TR-432

Enhanced Qualitative Physical Reasoning System

by

M. Ohki, K. Sakane,
J. Sawamoto(Mitsubishi) and Y. Fujii

October, 1988

ICOT

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

(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

Enhanced Qualitative Physical Reasoning System

Masaru Ohki*, Kiyokazu Sakane, Jun Sawamoto**, Yuichi Fuji

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

Abstract

Many expert systems have been used for diagnostic analysis and design using experiential knowledge. However, there are two problems when using only experiential knowledge: (1) when an expert system encounters unexpected problems, it cannot solve the problems by using only the knowledge which it has; (2) it is difficult to acquire experiential knowledge from human experts. The necessity of introducing general principles or basic knowledge to expert systems besides experiential knowledge to solve the above problems is proposed. We proposed Qupras (<u>Qu</u>alitative <u>p</u>hysical <u>r</u>e<u>a</u>soning <u>s</u>ystem), which grasps relations among physical objects and predicts the next state of a physical phenomenon, as a framework for basic knowledge. Qupras has two knowledge representations related to physical laws and objects.

However, there are some incomplete points in Qupras. We have enhanced Qupras; this paper describes the enhanced Qupras. The main enhanced features are (1) inheritance for representation of objects, (2) new primitive representations to describe discontinuous change, (3) meta predicates to evaluate conditions of physical rules, (4) Meta control feature for effective reasoning.

1. Introduction

Many expert systems have been used for diagnostic analysis and design using experiential knowledge. However, there are two problems when using only experiential knowledge.

* Current Address : Central Research Laboratory, Hitachi, Ltd. Higashikoigakubo, Kokubunji, Tokyo, 185, Japan

** Current Address : Computer Works, Mitsubishi Electric Corporation, 325, Kamiyamachiya, Kamakura, Kanagawa, 247, Japan

(1) When an expert system encounters unexpected problems, it cannot solve the problems by using only the knowledge which it has.

(2) It is difficult to acquire experiential knowledge from human experts.

The necessity to introduce general principles or basic knowledge to expert systems besides experiential knowledge to solve the above problems is proposed. General principles and basic knowledge are called deep knowledge in the basic sense. Experiential knowledge is called shallow knowledge, as opposed to deep knowledge. Reasoning using deep knowledge is called deep reasoning. The problem in deep reasoning is how deep knowledge is described and how it is used for reasoning.

One target area of many expert systems is engineering. Physical laws in physics textbooks are considered as one type of deep knowledge in engineering. Qualitative reasoning has been proposed as one way to realize deep reasoning, and much research has been done on it.1,2,3,4,5,6,7) However, in many qualitative reasoning systems other than QPT (Qualitative Process Theory), simultaneous qualitative equations for the system are given, and simultaneous qualitative equations cannot be built using deep knowledge. QPT uses two types of primitive knowledge which are processes to represent changing phenomena and individual views to represent static phenomena. However they do not correspond to physical laws. While determining active processes and active individual views, QPT builds simultaneous qualitative equations and reasons the state and transition of systems. However, the framework of QPT may not be suitable for the deep reasoning system based on physical laws for the following reasons.

(1) Physical laws must be represented by either processes or individual views in QPT.

When an attempt is made to describe physical laws in physics textbooks in QPT, physical laws related to change must be represented as processes, and physical laws related to static phenomena must be represented as individual views, because there are two possibilities of using processes or individual laws to represent physical laws in QPT. That is, there are two primitives to represent physical laws. Therefore, when we try to describe a physical law in QPT, we must decide whether processes or individual laws are used to represent it.

(2) Physical laws must be transformed to qualitative expressions.

Physical laws in physics textbooks are generally represented quantitatively, but not qualitatively. When we try to describe a physical law in QPT, we must transform it to a qualitative formula.

(3) Quantity spaces, which are semi-ordering relations of values which physical quantities change to, must be given in advance.

The necessity to represent, in advance, semi-ordering relations of values to which physical quantities change may not be suitable for a system which reasons the state transition of the system. Moreover, if more than one physical quantity is the same value after changing, and the value cannot be specified in advance, then the relation of the value cannot be described in quantity spaces.

(4) The relation that holds when a physical quantity is changing cannot be described in QPT.

The reason is that the change in a physical quantity is calculated, in QPT, after finding all active processes and all active individual views.

We proposed, in previous papers[8,9], Qupras (Qualitative physical reasoning system) which resolved the above points as follows:

(1) Physical laws are described in one framework for representation.

(2) Quantitative expressions are allowed, and physical quantities are dealt with both qualitatively and quantitatively.

(3) Quantity spaces are not necessary.

(4) Physical laws that holds when physical quantities are changing can be described.

Qupras has two primitive representations. One represents physical laws, called physical rules in Qupras. The other represents physical objects, called objects. Using these representations, Qupras reasons on: (1) relations among objects which are components of physical systems, and (2) the next states of the system following transition.

However, there were some incomplete points in the previous Qupras, which we have tried to resolve. This paper describes the enhanced Qupras. The main enhanced features are:

(1) Inheritance for representation of objects.

(2) New primitive representations to describe discontinuous change.

(3) Meta predicates to evaluate conditions of physical rules.

(4) Meta control feature for effective reasoning.

Although Qupras is a qualitative reasoning system, it may be possible to regard it as a knowledge representation system to represent the physical world with a reasoning mechanism of qualitative reasoning. Some of the above features are enhanced from this viewpoint. Qupras has a representation framework to represent physical laws and objects, and QPT has a representation framework to represent processes and individual views.

Section 2 describes the structure of Qupras, the representation of Qupras, and reasoning in Qupras. Section 3 describes an example analyzed by Qupras.

2. Qupras

2.1 Structure of Qupras

Qupras consists of a translation part and a reasoning part. The translation part translates representations in Qupras, which are mainly knowledge of physical laws and objects dependent on applications, to its internal representation. The reasoning part determines the physical relations that hold among the objects and predicts their next subsequent states. The structure of Qupras is given in Figure 1.

```
Representation section
  +----------------------------------------------------+
  |  1. Object: representation of objects              |
  |     (1) Applied conditions                         |
  |     (2) Relations                                  |
  |                                                    |
  |  2. Physics: representation of physical laws       |
  |     (1) Objects                                    |
  |     (2) Applied conditions                         |
  |     (3) Relations                                  |
  +----------------------------------------------------+

Reasoning section
  +--------------------------------------------------------------+
  |  1. Qualitative reasoning                                    |
  |     (1) Propagation                                          |
  |         Determine status of physical system                  |
  |                 at a given time.                             |
  |     (2) Prediction                                           |
  |         Determine the physical variables,                    |
  |         and values which should change.                      |
  |  2. Qualitative/quantitative evaluation of inequalities      |
  |     Evaluate inequalities, i.e., applied conditions,         |
  |     using relations.                                         |
  +--------------------------------------------------------------+
```

Figure 1 Structure of Qupras

The following is a brief syntax for the representation of Qupras translated to internal representations by the translation part. Its details will be given later. The representation of objects mainly consists of applied conditions and relations. The applied conditions correspond to conditions for the existence of the objects.

