TR-0820

Generation of Aggregated Knowledge in Qualitative Reasoning

by
H. Shinjo, M. Ohki, E. Oohira &
M. Abe (Hitachi)

December, 1992

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191 ~ 5
Telex ICOT J32964

Institute for New Generation Computer Technology

Generation of Aggregated Knowledge in Qualitative Reasoning

Hiroshi Shinjo, Masaru Ohki, Eiji Oohira, and Masahiro Abe

Central Research Laboratory, Hitachi, Ltd.,
Kokubunji, Tokyo 185, Japan

Abstract

Because knowledge acquisition is a very difficult process, some qualitative reasoning systems use deep knowledge representing principles. But using deep knowledge increases the complexity of reasoning because the grain size of reasoning that uses only deep knowledge is sometimes too small. We therefore propose a method for generating knowledge that has a larger grain size. This method generates "aggregated knowledge," representing the behavior of large components, from deep knowledge representing the behavior of small components. The generation process consists of three analysis steps. The first is qualitative simulation to find all possible behaviors of the target large component. The second is to find all the possible states from these behaviors. And the last is to find the transitional order of those states. These steps generate aggregated knowledge that has existential conditions, relations, and transitional orders for each possible state. Such aggregated knowledge can represent all kinds of components and is useful in applying qualitative reasoning to large and complex systems.

I. Introduction

Qualitative reasoning is one of the methods for simulating and explaining behaviors of dynamic systems [2], [3]. It can be used to build a model that uses a set of differential equations to describe the mechanism of behaviors, it can be used to analyze the results of qualitative simulation, and so on [5], [6], [7].

Conventional expert systems use shallow knowledge, which is based on expert's experience. But shallow knowledge has the following problems:

1) It is difficult to acquire knowledge from experts.
2) It is difficult to rearranging a knowledge base for application to other domains or tasks.

3) Reasoning systems using shallow knowledge cannot solve unpredictable problems.

Unlike shallow knowledge, deep knowledge represents principles, such as physical rules. Because principles are independent of specific tasks, one knowledge base can be used to build models of object systems in various domains. But if a qualitative reasoning system uses only deep knowledge, the grain size of the knowledge might sometimes be too small. For example, to solve mathematical problems, we use formulas rather than axioms. Since formulas are acquired by the process that uses axioms to solve a certain problem, the grain size of formulas is larger than that of axioms. Using only deep knowledge thus increases the complexity of computation.

To apply qualitative reasoning to complex systems, we use knowledge with a large grain size (like that of shallow knowledge), and we adjust the grain size of reasoning according to the situations. Falkenhainer and Forbus present one of the largest-scale systems that has ever been treated with qualitative reasoning [1]. Their method uses hierarchical knowledge and shifts the grain size of reasoning. That is, their method can build a smaller-grain-size model of the target system when a user selects both the small grain size of model description and the standard behaviors for the system. But it is difficult to acquire knowledge about the target domain without contradiction. Liu and Farley propose a set of rules that choose proper ontologies in reasoning about electronic circuits [4]. Their method is effective because the reasoning system can analyze the same problem from different perspectives, but it does not use complex circuits and it does not automatically give the knowledge or the rules for shifting ontologies. When more complex systems are treated, the acquisition of knowledge and rules might be an intrinsic problem. It is difficult for users to acquire large-grain-size knowledge because they must analyze all possible states in which the component specified by the large-grain-size knowledge can behave. In short, this knowledge must be able to take in the whole real number space of input variables. It is impossible to simulate the component numerically at regular input values ranging from minus infinity to plus infinity, so a the method for automatic knowledge acquisition is needed.

We propose a method for generating large-component knowledge by aggregating small-component (device) knowledge. This small-component knowledge is treated as deep knowledge, and the large-component knowledge is treated as task-dependent knowledge. In an electronic circuit, for example, small-component knowledge represents the function of devices like resistors and transistors, whereas large-component knowledge represents the behavior of small circuits like amplifiers or Schmitt triggers. In this paper, we call such large-component knowledge "aggregated knowledge."

Our method generates the aggregated knowledge representing the behaviors of large components, but it cannot decide the structure of components that consists of more than one small component (device). The user must therefore specify both the structure of a component and its input and output variables. Yoshida proposed a method that decides the

structure of components by using the results of qualitative simulations [13], [14]. His method can extract the structures of NOR gates and NOT gates from the inference pattern of shift registers. When we are not dealing with design problems, however, we usually know the structure of the target systems. Our method therefore assumes that their structure is known.

The method we propose here has the following advantages:

1) It reduces the complexity of reasoning, because the analysis of the internal conditions of the large component is omitted.
2) Aggregated knowledge can be used like deep knowledge because it has the same form.
3) Aggregated knowledge can represent discontinuous transitions, whereas ordinary qualitative reasoning systems can deal only with continuous transitions.

Therefore, in adopting qualitative reasoning to analyze an specific task, we can build a model with the proper grain size of knowledge. If we want to analyze a Schmitt trigger circuit, for example, we can use only device knowledge and Kirchoff's Law as deep knowledge. But if we want to analyze a large circuit including a Schmitt trigger, we can also use the large-component knowledge representing the function of a Schmitt trigger. The time for computation is therefore reduced. And this method can generate the knowledge of a much larger component by using large-component and small-component knowledge recursively.

The knowledge generation process consists of three analytical steps. The first is qualitative simulation, with small-component knowledge, to find all possible behaviors of the target large component. The second is to use these behaviors to find all possible states. And the last step is to find the transitional order of those states. Finally, we can generate the aggregated knowledge that includes the existential conditions, relations, and transitional order for each possible state.

In this way, we can get a lot of aggregated knowledge according to the tasks or domains. Thus we can have not only the common database consisting of deep knowledge, but also the effective library consisting of aggregated knowledge for specific tasks.

Section II of this paper gives an overview of our method, Sec. III outlines our qualitative reasoning system Qupras (for qualitative physical reasoning system), and Sec. IV describes the representation of aggregated knowledge. Section V shows the method for aggregating knowledge, and Sec. VI demonstrates two examples of aggregating knowledge about electronic circuits. In Section VII, we discuss the directions for future research, and finally, in Sec. VIII, we conclude by very briefly summarizing this paper.

