PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1988,
edited by ICOT. © ICOT, 1988

METHODS FOR PARTITION OF TARGET SYSTEMS IN QUALITATIVE REASONING

Kiyokazu Sakane, Masaru Ohki, Jun Sawamoto[1], Yuichi Fujii

Research Center
Institute for New Generation Computer Technology
Mita Kokusai Bldg. 21F, 1-4-28, Mita
Minato-ku, Tokyo 108 Japan

ABSTRACT

This paper proposes two methods for partition of target systems in qualitative reasoning and discusses their effect on the efficiency of execution. One of the problems of qualitative reasoning is that it cannot deal with large target systems. We propose two methods of partition to solve this problem: (1) partition of variables according to the independence of each component, and (2) partition of a system by the field of applicable rules. Then we formulate the computational complexities of two reasoning processes (propagation and prediction) for two types of qualitative simulators (qualitative modeling type and qualitative process theory type). We estimate the effect of the partition methods from the viewpoint of computational complexity. Finally, we describe the conditions under which the methods reduce computational complexity.

1 INTRODUCTION

Recently, there have been many reports on research related to qualitative reasoning [deKleer 84],[Forbus 84], and [Nishida 87]. However, we found some problems in them. One of the biggest problems is that qualitative reasoning cannot deal with large target systems, because large CPU time and a large memory are needed to simulate the behavior of such systems. The reason is that almost all currently available qualitative simulators aquire the system behavior as a whole. A few researches are related to partition of a system by time-scale [Kuipers 87, Tanaka 88]. They notices only the difference of time-scale of behavior among variables to structure a complex system hierarchically. When we consider a large target system that consists of many parts or subsystems, we divide it into some subsystems using heuristics about the structure or properties of the system. We consider the behaviors of all subsystems and integrate them to infer the behavior of a whole system. In this case, the method of partition by time-scale is not sufficient. Partition methods corresponding to such heuristics are necessary.

Assume a target system which consists of many parts that have close interactions between internal components and only weak interactions with external components. We predict the behavior of each subsystem independently, ignoring external circumstances other than the input state. Because we have heuristics in which each subsystem can be treated as a functional like component that responds to the input state. Assume another target system, each of whose components is designed and operated according to the rules of a different physical field (for example, electronic circuit, thermodynamics, electrostatics, and quantum mechanics) to satisfy a different function. If we know this, we only need to consider the behavior of each subsystem noticing rules in the specified physical field to infer the behavior of the whole system. Because there is little interaction between the rules of the different physical fields.

According to the above heuristics, we propose two methods for partition of target systems:

(1) A method that partitions the variables according to the independence of each component.

(2) A method that partitions a system into subsystems by restricting the field of applicable rules.

These two methods reduce the number of variables and/or rules to be considered

---

1) Current Address:
Computer Works, Mitsubishi Electric Corporation
325, Kamimachiya, Kamakura, Kanagawa 247 Japan

simultaneously and modularize each subsystem functionally. Therefore both methods have the following advantages.

(1) Computational complexity is reduced.

(2) These methods give good support in compiling simulated results of each modularized subsystem.

Here, we focus on the reduction of computational complexity. The effect of the two partition methods with this viewpoint changes according to the type of target system and simulator. It is very important to clarify some conditions for their application.

Section 2 explains the partition methods. Section 3 formulates the computational complexity of two reasoning processes (propagation and prediction) for two types of qualitative simulators (the qualitative modeling (QM) type such as [Kuipers 84], QSIM [Kuipers 85], and ENVISION [deKleer 84], and qualitative process theory (QPT) type such as GIZMO [Forbus 84], QPE [Forbus 86], and Qupras [Ohki 88]). Section 4 estimates the effect of partition methods with computational complexity. From this discussion, section 5 summarizes the conditions under which the partition methods improve the efficiency of qualitative reasoning. Section 6 shows an example of application of the partition method.

2 METHODS FOR PARTITION OF A TARGET SYSTEM

This section explains the two partition methods of target systems.

(1) *Method for partition of variables according to the independence of each subsystem*

If a target system consists of loosely connected parts, the whole system can be partitioned into subsystems corresponding to the parts. First, the parts which satisfy following condition are searched. That is, there are close internal interactions between physical variables in the same part, and there is no relationship between variables in different parts except for a small number of inputs and outputs. Such parts are collected as subsystems of the whole system. All variables belonging to only a part are assigned to the subsystem as internal variables. Each external variable, that is, an output from one part and/or an input to the other, is shared by both of these subsystems as a common variable.