Objects satisfying these conditions are called active objects. The relations are expressed as relative equations which include physical quantities when the objects are active. The representation of physical rules mainly consists of objects, applied conditions, and relations. The objects are those necessary to apply the physical rule. The representations of applied conditions and relations are similar to those of objects. Applied conditions are those required to activate a physical rule. Physical rules whose necessary objects are activated and which are satisfied with their conditions are called active physical rules. When a given physical rule is active, its relations are relative equations holding among physical quantities of objects specified in the physical rule, and they correspond to equations of physical laws in physics textbooks. The reasoning part consists of qualitative reasoning and qualitative/quantitative evaluation for inequalities. There are two kinds of qualitative reasoning, propagation and prediction, similar to those given in other research.2,4) However, in Qupras, both quantitative information and qualitative information are used in both types of reasoning. Propagation reasoning determines the state of the physical world at a given time (or during a given time interval). Prediction reasoning determines physical quantities that change with time and infers their values at the next given point in time. Next, propagation reasoning obtains the next states of the physical world using the results of prediction reasoning.

Physical rules are specified in qualitative/quantitative expressions, and qualitative/quantitative evaluation of inequalities (i.e., quantitatively as well as qualitatively) determines the conditions for objects and physical rules. These conditions are expressed as binomial relations or atoms. Conditions are evaluated using relations in active physical rules and active objects. The unit performing this qualitative/quantitative evaluation is called the expression evaluator. It determines conditions using quantitative values as much as possible.

Qupras differs from the earlier qualitative reasoning systems2,3,4) in that it quantitatively handles physical quantities. One of the advantages in to handling physical quantities quantitatively is that it is not always appropriate to handle all physical quantities qualitatively. Treating all variables qualitatively (even when the values of variables are quantitatively known) may lead to ambiguity, because quantitative information is abandoned to handle all variables qualitatively, although the quantitative data is more precise. Another advantage of quantitative treatment is that it is not necessary to transfer quantitative expressions to qualitative expressions.2) Qupras uses quantitative physical laws directly, so it does not need to translate the physical laws to qualitative representations. If only qualitative information is known, Qupras can use it, too.

2.2 Representations of Qupras

2.2.1 Representations of Qupras

In Qupras, physical objects are described by the predicate "object" and physical rules are described by the predicate "physics". Rules for discontinuous changes are described by the predicate "event" or "initial_event". The initial states, which specify objects involved and initial facts, are described by the predicate "initial_state". "initial_state" is regarded as a problem which Qupras has to solve.

First, we explain the syntax for the representation of Qupras. The syntax for the representation of objects is shown in Figure 2. The definition of "object"

```
<object definition> ::=
 "object" <object class name>":"<object variable>
   ["supers"
      <object class name> {"," <object class name>} ";"]
   ["parts_of"
      {<part name> "-" <object class name> ";"}]]
   ["attributes"
      {<attribute name> ["-" <property definition>] ";"}]
   ["initial_relations"
      {<relation> ";"}]
   ["conditions"
      {<simple relation> ";"}]
   ["relations"
      {<relation> ";" }]
   {"state" <state name>
      "conditions"
        {<simple relation> ";"}
      "relations"
        {<relation> ";"}}
  "end" "."

<object variable> ::= <logical variable>
```

Figure 2 Syntax for the definition of "object"

consists of definitions for super classes, parts, attributes, initial relations, relations and states. We call an instance of an object class an object. The super classes specify object classes of the super object. The parts specify object classes of components of the object with the part names. Attributes are physical quantities which the object has. If there are any properties of an attribute, the properties can be defined. The syntax for properties is shown in Figure 3. The conditions are applied conditions. When the conditions are satisfied, the object becomes active. The initial relations mainly specify the relations related to the amount of the attributes known from the first time, even if an object is not active. The relations describe relative equations among the physical quantities of the object, and describe atoms which mainly describe states of the object. When an object is active, its relations are available and they become known relations. The relations

```
<property definition> ::=
    <specifier>{ "(" <option> {"," <option>} ")" }

<specifier> ::= "constant" | "variable"

<option> ::=
    "initial_value" "(" <constant> ")" |
    "initial_sign" "(" <sign specifier> ")" |
    <number>
```

Figure 3 Syntax for properties

are constraints on the attributes of the object class, and cannot be changed. On the other hand, the initial relations do not change and hold unless they are changed by the prediction reasoning or events. The initial relations correspond to initial facts or conditions holding the first time. The representation for states also represents the phenomena of the phase transition. If the conditions of the definition for an object class are satisfied and the conditions in a state definition are satisfied, the object is active in the state and the relations in the state become known relations.

