ on subset of $\mathrm{typ}(T)$. A sequent $\langle \Gamma, \Delta \rangle$ of subset of $\mathrm{typ}(T)$ is said to be *constraint* of T if $\Gamma \vdash_T \Delta$, and *T-consistent* if $\Gamma \nvdash_T \Delta$. T is *inconsistent* if there is no T-consistent sequent in $\vdash_T$.

A theory T is *regular* iff T satisfies the following for all types α and all sets $\Gamma, \Gamma', \Delta, \Delta'$ of types:

1. **Weakening:** if $\Gamma \vdash_T \Delta$ then $\Gamma \cup \Gamma' \vdash_T \Delta \cup \Delta'$,

2. **Partition:** if $\Gamma \nvdash_T \Delta$ then there is a partition $\langle \Gamma', \Delta' \rangle$ with $\langle \Gamma, \Delta \rangle \leq \langle \Gamma', \Delta' \rangle$ such that $\Gamma' \nvdash_T \Delta'$.

Definition 2.4. Let T_1 and T_2 be regular theories. A *regular theory interpretation* $f : T_1 \to T_2$ is a function from $\mathrm{typ}(T_1)$ to $\mathrm{typ}(T_2)$ such that for each $\Gamma, \Delta \subseteq \mathrm{typ}(T_1)$

$$\Gamma \vdash_{T_1} \Delta \Longrightarrow f[\Gamma] \vdash_{T_2} f[\Delta].$$

Let $\boldsymbol{A}$ be a classification and let $\langle \Gamma, \Delta \rangle$ be a sequent of types of $\boldsymbol{A}$. A token a of $\boldsymbol{A}$ *satisfies* $\langle \Gamma, \Delta \rangle$ provided that if a is of type α for every $\alpha \in \Gamma$ then a is of type α for some $\alpha \in \Delta$, that is, for $a \in \mathrm{tok}(\boldsymbol{A})$

$$\forall \alpha \in \Gamma (a \models_{\boldsymbol{A}} \alpha) \Longrightarrow \exists \alpha \in \Delta (a \models_{\boldsymbol{A}} \alpha).$$

For a set Σ of functions, a sequent $\langle\Gamma, \Delta\rangle$ of Σ with $\Gamma \cap \Delta = \emptyset$ will be called a *specification* on Σ, in the sense that $\langle\Gamma, \Delta\rangle$ specifies an object having any function in Γ and no function in Δ. It is said to be a *complete specification* if $\Gamma \cup \Delta = \Sigma$.

Let $\boldsymbol{A}$ be a classification such that $\mathrm{typ}(\boldsymbol{A}) = \Sigma$. A specification $\langle\Gamma, \Delta\rangle$ on Σ is *realized* by a token a of $\boldsymbol{A}$ if $a \models_{\boldsymbol{A}} \alpha$ for every $\alpha \in \Gamma$ and $a \not\models_{\boldsymbol{A}} \alpha$ for every $\alpha \in \Delta$. It is also said a is *counterexample* for $\langle\Gamma, \Delta\rangle$ if a realizes $\langle\Gamma, \Delta\rangle$.

Definition 2.5. Given a classification $\boldsymbol{A}$, the theory $\mathrm{Th}(\boldsymbol{A})$ *generated* by $\boldsymbol{A}$ is the theory whose

1. types are the types of $\boldsymbol{A}$, i.e., $\mathrm{typ}(\boldsymbol{A})$, and
2. constraints are the set of sequent satisfied by every token in $\boldsymbol{A}$, i.e., $\vdash_{\mathrm{Th}(\boldsymbol{A})}$ satisfy the followings for all sets $\Gamma, \Delta \subseteq \mathrm{typ}(\boldsymbol{A})$:

$$\Gamma \vdash_{\mathrm{Th}(\boldsymbol{A})} \Delta \iff \forall a \in \mathrm{tok}(\boldsymbol{A})(\forall \alpha \in \Gamma(a \models_{\boldsymbol{A}} \alpha) \to \exists \alpha \in \Delta(a \models_{\boldsymbol{A}} \alpha)).$$

Definition 2.6.