The behavior of each subsystem is simulated as response to the input states. Each subsystem communicates its simulated results to others through common variables. Because the number of variables are reduced that a simulator has to deal with simultaneously, the problem we posed is solved.

(2) *Method for partition of a system by the field of applicable rules*

If there are some restrictions on the fields of dominant rules for each part, the whole system can be partitioned into subsystems. Each subsystem consists of the parts whose fields of dominant rules are identical. Because the kind of objects to apply a field of rules is restricted, the kind of objects in each subsystem should also be restricted. The simplified model is constructed ignoring rules of other fields. Because the rules of other fields have little effect on the behavior of the subsystem, a simulator acquires it with this simplified model to get a good approximate solution. Because the number of rules to be considered simultaneously are reduced, the problem we posed is solved.

However, these partitions are not usually effective. Sections 3 and 4 estimate the actual effect with the computational complexity when the above two methods are applied to the currently available qualitative simulators.

3 FORMULATING THE COMPUTATIONAL COMPLEXITY OF CURRENT SIMULATORS

This section formulates the computational complexity of the current simulators. Current qualitative simulators are roughly classified in two types: QMtype and QPTtype. In both type of simulators, qualitative behavior is simulated by two processes, propagation and prediction, in turn. We formulate approximately the computational complexity of the propagation and prediction process of QM-type and QPT-type simulators according to the algorithms of [Kuipers 84] and Qupras [Ohki 88] whose reasoning processes we know well. This discussion holds generally with many qualitative simulators.

3.1 Formulating the Computational Complexity of QM-type Simulators

In QM-type simulators, variables and constraints which express system dynamics are given and fixed. In computational complexity estimation, we use notations as shown in Table 1.

(1) *Propagation Process*

QM-type simulators use all constraints in a model in the propagation process. The simulator

Table 1 Notation in the Reasoning Process by QM-type Simulator

	Whole model	Partitioned model
Number of subsystems	1	W
For each subsystem:		
Size of a database	N	N/W+d
Number of constraints	M	M/W
Number of variables changing with time	N_Z	N_Z/W

d: number of common variables in each subsystem
c_0: cost of solving a constraint
c_1: cost of predicting the next qualitative state of a variable
c_2: cost of checking the consistency of a collection of candidates of all $\Gamma(z_i)$
$\Gamma(z_i)$: candidates of qualitative value of z_i at the next time
Y: average number of candidates in $\Gamma(z_i)$

propagates values of determined variables to undetermined variables through constraints. The simulator iterates this procedure until all variables in a model are determined. In the worst case, all M constraints are referred N times, until all N variables are determined. At least they are referred once. We introduce a new parameter H, which is the degree of iteration to refer each constraint. The computational complexity of the propagation process is expressed as:

$$CP_1 = c_0 \cdot M \cdot N^H \quad \text{where } 0<H<1 \qquad [1]$$

(2) *Prediction Process*

In prediction process, for each variable changing with time, z_i, candidates of qualitative value at next qualitative time, $\Gamma(z_i)$, is generated. $\Gamma(z_i)$ is acquired by finding the most neighboring limit points of the current value in the quantity space of z_i. Then, every collection of all changing variables in the next states is checked for consistency. The cost of prediction is represented as:

$$CD_1 = c_1 \cdot N_Z + c_2 \cdot Y^{N_Z} \qquad [2]$$

3.2 Formulating the Computational Complexity of QPT-type Simulators

(1) *Knowledge Representation and Generation of Instance Rules*

Qupras is similar to GIZMO (see APPENDIX). Basic representations of Qupras are objects and rules. In Qupras, physical variables are expressed as attributes of class objects. Constraints among objects are expressed as relations of template rules. Each class object and template rule has conditions that have to be satisfied for them to be active. Each template rule has a list of objects that have to be active before the rule is applicable. Instances for each class object and values for some attributes are given in the initial states.

At the beginning of the reasoning process, the simulator instantiates each template rule. The simulator generates instance rules for every collection of instance objects that satisfies the specification of the objects in the template rule (see Fig. 1). We use the simple model shown in Table 2.

