Chapter 5
Beyond Logical Approach to Systems Theory

Shingo Takahashi

Abstract Logical Approach to Systems Theory (LAST) provides a "meta"-framework to describe and investigate explicitly and deeply the similarity of system models based on general and formal definitions of system models and their structures. The main theorem of LAST is F-morphism theorem. It substantially enhances the concept of "isomorphism" between system models of the same type to those of "different types" in the sense that they can be described in different types of languages. After considering the limitation of LAST by interpreting Gödel's Incompleteness Theorem, it is clarified that the adaption and the structural change of a system would be beyond the description capability of LAST. Hence some new conceptual devices and effective models such as agent, internal model, and organizational learning are required to be developed. Agent-Based Organizational Cybernetics (AOC) could be a key model to describe the organizational learning that induces the adaptation and structural change of system models especially in social systems.

Keywords System model · Structure · Logic · Language · Homomorphism · F-morphism · Internal model · Agent · Organizational learning

5.1 Introduction

This paper describes the essence of Logical Approach to Systems Theory (LAST) (Takahashi & Takahara, 1995), and the framework of agent-based organization cybernetics (AOC) for consideration for adaptation that includes internal model and organizational learning as the main concepts (Takahashi, 2006).

S. Takahashi (✉)
School of Creative Science and Engineering, Waseda University, Tokyo, Japan
e-mail: shingo@waseda.jp

K. Kijima et al. (eds.), *Systems Research I*, Translational Systems Sciences 26,
https://doi.org/10.1007/978-981-16-9937-5_5

The similarity of systems models has been a central concept in systems theory as well as systems science. Basically, from the theoretical point of view, a system model is similar to another if a homomorphism can be defined between the two system models. The homomorphism theorem is one of the typical results on how isomorphic images can be constructed in terms of homomorphism. Though the homomorphism concept should be primally considered to give a similarity relation of systems models, it should be still clarified what properties are interpreted as similar between system models in the sense of "preserving" the properties by a homomorphism from one model to another, and how the system model should be expressed to define in a totally formal way the similarity by generalizing the homomorphism concept.

LAST will give an answer to these questions on general similarity.

LAST provides a "meta"-framework to describe and investigate explicitly and deeply the similarity of system models based on general and formal definitions of system models and their structures. Systems models are described in terms of model theory. Roughly speaking, the following relationship holds (Chang & Keisler, 1973):

$$\text{model theory} = \text{universal algebra} + \text{logic}.$$

Universal algebra is appropriate or proper for describing system models, especially general system models or abstract system models (Mesarovic & Takahara, 1975, 1989), each of which is expressed as a mathematical structure. Logic generally includes as the main body language, formation rules of formulas, deduction system, and satisfaction of formulas in a model. We sometimes use category theoretical formulation as well as universal algebra, which is often effective for the theory of general system models, when we consider a class of all systems satisfying some specific properties, for example, a class of all state space representations.

The main theorem of LAST is F-morphism theorem. It substantially enhances the concept of "isomorphism" between system models of the same type to those of "different types," which means that the two models described in "different languages" can be isomorphic in terms of F-morphism. The relation of being "isomorphic" of system models is the basis of the similarity in systems science. Hence the F-morphism concept in LAST should be considered as fundamental for the similarity in systems science, as well as homomorphism as a specific case.

The logical approach also clarifies the distinction of what type of properties of systems can be described in the theory and what type of them cannot. In particular, the adaptation of systems and the structural change of a system would be beyond the description capability of LAST. We need to develop some new conceptual devices such as agent, internal model, and organizational learning. This chapter focuses on considering the adaptation of social systems and introduces the agent-based organization cybernetics (AOC) as a key model.

5.2 General Systems

In the most general sense, a system can be defined as a relation on some attributes $V_1, \ldots, V_n$, the relation which is expressed by $S \subset V_1 \times \cdots \times V_n$ (Mesarovic & Takahara, 1975, 1989). Each set V_i represents the collection of alternative ways in which the corresponding object appears in the relation that defines the system. The object is identified in terms of a property and an attribute. A general form of an input-output system model is described as $S \subset X \times Y$, where X is the set of input attributes of concerns and Y the set of output attributes of concern, respectively. Starting from this general definition of a system, we develop systems theory by introducing into the attributes some structures such as linearity, stationarity, and so on that are suitable to our interests in objects as systems. Our definition of a system model realizes this concept of a system in as a formal and general way as possible.

5.3 Logical Approach to Systems Theory: LAST

Logical Approach to Systems Theory (LAST) provides a second-order framework to describe and investigate explicitly and deeply both system models and structures of them from a model theoretic point of view (Takahashi, 1995). LAST is characterized by type-free representation, distinction between model and structure, and hierarchical structure expansion.

1. Type-free representation of system models. The representation is not only independent of any specific formalisms such as differential equations or automata, but also capable of clarifying possibly different types of system models constructed from multifaceted aspects in modeling.
2. Distinction of system models from their structures. The logical approach provides a language and formal framework to describe the properties of each system model, to define its structure and specify the class. The language is determined by the structure of a system model.
3. Hierarchical structure expansion. The relations of inclusion among classes of system models are given as hierarchical relations of the structures of system models. That is, when a class of system models is included in another, the structure of system models of the former class is obtained by expanding the other. The expanded structure inherits the antecedent.

LAST places its emphasis mainly on complex systems such as information systems or social systems as concrete instances rather than traditional topics in control theory. So complex is even a single information system that includes system models of various types coming from diversity of individual objects. The design of a complex system requires to deal with such variety of types of system models and to specify a class of system models of any type separately from the description of the system models. Thus exploring the inter-relations among system models to describe a complex system is expected as a key feature in LAST. The primary purpose of LAST is to provide an effective framework for using basic concepts in systems theory to describe complex systems. The use of model theory rather than usual set theory would make this purpose easily attainable. LAST aims at a practical device for designing complex systems as well as theoretical development on system models and structures.

5.3.1 Basic Concepts of LAST

There are at least four basic concepts to be understood in applying LAST. We will illustrate in the following sections each of the concepts in detail.

1. **System model.** As stated previously, system models are objects of study in systems theory. LAST provides a formal framework for representing a system model to reflect systems recognition of a model builder. The representation should be fully independent of the types of system models, while individual system models employed in individual systems theories have their own specified types. Thus we have to specify the language in such a way that not only system models, but their types can be described, that is, what the types of system models are should be clearly defined. LAST gives a natural and suitable way to satisfy such requirement.
2. **Structure.** Since every system in the reality is recognized only as a system model, the structure of a system is equivalent to that of a system model. If it is allowed to use the term "structure" in defining a system, we could define a system as follows: "A system is a whole entity having its own structure."

   The concept of structure has been less well-defined than that of a system and rather controversial. However, in developing a meta-theory of systems, we cannot avoid making clear the concept of structure in a formal way. In our logical approach structure of a system model will be defined by a pair of a language to describe the system model and a set of formulas to specify the behavior of the system model. This definition comprehends essential parts of other definitions of structure.
3. **Morphism.** The morphism is a conceptual basis for considering similarities between system models. The similarity between two system models is often defined by some morphism between them, more precisely, by some homomorphism. The definition of similarity by homomorphism, however, depends on a particular representation or specification such as Mealy type automata used to describe the situation. Since morphism is both practically and conceptually significant as such in systems theory, we need to develop some general morphism independent from representation types so that it gives the similarity between system models not only of the same kind of type, but of different types. The type-freeness of representation of system models in LAST enables us to construct such a general morphism between system models of possibly different types including

homomorphism as a truly special case. For example we can construct a general morphism from a finite automaton to a Petri net.

4. **Universality.** Since an aim of LAST is to develop a meta-theory concerning "inter-models," we are interested in universal properties found in a class of system models or their structures rather than in individual models as instances. So far, some universal properties significant in systems theory have been examined. Here we will concentrate on the realization problem as universality, the problem which deals with how the minimal model in a given class of a structure can be constructed from a given set of input-output pairs. The algebraic specification is one of important examples of the realization as universality.

5.3.2 System Model

5.3.2.1 Definition of System Model

A system model is a whole entity with some interactions among its elements. A direct and natural representation idea of a system model is to express it as a mathematical structure that consists of a base set with relations and functions defined on it.

Definition 5.1 [System Model] A system model $\mathfrak{M}$ is composed of:

1. A base set M;
2. A set of $\lambda(i)$-ary relations on M, $\{R_i | \, i \in I\}$, where λ is a function such that $I \to N^+$ (positive integers);
3. A set of $\mu(j)$-ary functions on M, $\{f_j | \, j \in J\}$, where μ is a function such that $J \to N$ (non-negative integers).

Here an n-ary relation or function has n arguments, written as $R(a_1, \ldots, a_n)$, or $f(a_1, \ldots, a_n)$. The function λ (or μ) means that the arity of a relation R_i (or f_j) depends on its index i (or j), written as $R_i(a_1, \ldots, a_{\lambda(i)})$ or $f_j(a_1, \ldots, a_{\mu(j)})$. Nullary functions with no arguments are called *constants*. The pair $\langle \lambda, \mu \rangle$ is called the *type* of $\mathfrak{M}$.

We write $\mathfrak{M}$ as follows:

$$\mathfrak{M} = \left\langle M; \{R_i | i \in I\}, \left\{f_j | j \in J\right\}\right\rangle.$$

We sometimes write $| \mathfrak{M} |$ to indicate the base set M.

This definition of a system model has very wide applicability. Most systems representations we are interested in can be reformulated in the above form. We illustrate below some typical and significant examples of system models.

Example 5.1 [Input-Output System Model] A simple but quite important instance of system models is an input-output system model. Although we could describe it in some different representations, the following one is natural. An input-output system model is expressed by

$$M_{I/O} = \langle X \cup Y; S, X, Y \rangle ,$$

where $S \subset X \times Y$, X is the set of inputs and Y the set of outputs.

Example 5.2 [Linear System Model] A linear system model is a system model whose input set and output set are vector spaces and whose behavior has the linear property that $S(x,y)$ and $S(x',y')$ implies $S(\alpha x + \beta x', \alpha y + \beta y')$ for any α and β in a field F over which the input and output sets are the vector spaces. The linearity of the input set is represented by the following system model: $M_X = \langle F \cup X; F, X, +, -, ^{-1}, \cdot, 0_F, 1_F, 0_X \rangle$, where X is the set of inputs, F a unary relation that is the set of scalars, 0_X the zero vector, 0_F the zero scalar, 1_F the unit element of F, $+$ a binary function representing both scalar and vector addition, $-$ a unary function representing the additive inverse, $\cdot$ a binary function representing scalar multiplication and multiplication of a vector by a scalar and $^{-1}$ a unary function representing the multiplicative inverse. The linearity of the output set is similarly defined. A linear system model is defined as the union of, $\mathfrak{M}_Y$ and the input-output system model with the linear property of behavior, where the union of two system models $\mathfrak{M}_1$ and $\mathfrak{M}_2$ is the system model whose base set, functions, and relations are respectively the unions of the corresponding sets of the two system models.

