TR-0790

Design Support to Determine Range of Design Parameters by Qualitative Reasoning

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

August, 1992

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
(03)3456-3191～5
Telex ICOT J32964
Minato-ku Tokyo 108 Japan

Design Support to Determine Range of Design Parameters by Qualitative Reasoning

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

Central Research Laboratory, Hitachi, Ltd.,
1 - 280 Higashi-Koigakubo,
Kokubunji, Tokyo 185, Japan

Abstract

Qualitative reasoning has been applied in diverse fields of engineering, mainly for diagnosis but also for design. Here we present a new application to design: suggesting valid ranges for design parameters after structure has been determined. This important design step is implemented by using an envisioning mechanism that uses qualitative reasoning to determine all possible behaviors of a system. Our method: (1) all possible behaviors are found by envisioning with design parameters whose values are initially indeterminate and with whatever specifications the designer has; (2) if several behaviors are found, the designer selects those that are preferable.

The design-support system Desq (Design support system based on qualitative reasoning), based on an earlier qualitative reasoning system Qupras (Qualitative physical reasoning system), is improved in three features: (1) envisioning, (2) propagating new constraints on constant parameters, and (3) solving constraint in parallel.

Like Qupras, the Desq system can deal with quantities qualitatively and quantitatively. If the parameters can be expressed quantitatively, we may therefore be able to determine the quantitative ranges, which are more useful than qualitative values.

1 Introduction

Although many expert systems have been used recently in diverse fields of engineering, several problems still exist. One is the difficulty of building knowledge bases from the experience of human experts, and another is that these expert systems have not be able to deal with situations that cannot be imagined [10]. Reasoning methods using deep knowledge, which is the fundamental knowledge of a domain, are expected to solve these problems. Qualitative reasoning [2] determines dynamic behaviors, which are the states and state changes of a dynamic system by using deep knowledge of the dynamic system. Another feature of qualitative reasoning is that it can deal with quantities qualitatively. So far, there have been many applications of qualitative reasoning to engineering [12-14]. The main application has been to diagnosis [25], [20], but recently there have also been applications to design [11], [26].

In this paper, we show a new application to design that supports decisions by suggesting valid ranges for design parameters after the structure of the designed system has been determined. This application is not more innovative than the previous applications [11], [26] to design, but it is nonetheless one of the important steps of design [3]. Our previous paper [18] described the intermediate state of our research, and this paper shows the final state.

The key to the design support is using an envisioning mechanism, which finds possible behaviors of the dynamic system, to determine the ranges of those design parameters whose values are indeterminate. When the design parameters whose values a designer wants to determine are indeterminate, all possible behaviors under those indeterminate parameters can be predicted by the envisioning process. If the designer gives some specifications, the number of the possible behaviors may be reduced. In the envisioning, some hypotheses may be made to obtain each behavior. The main reason hypotheses are made is that conditions written into the definitions of objects and physical rules cannot be evaluated when the design parameters are indeterminate. Among the possible behaviors obtained, more than one behavior desired by the designer is expected to exist. The designer can therefore select the behaviors he prefers. Although the designer may not know the values of the design parameters, he knows the desired behavior. The values of the indeterminate parameters can be derived from the hypotheses that result in the desired behavior.

The method of determining valid ranges for design parameters can be summarized as follows:

(1) All possible behaviors are found by envisioning with design parameters whose values are initially indeterminate and with whatever specifications the designer has,

(2) If several behaviors are found, the designer selects those that are preferable.

We used a qualitative reasoning system Qupras (Qualitative physical reasoning system) [15-17] to construct a decision support system Desq (Design support system based on qualitative reasoning) that suggests valid ranges for design parameters.

Qupras, using knowledge about physical rules and objects after being given an initial state, determines the following:

(1) Relations between objects that are components of physical systems.

(2) The state transitions of the system.

To construct Desq, we extended Qupras as follows:

(1) Envisioning

In Qupras, if a condition of a physical rule or an object cannot be evaluated, Qupras asks the user to specify the condition. We extended Qupras to allow it to continue reasoning by assuming an unevaluated condition.