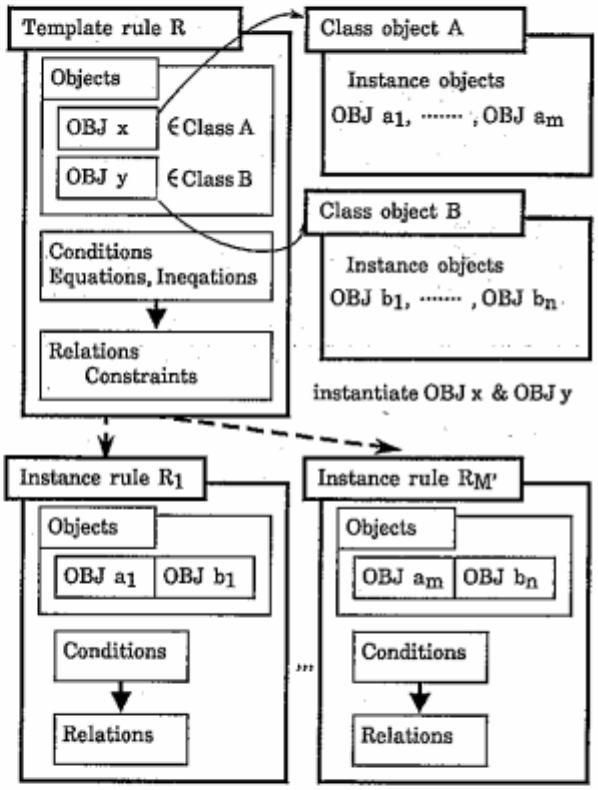


Fig. 1 Generation Process of Instance Rules from Template Rules

The number of instance rules generated for all template rule is:

$$M = S \cdot K^2 \qquad [3]$$

(2) *Propagation Process*

In the Qupras propagation process, the following procedure, exactly like forward chaining of

production rules, is iterated until no new active process or object is found.

For each instance rule,

if all objects in the rule are active and all applied conditions satisfy constraint propagation laws,

then the simulator solves all constraints in relations using constraint propagation laws.

We classify all instance rules into two groups:

(G1) The antecedent of the rule in this group is not satisfied until a propagation process ends; the simulator only checks the antecedent at each iteration of referring the rule.

(G2) The antecedent of the rule in this group is satisfied; the simulator checks the antecedent and propagates qualitative values through the constraints in the consequent.

Although a large number of instance rules are generated for each general template rule, only a few of them actually become active. Using the notation shown in Table 2, we formulate the total computational complexity in the propagation process as follows:

$$CP_2 = c_3 \cdot S \cdot K^2 \cdot R + c_4 \cdot F_P \qquad [4]$$

(3) *Prediction Process*

Because the quantity space is not declared explicitly in Qupras, the limit points of each variable are expressed in the applied conditions of rules and objects.

For each variable changing with time, z_i, the simulator finds the neighboring points of the current value from applied conditions of rules that contain $obj(z_i)$. $\Gamma(z_i)$ is selected from the neighboring points using state transition laws.

When we use the simple model shown in Table 2, the number of the instance rules which contain $obj(z_i)$ is $K \times I$. Then the $\Gamma(z_i)$ acquired for all z_i are checked for consistency. By the same token, we obtain the following computational complexity in the prediction process as well:

$$CD_2 = N_Z \cdot c_5 \cdot K \cdot I + c_2 \cdot Y^{N_Z} \qquad [5]$$

4 EFFECT OF PARTITION METHODS ON THE COMPUTATIONAL EFFICIENCY OF QUALITATIVE SIMULATORS

This section estimates the effects on the computational efficiency of qualitative simulators by the partition methods. Section 4.3 investigates the meanings of some formulae.

Table 2 Notation in the Reasoning Process by QPT-type Simulator

	Whole model	Partition of instance objects	Partition by applicable rules
Number of subsystems	1	W	W
For each subsystem:			
Number of template rules	$S=I \cdot J/r$	S	$S''=(I/D)\cdot(J/W+e)/r$
Number of class objects	J	J	$J/W+e$
Number of applicable template rules for each object	I	I	I/D
Number of instances in each object class	K	$K/W+f$	K
Number of instance rules in the propagation process			
In group G_1	$E_P \sim S \cdot K^2$	$E_P'/W \sim S \cdot (K/W+f)^2$	$E_P''/W \sim S'' \cdot K^2$
In group G_2	F_P	F_P'/W	F_P''/W
Number of iterations of referring rules	R	R'	R''
Number of variables changing with time	N_Z	N_Z/W	N_Z/W

r: number of objects referred in each template rule: $r=2$
e: number of common class objects belonging to some subsystems
f: number of common instance objects belonging to some subsystems
D: number of different fields of physical rules

c_3: cost of checking the activities of objects in a rule and verifying whether all conditions in a rule are satisfied
c_4: cost of solving constraints in relations of a rule

$obj(z_i)$: instance object that has z_i as an attribute
c_5: cost of searching the neighboring points of current value of z_i in conditions of a rule plus acquiring $\Gamma(z_i)$ by filtering the neighboring points

4.1 Effect of Partition Methods for QM-type Simulators

Because connections between variables and constraints are given and fixed statically in a domain model of QM-type simulators, the partition of variables partitions constraints to be applied to the variables, vice versa. Therefore, a whole model is partitioned into same set of subsystems by the two partition methods.