Example 5.3 [Discrete Event System Specification (DEVS) Model] The DEVS formalism provides a means of constructing simulation models and a formal representation of discrete event systems capable of mathematical manipulation just as differential equations serve this role for continuous systems (Zeigler, 1990). A DEVS model described in the DEVS formalism consists of a time base, inputs, states, outputs, and functions for determining next states and outputs given current states and inputs:

$$M_{\text{DEVS}} = \langle X \cup S \cup Y \cup R \cup \{\infty\} ; X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, t_{\text{a}}, Q, T \rangle ,$$

where X is a set of external event types, S a set of sequential state, Y a set of external event types generated as outputs, T the time base, t_{a} the time advance function from S to the non-negative reals with infinity:$t_{\text{a}} : S \rightarrow R^{+}_{0,\infty}$, Q the total state set defined by $Q = \{(s,e) | \, s \in S, 0 \leq e \leq t_{\text{a}}(s)\}$, δ_{int} the internal transition function: $\delta_{\text{int}} : S \rightarrow S$, δ_{ext} the external transition function: $\delta_{\text{ext}} : Q \times X \rightarrow S$, and λ the output function:$\lambda : Q \rightarrow Y$.

Some essential behavior of a system specified by DEVS, such as the property of the output function that generates an external output just before an internal transition takes place, should be considered to be included implicitly in $\mathfrak{M}_{\mathrm{DEVS}}$.

5.3.2.2 Language for Describing Systems Properties

The systems properties are the properties possessed by a system model such as linearity, stationarity, or causality. To investigate properties of these systems properties, which can be called the meta-treatment of system models, we introduce a formal language to describe systems properties. The use of the formal language characterizes LAST. In this section we give only the formal framework of the language.

Definition 5.2 [Language for a System Model] The language for a system model $\mathfrak{M}$ consists of:

1. $\lambda(i)$-ary predicate letters $\mathbf{R}_i$ for each $i \in I$, where λ is a function such that $I \to N^+$ (positive integers);
2. $\mu(j)$-ary function symbols $\mathbf{f}_j$ for each $j \in J$, where μ is a function such that $J \to N$ (non-negative integers).

We write $\mathfrak{L}(\mathfrak{M})$ also as $\mathfrak{L}(\mathfrak{M}) = \left\langle \{\mathbf{R}_i | i \in I\}, \left\{\mathbf{f}_j | j \in J\right\}\right\rangle$.

$\langle \lambda, \mu \rangle$ is also said to be the type of $\mathfrak{L}(\mathfrak{M})$. There is the one-to-one correspondence, denoted by *Cor*, between the boldface symbols in $\mathfrak{L}(\mathfrak{M})$ and the light face symbols for the relations and functions in $\mathfrak{M}$, i.e., $Cor(\mathbf{R}_i) = R_i$ for each $i \in I$ and $Cor(\mathbf{f}_j) = f_j$ for each $j \in J$. Then $\mathfrak{M}$ is said to be a *realization* of the language $\mathfrak{L}(\mathfrak{M})$ or *model* for $\mathfrak{L}(\mathfrak{M})$. The languages for two system models of the same type are, up to alphabetic variants, identical. Therefore every system model of the same type as a system model $\mathfrak{M}$ is a realization of the language $\mathfrak{L}(\mathfrak{M})$.

For example, an input-output system model,

$$\mathfrak{M}_{I/O} = \langle X \cup Y; S, X, Y \rangle$$

and all the system models of the same type as this system model are realizations of the language,

$$\mathfrak{L}\left(\mathfrak{M}_{I/O}\right) = \langle \mathbf{S}, \mathbf{X}, \mathbf{Y} \rangle .$$

We should notice that for language we customarily use boldface symbols with the same alphabets as a system model, for example, S and $\mathbf{S}$, so as not to confuse language with system models. This usage is only for the sake of convenience. However, we should notice that a language, e.g., $\mathfrak{L}\left(\mathfrak{M}_{I/O}\right)$, is purely syntactic construct. Other models than $\mathfrak{M}_{I/O}$, e.g.,$\mathfrak{M}' = \langle Z; T, V, W \rangle$, can be also realizations of $\mathfrak{L}\left(\mathfrak{M}_{I/O}\right)$, even if they have no property of an input-output system model. The desirable properties that every input-output system model should have are specified not only as a language but as a structure of the system model.

To describe systems properties, we need "grammar" that distinguishes "right sentences" from wrong sentences. In our logical approach, we assume that the properties of a system can be expressed as first-order sentences in first-order language that plays the role of the "grammar." The first-order language consists of the primitive symbols such as variables, logical connectives, quantifiers, identity symbols, parentheses and comma with the language $\mathfrak{L}(\mathfrak{M})$ for a system model, the formation rules of the terms, the atomic formulas and the well-formed formulas. The set of terms of the language $\mathfrak{L}(\mathfrak{M})$, denoted by $\mathrm{Term}(\mathfrak{L}(\mathfrak{M}))$, is recursively defined from the language $\mathfrak{L}(\mathfrak{M})$. Similarly the set of atomic formulas, denoted by $\mathrm{Atom}(\mathfrak{L}(\mathfrak{M}))$ and that of well-formed formulas, denoted by $\mathrm{Form}(\mathfrak{L}(\mathfrak{M}))$, are recursively defined from the language.

The logical connectives and universal quantifier as primitive symbols have no proper meanings such as "and," "not," and "for all." These intended meanings are realized only when these symbols are interpreted in a specific system model. This realization is called satisfaction. We will usually use other symbols, say $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, as individual variables. A term that has no variable is called a closed term. In first-order logic some abbreviations such as $\exists\mathbf{v}$, $\phi \vee \psi$, $\phi \rightarrow \psi$, and so on will be defined in the standard manner. Although the abbreviations actually intend to have the meanings of "for some (or there exist)," "or," and "imply" respectively, these meanings are only realized in a system model. Other logical concepts such as the *scope* of the quantifier, a bound variable, a free variable, and so on are introduced in the language, which can be found in the standard textbook of logic. A formula $\phi \in \mathrm{Form}(\mathfrak{L}(\mathfrak{M}))$ is said to be a *sentence* of $\mathfrak{L}(\mathfrak{M})$ if ϕ has no free variables. $\mathrm{Sent}(\mathfrak{L}(\mathfrak{M}))$ denotes the set of sentences.

In our logical approach every *systems property* of an individual system model is expressed by a sentence. For example, an input-output system model $\mathfrak{M}_{I/O}$ has a basic systems property: "every element of the system is a pair of an input and an output." This property can be expressed by the following sentence: $(\forall\mathbf{xy})(\mathbf{S}(\mathbf{x},\mathbf{y}) \rightarrow \mathbf{X}(\mathbf{x}) \vee \mathbf{Y}(\mathbf{y}))$. In ordinary mathematical notations, i.e., in set-theoretical language, this sentence means that $S \subset X \times Y$. As another example, an input-output system of function-type (Mesarovic & Takahara, 1989) can be expressed as: $(\forall\mathbf{x} \in \mathbf{X} \rightarrow (\exists\,!\,\mathbf{y} \in \mathbf{Y})\mathbf{S}(\mathbf{x},\mathbf{y}))$, where the notation $\exists\,!\,\mathbf{x}\phi(\mathbf{x})$ is the abbreviation of the sentence that means "there uniquely exists $\mathbf{x}$ such that $\phi(\mathbf{x})$."

We have noticed that a systems property is expressed by a sentence. Conversely, a sentence should be interpreted as a systems property in a system model so that the sentence obtains a concrete meaning in the system model.

Let us consider a system model $\mathfrak{M}_S = \langle\{a,b,c\}\,;S,X,Y\rangle$, where $S = \{(a,b),(a,c),(b,c)\}$ and $X = Y = \{a,b\}$. Is a formula $\mathbf{S}(\mathbf{x},\mathbf{y}) \rightarrow \mathbf{X}(\mathbf{x}) \vee \mathbf{Y}(\mathbf{y})$ true in this system model? If we assign a and b to the variables $\mathbf{x}$ and $\mathbf{y}$ respectively, the formula is true in that model. However if c to $\mathbf{y}$, then it is not true. To judge the truth of a formula containing some free variables, we need to assign an element of the base set of a system model to each variable. As will be stated later, since a sentence has no free variable, we can judge its truth without depending on the assignment of variables.

For example, a sentence $(\forall \mathbf{xy})(\mathbf{S}(\mathbf{x},\mathbf{y}) \to \mathbf{X}(\mathbf{x}) \vee \mathbf{Y}(\mathbf{y}))$ is not true in the system model $\mathfrak{M}_S$, so this model is not an input-output system model. An assignment to each variable is defined by an assignment function. Given a system model $\mathfrak{M}$ with a base set M, an *assignment function* ρ (or briefly *assignment*) is a function of the set V of variables to M. Then for a given assignment, sentences in terms of the language of the system model are interpreted into the system model. This interpretation is defined as the *denotation* of a term in $\mathfrak{L}(\mathfrak{M})$. Each variable $\mathbf{x}$ is replaced by the element $\rho(\mathbf{x}) \in M$ and each function symbol is replaced by the corresponding function. A denotation with respect to a given assignment can be regarded as a function of terms to the base set of a system model.

The concept that a systems property holds in a system model is defined as the *satisfaction* of formulas.

Definition 5.3 [Satisfaction] A formula ϕ holds in $\mathfrak{M}$ with an assignment function ρ, or ρ satisfies ϕ in $\mathfrak{M}$, written $\mathfrak{M} \models \phi\,[\rho]$, is defined recursively:

1. $\mathfrak{M} \models \mathbf{R_i}\left(t_1, \ldots, t_{\lambda(i)}\right)$ if and only if $\left\langle t_1^d\,[\rho], \ldots, t_{\lambda(i)}^d\,[\rho]\right\rangle \in R_i$;
2. $\mathfrak{M} \models \neg\phi\,[\rho]$ if and only if it is not the case that $\mathfrak{M} \models \phi\,[\rho]$;
3. $\mathfrak{M} \models \phi_1 \wedge \phi_2\,[\rho]$ if and only if $\mathfrak{M} \models \phi_1\,[\rho]$ and $\mathfrak{M} \models \phi_2\,[\rho]$;
4. $\mathfrak{M} \models \forall \mathbf{x}\phi\,[\rho]$ if and only if $\models\phi[\rho(y/\mathbf{x})]$ for any $y \in M$.