(2) Propagating new constraints on constant parameters

There are two types of parameters in Desq, variable and constant. Design parameters are constant because they do not change with time. In the envisioning process, constraints related to some constant parameters become stronger because conditions in the definitions of physical rules and objects are hypothesized. The constraints propagate to the subsequent states. To find constraints for constant parameters, Desq calculates the ranges of all constant parameters after determining the model.

(3) Solving constraint in parallel

Qupras uses a combined constraint solver consisting of three basic constraint solvers all written in ESP: a Supinf method constraint solver, an Interval method constraint solver, and a Groebner base method constraint solver. The processing load of the combined constraint solver was heavy, so we converted it to KL1, which is a parallel logic programing language supported by ICOT (Institute of New Generation Computer Technology), to speed up processing.

Like Qupras, Desq can deal with quantities qualitatively and quantitatively. If the parameters can be given as quantitative values, we may therefore be able to get quantitative ranges. Although the usual qualitative reasoning, such as that described in Ref. 8, gives qualitative ranges, quantitative ranges may be preferable for decision support.

Section 2 of this paper shows how Desq suggests ranges for design parameters, Section 3 describes the system organization of Desq, Section 4 shows examples of Desq suggesting the value of resistors in a DTL circuit and in a Schmitt trigger circuit, Section 5 describes related works, and Section 6 summarizes the paper.

2 Method of determining design parameters

In design, there are many cases in which a designer does not directly design a new device but instead simply modifies an old device. Sometimes designers only change parameters of components in a device to satisfy the requirements. In such cases, the designer knows the structure of the device and needs only to determine the new values of the components. This is a common situation in electronic circuit design, so we apply qualitative reasoning to these kinds of design decisions.

The key process used to determine design parameters is envisioning. Our method is as described in Section 1:

(1) All possible behaviors are found by envisioning with design parameters whose values are initially indeterminate and with whatever specifications the designer has.

(2) If several behaviors are found, the designer selects those that are preferable.

If a condition in the definition of a physical rule or an object cannot be evaluated, Desq hypothesizes one case in which the condition is valid and another in which it is not. Then Desq separately searches each case to find all possible behaviors. This method is called envisioning, and is the same as the reasoning described in Ref. 8. If a contradiction is detected, the reasoning is abandoned. If no contradiction is detected, the reasoning is valid. Finally, Desq finds several possible behaviors of a device. The specification a designer gives is used to prune undesired behaviors.

The characteristics of this approach are as follows:

(1) Only deep knowledge is used to determine design parameters.

(2) All possible behaviors with regard to indeterminate design parameters are found when any specifications are not given. This kind of information may be useful in safety design or danger estimation.

(3) Ranges of design parameters giving preferable behaviors are found. If a designer uses numerical CAD systems, SPICE for example, for more detail analyses he need not simulate values outside the ranges.