Next, we describe the syntax in relations and conditions. The syntax for relations and conditions is shown in Figure 4. A relation consists of an atom, an expression or an inequality. An atom is equal to an atom in logic. An expression is a quantitative or qualitative algebra expression. A qualitative expression is specified by qualitative equal ':=:'. The meaning of ':=:' is that the sign of the value and derivative of the left hand of a qualitative expression is equal to that of the right hand. A variable is a number, an attribute variable, a global constant, a local variable, a term constant or a derivative variable. A derivative variable is a time derivative of an attribute variable or a local variable. An attribute variable is a physical quantity of an object. An attribute variable is specified as "<attribute name>@<object definition>". <object definition> specifies an object. If an object is part of an other object which is a whole, it is specified as <partname>!<object definition>, where <partname> is the part name of the object in the whole object and <object definition> is the whole object. Global constants describe usual constants, for example, "room_temperature" specifies room temperature. Local variables specify variables which are not attribute variables. Term constants specify symbols. For example, "&off" specifies symbol "off". Conditions consist of an atom, an inequality, or two special predicates, "not" or "all_defined". "not" is satisfied when it is found that the atoms in "not" are not satisfied. The "all_defined" is used to implement the summation with "sum". The physical rule including the "all_defined" in its conditions is tested at the end to check whether the rest of its conditions are satisfied. If the conditions are satisfied, the physical rule becomes active, and the atoms specified by

```
<relation> ::=
    <atom> | <expression> | <inequality>

<conditions> ::= <atom> | <inequality> |
    "not" "(" <atom> ")" |
    "all_defined" "(" <object variable> "," <atom> "," <variables> ")"

<inequality> ::= <variable> <inequality symbol> <variable>
<inequality symbol> ::= ">" | ">=" | "<" | "=<"

<expression> ::=
    <variable> "=" <right expression> |
    <variable> ":=:" <right expression> | <inequality>

<right expression> ::=
    <arithmetic expression> |
    "sum" "(" <attribute name> "," <variables> ")"

<variable> ::=
    <real number> | <attribute variable> | <global constant> |
    <local variable> | <term constant> | <derivative variable>

<derivative variable> ::=
    "ddt" "(" <attribute variable> ")" |
    "ddt" "(" <local variable> ")" |

<attribute variable> ::= <attribute name> "@" <object definition>
<object definition> ::=
    {<part name> "!" {<part name> "!" }} <variable for object>

<global constant> ::= <constant name>

<local variable> ::= <local variable name> "#" "(" <arguments> ")"
<arguments> ::= <argument>{"," <arguments>}
<argument> ::= <object variable>

<term> ::= "&"<term name>
```

Figure 4   Syntax for the relation and conditions

"all_defined" are gathered and the summation of an attribute specified by "sum" related to the gathered objects is calculated.

Examples of "object" for electric power and batteries are given in Figure 5. The super objects of electric power are things, and the super objects of batteries is electric power. The Prolog variable "Battery" in the first line of the definition for batteries is used to describe itself in the "object" definition. The definitions for

```
object thing:Thing
   attributes
      temperature - variable(initial_value(room_temperature)) ;
      heat ;
  end.

object electric_power:Electric_power
   supers
      thing ;
   attributes
      capacity - variable(initial_value(10000));
      voltage_value - constant ;
      voltage ;
      current - variable(initial_value(0)) ;
      resistance - constant(0) ;
   initial_relations
      conduct(Electric_power) ;
  end.

object battery:Battery
   supers
      electric_power ;
   attributes
      total_resistance - constant(positive_zero) ;
      voltage_value - constant(5) ;
      current_max - constant ;
  end.
```

Figure 5 Description for battery

attributes of lower classes have higher priority than those of higher classes. The attribute of "voltage__value" is defined in the super class of "battery", "electric__power", but the definition of "voltage__value" in the class "battery" is used. The initial relations defined in the class "electric__power" are inherited in the class "battery".

The syntax for physical rules is shown in Figure 6. The syntax for physical rules is simpler than the syntax for objects other than the definition of object fields. The definition of physical rules contains the definition for domains. The definition of domain is used to specify domains of physical rules. The object field definition specifies object classes with some information. The first definition of the object field is the standard definition. The specified object class and its lower classes are applicable object classes. The second definition explicitly specifies the applicable object class. The third definition specifies the object classes which should be removed from the applicable object class. The fourth definition specifies the object class which is part of an other object. The fifth definition specifies the

```
<physics definition> ::=
 "physics" <physics name>
   ["domain"
      <domain name> ;]
   "objects"
      {<object field definition> ";"}
   ["conditions"
      {<simple relation> ";"}]
   "relations"
      {<relation> ";"}
  "end" "."

<object field definition> ::=
      <object variable> "-" <object class name> |
      <object variable> "-" "{" <object class name>
              {"," <object class name>} "}" |
      <object variable> "-" <object class name>
           "except" "(" <object class name> {"," <object class name>} ")" |
      <object variable> "-" <object class name> "part_of" <object variable> |
      <object variable> "-" {<object class name>}
           "partname" <name> "part_of" <object variable> |
      <object variable> "-" <object class name> "is_composite_object"
```

Figure 6 Syntax for the definition of "physics"

object class which is part of an other object with a specified part name. The last definition specifies that the object is a composite object.

Figure 7 shows three "physics" illustrating the specification in Qupras for the physical laws of the connection of resistors in a series circuit. The physical rules in Figure 7 can often be seen in physics textbooks as one physical law. The physical law is often represented as follows:

$$R = \sum_{i=1,n} Ri$$

In Qupras, however, it is difficult to represent the physical law as one physical rule, because the way to apply the physical law must be known in order to use the law. In the physical law, the connection of resistors is not explicitly represented. However, the physical rules for the connection of resistors in a series circuit must specify how resistors connect. The first physical rule "connection__start" represents the rule for the sum of resistors of an object belonging to "electric__power" and an object belonging to "electric__conductor". The conditions of the physical rule specify the conditions that the two objects connect and the two objects are conductible. The relations of the physical rule show that the two objects connect electrically, the accumulated value of resistances is equal

```
physics connection_start
   objects
      D - electric_conductor ;
      B - electric_power;
   conditions
      connect(B,D) ;
      conduct(D) ;
      conduct(B) ;
   relations
      electric_connect(B,D) ;
      accumulate_resistance#(B,D) = resistance@B + resistance@D ;
      current@D = current@B ;
  end.

physics connection_intermediate
   objects
      D1 - electric_conductor ;
      D2 - electric_conductor ;
   conditions
      electric_connect(_,D1) ;
      connect(D1,D2) ;
      conduct(D2) ;
   relations
      electric_connect(D1,D2) ;
      accumulate_resistance#(D1,D2) = accumulate_resistance#(_,D1) +
           resistance@D2 ;
      current@D2 = current@D1 ;
  end.

physics connection_end
   objects
      D1 - electric_conductor ;
      B - electric_power;
   conditions
      electric_connect(_,D1) ;
      connect(D1,B) ;
      conduct(B) ;
   relations
      electric_connect(D1,B) ;
      whole_connect ;
      total_resistance@B = accumulate_resistance#(_,D1) ;
      output_current@B = voltage_value@B / total_resistance@B ;
  end.
```

Figure 7 Description for the physical rule of the connection of resistorsin a series circuit in Qupras

to the sum of their resistances, and the electric current of an object belonging to "electric__battery" is equal to the electric current of an object belonging to "electric__power". The second physical rule specifies the rule for the sum of resistors of two objects belonging to "electric__conductor". The last physical rule represents the rule for the sum of resistors of an object belonging to "electric__conductor" and an object belonging to "electric__power". When the last physical rule is activated, it is found that all objects belonging to "electric__conductor" and "electric__power" electrically connect, and the total resistance of the object belonging to "electric__power" and the output of the electric current from the object belonging to "electric__power" can be obtained.