1. Given a regular theory T, the classification $\mathrm{Cla}(T)$ *generated* by T is the classification whose
   (a) tokens are the T-consistent partitions $\langle\Gamma, \Delta\rangle$ of $\mathrm{typ}(T)$,
   (b) types are the types of T, such that
   (c) $\langle\Gamma, \Delta\rangle \models_{\mathrm{Cla}(T)} \alpha$ iff $\alpha \in \Gamma$.
2. Given an interpretation $f : T \to T'$, we define an infomorphism

$$\mathrm{Cla}(f) : \mathrm{Cla}(T) \rightleftarrows \mathrm{Cla}(T')$$

   by
   (a) $\mathrm{Cla}(f)\hat{}(\alpha) = f(\alpha)$ for $\alpha \in \mathrm{typ}(T)$, and
   (b) $\mathrm{Cla}(f)\check{}(\langle\Gamma, \Delta\rangle) = \langle f^{-1}[\Gamma], f^{-1}[\Delta]\rangle$ for any token $\langle\Gamma, \Delta\rangle$ of $\mathrm{Cla}(T')$.

2.3 Functional Schemes

Now, we shall provide a mathematical framework for design, called *functional scheme*. First, we give a notion called a *information path*, then give a definition of functional scheme based on it. We may note, in passing, that since ADT is in progress, the notion of information path in this paper is still in premature stage. We have been working on the general form of the definition of the information path[2][3][1].

[1]For detailed arguments for this development, please refer to http://kurt.cla.kobe-u.ac.jp/˜kikuchi/adt.html.

Definition 2.7. A *scheme of information flow* is 3-tuple $\mathfrak{S} = \langle \boldsymbol{A}_{\mathfrak{S}}, \boldsymbol{B}_{\mathfrak{S}}, R_{\mathfrak{S}} \rangle$ where $\boldsymbol{A}_{\mathfrak{S}}$ and $\boldsymbol{B}_{\mathfrak{S}}$ are classifications, and $R_{\mathfrak{S}}$ is a binary relation between typ($\boldsymbol{A}$) and typ($\boldsymbol{B}$). We say that there is an *information path from a to be cloven by* $R_{\mathfrak{S}}$ if the condition

$$\forall \alpha \in \mathrm{typ}(\boldsymbol{A}_{\mathfrak{S}})(a \models_{\boldsymbol{A}_{\mathfrak{S}}} \alpha \Longleftrightarrow \exists \beta \in \mathrm{typ}(\boldsymbol{B}_{\mathfrak{S}})(\alpha R_{\mathfrak{S}} \beta \wedge b \models_{\boldsymbol{B}_{\mathfrak{S}}} \beta))$$

holds. A binary relation $\breve{R}_{\mathfrak{S}}$ between tok($\boldsymbol{B}_{\mathfrak{S}}$) and tok($\boldsymbol{A}_{\mathfrak{S}}$) is defined so that $b\breve{R}_{\mathfrak{S}}a$ iff there exists an information path from a to b cloven by $R_{\mathfrak{S}}$.

$$\begin{array}{ccc}
\mathrm{typ}(\boldsymbol{A}_{\mathfrak{S}}) & \xrightarrow{R_{\mathfrak{S}}} & \mathrm{typ}(\boldsymbol{B}_{\mathfrak{S}}) \\
\Big| \models_{\boldsymbol{A}_{\mathfrak{S}}} & & \Big| \models_{\boldsymbol{B}_{\mathfrak{S}}} \\
\mathrm{tok}(\boldsymbol{A}_{\mathfrak{S}}) & \xleftarrow{\breve{R}_{\mathfrak{S}}} & \mathrm{tok}(\boldsymbol{B}_{\mathfrak{S}})
\end{array}$$

With the notion laid above, we come to the central notion of the ADT, called a *functional scheme.* Let T be a theory of requirements, that represents our mental world, and $\boldsymbol{B}$ is a classification given by classifying entities in physical world by their behavior, that is, let tok($\boldsymbol{B}$) be a set of the entities and typ($\boldsymbol{B}$) be a set of behaviors and $b \models_{\boldsymbol{B}} \beta$ is defined by "an entities b has a behavior β". Mathematically speaking, it is no more than a notion between a regular theory T and a classification $\boldsymbol{B}$.

Definition 2.8. A *functional scheme* is 3-tuple $\mathfrak{S} = \langle \mathrm{Cla}(T_{\mathfrak{S}}), \boldsymbol{B}_{\mathfrak{S}}, R_{\mathfrak{S}} \rangle$ for which Cla(T) is a classification generated by a regular theory $T_{\mathfrak{S}}$ and $\boldsymbol{B}_{\mathfrak{S}}$ is a classification. $T_{\mathfrak{S}}$ and $\boldsymbol{B}_{\mathfrak{S}}$ are called the *theory of requirement* and the *functional classification* of $\mathfrak{S}$, respectively.