5.3.3 Structure

The structure of a system model characterizes the system model in the sense that the structure determines to which class of systems the system model pertain. In this sense if a system model is expressed by a collection of some differential equations, we can say that the matrices of the coefficients of the differential equations give a structure of the system model. However, from systems viewpoints, a class of system models should be specified not by the form of differential equations, but by a set of systems properties. Hence the structure of a system model should have at least the following features.

First, the structure of a system model generates its properties or behavior to be recognized.

Second, the representation of structure is based on a hierarchical construction. For example, the structure of an input-output linear system model is "hierarchically" constructed from both a linear structure and an input-output structure, in the sense that the input-output linear structure explicitly "inherits" the properties from the linear and input-output structure.

Third, the structure distinguishes the properties of the class of system models satisfying it from those of an individual system model in the class.

One way to fulfill the above requirements is to adopt a "language" that expresses systems properties, and to represent the structure as "axioms." This means that we should abstract basic properties from a class of system models as axioms that are common characteristics of the class. Thus the structure of a "family system" in the previous section is abstracted from concrete family models. For example, we can abstract some axioms such that every father is a male, every mother is a female, father and mother are married, all brothers have the same father and mother, and so on. The language such as "father," "male," "every," "is-a," etc., and some "grammar" to make legal sentences should be chosen before axioms are described. Then the axioms are expressed by some sentences in that language. We should notice that this example of the structure of family does not include all families at all; a family that has brothers whose mothers are different is not included.

A language and axioms are chosen from a systems viewpoint that reflects our current interest. In this sense the structure of a system model expresses fundamental interactions we recognize as the system model does. Thus a modeling process contains as its essential part some stages of specifying language and constructing axioms. Consequently the structure of a system model is defined as a pair, $(\mathfrak{L}; \Sigma)$, of language $\mathfrak{L}$ to define the system model and a set of axioms Σ to describe the class to which the system model pertains. In LAST every structure is defined in a formal language such as first-order language. Use of other formal languages than first-order is not restricted in LAST. We notice that there are some advantages and disadvantages of the use of first-order language.

The formal description of a structure of a system model has some technically outstanding advantages as well as conceptual ones. It enables us to point out what a systems property of a given system model is, and to distinguish the system properties from system models that "satisfy" the properties. This relation is provided as satisfaction relation that is one of the main characteristics of LAST: type-free representation. Thus we can construct and specify a class of system models without depending on the concrete descriptions of individual system models.

The formal definition of the structure of a system is defined below.

Definition 5.4 [Structure of a System] Let $\mathfrak{M}$ be a system model, $\mathfrak{L}(\mathfrak{M})$ the language for $\mathfrak{M}$, and Σ a set of sentences of $\mathfrak{L}(\mathfrak{M})$, where $\mathfrak{M} \models \Sigma$. Then the *structure* of a system as a prototype of the system model $\mathfrak{M}$ is defined by $(\mathfrak{L}(\mathfrak{M})\,; \Sigma)$.

A given system model necessarily determines $\mathfrak{L}(\mathfrak{M})$, unique up to alphabetic invariants. We should notice that $\mathfrak{L}(\mathfrak{M})$ is a collection of "symbols," therefore the role of $\mathfrak{L}(\mathfrak{M})$ in the systems recognition is to point out the names and types of the relations that are identified in the system we recognize.

On the other hand, Σ provides the rules how elements in a system model interact. Therefore the properties of a system implied by the structure of the system are expressed as the formulas derived from Σ; $T(\Sigma) = \{\phi \in \mathrm{Sent}(\mathfrak{L}(\mathfrak{M}))\,|\,\Sigma \vdash \phi\}$ is the whole of the properties characterized by the structure of the system, $(\mathfrak{L}(\mathfrak{M})\,; \Sigma)$. If Σ is complete, the properties satisfied by a system model having the structure $(\mathfrak{L}(\mathfrak{M})\,; \Sigma)$ accord with the properties of a system implied by

$(\mathfrak{L}(\mathfrak{M}) ; \Sigma)$); that is, let $\mathrm{Th}(\mathfrak{M}) = \{\phi \in \mathrm{Sent}(\mathfrak{L}(\mathfrak{M})) \,|\mathfrak{M} \models \phi, \mathfrak{M} \models \Sigma\}$, then $\mathrm{Th}(\mathfrak{M}) = T(\Sigma)$. Notice that it follows from the definition that $(\mathfrak{L}(\mathfrak{M}) ; \Sigma)$ cannot be uniquely determined for one system model $\mathfrak{M}$ since we can take another Σ as axioms for which $\mathfrak{M}$ is a model. This means that there may be plenty of system models satisfying a given structure $(\mathfrak{L}(\mathfrak{M}) ; \Sigma)$. For example, many models satisfy the Peano's Axioms well known as a structure of the natural number. They are not necessarily isomorphic to the natural numbers (Chang & Keisler, 1973).

As an example we define below the structure of input-output system.

Definition 5.5 [The Structure of Input-Output System] The structure of input-output system model is defined by $(\mathfrak{L}_{I/O};\Sigma_{I/O})$:

$$\mathfrak{L}_{I/O} = \{\mathbf{X}, \mathbf{Y}, \mathbf{S}\},$$

where

$\mathbf{X}$, $\mathbf{Y}$: unary relation symbols,
$\mathbf{S}$: a binary relation symbol;

$$\Sigma_{I/O} = \left\{\phi_{I/O}\right\},$$

where $\phi_{I/O} \equiv (\forall \mathbf{xy})(\mathbf{S}(\mathbf{x}, \mathbf{y}) \rightarrow \mathbf{X}(\mathbf{x}) \wedge \mathbf{Y}(\mathbf{y}))$.

An input-output system model $M_{I/O} = \langle X \cup Y; S, X, Y\rangle$ is a realization of $\mathfrak{L}_{I/O}$ and model for $\Sigma_{I/O}$.

5.3.4 Morphism

One of the most important purposes of systems science is to investigate the similarity between system models. The similarity can be divided into two types: structural similarity and behavioral similarity. So far, fixing the type of models, we have studied structural similarity in systems theory using modeling morphisms defined especially between input-output system models. However, as will be made clear in the subsequent discussions, these modeling morphisms are defined not in a general way in which we can deal with structural similarity between any system models, but in a specific way based only on homomorphisms. This section is devoted to the development of a general theory of structural similarity between system models.

5.3.4.1 Morphisms for Models of the Same Type

In this section we investigate morphisms between system models of the same type, following the three cases mentioned in the previous section.

Preservation of Generator: Homomorphism

A morphism between system models of the same type is usually given by a homomorphism, which preserves only atomic formulas as the generators of the language.

Definition 5.6 [Homomorphism] Let $\mathfrak{M}_1 = \left\langle M_1; \left\{ {R_i}^1 | i \in I \right\}, \left\{ {f_j}^1 | j \in J \right\} \right\rangle$ and $\mathfrak{M}_2 = \left\langle M_2; \left\{ {R_i}^2 | i \in I \right\}, \left\{ {f_j}^2 | j \in J \right\} \right\rangle$. Notice that $\mathfrak{M}_1$ and $\mathfrak{M}_2$ are of the same type. A function $h : M_1 \to M_2$ is called a *homomorphism* of $\mathfrak{M}_1$ to $\mathfrak{M}_2$ if for any $i \in I$, $j \in J$, $a_1, \ldots, a_{\lambda(i)}$, $a_1, \ldots, a_{\mu(j)}$, $a \in M_1$, $\left(a_1, \ldots, a_{\lambda(i)}\right) \in R_i^1$ implies $\left(h\left(a_1\right), \ldots, h\left(a_{\lambda(i)}\right)\right) \in R_i^2$ and $h\left(f_j^1\left(a_1, \ldots, a_{\mu(j)}\right)\right) = f_j^2\left(h\left(a_1\right), \ldots, h\left(a_{\mu(j)}\right)\right)$.

A bijective (i.e., one-to-one and onto) homomorphism is called an *isomorphism*.

From Definition, we can see that a homomorphism preserves only the atomic formulas, which is viewed as the generators of the language for a system model. In systems theory, the concept of a homomorphism is defined as a modeling morphism between input-output system models.

Definition 5.7 [Modeling Morphism (Mesarovic & Takahara, 1975, 1989)**]** Let $S \subset X \times Y$ and $S' \subset X' \times Y'$ be input-output system models. Let $h_x : X \to X'$ and $h_y : Y \to Y'$ be functions. $h = (h_x, h_y) : S \to S'$ is called a *modeling morphism of S to S'* if for any $(x, y) \in X \times Y$, $(x, y) \in S$ implies $(h_x(x), h_y(y)) \in S'$.

For example, let us consider input-output system models $\mathfrak{M} = \langle X \cup Y; S, X, Y \rangle$ and $\mathfrak{M}' = \left\langle X' \cup Y'; S', X', Y' \right\rangle$, where S, S': binary relations on $X \cup Y$ and X, X', Y, Y': unary relations on $X \cup Y$. Suppose that $\mathfrak{M}$ and $\mathfrak{M}'$ satisfy $\mathbf{S}(\mathbf{X}, \mathbf{Y}) \to \mathbf{X}(\mathbf{X}) \wedge \mathbf{Y}(\mathbf{Y})$ and $\mathbf{S}'(\mathbf{X}, \mathbf{Y}) \to \mathbf{X}'(\mathbf{X}) \wedge \mathbf{Y}'(\mathbf{Y})$, respectively. Then a homomorphism h of $\mathfrak{M}$ to $\mathfrak{M}'$ is regarded as a modeling morphism of S to S'. Notice that from the definition of a homomorphism, $h(x) \in X'$ for any $x \in X$ and $h(y) \in Y'$ for any $y \in Y$.

Preservation of Σ: Σ-Homomorphism

Next we consider homomorphisms preserving axioms Σ. Recall that the axioms Σ provide the structure of a system.

Grätzer defined such homomorphisms as Σ-homomorphisms (Grätzer, 1979). By a Σ-homomorphism the axioms Σ are preserved in a homomorphic image. We formulate a Σ-homomorphism directly based on this idea. This definition is different from Grätzer's original definition that uses the concept of $\Phi - l$ inverse.

Let h be a homomorphism of $\mathfrak{M}_1$ to $\mathfrak{M}_2$. *The homomorphic image of h* in $\mathfrak{M}_2$ is a submodel of $\mathfrak{M}_2$ whose domain is $h(\mathfrak{M}_1)$. We write $h(\mathfrak{M}_1)$ to indicate the homomorphic image of h in $\mathfrak{M}_2$ as follows.

$$h(\mathfrak{M}_1) = \left\langle h(M_1);\left\{R_i^2 \cap h(M_1)^{\lambda(i)} | i \in I\right\}, \left\{f_j^2 \parallel h(M_1)^{\mu(j)} | j \in J\right\}\right\rangle,$$

where $f_j^2 \parallel h(M_1)^{\mu(j)}$ denotes the restriction of f_j^2 to $h(M_1)^{\mu(j)}$.