(1) *Propagation Process*

Consider an equally partitioned model as shown in Table 1 and assume inequality [6]. We obtain the computational complexity of the propagation process using the partition method from the equation [1] as follows:

$$d \ll N/W \qquad [6]$$

$$CP_1' = c_0 \cdot M \cdot N^H/W^H \qquad [7]$$

Assumption [6] means that each subsystem has far fewer inputs and outputs than its internal variables. As H becomes greater, the effect of the partition method for the reduction of computational complexity becomes more effective.

(2) *Prediction Process*

Similarly, we estimate the computational complexity of the prediction process as follows:

$$CD_1' = c_1 \cdot (N_Z/W) \cdot W + c_2 \cdot [Y^{N_Z/W}]^W \qquad [8]$$

Because the right-side of formula [8] is equal to that of [2], the computational complexity of the prediction process cannot be reduced by partition methods alone.

4.2 Effect of Partition Methods for QPT-type Simulators

A domain model for QPT-type simulators is partitioned into different sets of subsystems by the two partition methods. We define two kinds of equally partitioned models as shown in Table 2. The instances of each class object are distributed among subsystems evenly by the method of partition of variables. It does not restrict the field of applicable rules in each subsystem. The partition by rules restricts the field of applicable rules in each subsystem. This restriction also restricts the class objects in each subsystem and reduces the number of template rules applicable to a class object. However instance objects are not partitioned.

4.2.1 Partition of Instance Objects

(1) *Propagation Process*

Consider the first equally partitioned model in Table 2, and assume two inequalities, [9] and [10]. We obtain formula [11].

$$f \ll K/W \qquad [9]$$

$$R' < R \qquad [10]$$

$$CP_2' = c_3 \cdot S \cdot K^2 \cdot R'/W + c_4 \cdot F_p' \qquad [11]$$

The meaning of assumption [9] is equal to that of [6]. Assumption [10] means that the iteration number are not increased by application of the partition method.

The first term in the right-side of [11] is the sum of the cost of checking activities of all objects and the cost of verifying the truth of applied conditions throughout a propagation process. This term is reduced 1/W times due to the reduction of the number of instance rules generated[2]. The second term shows the cost of solving constraints in all active rules. It does not decrease if the number of total active rules, Fp', is not reduced.

(2) *Prediction Process*

We also estimate the cost of the prediction process under assumption [9] as follows:

$$CD_2' = N_Z \cdot c_5 \cdot K \cdot I/W + c_2 \cdot Y^{N_Z} \qquad [12]$$

The first term in the right-side of [12] is the cost of collecting $\Gamma(z_i)$ from the applied conditions of the rules which contain $obj(z_i)$. This term is reduced 1/W times due to the reduction of the number of instance rules containing $obj(z_i)$. However we have the second term not reduced by this partition method, if the number of changing variables N_Z does not decrease.

---

2) The first term in the right side of [11] is proportional to the number of instance rules generated. The number of instance rules generated for each subsystem is approximately $S \cdot (K/W)^2$ under the assumption [9]. The total number of instance rules generated is $W \cdot S \cdot (K/W)^2$ ($= S \cdot K^2/W$). That is, it is reduced 1/W times by applying the partition method. Therefore, the term is reduced 1/W times by the partition method.

4.2.2 Restriction of Applicable Template Rules for Each Class Object

(1) *Propagation Process*

Consider the second equally partitioned model in Table 2, and assume inequalities, [13] and [14], just like [9] and [10]. We derive the following relations.

$$e \ll J/W \quad [13]$$

$$R'' < R \quad [14]$$

$$CP_2'' = c_3 \cdot S \cdot K^2 \cdot R''/D + c_4 \cdot F_p'' \quad [15]$$

In this partition, the first term is reduced 1/D times in the same way as the first equally partitioned model, but again, the second term does not decrease.