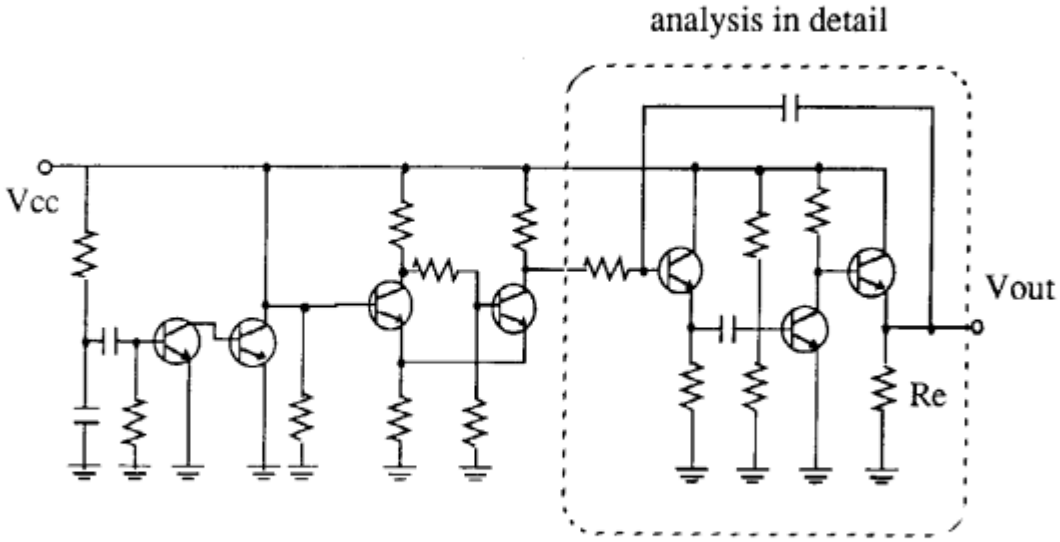
II. Overview of generating aggregated knowledge

We have developed a decision support system called Desq (for design support system based on qualitative reasoning) [10], [11], [12] based on the qualitative reasoning system Qupras [8], [9]. A designer often does not directly design a new circuit but instead simply modifies an old circuit. Sometimes designers can satisfy requirements simply by changing the parameters of components in an already-designed circuit. In these cases, the designer knows the structure of the circuit and needs only to determine the new values of the components. We want to apply qualitative reasoning to these kinds of design decisions, but if our support system uses only deep (device) knowledge, the complexity of reasoning would be too great for treating large circuits. Large-component knowledge is therefore indispensable for supporting the design of large circuits.

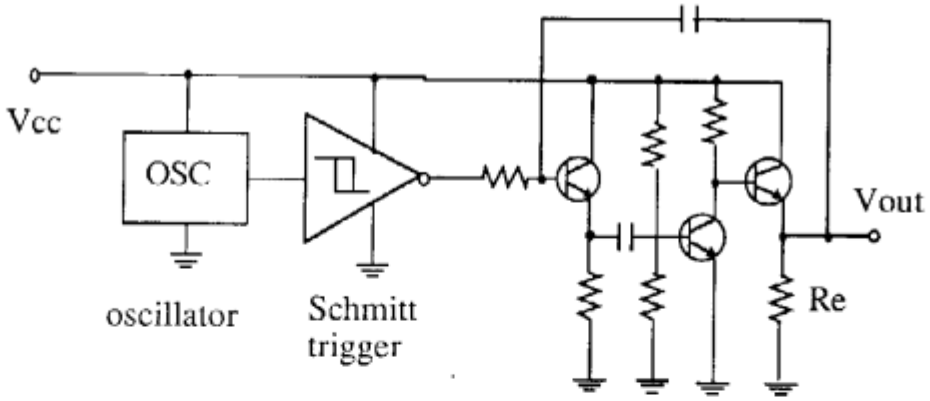
Suppose that we want to determine only the value of resistance "Re" in the circuit shown by Fig. 1. If Desq used only the device (small-component) knowledge, the complexity of reasoning would be extremely high because the grain size of reasoning would be too small. Our method can generate large-component knowledge, however, by aggregating small-component knowledge. This means that it can generate the large-component knowledge that represents the input and output function of an oscillator or a Schmitt trigger. Both large-component and small-component knowledge can be utilized similarly. When the input value for a large component is given, the output value is determined immediately. Complexity is therefore reduced because Desq need not analyze the internal conditions of oscillators and Schmitt triggers.

By using qualitative simulation, this method can automatically aggregate small-component knowledge to generate large-component knowledge. But a user has to specify the the structure of the target component and its input and output variables. Knowledge is aggregated in the following three steps:

1) All possible states in all behaviors are found out by simulating with Qupras, and the existential conditions and relations for each possible state is extracted from the results of reasoning.
2) Because many of the behaviors found in step 1 are redundant, the states that have the same characteristics are united.
3) Relations of the transitions from one state to other are determined from the results of reasoning.



(a) before aggregation



(b) after aggregation

Fig. 1. Overview of aggregated knowledge.

III. Qupras outline

Qupras (Fig. 2) is a qualitative reasoning system that uses knowledge from physics and engineering textbooks. Qupras can be executed on the parallel inference machines supported by Institute of New Generation Computer Technology (ICOT) - that is, on PIM, Multi-PSI, or Pseudo Multi-PSI.