By the property of a homomorphism, $f_j^2 \parallel h(M_1)^{\mu(j)}$ is well-defined.

We define a Σ-homomorphism as a homomorphism whose homomorphic image preserves Σ.

Definition 5.8 [Σ-Homomorphism] Let $\mathfrak{M}_1$ and $\mathfrak{M}_2$ be system models of the same type, and $\mathfrak{M}_1 \models \Sigma$ and $\mathfrak{M}_2 \models \Sigma$. A homomorphism h of $\mathfrak{M}_1$ to $\mathfrak{M}_2$ is called a Σ-*homomorphism* of $\mathfrak{M}_1$ to $\mathfrak{M}_2$, if $h(\mathfrak{M}_1) \models \Sigma$.

Our definition of Σ-homomorphism is slightly weaker than Grätzer's definition using the concept of $\Phi - l$ inverse. His definition requires that any "inverse" elements should be preserved. A homomorphism that has the $\Phi - l$ inverses is defined as a strong Σ-homomorphism. The axioms Σ are preserved in the homomorphic image on h in $\mathfrak{M}_2$ by a strong Σ-homomorphism h. So the homomorphic image of a strong Σ-homomorphism h, $h(\mathfrak{M}_1)$, is a model of Σ, i.e., $h(\mathfrak{M}_1) \models \Sigma$.

Preservation of Th $(\mathfrak{M})$: S-Homomorphism

That two system models are isomorphic or of the same structure implies that in a sense the properties of the two system models are equivalent. A usual homomorphism preserves the primitive properties, i.e., the generators. In this section we will define a homomorphism as an *S-homomorphism* that preserves all sentences satisfied in a system model (Th $(\mathfrak{M})$). Furthermore we will show that an induced homomorphism is an S-homomorphism, which is well known as the homomorphism theorem. From this theorem we can see that every morphism for the structural similarity should be an S-homomorphism.

Definition 5.9 [S-Homomorphism] Let $\mathfrak{M}_1$ and $\mathfrak{M}_2$ be system models of the same type, and $h : \mathfrak{M}_1 \to \mathfrak{M}_2$ a homomorphism of $\mathfrak{M}_1$ to $\mathfrak{M}_2$. Then h is called an *S-homomorphism* of $\mathfrak{M}_1$ to $\mathfrak{M}_2$ if for any sentence ϕ of $\mathfrak{L}(\mathfrak{M}_1)$

$$\mathfrak{M}_1 \models \phi \quad \text{if and only if } h(\mathfrak{M}_1) \models \phi.$$

From the definition we can immediately see that an S-homomorphism is a Σ-homomorphism. We should notice that if every sentence that holds in $\mathfrak{M}_1$ holds in $h(\mathfrak{M}_1)$ as well, then h is already an S-homomorphism. Indeed if a sentence ϕ holds in $h(\mathfrak{M}_1)$ and does not hold in $\mathfrak{M}_1$, then $\neg\phi$ holds in $\mathfrak{M}_1$. Thus, by the above condition, $\neg\phi$ holds in $h(\mathfrak{M}_1)$, which is a contradiction.

Let $h : \mathfrak{M}_1 \to \mathfrak{M}_2$ be a homomorphism of $\mathfrak{M}_1$ to $\mathfrak{M}_2$. Then we define *the quotient system model with respect to h*, written by $\mathfrak{M}_1/h$:

$$\mathfrak{M}_1/h = \left\langle M_1/h; \left\{R_i^1/h | i \in I\right\}, \left\{f_j^1/h | j \in J\right\}\right\rangle,$$

where $\mathfrak{M}_1 = \langle M_1; \{R_i{}^1 | i \in I\}, \{f_j{}^1 | j \in J\}\rangle$ and $\mathfrak{M}_2 = \langle M_2; \{R_i{}^2 | i \in I\}, \{f_j{}^2 | j \in J\}\rangle$; M_1/h is the partitioned set of M_1 by the equivalence relation defined by:

$$a \equiv b \quad \text{if and only if } h(a) = h(b) \text{ for any } a, b \in M_1,$$

$s\left([a_1], \ldots, [a_{\lambda(i)}]\right) \in R_i^1/h$ if and only if $\left(h(a_1), \ldots, h(a_{\lambda(i)})\right) \in R_i^2$, $f_j^1/h\left([a_1], \ldots, [a_{\mu(j)}]\right) = \left[f_j^1\left(a_1, \ldots, a_{\mu(j)}\right)\right]$.

$[a_i]$ denotes the equivalence class of a_i.

The quotient system model, $\mathfrak{M}_1/h$, is obviously well-defined. The well-known homomorphism theorem can be considered as a theorem by which an S-homomorphism, $h^\#$, is induced.

Theorem 5.1 [Homomorphism Theorem] *Let $\mathfrak{M}_1$ and $\mathfrak{M}_2$ be system models of the same type, and h a homomorphism of $\mathfrak{M}_1$ onto $\mathfrak{M}_2$. Then a map*

$$h^\# : \mathfrak{M}_1/h \to \mathfrak{M}_2$$

is an S-homomorphism, where $h^\#$ is defined by:

$$h^\#([a]) = h(a) \text{ for any } [a] \in M_1/h$$

$h^\#$ is called the induced homomorphism of h.

A difference between the well-known homomorphism theorem in the usual form in algebra and this theorem is that theorem shows that the induced isomorphism $h^\#$ is an S-homomorphism preserving sentences. One of the reasons why the concept of a homomorphism is important is because an S-homomorphism $h^\#$ can be constructed from a homomorphism h.

5.3.4.2 Morphisms for Models of Different Types

In this section we consider morphisms between system models of different types. Since a homomorphism as seen in the previous section can be defined only for system models of the same type, the concept of a homomorphism is not applicable to the class of system models of different types. We, therefore, introduce a new morphism, called F-morphism, which is a generalization of homomorphism and can be applied to the class of system models of different types as well as of the same type.

In this section we also consider the three cases for the preservation of properties—generators, Σ, and $\mathrm{Th}(\mathfrak{M})$. We will show F-morphism theorem as a theorem corresponding to the homomorphism theorem.

Preservation of Generator: F-Morphism

First we define a basic interpretation function and a basic morphism. An F-morphism is defined recursively by using these functions.

Definition 5.10 [Basic Interpretation Function] Let $\mathfrak{M}_1 = \langle M_1; \{R_i{}^1 | i \in I_1\}, \{f_j{}^1 | j \in J_1\}\rangle$ and $\mathfrak{M}_2 = \langle M_2; \{R_i{}^2 | i \in I_2\}, \{f_j{}^2 | j \in J_2\}\rangle$ be system models of possibly different types.

Then a function Bas of $\mathfrak{L}(\mathfrak{M}_1)$ to the set of formulas of $\mathfrak{L}(\mathfrak{M}_2)$ is said to be a *basic interpretation function of* $\mathfrak{L}(\mathfrak{M}_1)$ to $\mathfrak{L}(\mathfrak{M}_2)$ if the following conditions are satisfied.

1. For every relation symbol $\mathbf{R}_i{}^1 \in \mathfrak{L}(\mathfrak{M}_1)$, $\mathrm{Bas}(\mathbf{R}_i{}^1)$ is a $\lambda_1(i)$-ary formula of $\mathfrak{L}(\mathfrak{M}_1)$;
2. For every function symbol $\mathbf{f}_i{}^1 \in \mathfrak{L}(\mathfrak{M}_1)$, $\mathrm{Bas}(\mathbf{f}_i{}^1)$ is a $(\mu_1(j) + 1)$-ary formula of $\mathfrak{L}(\mathfrak{M}_2)$.

A basic interpretation function associates a formula of the second system model with each symbol of the language of the first one. The association is intended to give "interpretation" of the first system model to the second one. The basic interpretation function works as a meaningful interpretation only when a *basic morphism* with it is defined as follows.

Definition 5.11 [Basic Morphism] Let $\mathfrak{M}_1$ and $\mathfrak{M}_2$ be as above and Bas be a basic interpretation function of $\mathfrak{L}(\mathfrak{M}_1)$ to $\mathfrak{L}(\mathfrak{M}_2)$.

A function I_O of M_1 to M_2 is said to be a *basic morphism* of $\mathfrak{M}_1$ to $\mathfrak{M}_2$ with Bas if the following conditions are satisfied.

1. For every relation symbol $\mathbf{R}_i^1 \in \mathfrak{L}(\mathfrak{M}_1)$ and every assignment ρ, if $\mathfrak{M}_1 \models \mathbf{R}_i^1(\mathbf{x}_1, \ldots, \mathbf{x}_{\lambda_1(i)})[\rho]$, then $\mathfrak{M}_2 \models \mathrm{Bas}(\mathbf{R}_i^1)(\mathbf{x}_1, \ldots, \mathbf{x}_{\lambda_1(i)})[I_O \circ \rho]$, where $I_O \circ \rho$ denotes the composition of I_O andρ;
2. For every function symbol $\mathbf{f}_j^1 \in \mathfrak{L}(\mathfrak{M}_1)$ and every assignment ρ, if $\mathfrak{M}_1 \models \left(\mathbf{f}_j^1(\mathbf{x}_1, \ldots, \mathbf{x}_{\mu_1(j)}) = \mathbf{x}_{\mu_1(j)+1}\right)[\rho]$, then $\mathfrak{M}_2 \models \mathrm{Bas}\left(\mathbf{f}_j^1\right)(\mathbf{x}_1, \ldots, \mathbf{x}_{\mu_1(j)+1})$ $[I_O \circ \rho]$, and satisfies the following condition expressing that $\mathrm{Bas}\left(\mathbf{f}_j^1\right)$ is a function: $\mathfrak{M}_2 \models (\forall \mathbf{x}_1 \ldots \mathbf{x}_{\mu_1(j)})(\exists \mathbf{x}_{\mu_1(j)+1})(\forall \mathbf{y}_{\mu_1(j)+1})\left(\mathrm{Bas}\left(\mathbf{f}_j^1\right)\right.$ $\left.(\mathbf{x}_1, \ldots, \mathbf{x}_{\mu_1(j)}, \mathbf{y}_{\mu_1(j)+1}) \leftrightarrow \mathbf{x}_{\mu_1(j)+1} = \mathbf{y}_{\mu_1(j)+1}\right)$.

$\mathrm{Bas}(\mathbf{R}_i^1)$ and $\mathrm{Bas}\left(\mathbf{f}_j^1\right)$ are called *basic interpretations* of $\mathbf{R}_i^1$ and $\mathbf{f}_j^1$. The identity $=$ is interpreted as the identity of $\mathfrak{L}(\mathfrak{M}_2)$, that is, $\mathrm{Bas}\left(=_{\mathfrak{L}(\mathfrak{M}_1)}\right) \equiv =_{\mathfrak{L}(\mathfrak{M}_2)}$.