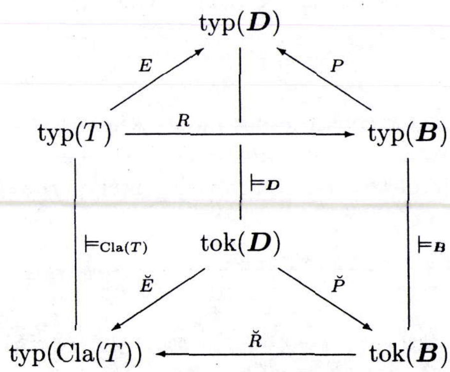
A functional scheme is depicted by means of a diagram as follows.

$$\begin{array}{ccc}
\mathrm{typ}(\mathrm{Cla}(T_{\mathfrak{S}})) & \xrightarrow{R_{\mathfrak{S}}} & \mathrm{typ}(\boldsymbol{B}_{\mathfrak{S}}) \\
\Big| \models_{\mathrm{Cla}(T_{\mathfrak{S}})} & & \Big| \models_{\boldsymbol{B}_{\mathfrak{S}}} \\
\mathrm{tok}(\mathrm{Cla}(T_{\mathfrak{S}})) & \xleftarrow{\breve{R}_{\mathfrak{S}}} & \mathrm{tok}(\boldsymbol{B}_{\mathfrak{S}})
\end{array}$$

2.4 Medium Classification

It is often not enough to realize an entity, even if we have a classification of requirements Cla($T_{\mathfrak{S}}$) and a classification of physical entities $\boldsymbol{B}_{\mathfrak{S}}$, since we sometime do not know how to obtain a relation $R_{\mathfrak{S}}$ which make it possible to classify entities by requirements through their behaviors. Therefore, we introduce a classification called a *medium classification* that connects two classifications, i.e., Cla($T_{\mathfrak{S}}$) and $\boldsymbol{B}_{\mathfrak{S}}$. A medium classification can be seen as a kind of *drawings* in design activity, since they are supposed to depict a way to realize entities indicated by requirements.

Let T be a regular theory and $\boldsymbol{D}, \boldsymbol{B}$ be classifications. Let E be a binary relation between typ(T) and typ($\boldsymbol{D}$), and let P be a binary relation between typ($\boldsymbol{B}$) and typ($\boldsymbol{D}$). When $\mathfrak{E} = \langle \text{Cla}(T), \boldsymbol{D}, E \rangle$ and $\mathfrak{P} = \langle \boldsymbol{B}, \boldsymbol{D}, P \rangle$ are schemes of information, we call $\boldsymbol{D}$ a *medium classification* between Cla(T) and typ($\boldsymbol{B}$).



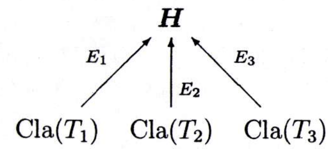
3 Heterogeneous Logic

A specification usually has multiple forms of representations, such as drawings and sentential specifications, and they are often closely interrelated to give information that should be satisfied by the entities. To be able to find the entities, there must be a logical relation between these specifications with different representations. Thus, in this section, we formulate the logical relation between heterogeneous specifications as *heterogeneous logic* by using a reasoning system called *Hyperproof*[4] as an example.

3.1 Background

Heterogeneous logic is a heterogeneous reasoning system where inference proceeds from information represented in more than one form. In mathematics, representations other than sentential representations, especially visual one, still remain second-class citizens, and at best, they have been regarded as teaching tools or heuristics for mathematical discoveries. In this context, Barwise emphasized in [7] that efficient reasoning is inescapably heterogeneous (or "hybrid") in nature, and gave some examples such as Venn diagrams and proof of Pythagorean theorem with diagrams where visual information can be integral to the reasoning itself. As mentioned earlier, design activities are certainly among them.

Mathematically, these notions are expressed by relations between a classification (*core*) and multiple theories. Suppose there are several systems of concepts modeled by means of theories T_i for i in some index set I, a *heterogeneous logic* is a classification $\boldsymbol{H}$ with an binary relation E_i between the element of typ(Cla(T_i)) and typ($\boldsymbol{H}$), one for each $i \in I$.