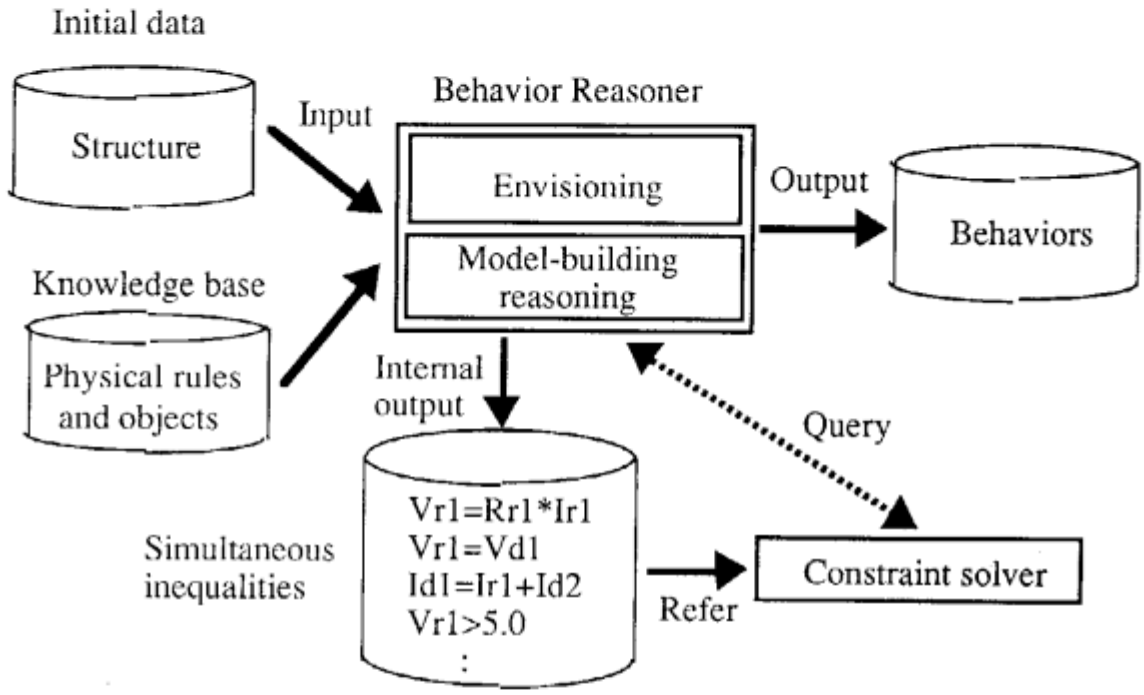


Fig. 2. Qupras.

Qupras has the following characteristics:

1) It has three primitive representations defined as deep knowledge: physical rules (laws of physics), objects, and events.
2) It determines the dynamic behaviors of a system by using knowledge of physical rules, objects, and events to construct all the equations for the system. The user therefore need not enter all the equations of the system.
3) It deals with equations that describe basic laws of physics qualitatively and quantitatively.
4) It does not require quantity space to be given in advance. Because the constraint solver can calculate all parameters included in the target system, Qupras finds the quantity spaces itself during reasoning.

5) It has the function of "envisioning." This means that even if the conditions of physical rules and objects cannot be evaluated, Qupras can continue to reason by assuming unevaluated conditions.

6) Objects in Qupras can inherit definitions from their superobjects. Physical rules can thus be defined generally by using superobjects to specify the definitions of object classes.

Qupras is similar to QPT [2] but does not use influence: in Qupras, only equations are used to describe the relations of values. This is because Qupras represents the laws of physics given in physics and engineering textbooks, and there these laws are generally described by using equations rather than influences. Moreover, Qupras can deal not only with qualitative but also quantitative values for representations. This makes the results of reasoning less ambiguous than when only qualitative values are used.

The representation of objects (one kind of deep knowledge) mainly consists of existential conditions and relations. Existential conditions are those needed for the objects to exist, and objects satisfying these conditions are called active objects. Relations are expressed as relative equations that include physical variables. If the existential conditions for a state are satisfied, the relations for that state become known as relative equations that hold for physical variables of the objects specified in the physical rule definition.

The representation of physical rules (another kind of deep knowledge) mainly consists of objects, applied conditions, and relations. The objects are those necessary to apply a physical rule. The representations of applied conditions and relations are similar to the representations of objects. Applied conditions are those required to activate a physical rule, and relations correspond to the laws of physics. Physical rules whose necessary objects are activated and whose conditions are satisfied are called active physical rules. If a given physical rule is active, its relations become known as in the case of objects.

Qualitative reasoning in Qupras involves two forms of reasoning: propagation reasoning and prediction reasoning. Propagation reasoning determines the state of the physical system at a given moment (or during a given time interval). Prediction reasoning determines the physical variables that change with time, and it predicts their values at the next given point in time. Moreover, the propagation reasoning uses the results from the prediction reasoning to determine the subsequent states of the physical system.

The function of envisioning is useful because even if initial data is given incompletely, Qupras can analyze a behavior of a target system by hypothesizing. For example, if the state of a certain transistor is unknown, Qupras hypothesizes both "on state" and "off state" and builds the model based on each hypothesis. Moreover, Qupras can determine the value (or range) of unknown parameters because the constraint solver calculates a model that is described by a set of simultaneous equations.

The reasoning process is as follows: When the initial data of a target system is given, the behavior reasoner builds its model corresponding to the initial state by evaluating the conditions of physical rules and objects. The rules and objects are stored in the knowledge base, and model-building reasoning generates the simultaneous inequalities. Simultaneous inequalities are passed to the constraint solver to check their consistency and to store them. If an inconsistency is detected, the reasoning process is abandoned. Conditions in the definitions of physical rules and objects are checked by the constraint solver. If the conditions are satisfied, the inequalities in the consequences of the physical rules and objects are added to the simultaneous inequalities in the constraint solver. Conditions that cannot be evaluated by the constraint solver are hypothesized. After determining the model of the target system on one state, the behavior reasoner calculates all parameters and predicts the following state. When predicting the following state, the restrictions for constant parameters are passed and the information on the variable parameters used to predict the following state is generally passed to the following state.

IV. Aggregated knowledge representation

A. Classification of component

Although there are various types of input and output functions, we want to generate their component knowledge by using the same method. We therefore analyzed the classification of large components to determine the syntax of aggregated knowledge. In this paper, these functions are classified as follows:

1) Type 1: The function can be represented by a single equation.
2) Type 2: The function representation requires more than one equation.
3) Type 3: The function representation requires more than one equation and is overlapped.

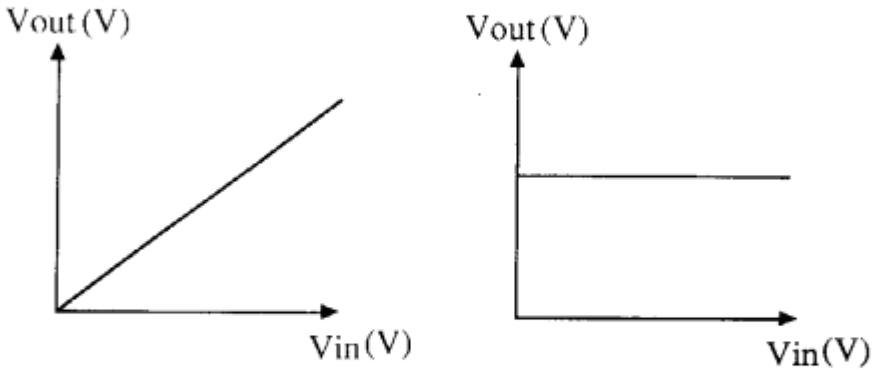
Each equation in the function means one state of behavior in qualitative reasoning. A function can be represented concisely by mutually substituting simultaneous equations that represents the model of one state. Qupras repeatedly builds a model and changes it according to the predicted following states. When conditions in the following state are changed, model is also changed. The equation representing the model is therefore also changed according to changing state.