(2) *Prediction Process*

We also estimate the cost of the prediction process as follows:

$$CD_2'' = N_Z \cdot c_5 \cdot K \cdot I/D + c_2 \cdot Y^{N_Z} \quad [16]$$

Again, there is a term not reduced by partition method alone.

4.3 Investigation of the Meanings of Some Formulae

4.3.1 Assumptions for Obtaining the Computational Complexity to Be Reduced

Assumption [10] that requires the iteration number not to be increased, derives the sufficient condition that a domain model should not have feedback loops.

Consider an example of a model shown in Fig. 2a. A whole model is partitioned into two subsystems by our partition method. A feedback loop between the two subsystems is generated by the partition (see Fig. 2b). There are some dependency among variables in the above two models. Assume that the simulator refers constraints to propagate variable values in the sequential manner.

In the simulation as a whole (Fig. 2a), the simulator determines the values of x and y in order in the first step, and use the y value to determine the rest of the variable values in the next step. However, the partition (Fig. 2b) causes a redundant computation. Because subsystem A determines the x value and wastes time in trying to determine variable values other than x in subsystem A without input value y in the first step, then subsystem B obtains the y value, finally subsystem A determines

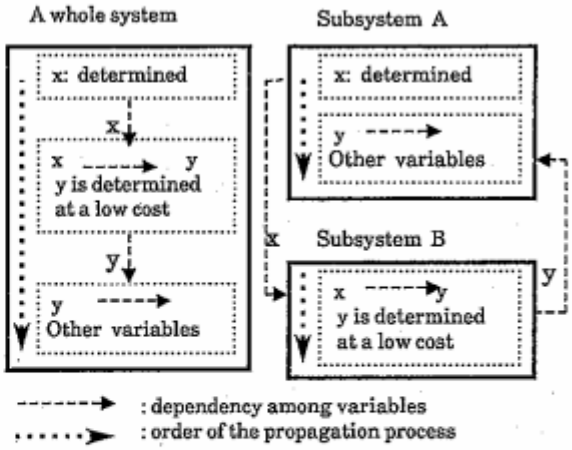


Fig. 2a Propagation for a Whole System without Feedback Loop

Fig. 2b Propagation for a System Partitioned into Subsystems with a Feedback Loop

the rest of the variables from the y value in the second step of propagation.

This is caused by the feedback loop in the structure. The same conditions are derived from assumption [13].

4.3.2 Terms Unreduced by the Partition Methods in the Propagation Process

The existence of terms not reduced by our methods in the propagation process requires the expression of rules to be more general.

Let us assume an appropriate size domain model (see Fig. 3) to estimate the effect of the ratio of the

Fig. 3 An Example of a Domain Model to be Partitioned

$I = 10, J = 10, r = 2, \quad R' = 1, \quad a_3 \simeq a_4$
$S = I \cdot J/r = 50, K = 10 \rightarrow 50, \quad W = 1 \rightarrow 5$
$\alpha = F_p'/(S \cdot K^2) : 0.1 \rightarrow 0.5$

α: ratio of the number of active rules to the total instance rules

second term in the formula [11]. That is, the costs, c_3 and c_4 are equal. Both the cost of verifying a constraint and the cost of solving a constraint are

much larger the cost of checking activities of objects in a rule. Because they need a massive search in the list of propagation laws or heavy computation like the SUP-INF method [Robert 77].

We introduce a new parameter, α, which is the ratio of the number of active instance rules to the total instance rules. We plot the computational complexities of the propagation process in Fig. 4,

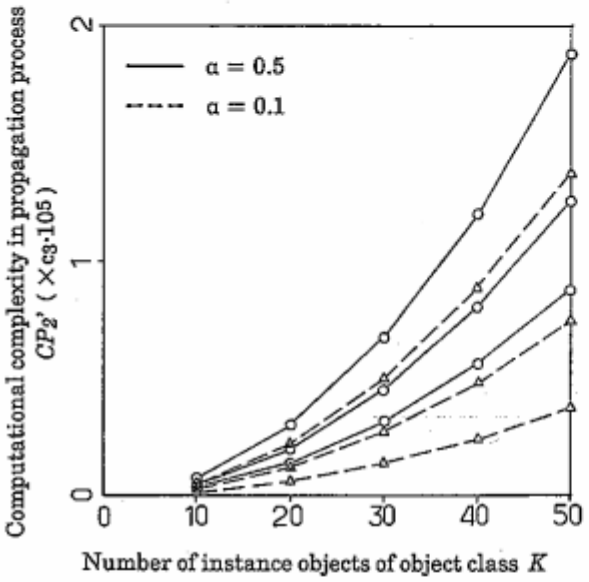


Fig. 4 Evaluation of the Computational Complexity of the Propagation Process of the Model Shown in Fig. 3