3.2 Hyperproof

Hyperproof is a system for constructing proofs where the information is given in two different forms: graphical and sentential. With these information, the system allows users to solve simple reasoning problem. In Hyperproof, to construct proofs

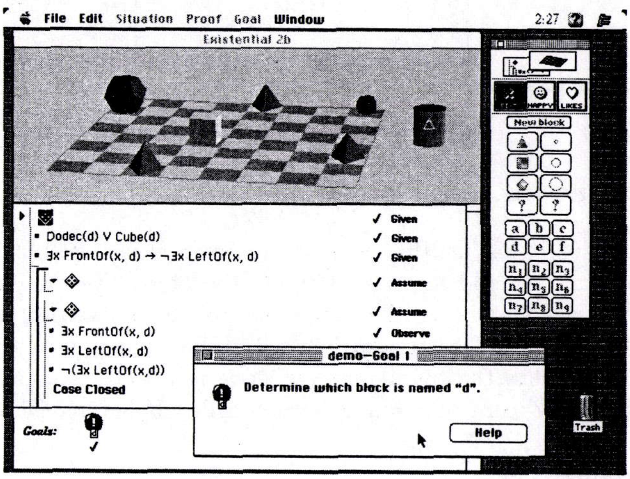


Fig. 1: Screen image of Hyperproof

of nonconsequence, users are asked to invent a situation, i.e., graphical information, that follows from given information but it neglects claim in question. We will focus on this type of proofs in Hyperproof and explain it in the framework of ADT.

3.2.1 Proof System in Hyperproof

In Hyperproof, a proof typically begins with some initial information in the form of a diagram depicting a blocks world and some sentences expressed in the language of first-order logic. The diagrams in Hyperproof are more or less information about the blocks world such as depicted in Fig. 1. From this initial information, users are asked to demonstrate that requested characteristics hold with the given information. The proof system in Hyperproof is an extension of the *Fitch-style* deductive system(Fig.2).

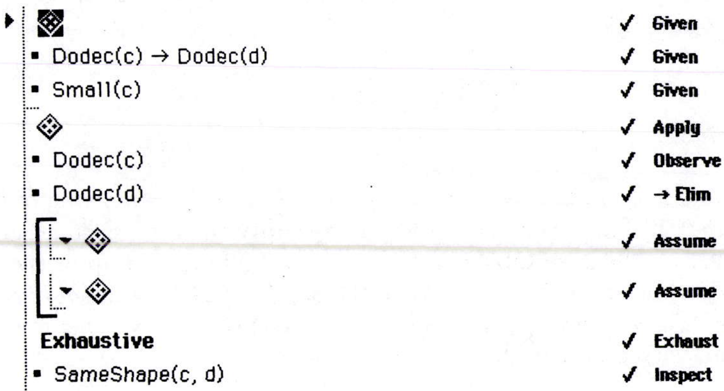


Fig. 2: Example of extended Fitch-style deductive system in Hyperproof

Following the definitions of syntax, semantics, logical notions, and the rule called **Observe** and **Apply** of Hyperproof in [7], the proof is defined much as the same way as the Fitch-style system of deduction. Main differences between the Fitch system and the proofs in Hyperproof is the introduction of the diagrams and inference rule, i.e., **Observe** and **Apply**[2]. The rule **Observe** allows users to extract sentential information from diagrammatic information and **Apply** allows them to extract information in the opposite way.

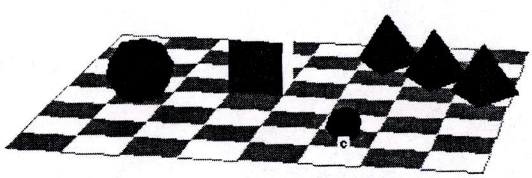


Fig. 3: A diagram

[2]Actually, there are more additional rules in the proof of Hyperproof. Because of the limited space here, we focus on these two rules.

Fig. 4: A proof

For example, if you have a diagram such as Fig.3 at a certain step of inference indicated by > in Fig.4, you can use the rule **Apply** to assert a dodecahedron in the diagram is c, then use **Observe** to extract the information from the diagram that c is a dodecahedron. Note that ■ in the proof indicate a *current diagram* at a step, that is, Fig.3 at the step indicated by >. A *current subproof* is the subproof that has the last sentence or diagram in a proof.