The Type 1 - 3 functions can cover with behaviors of all kinds of components. Since a Type 2 function consists of more than one equation, it is regarded as a set of Type 1 functions. The Type 3 function is regarded as a particular pattern of Type 2 functions that overlap each other. Our method can therefore generate the aggregated knowledge to all kinds of components by dealing with these three types.

Figure 3 shows two examples of Type 1 functions: a direct proportionality, and a constant. The aggregated knowledge of an amplifier that has no saturated state, for example, can be described simply by the relation between input and output in the same way that describing Ohm's Law is enough to represent a resistance.

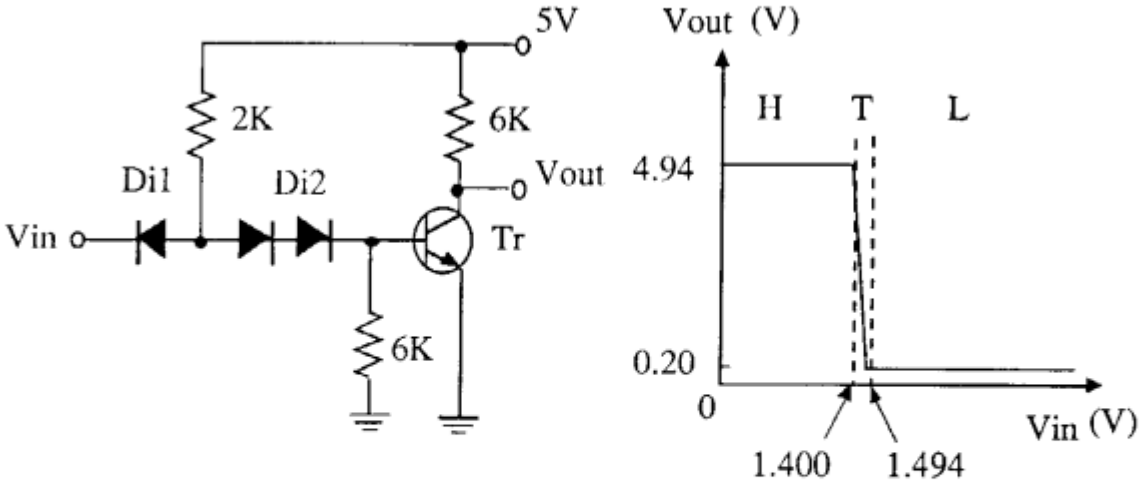
As an example of a Type 2 component, Fig. 4 shows diode transistor logic, an inverter that has three states. Its output value is high when the input value is low, and the output value is low when the input value is high. Here these are called "H" (high) state and "L" (low) state. There is also a transitional ("T") state between these two states. This type of function is represented in the same way that a diode is represented: as "on" state when the voltage between the terminals is under 0.7 volts and "off" state when this voltage is above 0.7 volts. This type of component can therefore be represented by describing not only the relations but also the existential conditions for each state.

A Schmitt trigger (Fig. 5) is an example of a Type 3 component. It has four states: a low-level state, a high-level state, and two transitional states. Here these are called the "L" state, "H" state, "Tu" (upward transition) state, and "Td" (downward transition) state. The characteristics of a Schmitt trigger show hysteresis, so the "H" and "L" states are overlapped partially. Because of this overlap, a Type 3 component cannot be represented only by existential conditions and relations. When the input value is between 5.8 and 7.4 volts, the state may be either "H" or "L." In addition to existential conditions and relations, the transitional information of a Schmitt trigger is needed to determine its state exactly. When the input value is in the overlapped range, the state is determined by the preceding state. If the preceding state is "L" or "Td," then the current state is "L." But if the preceding state is "H" or "Tu," the current state is "H." The representation for all kinds of aggregated knowledge therefore needs the following three factors: relations, conditions, and transitional information.



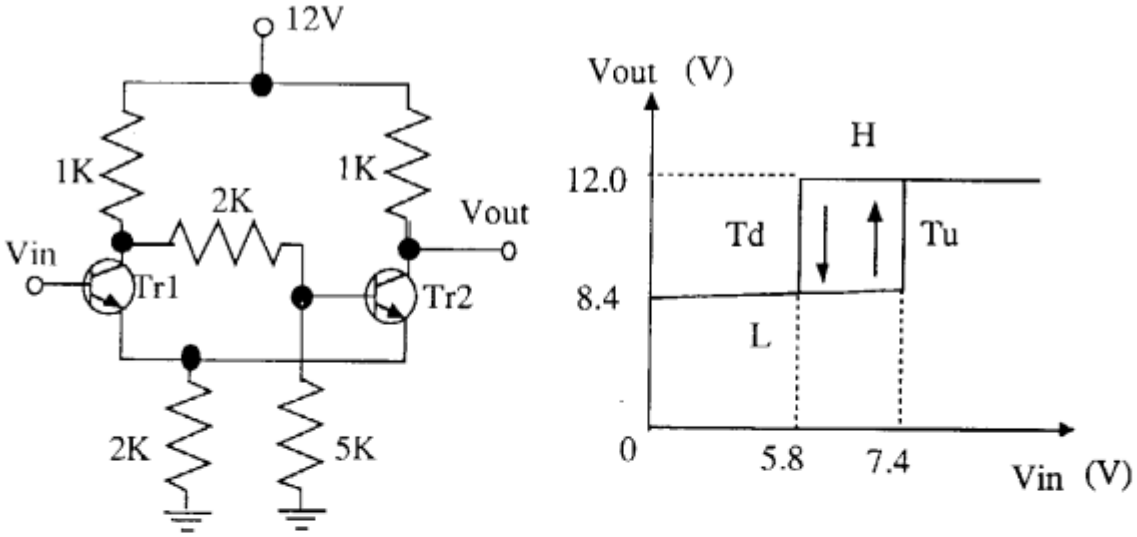
(a) direct propotionality (a) constant

Fig. 3. Example of Type 1 functions.



(a) structure of circuit (b) function

Fig. 4. Diode transistor logic.



(a) structure of circuit (b) function

Fig. 5. Schmitt trigger.