Definition 5.12 [F-morphism] Let $\mathfrak{M}_1$ and $\mathfrak{M}_2$ be as in Definition of Basic Morphism]. An *F-morphism*, $I : \mathfrak{M}_1 \to \mathfrak{M}_2$, is a pair of functions $\langle I_{\mathrm{O}}, I_F \rangle$, where I_{O} is a basic morphism of with Bas and I_F is a function of the set of formulas of $\mathfrak{M}_1$ to the set of the formulas of $\mathfrak{M}_2$, which is defined as follows.

For any formula Φ of $\mathfrak{M}_1$

1. If Φ is an atomic formula of the form $\mathbf{f_j}(u_1, \ldots, u_{\mu(j)}) = \mathbf{x}$ or $\mathbf{x} = \mathbf{f_j}(u_1, \ldots, u_{\mu(j)})$, then

$$I_{\mathrm{F}}(\Phi) = \begin{cases} \mathrm{Bas}\left(\mathbf{f_j}\right)\left(u_1, \ldots, u_{\mu(j)}, \mathbf{x}\right), & \text{if } T(\Phi) \text{ is the empty set} \\ \left(\exists \mathbf{x_{k_1}} \ldots \mathbf{x_{k_m}}\right) & \\ \times \left(\mathrm{Bas}\left(\mathbf{f_j}\right)\left(\mathbf{x_1}, \ldots, \mathbf{x_{\mu(j)}}, \mathbf{x}\right)\right. & \\ \times \wedge \left(\wedge \left(I_F\left(\mathbf{x_{k_i}} = u_{k_i}\right) | u_{k_i}\right.\right. & \\ \in T(\Phi)))), & \text{if } T(\Phi) = \left\{u_{k_1}, \ldots, u_{k_m}\right\} \end{cases},$$

where every $\mathbf{x_{k_p}}$ is a variable not occurring in Φ, and $\mathbf{x_i}$ is u_i for $u_i \notin T(\Phi)$.

2. If Φ is an atomic formula $\mathbf{P}(t_1, \ldots, t_n)$ other than of the form in (1), then

$$I_{\mathrm{F}}(\Phi) = \begin{cases} \mathrm{Bas}(\mathbf{P})(u_1, \ldots, u_n), & \text{if } T(\Phi) \text{ is the empty set} \\ \left(\exists \mathbf{x_{k_1}} \ldots \mathbf{x_{k_m}}\right) & \\ \times (\mathrm{Bas}(\mathbf{P})(\mathbf{x_1}, \ldots, \mathbf{x}_n) & \\ \wedge \left(\wedge \left(I_F\left(\mathbf{x_{k_i}} = u_{k_i}\right) | u_{k_i}\right.\right. & \\ \in T(\Phi)))), & \text{if } T(\Phi) = \left\{u_{k_1}, \ldots, u_{k_m}\right\} \end{cases},$$

where every $\mathbf{x_{k_p}}$ is a variable not occurring in $\mathbf{P}$, and $\mathbf{x_i}$ is u_i for $u_i \notin T(\Phi)$.

3. Otherwise,

$$I_{\mathrm{F}}(\neg\Phi) = \neg(I_{\mathrm{F}}(\Phi)),$$

$$I_{\mathrm{F}}(\Phi_1 \wedge \Phi_2) = (I_{\mathrm{F}}(\Phi_1)) \wedge (I_{\mathrm{F}}(\Phi_2)),$$

$$I_{\mathrm{F}}(\forall \mathbf{x} \Phi) = (\forall \mathbf{x})(I_{\mathrm{F}}(\Phi)).$$

Since $T(\Phi)$ eventually becomes empty, I_{F} is well-defined.

Let us consider an example of F-morphisms.

Example 5.4 We can define an F-morphism of $(N; \leq)$ to $(N; +)$, where N is the set of natural numbers, $\leq$ the linear ordering on N and $+$ addition. If we define Bas by $\mathrm{Bas}(\leq) = (\exists \mathbf{z})(\mathbf{x} + \mathbf{z} = \mathbf{y})$ and I_{O} by the identity, then $\langle I_{\mathrm{O}}, I_{\mathrm{F}} \rangle$ with Bas is an F-morphism of $(N; \leq)$ to $(N; +)$.

We should notice that the function I_{F} is "automatically" defined according to the definition if I_{O} with Bas is already defined. Furthermore it is clear that if $(N; \leq) \vDash \phi$, then$(N; +) \vDash I_F(\phi)$. For example, let ϕ be a sentence $(\forall \mathbf{xyz})(\mathbf{x} \leq \mathbf{y} \wedge \mathbf{y} \leq \mathbf{z} \to \mathbf{x} \leq \mathbf{z})$.

Then

$$\begin{aligned} I_{\mathrm{F}}(\phi) &= I_{\mathrm{F}}\left((\forall \mathbf{xyz})\left(\mathbf{x} \le \mathbf{y} \wedge \mathbf{y} \le \mathbf{z} \to \mathbf{x} \le \mathbf{z}\right)\right) \\ &= (\forall \mathbf{xyz})\left(I_{\mathrm{F}}(\mathbf{x} \le \mathbf{y}) \wedge I_{\mathrm{F}}(\mathbf{y} \le \mathbf{z}) \to I_{\mathrm{F}}(\mathbf{x} \le \mathbf{z})\right) \\ &= (\forall \mathbf{xyz})\left((\exists \mathbf{z_1})(\mathbf{x} + \mathbf{z_1} = \mathbf{y}) \wedge (\exists \mathbf{z_2})(\mathbf{x} + \mathbf{z_2} = \mathbf{y}) \to (\exists \mathbf{z_3})(\mathbf{x} + \mathbf{z_3} = \mathbf{y})\right). \end{aligned}$$

So $(N; +) \vDash I_{\mathrm{F}}(\phi)$.

Example 5.5 [Automaton] A Moore type automaton, $M_{\mathrm{r}} = (A, B, C, \phi_{\mathrm{r}}, \lambda_{\mathrm{r}})$ with A, B, C: finite sets, $\phi_{\mathrm{r}} : C \times A \to C$ and $\lambda_{\mathrm{r}} : C \to B$, is regarded as equivalent to a Mealy type automaton, $M_{\mathrm{e}} = (A, B, C, \phi_{\mathrm{e}}, \mu_{\mathrm{e}})$ with A, B, C: finite sets, $\phi_{\mathrm{e}} : C \times A \to C$ and $\mu_{\mathrm{e}} : C \times A \to B$. Let

$$\text{Moore} = \left\langle A \cup B \cup C;\ A,\ B,\ C,\ \hat{\phi}_r,\ \hat{\lambda}_r \right\rangle$$

and

$$\text{Mealy} = \left\langle A \cup B \cup C;\ A,\ B,\ C,\ \hat{\phi}_e,\ \hat{\mu}_e \right\rangle,$$

where A, B, C: unary relations, $\hat{\phi}_{\mathrm{r}}$, $\hat{\lambda}_{\mathrm{r}}$, $\hat{\phi}_{\mathrm{e}}$, $\hat{\mu}_{\mathrm{e}}$: arbitrary extensions of ϕ_{r}, λ_{r}, ϕ_{e}, μ_{e} respectively, and $\hat{\phi}_{\mathrm{r}} = \hat{\phi}_{\mathrm{e}}$, $\hat{\mu}_{\mathrm{e}}(c, a) = \hat{\lambda}_{\mathrm{r}}\left(\hat{\phi}_{\mathrm{r}}(c, a)\right)$ for any c, $a \in A \cup B \cup C$.

Then we can define an F-morphism of Mealy to Moore as follows.

$$\left. I_{\mathrm{O}} : \text{the identity} \right\},$$

$$\text{Bas}\left(\hat{\phi}_{\mathbf{e}}\right) = \hat{\phi}_{\mathbf{r}}(\mathbf{x}, \mathbf{y}, \mathbf{z}),$$

$$\text{Bas}\left(\hat{\boldsymbol{\mu}}_{\mathbf{e}}\right) = \left(\hat{\boldsymbol{\lambda}}_{\mathbf{r}}\left(\hat{\phi}_{\mathbf{r}}(\mathbf{x}, \mathbf{y})\right) = \mathbf{z}\right),$$

$$\text{Bas}(\mathbf{A}) = \mathbf{A}, \quad \text{Bas}(\mathbf{B}) = \mathbf{B}, \quad \text{Bas}(\mathbf{C}) = \mathbf{C}.$$

These definitions obviously satisfy the conditions of basic morphisms. The following corollary shows that an F-morphism is an extension of a homomorphism.

Corollary 5.1 *Let $\mathfrak{M}_1$ and $\mathfrak{M}_2$ be system models of the same type, and h a homomorphism of $\mathfrak{M}_1$ to $\mathfrak{M}_2$. Then h is a basic morphism of $\mathfrak{M}_1$ to $\mathfrak{M}_2$, and $\langle h, I_{\mathrm{F}} \rangle$ is an F-morphism of $\mathfrak{M}_1$ to $\mathfrak{M}_2$, where I_{F} is a function uniquely determined by h in Definition of F-morphism.*

From the definition of basic morphism we can see that an F-morphism preserves generators. Notice that it is necessary to give an interpretation I_{F} of the generators in defining an F-morphism, while in the case of a homomorphism I_{F} is trivially defined, and is not explicitly given in usual algebra.

Preservation of Σ: Σ_{F}-Morphism

In this section we will define an F-morphism preserving axioms Σ as a Σ_{F}-morphism. By a Σ_{F}-morphism, Σ is preserved in an image of an F-morphism. First we define the image of a basic morphism.

Let $\mathfrak{M}_1$ and $\mathfrak{M}_2$ be system models of possibly different types. Let $I = \langle I_{\mathrm{O}}, I_{\mathrm{F}} \rangle : \mathfrak{M}_1 \to \mathfrak{M}_2$ be an F-morphism. Then *the image of a basic morphism*, written $I(\mathfrak{M}_1)$, is defined by:

$$\begin{aligned} I(\mathfrak{M}_1) = \Big\langle I_{\mathrm{O}}(M_1);\ &\Big\{R_i^2 \cap I_{\mathrm{O}}(M_1)^{\lambda_2(i)} | i \in I_2\Big\}, \\ &\Big\{f_j^2 \parallel I_{\mathrm{O}}(M_1)^{\mu_2(j)} | f_j^2\left(a_1, \ldots, a_{\mu_2(j)}\right) \in I_{\mathrm{O}}(M_1) \\ &\times \text{ for any } a_1, \ldots, a_{\mu_2(j)} \in I_{\mathrm{O}}(M_1),\, j \in J_2\Big\}\Big\rangle. \end{aligned}$$

Notice that if there exists a j such that $f_j^2(a_1, \ldots, a_{\mu_2(j)}) \notin I_{\mathrm{O}} M_1$ for $a_1, \ldots, a_{\mu_2(j)} \in I_{\mathrm{O}}(M_1)$, then $I(\mathfrak{M}_1)$ is of a different type from $\mathfrak{M}_2$. An F-morphism is called an *onto F-morphism* if its basic morphism is onto.