3.2.2 Theory on Hyperproof Representations

Let W, S and D be a set of block worlds, diagrams and sentences in Hyperproof, respectively. Let $P \subseteq D \cup S$ and q be a single Hyperproof representation, then q is *logical consequence* of P, written $P \models q$ iff

$$\forall w \in W \forall r \in P(w \models r \rightarrow w \models q),$$

and P is *consistent* iff

$$\exists w \in W \forall r \in P(w \models r).$$

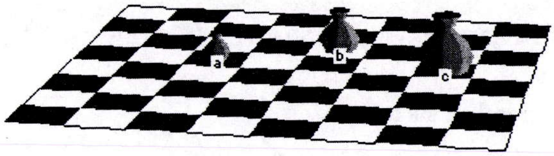
Based on this logical consequence relation, theories on a set S of sentences and a set D of diagrams is defined. Let $T_S = \langle S, \vdash_{T_S} \rangle$ and T_D be theories on S and D, respectively, and if $\Gamma, \Delta \subseteq S$ then

$$\Gamma \vdash_{T_S} \Delta \Longleftrightarrow \forall w \in W(\forall r \in \Gamma(w \models r) \rightarrow \exists r \in \Delta(w \models r)),$$

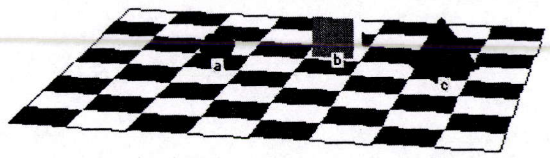
and similarly if $\Gamma, \Delta \subseteq D$ then

$$\Gamma \vdash_{T_D} \Delta \Longleftrightarrow \forall w \in W(\forall r \in \Gamma(w \models r) \rightarrow \exists r \in \Delta(w \models r)).$$

On diagram, there is a important notion called *extension*. One diagram is an extension of another if it can be obtained by assigning definite values for attributes that were not determined by the original situation. For example, in Fig.5, (a) is an extension of (b).



(a) Original diagram



(b) Extension

Fig. 5: Extension of an diagram

3.2.3 Hyperproof as Heterogeneous Logic

Let $\boldsymbol{H}_{Hyp}$ be a Hyperproof classification such that the types of $\boldsymbol{H}_{Hyp}$ are the disjoint union of the types of T_S and T_D, and the tokens of $\boldsymbol{H}_{Hyp}$ are the extended Fitch-style proofs as explained above. A binary relation $\models_{\boldsymbol{H}_{Hyp}}$ between typ($\boldsymbol{H}_{Hyp}$) and tok($\boldsymbol{H}_{Hyp}$) is defined by

1. if η is of types T_S,

   $h \models_{\boldsymbol{H}_{Hyp}} \eta$ iff "η is consistent with all sentences in a current subproof of h", and

2. if η is of types T_D,

   $h \models_{\boldsymbol{H}_{Hyp}} \eta$ iff "the last diagram in a current subproof of h is a extension of η"

for each $h \in$ typ($\boldsymbol{H}_{Hyp}$) and $\eta \in$ tok($\boldsymbol{H}_{Hyp}$).

By above definitin, the classification $\boldsymbol{H}_{Hyp}$ is a heterogeneous logic where there are binary relations between typ(T_S) and typ($\boldsymbol{H}_{Hyp}$), i.e., E_S, and between typ(T_D) and typ($\boldsymbol{H}_{Hyp}$), i.e., E_D. Here, E_S is defined as an identity map from typ(T_S) to typ($\boldsymbol{H}_{Hyp}$). Let $s \in$ tok(Cla(T_S)) and $h \in$ tok($\boldsymbol{H}_{Hyp}$) then there is an information path from s to h cloven by E_S if condition

$$\forall \sigma \in \text{typ}(T_S)(s \models_{\text{Cla}(T_S)} \sigma \iff \exists \eta \in \text{typ}(\boldsymbol{H}_{Hyp})(\sigma = \eta \wedge h \models_{\boldsymbol{H}_{Hyp}} \eta))$$

holds. On the other hand, E_D is also defined as an identity map from $\mathrm{typ}(T_D)$ to $\mathrm{typ}(\boldsymbol{H}_{Hyp})$. Let $d \in \mathrm{tok}(\mathrm{Cla}(T_D))$ and $h \in \mathrm{tok}(\boldsymbol{H}_{Hyp})$ then there is an information path from d to h cloven by E_D if condition