moving α and W as parameters. From this result, we conclude: (1) The reduction of computational complexity becomes greater, as α becomes smaller. In this appropriate model, if α is less than 10 %, then the computational complexity, CP_2', is nearly inversely proportional to the number of partitions, W. (2) The magnitude of computational complexity reduction increases sharply with the increase of K. If the number of active instance rules for an instance object is not changed by the increase of instance objects, then an increase in K causes α to decrease. Therefore, the reduction of CP_2' is more accelerated.

As the range of objects to which a template rule is applicable becomes wider, the ratio of instance rules which is active, α, becomes smaller. That is, the partition method has a strong effect. The same condition is needed for partition by rules to be efficient.

4.3.3 Terms Unreduced by the Partition Methods in the Prediction Process

Note that the term, $c_2 \cdot Y^{N_Z}$, in equations [8],[12], and [16] is not reduced by partition methods. Even if c_2 is much smaller than costs c_1 and c_5, this term increases sharply according to the increase of N_Z. Unless N_Z is reduced, system partition alone cannot reduce computational complexity in the prediction process. Because the propagation process is performed for each candidate of state which is not filtered out in the consistency check, the total cost of simulation may diverge exponentially without N_Z decreasing.

5 CRITERIA TO APPLY OUR METHODS

This section summarizes the criteria to apply the methods proposed in section 2 with the propagation process and prediction process.

5.1 Criteria to Apply the Method in the Propagation Process

If all of the following three conditions are satisfied, our methods of partition are applicable to the system and reduce the computational complexity.

(1) *Use of QPT-type simulators*

The reduction of computational complexity is not so remarked in QM-type simulators. Because the numbers of constraints and variables used for simulation are fixed, the reduction of computational complexity is due to only the reduction of iteration number to refer each constraint, N^H. In addition, H in equation [7] for QM-type simulators is usually much less than 1.0.

The methods are useful for QPT-type simulators, because the first terms in equations [11] and [15] are reduced to 1/W and 1/D times, respectively. The reduction of computational complexity is caused mainly by the reduction of the number of total instance rules generated. Therefore, we mainly investigate QPT-type simulators.

(2) *Application to large systems without feedback loops consisting of relatively independent subsystems*

Assumptions, [9] and [10], are required to reduce the computational complexity of the propagation process to the degree expressed as [11]. Assumption

[9] is the condition that each subsystem has far fewer inputs and outputs than its internal variables. For large target systems consisting of relatively independent subsystems, this condition is satisfied usually. Assumption [10] requires the target system to have no feedback loop, as discussed in section 4.3.1. This condition is not very difficult to fulfill in many artifacts, because dataflow in them is usually planned to flow in one direction.

(3) *Expression of rules in a general form*

As discussed in section 4.3.2, the expression of rules must be more general for the effect of our methods with the reduction of computational complexity in the propagation process to be more powerful.

5.2 Criteria to Apply the Method in the Prediction Process

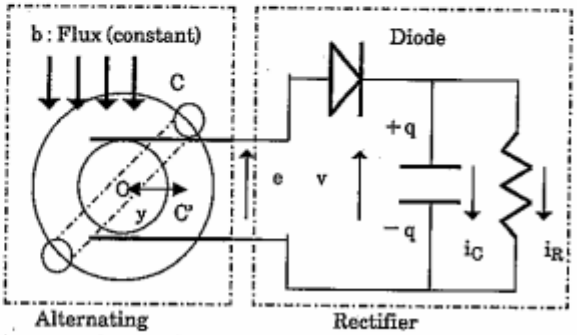
As investigated in 4.3.3, although assumptions [9] and [13] assumed to derive [12] and [16] are easily satisfied, the existence of terms not reduced by our partition methods is critical. Some knowledge about the order of changing is needed to decrease N_Z.

Fortunately, the partition methods have also advantages with acquiring this kind of knowledge. When the simulator simulates the system behavior as a whole, only the knowledge with the order of changing among variables is available. However, it is very rare in practice that sufficient knowledge is given to specify a variable to be changed first. When the system is partitioned into subsystems by the partition method, the knowledge with the order of variable changes among subsystems is also available[3]. This knowledge is likely to be known, even if the orders of changing among variables are not known. For example, (a) variables in subsystem A are known to change faster than variables in subsystem B, (b) subsystem A is known to be able to follow up subsystem B, and so on.