B. Representation for discontinuous behaviors

Qualitative reasoning generally deals with continuous behaviors. That is, the prediction of the following state is based on analysis of the continuous change of variables. But discontinuous behavior can result from positive feedback, so aggregated knowledge must also be able to deal with discontinuous behavior. For example, although a Schmitt trigger has four states, two of them instantly transit to another state because both transistors in the circuit are "on" state. Since these "Tu" and "Td" states are instantaneous, they can be omitted. In other words, a Schmitt trigger would actually have only "H" and "L" states. In describing the knowledge for a Schmitt trigger then should we exclude or include the instantaneous states?

In this paper, we include the instantaneous states because otherwise the reasoning could contradict itself. Figure 6 shows an example of contradiction, resulting from using knowledge of only the two states. Here, shown in Fig. 6.1, Component 1 (C1) and Component 2 (C2) are connected. The input and output of C1 are respectively x1 and y1, and those of C2 are x2 and y2. Because of the connection, y1 is equal to the x2. Figure 6.2(a) shows the behavior of C1: its output is first proportional to its input, then it increases instantaneously (discontinuously), and finally it is saturated. Figure 6.2(b) shows the behavior of C2: first its output is proportional to its input and then it is saturated. Finally, Fig. 6.2(c) shows the behavior of the connected components. When the input x1 is increased from zero to "a," the output y2 is between "s" and "t." Because x2 is in an increasing state and the qualitative reasoning system automatically finds the change of state, the next analysis is for the point at which x2 is equal to "r" and the behavior of C2 changes. But because of the discontinuous description, there are no cases where y1 is equal to "r."

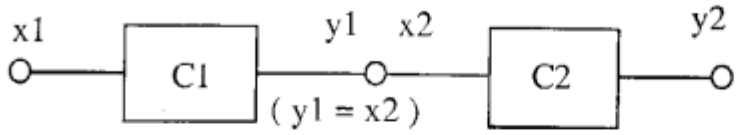
To represent this instantaneous state, the description must be as follows:

conditions: x1 = a

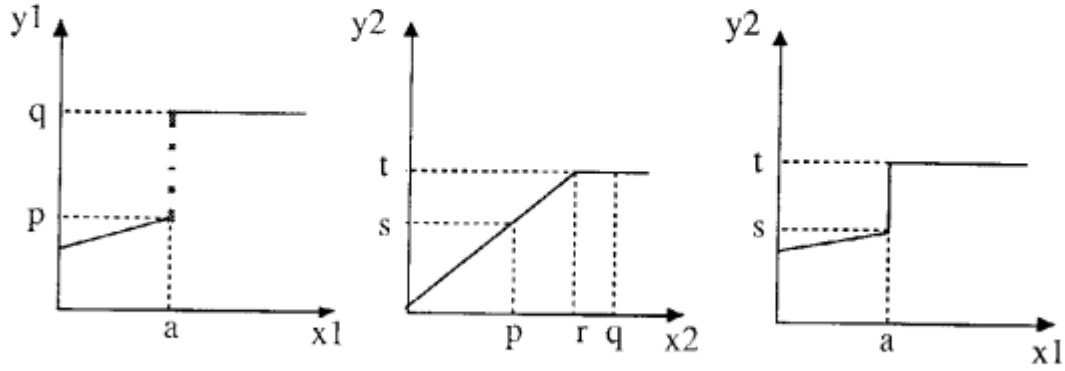
relations: p < y1 < q

C. Description of knowledge

Based on the discussions in Secs. 4.1 and 4.2, Fig. 7 shows a part of the description of a Schmitt trigger. This description is similar to the object definition in Qupras, but one difference is the description for the transitional information. The Schmitt trigger is defined as an "object," and the "attributes" field specifies internal variables in this object. The four states of the Schmitt trigger, "H," "L," "Tu," and "Td," are respectively equivalent to "s1," "s2," "s3," and "s4." Each state has conditions, relations, and transitional information. The "@" indicates a parameter, and the "vin@Schmitt" represents the input voltage value of the Schmitt trigger. The "vin@Schmitt > 5.8255911" specifies that the input voltage is greater than 5.8255911 volts. The "transitions" field specifies transitional information that describes the candidate following state and the conditions to reach that candidate state.



6.1 structure



(a) C1 (b) C2 (c) connected

6.2 function

Fig. 6. An example of contradict for discontinuous behavior.

V. Process of aggregation

A. Simulating all possible behaviors

To generate the aggregated knowledge for a large component, all its possible behaviors must be found. This knowledge must therefore be able to include the whole real number space of input variables. Since it would be impossible to simulate the component numerically at all input values from minus infinity to plus infinity, we simulate it qualitatively by using Qupras to find all possible behaviors. Qupras can do this because it regards as one state the interval over which the target system behaves as described by one model. Standard qualitative reasoning, however, cannot find all behaviors. In this section, we first describe why the standard reasoning is not appropriate, and then we describe our method.

To see why the standard qualitative simulation cannot find all possible states, consider the Schmitt trigger shown in Fig. 8. In this case, all states cannot be analyzed even if the behavior simulations are based on both initial conditions: increasing and decreasing input values. The "L1" and "H1" states exist for both increasing and decreasing conditions, but "L2" or "H2" state exists for either an increasing or decreasing condition. If aggregated knowledge is based on the behaviors of these simulations,

```
object thing:Schmit
  attributes

    vr_register_1 - variable;
    ir_register_1 - variable;
        ........
    vout - variable;
    vin - variable;

 state s1
 conditions
   vin@Schmit > 5.8255911 ;
 relations
   vout@Schmit = 12.0;
 transitions
   transition_conditions
     vin@Schmit > 5.8255911 ;
   transition_state
     s1 ;
 transitions
   transition_conditions
     vin@Schmit = 5.8255911 ;
   transition_state
     s4 ;


state s2
conditions
  vin@Schmit >= 0.0;
  vin@Schmit < 7.4301796;
relations
  ir_register_7@Schmit = ir_tanshi_19@Schmit + ir_tanshi_23@Schmit;
  ir_register_1@Schmit = ir_register_13@Schmit + ir_tanshi_22@Schmit;
        ........
  vr_register_1@Schmit = -1.0 * vr_tanshi_3@Schmi + 12.0;

transitions
  transition_conditions
    vin7@Schmit >= 0.0;
    vin@Schmit < 7.4301796;
  transition_state
    s2;
transitions
  transition_conditions
    vin@Schmit = 7.4301796;
  transition_state
    s3;

state s3
      ........


state s4
      ........


end.
```

Fig. 7. A Schmitt trigger described by aggregated knowledge.