The syntax for "event" and "initial__event" represents discontinuous changes. "event" specifies a general rule for discontinuous change. The syntax for "event" is represented in Figure 8. Its syntax is similar to the syntax of "physics". The

```
<event definition> ::=
 "event" <event name>
   ["domain"
      <domain name> ";"]
   "objects
      {<object field definition> ";"}
   "conditions"
      {<simple relation> ";"}
   "actions"
      {<action> ";"}
  "end" "."

<action> ::=
    "remove" "(" <relation> ")" | "add" "(" <relation> ")"
```

Figure 8 Syntax for the definition of "event"

actions in "event" correspond to the relations in "physics". The actions specify that meta operations perform destructive updating to known initial relations as follows:

(1) Removal of a relation from the initial relations.

(2) addition of a relation to the initial relations.

The syntax for "initial__event" is equal to the syntax for "event" except when the predicate name "event" is "initial__event". "initial__event" is used in specific, not general, discontinuous changes. When the name of "initial__event" is specified in "initial__state", described later, the definition of "initial__event" becomes useful and the event can become active only once. Even if the conditions of "initial__event" are satisfied more than once, only the first satisfaction is available. However "event" is activated whenever its conditions are satisfied.

Next, the syntax for "initial__state" is explained. It is shown in Figure 9.

```
<initial_state definition> ::=
 "initial_state" <initial_state name>
   ["supers"
      <initial_state name> {"," <initial_state name>} ";"]
   "objects"
      {<variable for object> "-" <object class name> ";"}
   ["initial_relations"
      {<simple relation> ";"}]
   ["events"
      {<event name>"("<variable for object> {"," <variable for object>}")" ";"]
   ["controls"
      <control definitions>]
  "end" "."

<control definitions> ::=
   ["no_interesting" "(" "[" <attribute variable>
      {"," <attribute variable> } "]" ")" ";" ]
      ["change_with_priority" "(" "[" "(" <attribute variable>
             "," <priority> ")"
      {"," "(" <attribute variable> "," <priority> ")" } "]" ")" ";"]
   ["physical_domain" "(" "[" <domain name>
      {"," <domain name>} "]" ") ";"]
```

Figure 9 Syntax for the definition of "initial__state"

"initial__state" specifies an initial state which specifies objects involved, initial relations, initial events and control information. Initial relations define simple relations and set initial values of some attributes of objects. Control information is used to improve the execution efficiency of reasoning.

(1) no__interesting: The changes of the specified attribute variables are ignored even if they are changing.

(2) change__with__priority: The priorities of attribute variables for changes are represented by pairs of an attribute variable and its priority. The attribute variables with the highest priority are preferentially used as changing variables whenever the attribute variables are changing.

(3) physical__domain: Physical rules to be used are restricted to physical rules with the specified domain names.

Definitions of "initial__state" are regarded as problems which Qupras must solve.

2.2.2 Translation to Internal Representation

The translation part translates the representation of Qupras to internal representations in order to improve the efficiency of reasoning. The main features of the translation part are as follows:

(1) To make instance representations for object definitions and template rules for physical rules.

(2) To differentiate numerical expressions.

(3) To make a variable dictionary.

(4) To translate attributes to internal variable representations to access a variable dictionary.

Each instance definition for all objects specified in the "initial__state" in which they are used is translated to an internal representation. Even if there are definitions of objects which are not specified in the "initial__state" in which they are used, the internal representation for these definitions is not made.

An internal template rule for a physical rule is made when the objects belonging to the defined class or its lower classes for all objects specialized in its object field definitions are specified in "internal__state". Even if only one object belonging to the defined class or its lower classes is not specified in "internal__state", no internal template rule is made, because the physical rule must not be activated. All instance definitions for all objects specified in the "internal__state" are made, but no internal physical rules corresponding to objects are made. Internal physical rules are made corresponding to physical rules. Even if there are several combinations of objects which satisfy object class definitions of a physical rule, only one internal physical rule is made, because if internal representations for all combinations of objects are made, the memory required for internal physical rules becomes huge.

Numerical expressions in the relations and the initial relations of the object definitions are differentiated. When differentiating, the declaration of an attribute is referenced to show whether it is constant or variable. If the attribute is constant, its time derivative is not made. However, numerical expressions in physical rules are not differentiated when internal physical rules are made, because objects for object classes defined in a physical rule have not been precisely determined yet. They are determined in reasoning, and the numerical expressions are differentiated at the time.

Attributes of all the objects declared in "initial__state" are translated to internal variable representations and entered in a dictionary for variables. Values of all attributes are stored in the variable dictionary. To find the value of an attribute, the value is obtained by looking up the variable dictionary.

2.3 Reasoning in Qupras

There are two reasoning mechanisms in Qupras as described in Section 2.1 above. One is qualitative reasoning and the other is qualitative/quantitative evaluation for inequalities performed by the expression evaluator. We describe the latter first.

The expression evaluator tests whether the conditions in the definitions of the objects, physical rules and events are proven by the known relations obtained from active objects and active physical rules, and from initial relations. Relations are given as expressions and terms. We ignore the solution of terms in this discussion, because their evaluation is very simple.

We wish to solve nonlinear simultaneous inequality expressions to test the conditions in the objects, physical rules and events. We have used more than one algorithm to build the expression evaluator, because we do not know any efficient algorithms for nonlinear simultaneous inequality expressions. The expression evaluator consists of three parts:

(1) Nonlinear inequality solver based on the interval method 11) This solver can deal with nonlinear inequality expressions, but not completely. For example, from these inequality expressions:

$X > Y, Z = X - Y$

the following result cannot be obtained by this solver:

$Z > 0.$

(2) Linear inequality solver based on the SupInf method 10) This solver solves linear inequality expressions completely, for example, the above example can be solved by this solver. However, it cannot deal with nonlinear inequality expressions. For example, from these inequality expressions:

$X > 0, Y > 0, Z = X * Y$

the following result cannot be obtained by this solver:

$Z > 0.$

However, this example can be solved by the solver of (1).

(3) Nonlinear simultaneous equation solver based on the Grobner base This solver can process nonlinear simultaneous expressions and reduce the expression as far as possible, but cannot deal with inequalities.