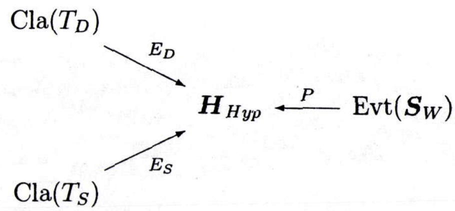
$$\forall \delta \in \mathrm{typ}(T_D)(d \models_{\mathrm{Cla}(T_D)} \delta \Longleftrightarrow \exists \eta \in \mathrm{typ}(\boldsymbol{H}_{Hyp})(\delta = \eta \wedge h \models_{\boldsymbol{H}_{Hyp}} \eta))$$

holds.

P is a classification $\models_P$ such that

$$w \models_P \eta \Longleftrightarrow \text{“}\eta \text{ is satisfied in } w\text{”}$$

for each $w \in W$ and $\eta \in \mathrm{typ}(\boldsymbol{H}_{Hyp})$.



3.3 Proofs of Nonconsequence as Abstract Design

In this section, we explain the construction of a proof of nonconsequence in Hyperproof based on the functional schemes in ADT.

When you constructs this type of proof in Hyperproof, you must create an extension of the given diagram, in which the given sentences and the diagram are all true but the goal sentence is false. This task can be described in the form of a specification. Let S' be a set of the given sentences and D' be a singleton that has the given diagram as an element, then the specification $\langle \Gamma, \Delta \rangle$ is such that Γ is a disjoint union of S' and D' and Δ is a set that has the goal sentence as the only element so that $\Gamma \cap \Delta = \emptyset$ and $\Gamma \cup \Delta \subseteq \mathrm{typ}(\boldsymbol{H}_{Hyp})$. And the task is to find an extension δ' of $\delta \in D'$ such that

$$\exists w \in W(\forall \eta \in \Gamma(w \models \eta) \wedge \forall \eta \in \Delta(w \not\models \eta) \wedge w \models \delta').$$

Let us consider this type of proof in the functional schemes. Since typical users have limited knowledge about proofs of Hyperproof, let $N \subseteq \mathrm{tok}(\boldsymbol{H}_{Hyp})$ be a set of proof which they are already familiar with. Then, their knowledge about sentences about h can be represented by the pair $\langle \Gamma_{h_S}, \Delta_{h_S} \rangle$ of disjoint subsets of $\mathrm{typ}(T_S)$ for each proof $h \in N$ such that

$$\forall \sigma \in \Gamma_{h_S}(h \models_{\boldsymbol{H}_{Hyp}} \sigma) \wedge \forall \sigma \in \Delta_{h_S}(h \not\models_{\boldsymbol{H}_{Hyp}} \sigma).$$

Clearly, $\Gamma_{h_S} \cap \Delta_{h_S} = \emptyset$ and $\Gamma_{h_S} \cup \Delta_{h_S} \subseteq \mathrm{typ}(\boldsymbol{H}_{Hyp})$. The set $K_S = \{\langle \Gamma_{h_S}, \Delta_{h_S} \rangle | h \in N\}$ represents their knowledge about sentences in Hyperproof. By this knowledge K_S, a theory $T(K_S)$ is defined such that

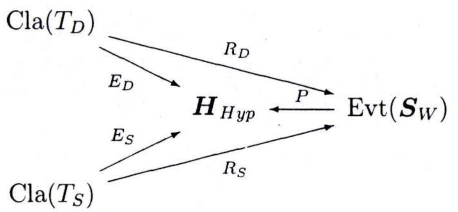
1. $\mathrm{typ}(T(K_S))$ is a set of sentences concerning N,
2. Let $\bar{K}_S$ be the set of partitions of $\mathrm{typ}(T(K_S))$ such that $\langle \Gamma, \Delta \rangle \in \bar{K}_S$ iff $\langle \Gamma', \Delta' \rangle \leq \langle \Gamma, \Delta \rangle$ for some $\langle \Gamma', \Delta' \rangle \in K_S$, and for each $\Gamma, \Delta \subseteq \mathrm{typ}(T(K_S))$, $\Gamma \vdash_{K_S} \Delta$ iff $\langle \Gamma, \Delta \rangle \not\leq \langle \Gamma', \Delta' \rangle$ for each $\langle \Gamma', \Delta' \rangle \in K_S$.