Definition 5.13 [Σ_{F}-Morphism] Let $\mathfrak{M}_1$ and $\mathfrak{M}_2$ be system models and $I = \langle I_{\mathrm{O}}, I_{\mathrm{F}} \rangle$ an F-morphism of $\mathfrak{M}_1$ to $\mathfrak{M}_2$. Suppose$\mathfrak{M}_1 \models \Sigma$. Then I is called a Σ_{F}*-morphism* of $\mathfrak{M}_1$ to $\mathfrak{M}_2$ if

$$I(\mathfrak{M}_1) \models I_{\mathrm{F}}(\Sigma),$$

where $I_{\mathrm{F}}(\Sigma) = \{I_{\mathrm{F}}(\Phi) | \Phi \in \Sigma\}$.

A Σ_{F}-morphism is a kind of extension of a Σ_{F}-homomorphism. The following corollary says that a Σ_{F}-morphism between system models of the same type accords with a Σ-homomorphism.

Corollary 5.2 *Let $\mathfrak{M}_1$ and $\mathfrak{M}_2$ be system models of the same type. Let $h : \mathfrak{M}_1 \to \mathfrak{M}_2$ be a homomorphism of $\mathfrak{M}_1$ to $\mathfrak{M}_2$. Suppose$\mathfrak{M}_1, \mathfrak{M}_2 \models \Sigma$. Then h is a Σ-homomorphism of $\mathfrak{M}_1$ to $\mathfrak{M}_2$ if and only if $\langle h, I_{\mathrm{F}} \rangle$ is a Σ_{F}-morphism of $\mathfrak{M}_1$ to $\mathfrak{M}_2$.*

Preservation of Th $(\mathfrak{M})$: S_{F}-Morphism

In this section we define a morphism between system models of different types, which preserves Th$(\mathfrak{M})$ of a system model. Furthermore we will show the F-morphism theorem corresponding to the homomorphism theorem in the case of the same type. The F-morphism theorem gives a relationship between an F-morphism and an S_F-morphism. Unless mentioned explicitly, in the sequel let $\mathfrak{M}_1$, $\mathfrak{M}_2$ be system models and $I = \langle I_O, I_F \rangle$ an F-morphism of $\mathfrak{M}_1$ to $\mathfrak{M}_2$.

Definition 5.14 [S_F-Morphism] An F-morphism I is called an *S_F-morphism* of $\mathfrak{M}_1$ to $\mathfrak{M}_2$ if for any sentence Φ of $\mathfrak{L}(\mathfrak{M}_1)$, $\mathfrak{M}_1 \models \Phi$ if and only if $I(\mathfrak{M}_1) \models I_F(\Phi)$.

From the definition an S_F-morphism is a Σ_F-morphism. In general, we can check whether an F-morphism is an S_F-morphism by the way based on a structural induction on sentences. However in most cases, they are more than routine tasks.

For an onto F-morphism I, we define t*he quotient system model with respect to I* by:

$$\mathfrak{M}_1/I = \left\langle M_1/I_O : \left\{ R_i^1/I | i \in I_1 \right\}, \left\{ f_j^1/I | j \in J_1 \right\} \right\rangle,$$

where M_1/I_O is the partitioned set of M_1 by the equivalence relation $\equiv_{I_O}$ defined by:

$$a \equiv_{I_O} b \quad \text{if and only if} \quad I_O(a) = I_O(b) \quad \text{for any} \quad a, b \in \mathbf{M_1},$$

$$R_i^1/I\left([a_1], \ldots, [a_{\lambda_1(i)}]\right) \quad \text{if and only if} \quad \mathrm{Bas}\left(R_i^1\right)\left(I_O(a_1), \ldots, I_O\left(a_{\lambda_1(i)}\right)\right),$$

$$f_j^1/I\left([a_1], \ldots, [a_{\mu_1(j)}]\right) = \left[f_j^1\left(a_1, \ldots, a_{\mu_1(j)}\right)\right],$$

where $[a]$ represents an equivalence class in M_1/I_O by $\equiv_{I_O}$.

The above definition of f_j^1/I is well-defined. Indeed, suppose $I_O(a_1) = I_O(b_1), \ldots, I_O\left(a_{\mu_1(j)}\right) = I_O\left(b_{\mu_1(j)}\right)$ and $f_j^1\left(a_1, \ldots, a_{\mu_1(j)}\right) = c_1$, $f_j^1\left(b_1, \ldots, b_{\mu_1(j)}\right) = c_2$ hold in $\mathfrak{M}_1$. Then since I is an F-morphism, $\mathrm{Bas}\left(\mathbf{f_j^1}\right)\left(I_O(a_1), \ldots, I_O\left(a_{\mu_1(j)}\right), I_O(c_1)\right)$ and $\mathrm{Bas}\left(\mathbf{f_j^1}\right)\left(I_O(b_1), \ldots, I_O\left(b_{\mu_1(j)}\right), I_O(c_1)\right)$ hold in $\mathfrak{M}_2$. Since $\mathrm{Bas}\left(\mathbf{f_j^1}\right)$ is a function from the definition of basic morphism, we have $I_O(c_1) = I_O(c_2)$.

The following is one of the main theorems about F-morphisms.

Theorem 5.2 [F-Morphism Theorem] *Let $I : \mathfrak{M}_1 \to \mathfrak{M}_2$ be an onto F-morphism. Then*

$$I_O^{\#} : \mathfrak{M}_1/I \to \mathfrak{M}_2$$

is a one-to-one basic morphism, furthermore

$$I^{\#} = \left\langle I_O^{\#}, I_F^{\#} \right\rangle$$

is an S_F-morphism, where $I_{\mathrm{O}}^{\#}$ is defined by

$$I_{\mathrm{O}}^{\#}([a]) = I_{\mathrm{O}}(a) \quad \text{for} \quad [a] \in M_1/I_{\mathrm{O}}$$

and the basic interpretations are

$$\mathrm{Bas}^{\#}\left(\mathbf{R}_{\mathbf{i}}^{\mathbf{1}}/I\right) = \mathrm{Bas}\left(\mathbf{R}_{\mathbf{i}}^{\mathbf{1}}\right) \quad \text{for} \ \ i \in I_1,$$
$$\mathrm{Bas}^{\#}\left(\mathbf{f}_{\mathbf{j}}^{\mathbf{1}}/I\right) = \mathrm{Bas}\left(\mathbf{f}_{\mathbf{j}}^{\mathbf{1}}\right) \quad \text{for} \ \ i \in J_1.$$

$I^{\#}$ is called *the induced F-morphism* of *I*.

F-morphism enhances the concept of "isomorphism" between system models to those of different types, which means that the two models described in "different languages" are isomorphic in terms of F-morphism.

5.3.4.3 Application of F-Morphisms

A typical way to apply the F-morphism concept to concrete system models would be to construct an F-morphism between them. In this section, as an application of F-morphisms, we construct an F-morphism of a given finite automaton structure to a Petri net structure, and show the equivalence between a finite automaton and a Petri net. The equivalence means here that we can show that there is a Petri net that preserves all the properties of a given finite automaton. It is well known that Petri nets can represent finite automata (Peterson, 1981). The emphasis in this section, however, is on that the use of the F-morphism concept in considering the equivalence between a finite automaton and a Petri net reveals that each property of the finite automaton precisely (in a formal way) corresponds to some property of the Petri net, and the first-order sentences satisfied in the finite automaton are all preserved in the Petri net as corresponding sentences transformed by an F-morphism. Thus the F-morphism concept provides a formal meaning of "equivalence" between system models of different types, while the judgment of the equivalence "without F-morphisms" would depend fully on the intuition of a modeler constructing the correspondence between them.

Equivalence Between a Finite Automaton and a Petri Net

In this section we construct an F-morphism of a given finite automaton structure to a Petri net structure, and show that all the properties holding in the finite automaton also hold in the Petri net.

Definition 5.15 A finite automaton structure FA is the following system model.

$$\mathrm{FA} = \langle A \cup B \cup C;\ A, B, C, \phi, \rho \rangle\ ,$$

where

A, B, C: unary relations
ρ: binary functions such that

$\phi(a, b) \in C$, if $a \in C$ and $b \in A$
$\phi(a, b) = a$, otherwise;
and $\rho(a, b) \in B$, if $a \in C$ and $b \in A$
$\rho(a, b) = a$, otherwise.

The conditions on $a \notin C$ or $b \notin A$ for ϕ and ρ are imposed only to make the functions ϕ and ρ total, since the first-order language we use does not allow partial functions. However, since we will restrict the sentences to the extent as defined later, when we describe the properties of system models, we can regard ϕ and ρ intrinsically as $\phi : C \times A \to C$ and $\rho : C \times A \to B$.

Definition 5.16 [Petri Net Structure] A Petri net structure PN is the following system model.

$$\mathrm{PN} = \left\langle P \cup T \cup N;\ P, T, I, O, \hat{N} \right\rangle,$$

where

P, T: unary relations
N: the set of natural numbers
$\hat{N}$: the set of constants corresponding to N
$I, O \subset P \times T \times N$

P denotes the set of places and T the set of transitions. $I(p, t, n)$ means that there are n arcs from the place p to the transition t. $O(p, t, n)$ means that there are n arcs from the transition t to the place p.

There are some ways to construct PN that is considered to have an equivalent structure to FA (Peterson, 1981). Here following Peterson with some modification, we define PN considered as equivalent to FA. Then our aim is to construct an F-morphism between FA and PN, and to show that the constructed PN preserves all the properties satisfied in FA.

Definition 5.17 Given a finite automaton structure FA. We define the corresponding Petri net structure PN as follows.

$$\mathrm{PN} = \left\langle P \cup T \cup N;\ P, T, I, O, \hat{N} \right\rangle,$$

where

$P = C \cup A \cup B$;
$T = \{t_i | \, i \in (C \times A) \cup A \cup B\}$;
$I = I_1 \cup I_0$,

where

$I_1 = \{(p, t_i, 1) | \, i = (c, a) \in C \times A \wedge (p = c \vee p = a)\} \cup \{(p, t_i, 1) | i = p \in B\}$,
$I_0 = \{(p, t_i, 0) | (p, t_i, 1) \notin I_1, p \in P, t_i \in T\}$,
$O = O_1 \cup O_0$,

where

$O_1 = \{(p, t_i, 1) | \, i = (c, a) \in C \times A \wedge (p = \phi(c, a) \vee \rho(c, a))\} \cup \{(p, t_i, 1) | \, i = p \in A\}$,
$O_0 = \{(p, t_i, 0) | (p, t_i, 1) \notin O_1, p \in P, t_i \in T\}$.