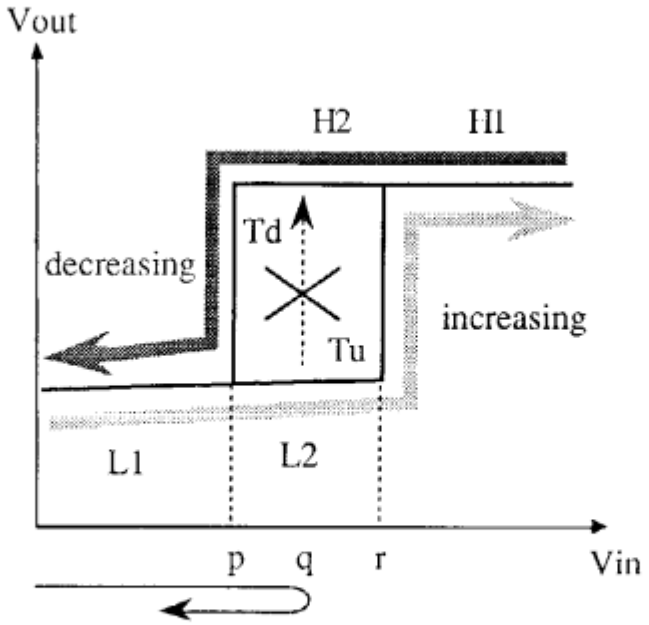


Fig. 8. Example showing that standard simulations cannot find all states.

the reasoning will sometimes be inconsistent. When the simulation is based on the condition that input value decreases from "q" after it increases from zero to "q," the correct behavior is to keep the low-level state. But the low-level state does not exist under the condition that input value is decreasing between "p" and "r." The simulation therefore passes through the "L1," "L2," "H2," "Td," and "L1" states even though the "H2" and "Td" states are incorrect.

Our method first finds out all possible states and then simulates the states following from each. All possible behaviors can be acquired by the transitional relation between every possible state and its following state. To find out all states, the initial conditions of qualitative simulation differ from those of the above-mentioned method: the initial input value is not defined in the initial conditions, but the changing states of input variables are defined as increasing, decreasing, or constant.

To acquire the transitional relation between two states, our method uses simulations to find all the following states from each possible state. These simulations need not be completed, and they are interrupted when the following state is found.

"Envisioning" is used to find all states. To compensate for the incomplete information (the initial values of input variables are not defined in the initial conditions), Qupras hypothesizes all possible cases. According to each hypothesis, Qupras builds models that represent all possible states and that consist of a set of equations. Then Qupras solves those ranges of input variables that lead to each possible state. The

relations between input and output, on the other hand, can be acquired from the equations in the model. As a result, Qupras can get the existential conditions and relations for all possible state.

Two kinds of description are used to represent relations. If the output is constant during the state, the relation directly describes the output value as a certain constant because Qupras can get the value of the output variable by solving the equations of the model. Since the output value is constant independent of the value of the input variable in the state, the value of the input variable is not needed to get the value of the output variable. If the output depends on the input, the relation is described by all equations of the model. When the input value is given, Qupras can get the value of output variables by substituting that value as the input variable.

As input variables change increasingly, decreasingly, or constantly, all the states that follow from all acquired states can be found out. Therefore the results of all the qualitative simulations by Qupras cover all possible behaviors.

B. Uniting and arranging the states

Because the results of simulations are redundant, a second process unites the states that have the same characteristics. Under certain conditions, this process unites two states that have one of the following features:

1) Both states have the same constant output value.
2) Both states have the same model (which is described by simultaneous equations).

Feature 1 has priority over feature 2. The characteristics of a state are determined by its own model describing the internal conditions. If two states have the same output value, they are united even if they have different models.

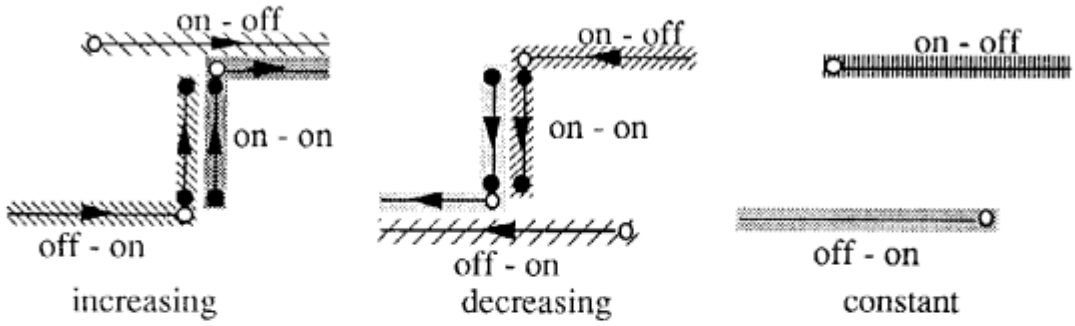
But even though two states may have the same model or output values, sometimes their existential conditions are different. When those conditions are overlapped, those states are united. Otherwise they are not united. The new united conditions are extended to the range over which the two conditions cover each other.

To acquire the transitional information, the last process arranges the united states according to the transitional order as analyzed by simulation.