We have connected the three solvers as shown in Figure 10. Linear and nonlinear expressions and inequalities are entered in the solver based on the interval method. The results on inequalities are entered in the solver based on the SupInf method. Linear and nonlinear equations are entered in the solver based on the

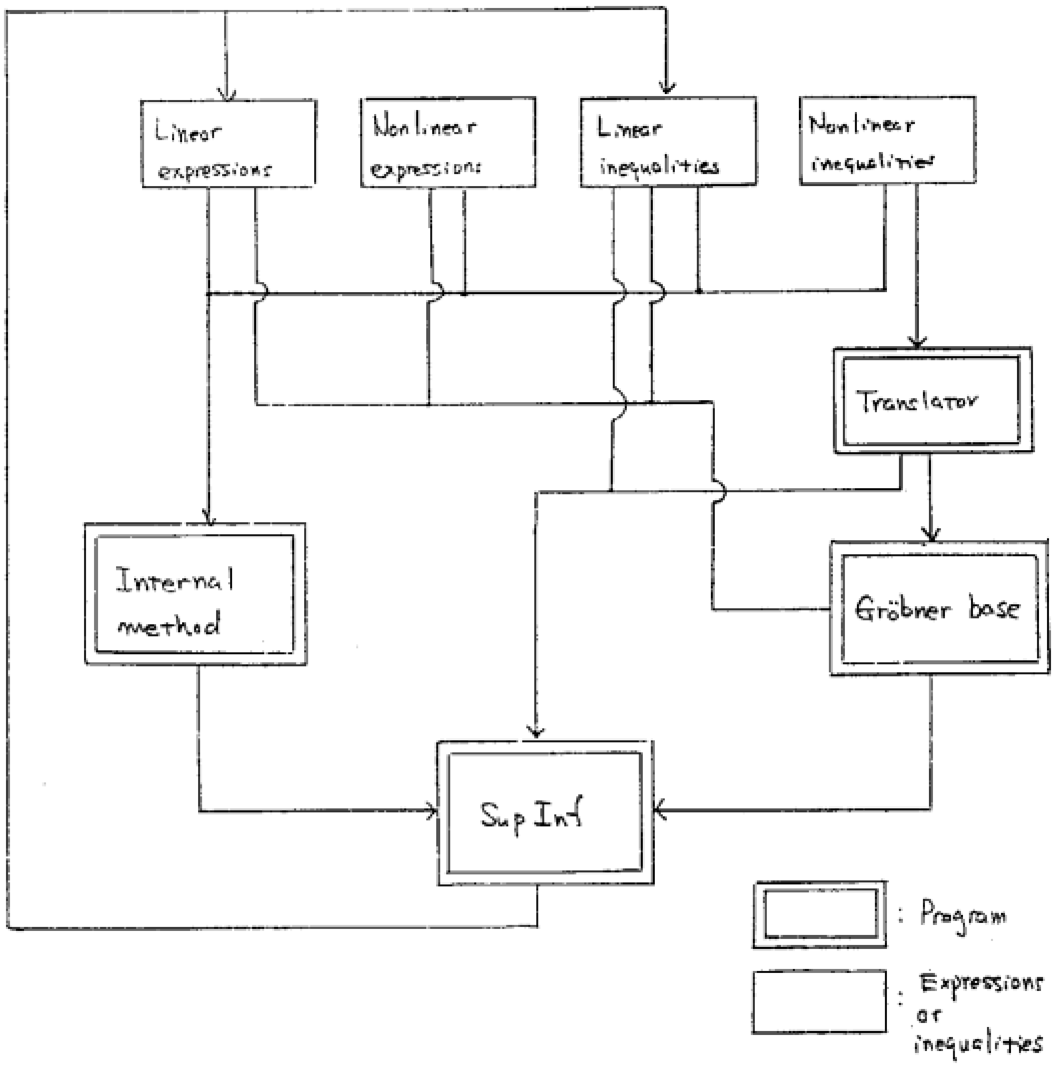


Figure 10 Expression evaluator

Grobner base[12)]*. The linear expressions which are output from the solver are entered in the solver based on the SupInf methcd. Linear inequality expressions are entered in the solver based on the interval method and the solver based on the SupInf method. Nonlinear inequalities are entered in the solver based on the interval method, and translated to nonlinear equations and linear inequalities.

* We would like to thank Dr. Sakai and Dr. Aiba for their willing agreement to use their Grobner base program and for their cooperation.

Some of the nonlinear equations are entered in the solver based on the Grobner base, and some of the linear inequalities are entered in the solver based on the SupInf method. If a new result is obtained, the result from the solver based on the SupInf method is entered in the solver based on the interval method. An expression evaluator connected like this can solve more broad expressions than ones which each solver alone can solve.

Now let us turn to qualitative reasoning in Qupras. We have already stated that there are two types of qualitative reasoning in this system, propagation and prediction. Before performing the two reasonings, all initial relations of the objects defined in the initial state are set to the known relations. Initial relations are mainly used to set initial values of attribute variables. If there is not an explicit change to an initial relation, the initial relation hold. The explicit changes are the prediction of the next value in the prediction reasoning or the update for variables in processing events.

Propagation reasoning is used to find active objects whose conditions are satisfied by the known relations, and the physical rules that hold among the active objects at one time or during a time interval. Qupras performs this reasoning as follows:

(1) Try to find inactive objects whose conditions are satisfied by the known relations using the expression evaluator.

(2) If such inactive objects are found, change them to active objects and add their relations to the known relations.

(3) Next, try to find the inactive physical rules whose necessary objects are active and whose conditions are satisfied by the current known relations.

(4) If such inactive physical rules are found, change them to active physical rules and add their relations to the known relations.

(5) If any remaining inactive object or physical rule was activated in the last sequence through steps (1) to (4), repeat steps (1) to (4); otherwise, go to (6).

(6) If there are some physical rules using "not" in their conditions and the atoms in "not" are not provable, the conditions including "not" are satisfied. Again, repeat steps (1) to (5) until any physical rule or object is not activated.

(7) Finally, test whether physical rules using "all_defined" are active. If the physical rules are active, the atoms specified by "all_defined" are gathered and the summation of an attribute specified in "sum" related to the gathered objects is calculated. (The current implementation for "not" and "all_defined" is sound, because an undetected contradiction is caused.)

If a contradiction is detected while propagating, the propagation reasoning returns the null next state; otherwise, the propagated result is returned.

Prediction reasoning is used to determine the attribute variables changing with time from the known relations which are the result of propagation reasoning. Then the new values or the new intervals of the changing variables at the next specified time or during the next time interval are sought. Qupras updates the values of changing variables according to the sought values or intervals. The updated values are used as the initial relations at the beginning of the next propagation reasoning. We describe the procedure of this reasoning briefly below.