In the same way, a theory $T(K_D)$ is defined based on their concept K_D about diagrams.

Then, the users realize that for each sentence σ and diagram δ, since E_S and E_D are identity functions, there might be a proof $h \in N$ such that $h \models_{\boldsymbol{H}_{Hyp}} \sigma$ and $h \models_{\boldsymbol{H}_{Hyp}} \delta$, respectively. From the functional scheme, in the case of sentences, this leads to a function g_{E_S} such that $g_{E_S}(h) = \langle \{\sigma \in \mathrm{typ}(T_S) \mid \sigma \models_{\boldsymbol{H}_{Hyp}} h\}, \{\sigma \in \mathrm{typ}(T_S) \mid \sigma \not\models_{\boldsymbol{H}_{Hyp}} h\} \rangle$ for a proof h, and $\langle \Gamma_{h_S}, \Delta_{h_S} \rangle \leq g_{E_S}(h)$. In the case of diagrams, $\langle \Gamma_{h_D}, \Delta_{h_D} \rangle \leq g_{E_D}(h)$. On the specification $\langle \Gamma, \Delta \rangle$ mentioned above, this h is a proof such that $\langle \Gamma, \Delta \rangle \leq g(h)_{E_S}$ and $\langle \Gamma, \Delta \rangle \leq g(h)_{E_D}$. At the same time, $\exists w \in W (w \models h)$. This makes $\boldsymbol{H}_{Hyp}$ a medium classification between $\mathrm{Cla}(T_S)$ and $\mathrm{Evt}(\boldsymbol{S}_W)$, and between $\mathrm{Cla}(T_D)$ and $\mathrm{Evt}(\boldsymbol{S}_W)$ such that in the case of sentences,

$$\begin{aligned} \sigma R_S w \iff \forall h \in N (\exists \eta \in \mathrm{typ}(\boldsymbol{H}_{Hyp})(\sigma E_S \eta \wedge h \models_{\boldsymbol{H}_{Hyp}} \eta) \\ \rightarrow \exists \eta \in \mathrm{typ}(\boldsymbol{H}_{Hyp})(w P \eta \wedge h \models_{\boldsymbol{H}_{Hyp}} \eta)) \end{aligned}$$

for each $\sigma \in \mathrm{typ}(T_S)$ and $w \in W$.



4 Conclusion

Thus, in this paper, we formulate the logical relation between heterogeneous specifications by using a reasoning system called Hyperproof as an example. We focus on the proof on nonconsequence in Hyperproof, explain it in the framework of ADT, and finally, show that the creative design process is essentially the same as the process of proofs of nonconsequence.

Acknowledgments

This research is partly supported by Grant-in-Aid for Scientific Research (C) 13650068 of Japan Society for the Promotion of Science (JSPS). The authors would like to thank Makoto Kikuchi and Hirofumi Miki for their valuable comments on the heterogeneous logic and design.

References

[1] Yuzuru Kakuda and Makoto Kikuchi (2001) Abstract Design Theory, Annals of the Japan Association for Philosophy of Science, vol. 10, no. 3.

[2] Yuzuru Kakuda (2002) On Directions of Flow of Information –INFORMATION PATHS, AGENTS, INFORMATION CHANNELS–, Proceedings of 4th International Workshop on Emergent Synthesis, to appear.

[3] Y. Kakuda, M. Kikuchi, I. Nagasaka Information paths and Agents, To appear.

[4] Jon Barwise and John Etchemendy (1994) Hyperproof, CSLI Lecture Notes No. 42, CSLI Publication.

[5] Yoshikawa, H. (1981) General design theory and a CAD system, Man-machine Communication in CAD/CAM, T. Sata, E. Warman (editors), pp 35 - 57, North-Holland.

[6] Jon Barwise and Jerry Seligman (1997) Information Follow, Cambridge Tracts in Theoretical Computer Science 44, Cambridge University Press.

[7] Jon Barwise and John Etchemendy (1995) Heterogeneous Logic, in Diagrammatic Reasoning: Cognitive and Computational Perspectives, Janice Glasgow, N. Hari Narayanan and B. Chandrasekaran, eds., Cambridge, Mass: The MIT Press, and Menlo Park, CA: AAAI Press, 211-234.