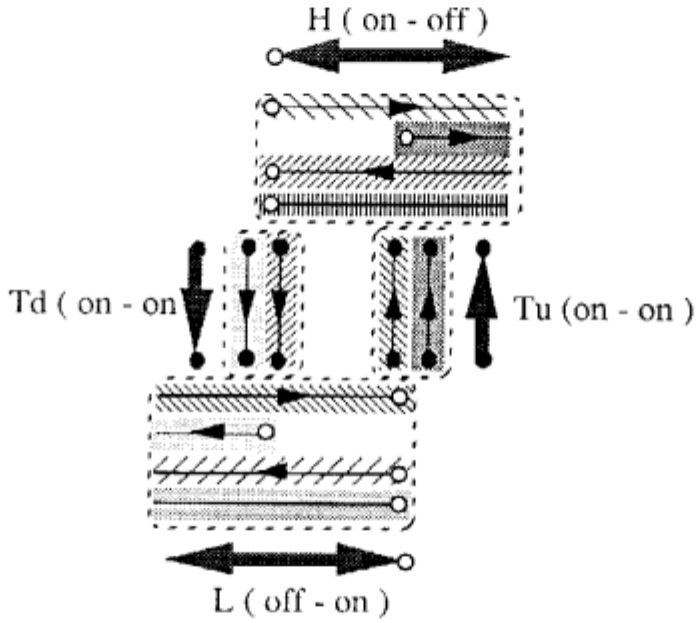
VI. Example

A. Generation

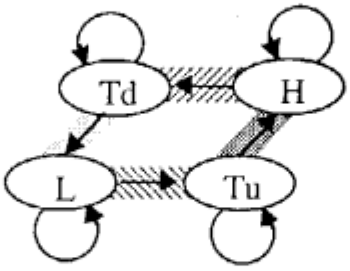
We have generated the two knowledge: Type 2 represents diode transistor logic, and Type 3 represents a Schmitt trigger. To demonstrate the total process of generating the knowledge, Fig. 9 shows the example of a Schmitt trigger. Figure 9(a) corresponds to Sec. V.A of this paper, and



(a) simulating all possible states



(b) uniting states represented by the same model.



(c) arranging united states

Fig. 9. Process of generating aggregated knowledge.

Figs. 9(b) and 9(c) correspond to Sec. V.B. The description "on - off" refers to the states of the transistors in Fig. 5(a) : Tr1 is in the "on" state and Tr2 is in the "off" state. The descriptions of "on - on" and "off - on" similarly refer to these transistors. The "on - on" states are in positive feedback and instantaneously change to another state. In Fig. 9(a), all possible states and behaviors are found out by several kinds of simulations. Each simulation is expressed by a different hatching pattern. The conditions for these simulations are as follows: the value of input variable (Vin) is not determined and the changing state of the input variable (dVin/dt) is defined as increasing, decreasing, or constant. The undefined variable Vin is incomplete information for "envisioning." In Fig. 9(b), the states that have the same model are united: "H" and "L" states are united and the existential conditions are extended. Since these two states exist in all conditions (Vin increasing, decreasing, or constant), the changing states of Vin are not described in the existential conditions for these two states. The "Tu" and "Td" states have the same model representing the "on - on" state, but they cannot be united because the ranges of Vin do not overlap. The "Tu" state therefore exists in the condition that Vin is only increasing, and the "Td" state exists in the condition that it is only decreasing. In Fig. 9(c), the states are arranged according to the simulations shown in Fig. 9(a). The arrows represent the directions of transitions, and the hatching patterns correspond to those in Fig. 9(a). The results of these process are thus described as the knowledge of the Schmitt trigger.

As to uniting states, the diode transistor logic example is used to explain another case. Because the output from this kind of component sometimes has the same value even if the models are different, we use the output value instead of a model to represent the relations of aggregated knowledge. Table I lists the results of simulating the initial condition that the input value is increasing. The terms "on," "off," and "sat" represents the states of the diode and transistor, and "sat" means saturated state. Qualitative simulation finds out continuous states ranging from minus infinity to plus infinity and shows that two kinds of states are repeated by turns. One kind of states continues for a certain period, and the other changes instantly. The behavior consists of nine states, but the second to fourth states have the same model because of the same internal condition. The fifth and sixth states also share a model, as do the seventh and eighth states. So this behavior consists of five states based on models (internal conditions). But because of the output values, the aggregated knowledge consists of only three states. The "H" state unites the first to fourth states, the "T" state is the fifth state, and the "L" state unites the sixth to nine states. Even if the states have different models, the function of states is the same if their output values are the same.

B. Effects

The purpose of aggregated knowledge is to reduce the complexity of reasoning, and this section mainly demonstrate this effect. Table II shows how complexity is reduced when Qupras simulates diode transistor logic or a Schmitt trigger. This data is for simulations of the condition that the input variables are increasing from zero. Small-component knowledge is a kind of standard deep knowledge and represents the function of devices like resistors and transistors. Large-component knowledge is an aggregated knowledge and represent the relations between input and output.

Table I
Simulated behavior of diode transistor logic.

state	internal condition (Di1 - Di2 - Tr)	input Vin (V)	output Vout (V)	united state
1	ON-OFF-OFF	(- ∞, 0.860)	4.940	H
2	ON-ON-OFF	0.860	4.940	
3	ON-ON-OFF	(0.860 , 1.400)	4.940	
4	ON-ON-OFF	1.400	4.940	
5	ON-ON-ON	(1.400 , 1.401)	(0.200 , 4.940)	T
6	ON-ON-ON	1.401	0.200	L
7	ON-ON-SAT	(1.401 , 1.494)	0.200	
8	ON-ON-SAT	1.494	0.200	
9	OFF-ON-SAT	(1.494 , + ∞)	0.200	

(a, b) : a < x < b

Table II
Effects of reducing complexity.

	diode transistor logic			Schmitt trigger		
	small component	large component	ratio small / large	small component	large component	ratio small / large
number of states in behavior	10	6	1.83	13	7	1.86
number of equations in one state (average)	107	25	4.28	85	21	4.05
total number of equations in behavior	1070	149	7.90	1105	237	4.66
processing time (sec)	10702	232	46.13	7908	704	11.23

The first effect is to reduce the number of states in the behavior. As shown in Table I, when using small-component knowledge, the states change according to the internal conditions. When using large-component knowledge, Qupras need not evaluate the internal conditions because the states change only according to the relations between input and output. As a result, the number of the states in behavior is reduced by about half.

The second effect is to decrease the number of equations that Qupras evaluates in one state. The fewer equations, the less the complexity. There are two kinds of equations, and we will consider them separately. One kind is described in the existential conditions of knowledge. When using small-component knowledge, Qupras evaluates the conditions of all devices: the more devices in the circuit, the more equations evaluated. When using large-component knowledge, Qupras evaluates the conditions of large components only once. Therefore the number of equations described in the conditions is reduced. The other kind of equation is described in the relations of knowledge. When using small-component knowledge, Qupras must solve simultaneous equations that describe models, and this is a very hard process. Even if Qupras uses large-component knowledge, this situation is the same. When the state has constant output, however, Qupras need not solve the equations because that output value is described in the knowledge. Therefore the equations described in the relations of knowledge are greatly reduced. As a result, both the number of states and the number of equations are reduced. This reduces processing time almost fifty-fold for diode transistor logic and over ten-fold for a Schmitt trigger.