(1) Find the attribute variables changing with time. They are the variables, which are non zero values, enclosing the ddt operator in the known relations.

(2) Next, try to find the values or the range of values to which the variables change. They are as follows:
(a) The values required for currently inactive objects and inactive physical rules to become active.
(b) The values required for currently active objects and active physical rules to become inactive.

(3) Select the values nearest to the current values from the values found in (2) using the expression evaluator with the current known relations.

(4) If there are any changes which are transitions from concrete values, it is assumed that these changes are instantly performed, and all the changes are instantly performed (instant change). If there is no instant change and there are any changes which are transitions from intervals, it is assumed that time is necessary for the transition (non-instant change).

Instant change:

(5) Change the value of the changing attribute variable to the nearest value, and remove the initial relations contradicting the nearest value.

Non-instant change:

(6) Obtain a combination of all changing variables. Steps (7) and (8) are performed with every combination.

(7) Change the value of the changing variable to the nearest value, and remove the initial relations contradicting the nearest value.

(8) If a contradiction is found while propagating, the combination is abandoned.

Steps (6) to (7) are similar to QSIM [13], and we reduce the inconsistent prediction in the propagation reasoning.

Qupras does not use quantity spaces as in QPT.[3] It finds the information corresponding to quantity spaces in step (3). If there are several changing variables in step (4), there may be an ambiguity in reasoning which generates several next states. However there is the possibility that Qupras can decrease the ambiguity by using quantitative information of attribute variables.

The reasoning for events is similar to the prediction reasoning. The conditions of events and initial events are checked after propagation reasoning. If the conditions of the definitions for events are satisfied, the actions in the definitions are performed, and when the process for the event is finished, propagation reasoning states with the result of the event. It is assumed that actions of events take precedence over changing variables. If there are any activated events, the process for the prediction reasoning is not performed. The procedures for events are as follows:

(1) Remove initial relations which are specified to be removed in the actions from the old known relations.

(2) Add initial relations which are specified to be added in the actions to the new known relations.

(3) Select one initial relation from the old known relations, and enter it in the new known relations.

(4) If a contradiction by entering the relation is detected, the relation is abandoned. Go back to (3), where the previous new relations hold.

(5) If all initial relations are entered in the new known relations, the process for events is terminated.

3. Example

We discuss the example of an flashlight.

Consider the flashlight shown Figure 11. Qupras infers the existing objects in the flashlight and the physical rules holding among the objects, and predicts the next states of the flashlight. There are three object descriptions for the flashlight; an electric light bulb, a battery and a switch. The switch description in Figure 12 is given as another example of an object description. Some physical rules for the flashlight were shown in Figure 3. There are fourteen physical rules and event definitions for the flashlight, as follows:

(1) switch on
This event definition describes the event when a switch is turned on. Its description is shown in Figure 13.

```
+-----------------------+
|                       |
|-                     +|
o                      | /-\
\                      +--$ |
 \  switch              --$ | small light bulb
o                      | \-/
|+                     -|
|                       |
+--------|:|:|:---------+
       + battery -
```

Figure 11 Structure of a flashlight

```
object control_equipment:Control_equipment
   supers
      thing :
   attributes
      voltage ;
      current ;
      resistance - constant(0) ;
  end.

object switch:Switch
   supers
      control_equipment ;
   attributes
      button - nonnumerical_variable ;
      state_of_button - nonnumerical_variable ;
  end.
```

Figure 12 Description of switch

(2) switch off

This event definition describes the event when a switch is turned off.

(3) electrical connection

Shown in Figure 7.

(4) increase current

This physical rule describes the increase of electric current when it is less than the maximum electric current. Here, the change of the electric current is not regarded as an instantaneous phenomenon. If the change is regarded as an instantaneous phenomenon, an event definition should be used.

```
event switch_on
   objects
      Switch - switch ;
   conditions
      button@Switch = &on_button ;
      state_of_button@Switch = &off_state ;
   actions
      remove(state_of_button@Switch = &off_state) ;
      add(state_of_button@Switch = &on_state) ;
      add(conduct(Switch)) ;
  end.
```

Figure 13 Description of the event "switch__on"

(5) steady current
This physical rule describes that an electric current does not change when it is equal to the maximum electric current.

(6) decrease current
This physical rule describes the decrease of an electric current when a circuit is not closed.

(7) using battery
This physical rule describes a phenomenon in which a battery consumes power and its capacity decreases.

(8) emptiness of battery
This physical rule describes the decrease of an electric current when the battery is empty.

(9) heat generation by electricity
This physical rule specifies the relation between heat generation and the electric current.

(10) heat
This physical rule specifies the relation between heat and heat generation.

(11) temperature and heat
This physical rule specifies the relation between heat and temperature.

(12) light generation
This physical rule describes that generation of light in an electric lamp increases when its temperature is greater than the temperature required to generate light.

Qupras must be supplied with a description of an initial state to begin reasoning. An example of an initial state is shown in Figure 14. It specifies that

```
initial_state normal_environment
   initial_relations
      room_temperature = 20 ;
  end.

initial_state light_bulb_eq_26
   supers
      normal_environment ;
   objects
      Light_bulb - light_bulb_type1 ;
      Switch - switch ;
      Battery - battery ;
   initial_relations
      connect(Battery,Switch) ;
      connect(Switch,Light_bulb) ;
      connect(Light_bulb,Battery) ;
      state_of_button@Switch = &off_state ;
      button@Switch = &on_button ;
   events
      initial_event1(Light_bulb,Switch) ;
  end.

initial_event initial_event1
   objects
      Light_bulb - light_bulb_type1 ;
      Switch - switch ;
   conditions
      temperature@Light_bulb >= light_generating_temperature@Light_bulb ;
   actions
      remove(button@Switch = &on_button) ;
      add(button@Switch = &off_button) ;
  end.
```

Figure 14 Description of an initial state

a battery, a switch and a light bulb are connected to each other, the state of the switch button is off, and the button is pushed. It also specifies an initial event named "initial__event1". The initial event describes that if the temperature of the light bulb is greater than the temperature required to generate light, then the switch button is pulled.