Definition 5.18 Let FA and PN be as in the above definitions, respectively. An F-morphism $I = \langle I_{\mathrm{O}}, I_{\mathrm{F}} \rangle$: FA $\to$ PN is defined as follows.

I_{O}: the inclusion map;
$I_{\mathrm{F}}(\mathbf{A}(\mathbf{x})) = (\mathbf{P}(\mathbf{x}) \wedge (\exists \mathbf{t} \in \mathbf{T})((\forall \mathbf{p} \in \mathbf{P})(\mathbf{I}(\mathbf{p}, \mathbf{t}, \mathbf{0}) \wedge \mathbf{O}(\mathbf{x}, \mathbf{t}, \mathbf{1}))))$;
$I_{\mathrm{F}}(\mathbf{B}(\mathbf{x})) = (\mathbf{P}(\mathbf{x}) \wedge (\exists \mathbf{t} \in \mathbf{T})((\forall \mathbf{p} \in \mathbf{P})(\mathbf{O}(\mathbf{p}, \mathbf{t}, \mathbf{0}) \wedge \mathbf{I}(\mathbf{x}, \mathbf{t}, \mathbf{1}))))$;
$I_{\mathrm{F}}(\mathbf{C}(\mathbf{x})) = (\mathbf{P}(\mathbf{x}) \wedge \neg I_{\mathrm{F}}(\mathbf{A}(\mathbf{x})) \wedge \neg I_{\mathrm{F}}(\mathbf{B}(\mathbf{x})))$;
$I_{\mathrm{F}}(\phi(\mathbf{x}, \mathbf{y}) = \mathbf{z}) = \Big((I_{\mathrm{F}}(\mathbf{C}(\mathbf{x})) \wedge I_{\mathrm{F}}(\mathbf{A}(\mathbf{y})) \to (\exists \mathbf{t} \in \mathbf{T})(\mathbf{I}(\mathbf{x}, \mathbf{t}, \mathbf{1}) \wedge \mathbf{I}(\mathbf{y}, \mathbf{t}, \mathbf{1})$
$\wedge \mathbf{O}(\mathbf{z}, \mathbf{t}, \mathbf{1}) \wedge I_{\mathrm{F}}(\mathbf{C}(\mathbf{z})))) \wedge (\neg I_{\mathrm{F}}(\mathbf{C}(\mathbf{x})) \vee \neg I_{\mathrm{F}}(\mathbf{A}(\mathbf{y})) \to \mathbf{z} = \mathbf{x}) \Big)$;
$I_{\mathrm{F}}(\rho(\mathbf{x}, \mathbf{y}) = \mathbf{z}) = \Big((I_{\mathrm{F}}(\mathbf{C}(\mathbf{x})) \wedge I_{\mathrm{F}}(\mathbf{A}(\mathbf{y})) \to (\exists \mathbf{t} \in \mathbf{T})(\mathbf{I}(\mathbf{x}, \mathbf{t}, \mathbf{1}) \wedge \mathbf{I}(\mathbf{y}, \mathbf{t}, \mathbf{1})$
$\wedge \mathbf{O}(\mathbf{z}, \mathbf{t}, \mathbf{1}) \wedge I_{\mathrm{F}}(\mathbf{B}(\mathbf{z})))) \wedge (\neg I_{\mathrm{F}}(\mathbf{C}(\mathbf{x})) \vee \neg I_{\mathrm{F}}(\mathbf{A}(\mathbf{y})) \to \mathbf{z} = \mathbf{x}) \Big)$.

This definition clearly satisfies the condition required for F-morphisms. Also we can see, as the following lemmas show, that the image of the above F-morphism preserves the structure of FA.

The following theorem shows a typical type of equivalence between PN and FA.

Theorem 5.3 [Equivalence of the Structures of Finite Automaton and Petri Net] Let $I = \langle I_{\mathrm{O}}, I_{\mathrm{F}} \rangle$ be the F-morphism defined as in the above Definition. Then for any many-sorted sentence Φ of $\mathfrak{L}$(FA), FA $\models \Phi$ if and only if PN $\models I_{\mathrm{F}}(\Phi)$.

This theorem implies that the structure of FA is embedded in PN constructed in Definition, and all the properties of FA are preserved there. We should notice that the dynamic behavior of PN by the transition of marking is implied by the relation O of PN, which can also represent the firing of the transitions.

5.4 Structure and Adaptation

5.4.1 Implications of Gödel's Incompleteness Theorem for Structural Change

The definition of the structure of a system model implies that there are two ways to represent a change of the structure: the changes of the symbols $\mathfrak{L}$ and of the axioms Σ. Here we deal with the change of the axioms. Then we can naturally say that a system has changed its structure from Σ to $\Sigma^{'}$, if $\Sigma^{'} = \Sigma + \{\psi\}$, where ψ nor $\neg\psi$cannot be derived from Σ, i.e.,$\Sigma \vdash \psi$ nor $\Sigma \vdash \neg\psi$, under some deduction system.

This new property ψ cannot be recognized in the old structure Σ. Classical deduction systems such as first-order logic cannot deal with this situation effectively. Also in ordinal systems theory based on set theory without logical language and deduction system we hardly develop comprehensive consideration on this matter. We need to construct a meta-framework in which the structure of a system ($\mathfrak{L}$; Σ) can be referred an "object" and to extend the concept of system model by adding extra domain to it.

Gödel proved incompleteness theorem that there is a sentence that cannot be inferred from the logical system including the primitive recursive arithmetic (PRA) such as Peano's axioms. From the Gödel's incompleteness theorem, we see that there is a formula such that PRA $\vdash \phi \leftrightarrow \neg \Box \phi$. The symbol $\Box$ expresses a modal operator and the formula $\Box\phi$ is interpreted to be "ϕ is provable" (Symoryński, 1985). Then $\neg \Box \phi$ means that ϕ is not provable. Furthermore $\phi \leftrightarrow \neg \Box \phi$ means that "I am not provable." This formula shows a self-referential sentence. Hence if the structure of a system includes PRA, the system has a property ϕ such that ϕ cannot be implied from the structure, and nor identified as behavior of the system.

As stated in the previous paragraph, if the structure of a system "moves" to a new structure that includes the old structure and the above self-referential sentence ϕ that is never proved from the old one, the system actually "change" its structure: the move from the structure Σ to $\Sigma^{'} = \Sigma + \{\phi\}$ represents a structural change of the system.

5.4.2 Adaptation in Social System: Agent-Based Organizational Cybernetics

There are two kinds of adaptive behavior of a system: first-order adaptive behavior and second-order adaptive behavior. The first-order adaptive behavior seeks to make the system stable by regulating the gap between the goal of the system and the systems output observed. It is realized as a negative feedback mechanism, which can be formulated in the structure of the system, i.e., in LAST. The second-order adaptive behavior requires to change the field of systems behavior, which can be represented as the concept of positive feedback or second-order cybernetics. The second-order adaptive behavior cannot be described as any properties derived from the structure of the system. As seen in the above, the self-referential sentence plays a key role in the adaptation by the structural change. We need to introduce some new conceptual devices to describe models of the second-order adaptive behavior: agent, internal model, and organizational learning.

This section focuses on adaptation in social systems and provides the theoretical framework for describing it. Here a comprehensively hybrid model is introduced, which combines conventional organizational cybernetic framework and computational organization theoretic approach, especially agent-based computational learning model. The framework of organizational cybernetics includes no agent concept innately, but originally aims to contribute to the diagnosis of organizational failure based on Ashby's law of variety (Espejo et al., 1996). On the other hand, computational organization theoretic approach contains agent-based task resolution processes in detail operational manner, but describes only a "flat" organization that has no hierarchical relationship between subsystems. The hybrid model presented here is comprehensive in understanding organizational learning in the sense that the learning process includes essentially the following steps: each agent resolves tasks in every functional layer in an organization; the results of the resolutions of tasks are unified to be organizational output performance; the organizational performance should be evaluated from environment; each agent change its internal model based on the evaluation results.

We call our newly developed approach Agent-based Organizational Cybernetics (AOC) (Takahashi, 2006). An organization considered in AOC is formulated to have four functional layers defined in organizational cybernetics: process, coordination, adaptation, and self-organization. The organizational cybernetic model has originally no concept of an agent. AOC introduces the concepts of agent and communication process among agents into each functional layer of an organizational cybernetic model. An agent is characterized by individual situatedness and internal model principle. The agent is defined as an autonomous decision-maker who constructs individually its internal model to describe its recognition of the situation surrounding it.

The basic features of AOC can be listed below.

1. Interaction between environment and decision-makers (from organizational cybernetic viewpoint).
2. An autonomous decision-maker makes a decision according to his decision principle.
3. An organization is structured in a multi-layer hierarchical form with some functional subsystems.
4. In each layer of the hierarchy, some agent groups are involved and interact one another.
5. Every agent has its own internal model that describes the situation surrounding the agent (called individual situatedness).

6. Every agent can learn its internal model and the organization can learn by sharing agents' internal models. The process of learning represents single- and double-loop learnings in organizational learning.

AOC allows us to deal with organizational problems such as organizational learning in essentially operational manners. Results analyzed using AOC could suggest how we should effectively and operationally manage complex problems on the organization concerned. The principal target of AOC is to provide design criteria of prescription, especially which has not yet been validated in actual situations, on how an organization of concern should make a decision and take an action to adapt itself to a dynamically changing environment. AOC can also provide an effective way to evolve new design of functions working in an organization by re-combination of subsystems.

5.4.3 Components of Organizational Learning

The concept of organizational learning we use for our framework has similar aspects to the Argyris' concept that individual learning processes are innately connected to organizational learning process. Our framework explicitly distinguishes individual levels of learning and organizational ones, and also does the levels of single-loop learning and double-loop one. We can see the explicit distinctions of the four types of learning loops.

The distinction by Argyris of single-loop learning and double-loop one is originated, as Argyris stated (Argyris & Schön, 1996), from similar notions in cybernetics developed by Ashby (1960). Based on organizational cybernetic approach, Espejo has provided a basis for the way how to apply the double-loop learning notion and process to actual organizations.

In AOC the concept of organizational learning, especially the learning-loop processes are realized in operational ways that each agent evaluates and revises its internal model. By actually implementing mechanisms of organizational learning in agents, the micro-macro problems can be explored effectively so as to tackle complex organizational systems.

The essential elements of organizational learning in AOC are the four learning loops: individual−/organizational- and single−/double-loop learning. AOC implements the learning processes as evaluating, revising, and sharing processes of internal models possessed by agents.

1. Individual single-loop learning.

   An agent builds its internal model to describe the environmental structure and the problem situation recognized, which includes some decision variables and decision criteria. The agent uses its internal model to optimize the decision variables. This learning does not enhance any ability to make organizational decisions.
2. Organizational single-loop learning.