VII. Future work

This aggregated knowledge is useful for applying qualitative reasoning to a large and complex system, but two problems remain:

1) The parameters in the target large component must be determined previously.
2) The structure of the component must be determined previously.

In principle, it is easy to solve the first problem. We want to have variable parameters in the aggregated knowledge. When users determine the values of the parameter according to the intention of their simulations, they can get the desirable specifications for the large component. To generate knowledge that has variable parameters, we simulate the target large component in special conditions. Just as we do not define the initial input value when we simulate the component to find all possible behaviors, we do not define the parameter value in the initial conditions. Like input variables, all the possible states are found according to the values of the parameters. But this problem results from the ability of our constraint solver, which solves simultaneous equations or inequations. Our constraint solver needs much time to solve them and presently can solve only some kinds of simultaneous quadratic equations or inequations. Therefore if the number of the variables are increased or

the equations are more complex, the constraint solver cannot solve the equations. To generate knowledge that has variable parameters, we need a much higher performance system for solving equations.

We do not think the second problem is important. Except in design problems, we generally know the structure of the target systems we want to simulate. The condition that the structure is known previously is therefore appropriate.

We think that the key to improving our method will be the users skill in judging which details can be neglected under which circumstances. To build a large and hierarchical knowledge base that can be applied to qualitative reasoning about complex systems in various domains, it will be necessary to represent one component from different perspectives. This will require consideration of the applied conditions as well as the existential conditions. If a circuit is used at high frequencies, for example, some of the condensers can be omitted. The aggregated knowledge should therefore have applied conditions and modified relations in which some of the variables in the simultaneous equations are neglected. We think, however, that such knowledge cannot be generated automatically.

VIII. Conclusion

We have described a method for generating aggregated knowledge for use in qualitative reasoning. Aggregated knowledge represents the function of large components, and it is made of the deep knowledge that represents the functions of small components. This kind of aggregated knowledge is useful when applying qualitative reasoning to large and complex systems. By using the deep knowledge in the database, we can get the knowledge for specific components automatically. As a result, we can get the library for the target domain. Moreover, since the aggregated knowledge has the same format as the standard deep knowledge of Qupras, we can aggregate the knowledge of much larger components by using large-component and small-component knowledge.

To make aggregated knowledge, we have to specify the structure of the target component and the names of input and output variables. The method of aggregating knowledge is as follows:

1) Qupras finds out all possible behaviors that the large component can transit.
2) The simulated states that have the same model or the same output value are united.
3) The united states are arranged according to the order of behaviors.

The description of aggregated knowledge has three factors for each possible state: existential conditions, relations, and - to allow representation of behaviors like hysteresis - information describing the transition to following states. Moreover the descriptions of relations are

extended for representing discontinuous behavior. As a result, the aggregated knowledge can represent all kinds of components.

Acknowledgements

This research was supported by the ICOT. We express our thanks to Dr. Fuchi, Director of the ICOT Research Center, who provided us with the opportunity to perform this research in the Fifth Generation Computer Systems Project. We also thank Dr. Nitta in the seventh research laboratory for many comments and discussions regarding this study. And we thank Professor Nishida of Kyoto University for his discussions, Mr. Kawaguchi, Miss Toki, and Miss Isokawa of Hitachi Information Systems for their help in the implementation of our system, and Mr. Yoshida of Advanced Research Laboratory of Hitachi for his discussions and suggestions.

References

[1] B. Falkenhainer and K.D. Forbus, "Setting Up Large-Scale Qualitative Models," In Proceedings AAAI-88, pp. 301-306, American Association for Artificial Intelligence, 1988.

[2] K.D. Forbus, "Qualitative Process Theory," Artificial Intelligence, 24, pp. 85-168, 1984.

[3] B. Kuipers, "Commonsense Reasoning about Causality: Deriving Behavior from Structure," Artificial Intelligence, 24, pp. 169-203, 1984.

[4] Z.Y. Liu and Arthur M. Farley, "Shifting Ontological Perspectives in Reasoning about Physical Systems," In Proceedings AAAI-90, pp. 395-400, American Association for Artificial Intelligence, 1990.

[5] T. Nishida, "Recent Trend of Studies with Respect to Qualitative Reasoning (I) Progress of Fundamental Technology," pp. 1009-1022, 1988 (in Japanese).

[6] T. Nishida, "Recent Trend of Studies with Respect to Qualitative Reasoning (II) New Research Area and Application," Journal of Information Processing Society of Japan, pp. 1322-1333, 1988 (in Japanese).

[7] T. Nishida, "Qualitative Reasoning and its Application to Intelligent Problem Solving," Journal of Information Processing Society of Japan, pp. 105-117, 1991 (in Japanese).

[8] M. Ohki, Y. Fujii, and K. Furukawa, "Qualitative Reasoning based on Physical Laws," Trans. Inf. Proc. Soc. Japan, 29, pp. 694-702, 1988 (in Japanese).

[9] M. Ohki, K. Sakane, J. Sawamoto, and Y. Fujii, "Enhanced Qualitative Physical Reasoning System: Qupras," New Generation Computing,10, pp. 223-253, 1992.

[10] M. Ohki, E. Oohira, H. Shinjo, and M. Abe, "Range Determination of Design Parameters by Qualitative Reasoning and its Application to Electric Circuits," In proceedings of the international conference on Fifth Generation Computer Systems 1992, pp. 1022-1029, ICOT, 1992.

[11] M. Ohki, N. Toki, S. Isokawa, E. Oohira, H. Shinjo, T. Kawaguchi, and M. Abe, "Nonlinear inequalities constraint solver combined multiple constraint solvers," Journal of Japanese Society for Artificial Intelligence, 1992 (submitted, in Japanese).

[12] E. Oohira, M. Ohki, H. Shinjo, and M. Abe, "A Reasoning Method for Computer-aided Circuit Design using Qualitative Reasoning for Circuits involving Discontinuous Changes," Transactions of IEICE, 1992 (submitted, in Japanese).

[13] K. Yoshida, and H. Motoda, "CLIP: Concept Learning from Inference Pattern (1) - A Method to Find Typical Inference Pattern," Journal of Japanese Society for Artificial Intelligence, Vol. 7, No. 4, pp. 675-685, 1992 (in Japanese).

[14] K. Yoshida, and H. Motoda, "CLIP: Concept Learning from Inference Pattern (2) - Concept Hierarchy Formation," Journal of Japanese Society for Artificial Intelligence, Vol. 7, No. 4, pp. 686-696, 1992 (in Japanese).