Using the description of the objects, physical rules, and the initial state, Qupras performs propagation reasoning and finds the active objects and their relations in the flashlight at the initial time. In the initial state, the event "switch__on" is activated because its conditions are satisfied. Qupras changes the switch to conductible, and performs the propagation reasoning. The initial state and its second state are shown in Figure 15. There are three active objects

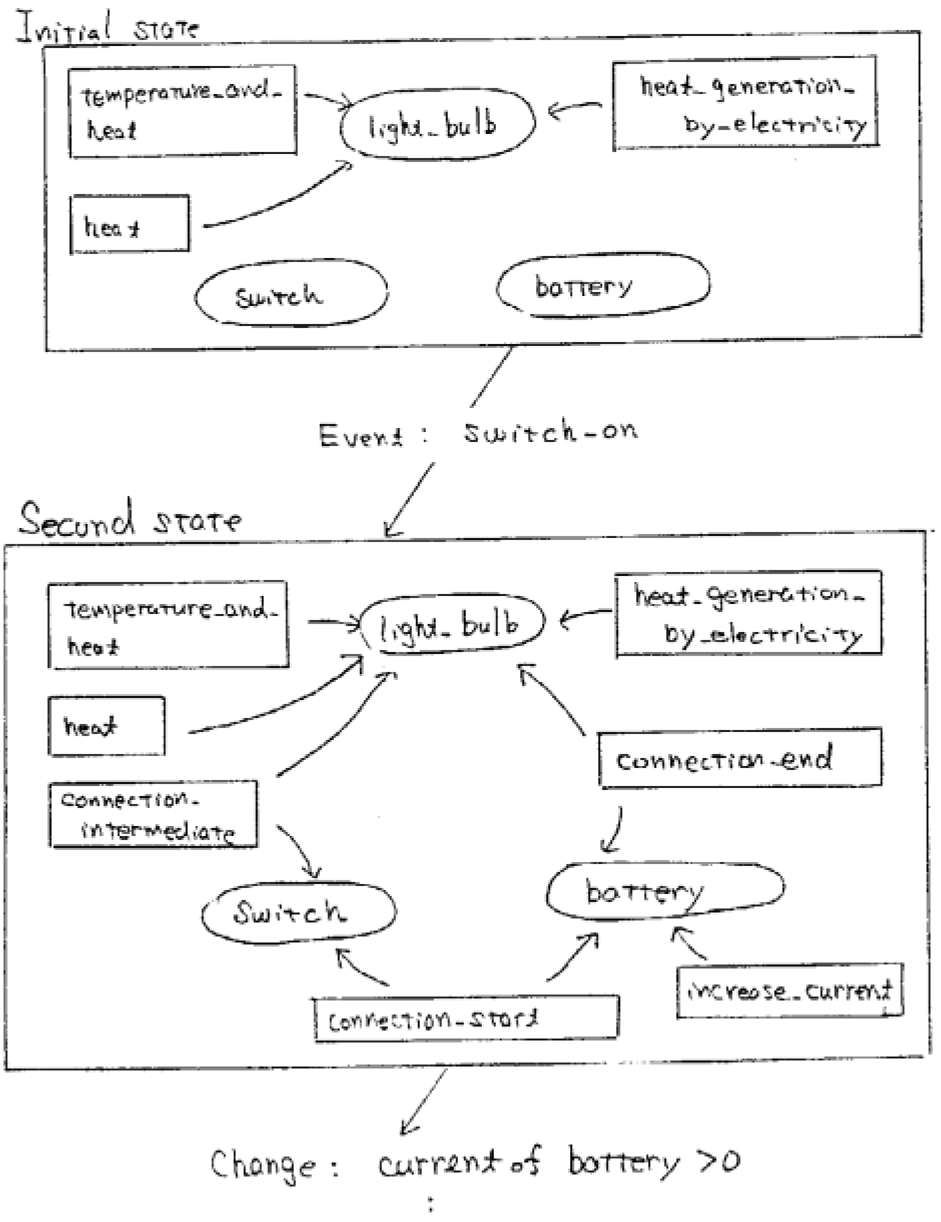
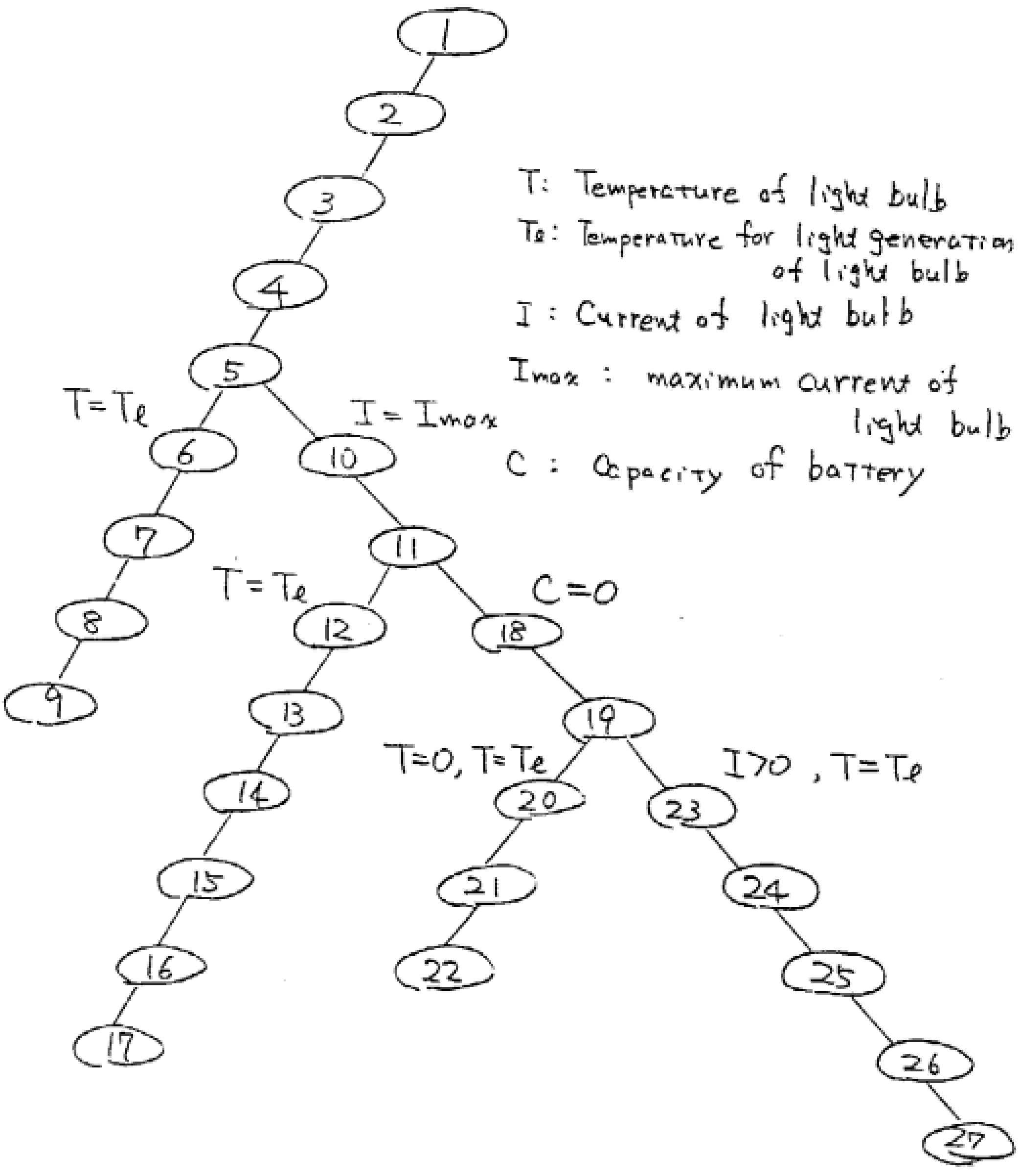