To achieve the given organizational goal, subgoals are specified to agents in inferior subsystems The values of the individual decision variables, which must be the results of the individual single-loop learning, are unified by the organization. The organization makes a decision based on the unified results.

3. Individual double-loop learning.

   Each agent evaluates its internal model, based on the results of the decision performed just before. Then the internal model is revised.

   This process of revising internal models by agents can be implemented effectively by using genetic algorithm (GA). The evaluation is defined by a fitness function that indicates what kind of information is available and how it should be utilized for the evaluation.

   After the evaluations of the internal models, applying GA operators such as selection, crossover, and mutation, the internal models are revised for the subsequent decisions.

4. Organizational double-loop learning.

   As the result of "good" individual double-loop learning, the agents share in the organization their good internal models that provide them with better decision capability and allow them to keep the organization viable.

5.5 Basis of Agent-Based Organizational Cybernetics

5.5.1 Hierarchical Organization Model in AOC

Combining the hierarchical model of organizational cybernetics and the agent model in computational organization theory, AOC consists of two basic models: hierarchical organization model and situated agent model.

A hierarchical organization model is a multi-layer system that has basically adaptive, coordination, and operational levels (Fig. 5.1).

In AOC the function of each level is realized by a group of agents. Every agent belongs to one of the subsystems of the hierarchy. Each subsystem seeks a possibly different goal from other subsystems. Hence an agent is conducted based on the goal of the subsystem of which the agent is a member.

The adaptive level is composed of intelligence and institutional functions. In this level, based on environmental information observed by the intelligence function, the organization creates policy or strategy that could achieve a given organizational goal. If the organizational goal is recreated, the organization would go to a self-organization phase.

The coordination level has a function that determines coordination variables to control inferior subsystems in a decentralized manner. Coordination principles, which define how to coordinate the inferior subsystems, are essential to achieve a coordination goal.

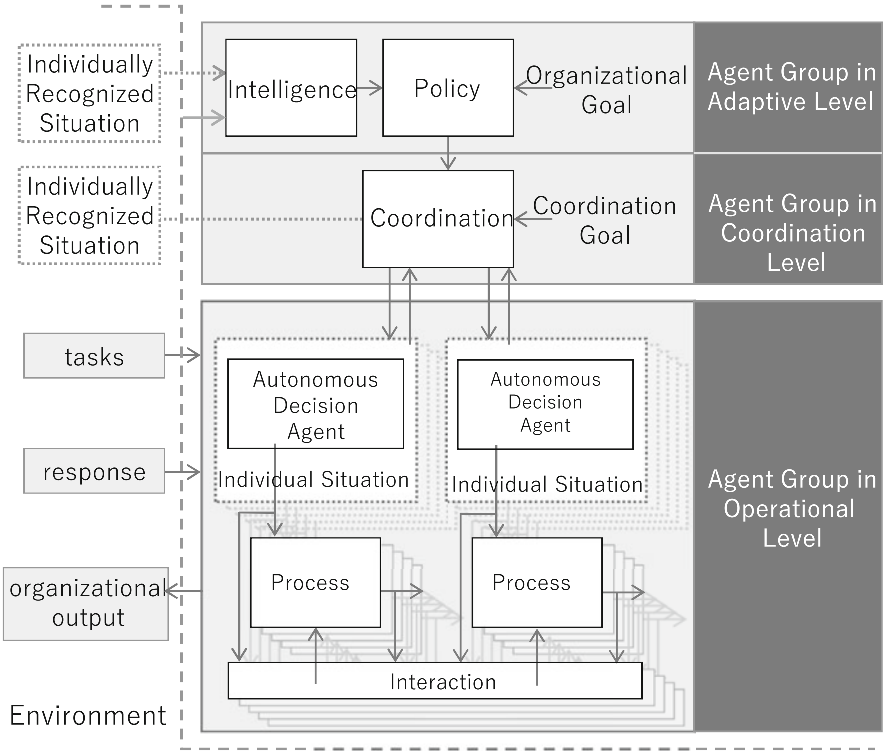
Individually Recognized Situation
Intelligence
Policy
Organizational Goal
Agent Group in Adaptive Level
Individually Recognized Situation
Coordination
Coordination Goal
Agent Group in Coordination Level
tasks
response
Autonomous Decision Agent
Individual Situation
Autonomous Decision Agent
Individual Situation
Agent Group in Operational Level
organizational output
Process
Process
Interaction
Environment

Fig. 5.1 Basic hierarchical model in AOC

In the operational level, agents determine decision variables autonomously, each of whom aims to optimize the process assigned to him. The processes interact with one another. The optimization process is given as a task resolution one, the result of which is reported to the superior subsystem, i.e., coordination level.

Computational Organization Theory has focused so far on models of the operational level. AOC formulates the operational level as a layer of the hierarchical subsystems of the overall organization model.

5.5.2 *Situated Agent Model in AOC as Autonomous Decision-Maker*

An agent concept in AOC as an autonomous decision-maker has basically the following features (Fig. 5.2).

1. An agent recognizes a process as a target of its decision-making, and builds its model internally, which is called an internal model.

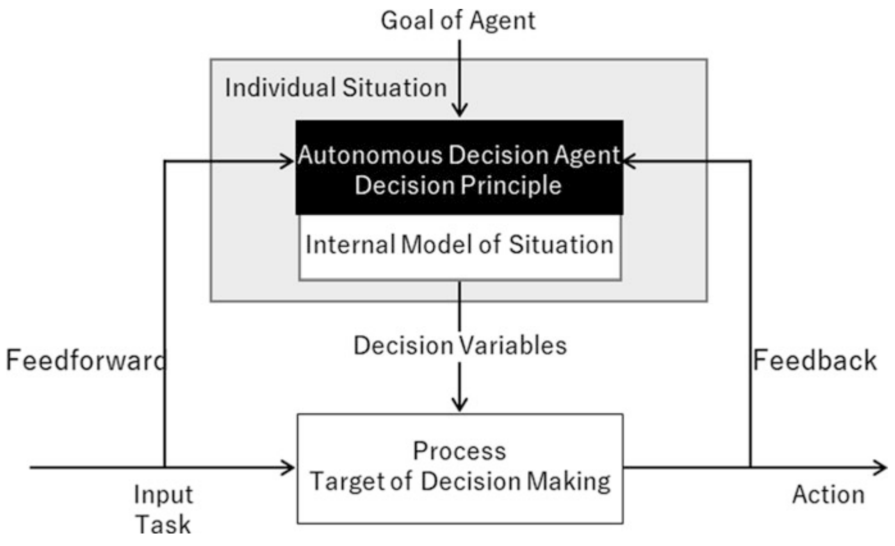

Fig. 5.2 Situated agent model as decision-maker

The internal model describes the behavior of the process and external inputs from environment, which can include agent's recognition of the surrounding situation. An agent has its own internal model to describe the surrounding situation. Every agent is considered to be involved in its situation that is individually perceived by that agent. We call it individual situatedness.

2. An agent applies a decision principle to a problem concerning the process so that the agent evaluates options or alternatives to solve the problem.

The decision principle represents a criterion for preference ordering of the alternatives. It can be formally defined as a function from the class of problems to be solved into the ordering structures of preferences.

An eminent feature of AOC is to deal with, in an operational manner, micro-macro link problems such as a problem of the relationship between individual learning process of each agent and organizational one. An agent is typically a member of one of the autonomous decision groups defined in the multi-layers of an organization. The overall environment can be recognized as interpreted information from shared internal models of individually perceived situations.

5.5.3 Typical Internal Models

We here consider typical internal models in each hierarchical level.

A typical internal model that an agent in adaptive level has its recognition of the environmental structure, especially the recognition how the environment makes responses to an agent and the organization in the form of cost-profit function.

Another typical internal model in this level is the decision principle that an agent uses to make its decision. An internal model used in coordination level is the recognition of the process in which assigned tasks should be actually resolved.

In the operational level, a typical internal model can be how an agent recognizes tasks to be resolved as well as the task resolution process itself.

The point in considering learning problems in an organization is how each agent should evaluate its internal model, based on which the agent revises its internal model, i.e., the recognition of its individual situation and shares it among agents.

5.6 Conclusion: Beyond LAST

Logical Approach to Systems Theory (LAST) provides a way to investigate the similarity and structure of system models in type-free representation. Hence LAST clarifies the similarity of systems models in different representations as "isomorphic relation" with F-morphism. LAST explicitly gives the concept of the structure of a system model as a pair of the description language and the axioms characterizing the system model. Then LAST clarifies the distinction of what type of properties of systems can be described in the theory and what type of them cannot. Adaptation, which has been a central concern of systems theory, is a typical type of properties logical approaches hardly describe. One possibility of describing adaptation in logical frameworks would be to exploit modal logic, which could express the self-referential concept. Then the concepts of agent, internal model, and organizational learning would provide a breakthrough way to describe the adaptation of systems.

Agent-based organizational cybernetics involves as its significant part an adaptive mechanism in which agents revise intrinsically their own internal models and then share them with the other agents of the system through the organizational learning process the system constructs. The adaptive mechanism has been developed using many types of evolutionary computation algorithms such as genetic algorithm and so on. Tons of research results concerning them have been accumulated. Our future target would construct a way to formulate such adaptive processes as clearly as possible based on these outcomes, which would give a landscape beyond LAST.

References

Argyris, C., & Schön, D. A. (1996). *Organizational learning II*. Addison-Wesley.
Ashby, W. R. (1960). *Design for a brain* (2nd ed.). Wiley.
Chang, C. C., & Keisler, H. J. (1973). *Model theory*. North-Holland.

Espejo, R., Schuhmann, W., Schwaninger, M., & Bilello, U. (1996). Organizational transformation and learning. In *A cybernetic approach to management*. Wiley.

Grätzer, G. (1979). *Universal algebra*. Springer.

Mesarovic, M. D., & Takahara, Y. (1975). *General systems theory: Mathematical foundation*. Academic Press.

Mesarovic, M. D., & Takahara, Y. (1989). *Abstract systems theory*. Springer.

Peterson, J. L. (1981). *Petri net theory and the modeling of systems*. Prentice-Hall.

Symoryński, C. (1985). *Self-reference and modal logic*. Springer.

Takahashi, S. (1995). Self-referential systems representation with modality in structure change. *Advances in Systems Science and Applications*, Special Issue, The International Institute for General Systems Studies, pp 19–24.

Takahashi, S. (2006). Agent-based organizational cybernetic approach to organizational learning. In *SICE-ICASE International Joint Conference 2006(SICE-ICCAS 2006).*

Takahashi, S., & Takahara, Y. (1995). *Logical approach to systems theory*. Springer.

Zeigler, B. P. (1990). Object-oriented simulation with hierarchical. In *Modular models: Intelligent agents and endomorphic systems*. Academic Press.