If this knowledge is given, candidates of variables to change are reduced and the computational complexity in the prediction process is very much reduced.

6 AN EXAMPLE OF THE APPLICATION OF THE PARTITION METHOD IN QUALITATIVE REASONING

Here we show that the computational complexity of prediction process can be reduced by the partition method using an example system (Fig. 5). The system consists of two subsystems, an alternating current generator (subsystem A) and a rectifier (subsystem B). As above mentioned, the partition method is equally effective to the QM-type simulator and QPT-type simulator on the computational complexity of the prediction process. We construct a model of this system for a QM-type simulator to make the representation simple.



y : abscissa of conductor C
C' : projection of C on axis y
u : velocity of C'
a : acceleration of C'
e : output terminal voltage
v : terminal voltage of capacitor
q : capacitor charge
i_C : capacitor current
i_R : resistance current

1.Current generator	2. Rectifier	
$e = M_Z^+(y) \times b$	(a) $e \geqq v$ (Diode: on)	(b) $e < v$ (Diode: off)
$\partial y = u$	$\partial q = i_C$	$\partial q = i_C$
$\partial u = a$	$i_C = e - v$	$i_C = -i_R$
$a = M_Z^-(y)$	$i_R = M_Z^+(e)$	$i_R = M_Z^+(e)$
	$v = M_Z^+(q)$	$q > 0$

∂ : qualitative derivative of a variable
M_Z^+, M_Z^-: functional relations of monotonic increase and decrease

Fig. 5 Example of a Target System where the Partition Method Is Effective

We have following heuristics: subsystem A is not affected by subsystem B; subsystem B follows up the

3) The partition method by time-scale [Kuipers 87], [Tanaka 88] partitions the target system using the knowledge with the order of changing among variables. Our partition method uses the order of variable change among subsystems. These two kinds of knowledge are approximately independent. Therefore the availability of these two kinds of knowledge changes with the properties of the target system.

changes of states in subsystem A, and variables in subsystem B depend on the input e determined by subsystem A. Using this knowledge, the variables and constraints are partitioned into two subsystems by our partition method. The behavior of the whole system is simulated as follows. First, the behavior of subsystem A is simulated over an appropriate time interval independently (see Fig. 6a). Then the behavior of input e simulated in step 1 is given to subsystem B. The response behavior of subsystem B is simulated for each state of input e independently (see Fig. 6b).

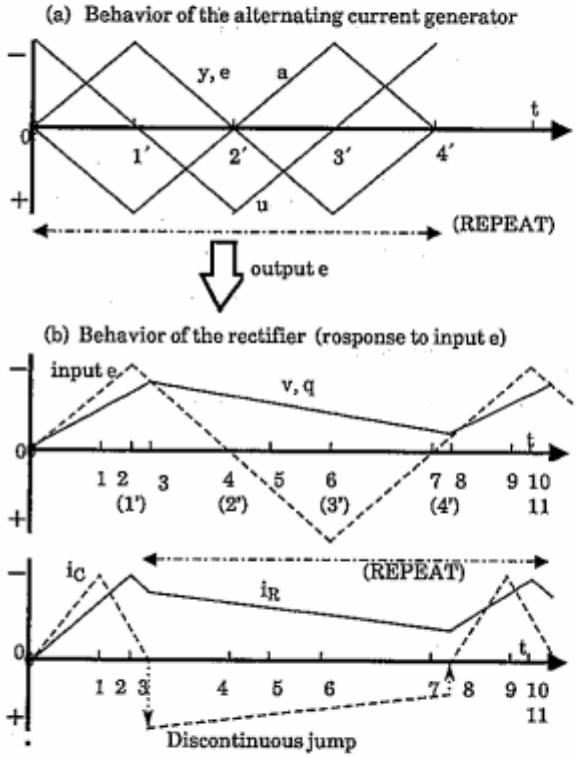


Fig. 6 Simulated Behavior of an Alternating Current Generator and a Rectifier System

In this example, candidates of the next state generated in the prediction process are reduced, because the behavior of each subsystem is separately simulated ignoring the order of changes among variables in the different subsystems. For example, in the simulation as a whole, qualitative state at qualitative time interval, t=(t2,t3), has 8 variables changing with time. 6 out of the 8 variables have multiple candidates of next qualitative values. The state has 96 candidates of the combinations of next qualitative state. In the simulation of subsystem B using the partition method, the same state has only 4 variables changing with time and candidates of next states are reduced to 8.