Figure 15 Initial state and second state of the flashlight

(circled) and seven physical rules (in broken line squares) in the second state. Qupras tries to find attributes of objects that vary over time, because no event is activated in the second state. There are four changing attributes of objects: the battery current, a switch and a light bulb, and the heat generation of the light bulb. Qupras predicts that their next values are greater than 0, because all their values are 0. The following states are shown in Figure 16. There are several forks because the prediction is ambiguous. For example, the first fork results from either the change when the temperature of the light bulb becomes the temperature to generate light or the change when the electric current becomes the maximum electric current, whichever happens first. The correct sequence is that the current of the circuit becomes its maximum current first, the battery



T: Temperature of light bulb
Te: Temperature for light generation of light bulb
I: Current of light bulb
Imax: maximum current of light bulb
C: Capacity of battery
T=Te
I=Imax
T=Te
C=0
T=0, T=Te
I>0, T=Te

Figure 16 Flashlight transitions

capacity becomes zero, the temperature of the light bulb becomes the temperature to generate light, and the initial event is activated and the circuit becomes the state to be opened.

4. Discussion

We have enhanced the previous Qupras related to the following points to describe physical laws as deep knowledge and to reason about the physical world using deep knowledge:

(1) Inheritance for representation of objects.

(2) New primitive representations to describe discontinuous change.

(3) Meta predicates to evaluate conditions of physical rules.

(4) Meta control feature for effective reasoning.

Most of these features are necessary to write the example in Section 3. If we would like to decrease the ambiguities of the reasoning as shown in Figure 16, we can do so by using the meta control feature. When we specify the priority of change for attributes according to common sense as follows:

electric current > temperature > battery capacity

all ambiguities can be removed in Figure 16.

Enhanced Qupras is implemented in Quintas-Prolog.[14] The total number of program lines is about 14 k lines with comments. The rough number of lines in each part is as follows:

(1) Translation part ------------------ 7 k lines

(2) Qualitative reasoning part ---- 3 k lines

(3) Evaluation part ------------------ 4 k lines

The value for the evaluation part includes the number of the line of the Grobner base program, which is about 0.7 k lines.

We tried to write several systems in Qupras to find new primitives and new features to represent the physical world. When we used Qupras to reason about the physical world, heuristic knowledge was often necessary. For example, the heat loss should be negligible. We think that new primitive representations are necessary to reduce the necessity for writing heuristic knowledge, and new features are necessary to write heuristic knowledge and to reason about the physical world using both deep knowledge and heuristic knowledge. We will try to use Qupras in several applications. For example, we plan the following:

(1) Knowledge compilation from knowledge of Qupras to diagnostic rules.

(2) Design support to check whether users' designs are safe by simulation.

(3) Automated designs from features using knowledge of Qupras.

Acknowledgment

This research is based on discussions in the ICOT FAI (Foundations of Artificial Intelligence) working group. We wish to express our thanks to its head, Professor Mizoguchi (Science University of Tokyo), and its members. We also wish to thank researchers in the fifth research laboratory in ICOT. Finally, we must express our thanks to Dr. Kazuhiro Fuchi, Director of the ICOT Research Center, who provided us with the opportunity of doing this research in the Fifth Generation Computer Systems Project.

References

1) Bobrow, D.G. : Special Volume on Qualitative Reasoning about Physical Systems, Artificial Intelligence, 24, 1984.

2) de Kleer,J. and Brown,J.S. : Qualitative Physics Based on Confluence, Artificial Intelligence 24, pp.7-83, 1984.

3) Forbus, K.D. : Qualitative Process Theory , Artificial Intelligence 24, pp.85-168, 1984.

4) Kuipers, B. : Commonsense Reasoning about Causality:Deriving Behavior from Structure, Artificial Intelligence 24, pp.169-203, 1984.

5) Nishida, T. and Doshita, S. : Reasoning about Discontinuous Change, AAAI-87, pp.643-648,1987.

6) Yamaguchi, T., Mizoguchi R., Taoka N., Kodaka H., Nomura Y. and Kakusho O. : Basic Design of Knowledge Compiler Based on Deep Knowledge, J. of Japanese Soc. for Artif. Intel., 2, pp.333-340, 1987. (in Japanese)

7) Ohwada H., Mizoguchi F. and Kitazawa Y. : A Method for Developing Diagnostic Systems based on Qualitative Simulation, J. of Japanese Soc. for Artif. Intel., 3, pp.617-626, 1988. (in Japanese)

8) Ohki, M. and Furukawa, K. : Toward Qualitative Reasoning, ICOT-TR 221, 1986.

9) Ohki, M., Fuji, Y. and Furukawa, K. : Qualitative Reasoning based on Physical Laws, Trans. Inf. Proc. Soc. Japan, 29, pp.694-702, 1988. (in Japanese)

10) Ohki, M., Sawamoto, J., Sakane, K. and Fuji, Y. : A Constraint Logic Programming Language based on the Sup-Inf Method, Proc. of 5th Conf. JSSST, pp.49-52, 1988. (In Japanese)

11) Simmons, S. : Commonsense Arithmetic Reasoning, AAAI-86, pp.118-128, 1986.

12) Sakai K. and Aiba A. : CAL : A Theoretical Background of Constraint Logic Programming and Its Application, ICOT TR-364, 1988.

13) Kuipers, B. : Qualitative Simulation of Mechanisms, MIT LCS TM-274, 1985.

14) Quintas Computer System Inc. : Quintas Prolog Reference Manual, 1985.