The total cost of propagation process throughout the simulation is reduced due to the decrease of candidates of next states generated. However the computational complexity of each propagation process in QM-type simulator cannot be reduced.

7 CONCLUSION

This paper proposed two methods for the partition of target systems of qualitative reasoning: (1) partition of variables according to the independence of each subsystem, and (2) partition of a system by the field of applicable rules. They are based on heuristics about the structure and properties of the system. These methods are efficient for enhancing the computational efficiency of qualitative reasoning under the following conditions.

(1) *Conditions for application to the propagation process*

(a) Use of QPT-type simulators

(b) Application to large systems without feedback loops consisting of relatively independent subsystems

(c) Expression of rules in a general form

(2) *Conditions for application to the prediction process*

(a) Use of the order of variable changes among subsystems

Consequently, it is clear that our methods of partition are good supports in simulating the qualitative behavior of large target systems.

There are still remaining problems. One of them is that our methods cannot deal with systems with feedback loops. To eliminate this problem, the simulator must control the order of propagation among subsystems utilizing some methods, for example; (1) least commitment method, and (2) method where the propagation process of each subsystem should be triggered by the determination of input variables.

In future study, our methods must be useful for compiling the simulated results of each subsystem. Each subsystem is functionally modularized, because the two methods partition a system into relatively independent subsystems. The simulated results are

compiled as a function that responds to input states and generates output values to other modules.

ACKNOWLEDGMENT

We would like to thank to researchers of the Fifth Research Laboratory at ICOT for their valuable comments and discussions. Our thanks must also go to Dr. K. Fuchi, Director of the ICOT Research Center, who gave us the opportunity to conduct this research in the Fifth Generation Computer System Project.

REFERENCES

[deKleer 84] Johan de Kleer & John Seely Brown, Qualitative physics based on confluence, Artificial Intelligence 24, pp.7-83 (1984)

[Forbus 84] Kenneth D. Forbus, Qualitative Process Theory, Artificial Intelligence 24, pp.85-168 (1984)

[Forbus 86] Kenneth D. Forbus, The Qualitative Process Engine, Illinois Univ. UIUCDCS-R-86-1288 (Dec. 1986)

[Kuipers 84] Benjamin Kuipers, Commonsense reasoning about causality: Deriving behavior from structure, Artificial Intelligence24, pp.169-203 (1984)

[Kuipers 85] Benjamin Kuipers, Qualitative Simulation of Mechanisms, MIT LCS TM-274 (1985)

[Kuipers 87] Benjamin Kuipers, Abstraction by Time-scale in Qualitative Simulation, Proceedings AAAI-87, pp.621-625 (1987)

[Nishida 87] Toyoaki Nishida, Qualitative Analysis of Discontinuous Changes in Simple Pulse Circuits, Journal of Japanese Society for Artificial Intelligence, Vol.2, No.4, pp.501-510 (in Japanese) (1987)

[Ohki 88] Masaru Ohki, Towards Qualitative Physics, ICOT-TR-221 (1988)

[Robert 77] Robert E. Shostak, On the SUP-INF Method for Proving Presburger Formulas, Journal of the Association for Computing Machinery, Vol.24, No.4, pp.529-543 (Oct. 1977)

[Tanaka 88] Hiroshi Tanaka, Temporal-hierarchical Qualitative Reasoning and its Application to Medicine, Proceedings of Logic programming Conference '88, pp.11-17 (1988)

APPENDIX Differences Between Qupras and the Simulators Based on QPT

Qupras is similar to the simulators based on qualitative process theory (QPT), GIZMO and QPE . All of them aim to solve high-level physical problems. There are some differences of the knowledge representation between them: (a) In Qupras, static relations and dynamic physical laws among objects are described in the unified form of template rule, "physics". (b) Quantity space is not described explicitly in Qupras. (c) Qupras handles physical variables quantitatively as well as qualitatively.

Table 3 compares the terminology of knowledge representation between Qupras and the simulators based on QPT.

Table 3 The Comparison between Qupras and QPT with the Terminology of Knowledge Representation

	Qupras	QPT
Rules to express static relations among objects	Physics	Individual view
Rules to express dynamic changes of objects	Physics	Physical process
Objects to exist before the rule is applicable	Objects	Individuals
Conditions for the rule to be active	Conditions	Precnditions & QuantityConditions
Constraints(functional)	Relations	Relations
Constraints(derivative)	Relations	Influences