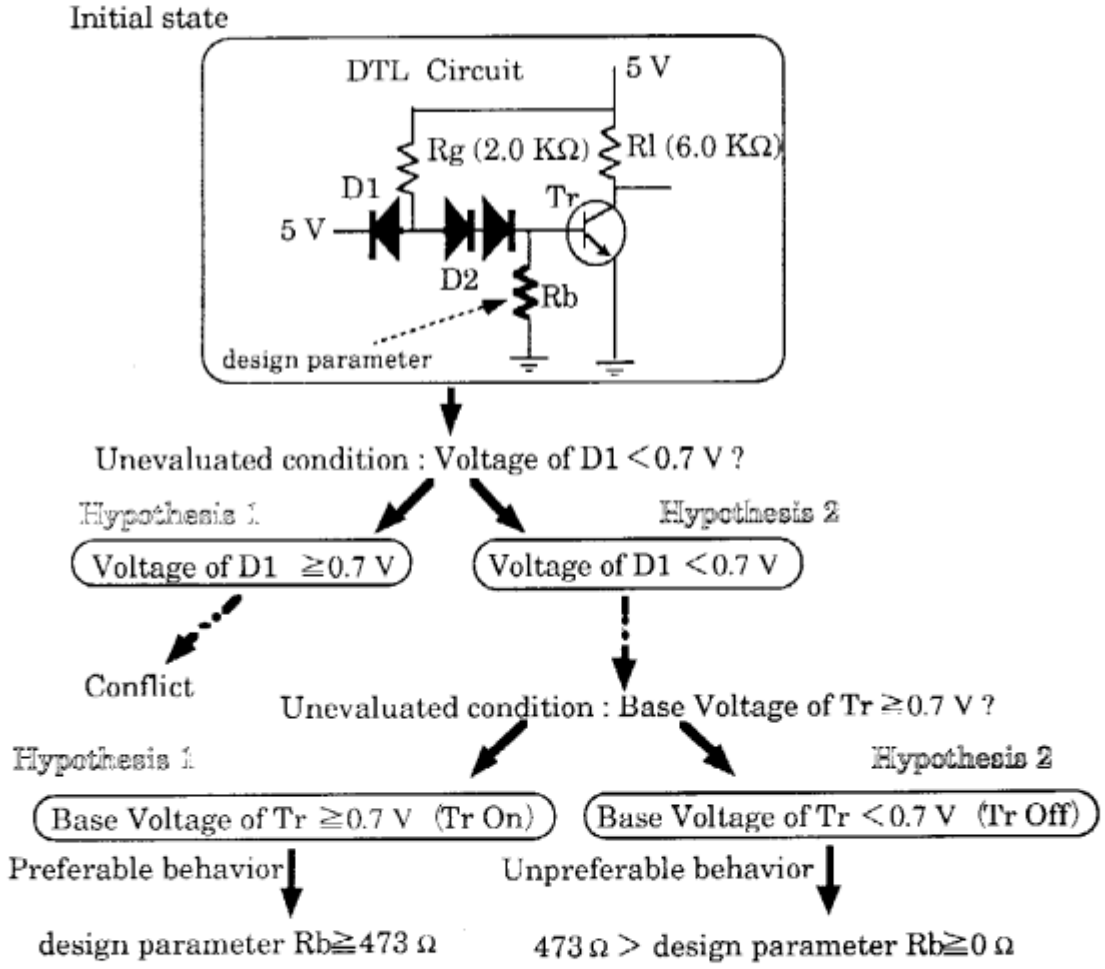


Fig. 1. An example of deciding a of design parameter.

Figure 1 shows an example of suggesting ranges for a design parameter. This example illustrates the determination of a resistance value in a DTL circuit. The designer inputs the DTL structure and the value for the parameters of all components except the resistance Rb, but he does not give any specification.

Desq tries to build the total model for the DTL, which model is described as a set of simultaneous inequalities. Desq checks the conditions in the definitions of physical rules and objects. The definitions for objects specified in the initial state for the DTL circuit are checked. The definitions for physical rules in knowledge base of Desq are checked. If their conditions are satisfied, the equations of their consequences are added to the model of the DTL and they are sent to the parallel constraint solvers. But because resistance Rb is indeterminate, it is not known what state the diode D1 is in. The first condition is whether the voltage of D1 is less than 0.7 volts. Desq hypothesizes two cases: in the first the condition is not satisfied, and in the second it is. The first hypothesis is abandoned because the parallel constraint solver detects a conflict with the other equations. In the second hypothesis, no conflict is detected. After some more hypotheses are made, a state is detected in which it is not known whether or not the condition giving the state of the transistor Tr is satisfied. Desq similarly hypothesizes these two conditions. Finally, Desq finds two possible behaviors for the initial data. Then Desq calculates the resistance Rb. The resistance must be greater than 473 ohms to give the desired behavior, that the circuit acts as a NOT circuit because the transistor is "on". If the resistance is less than 473 ohms, the circuit shows an undesired behavior. We can know that the resistance Rb must therefore be greater than 473 ohms.

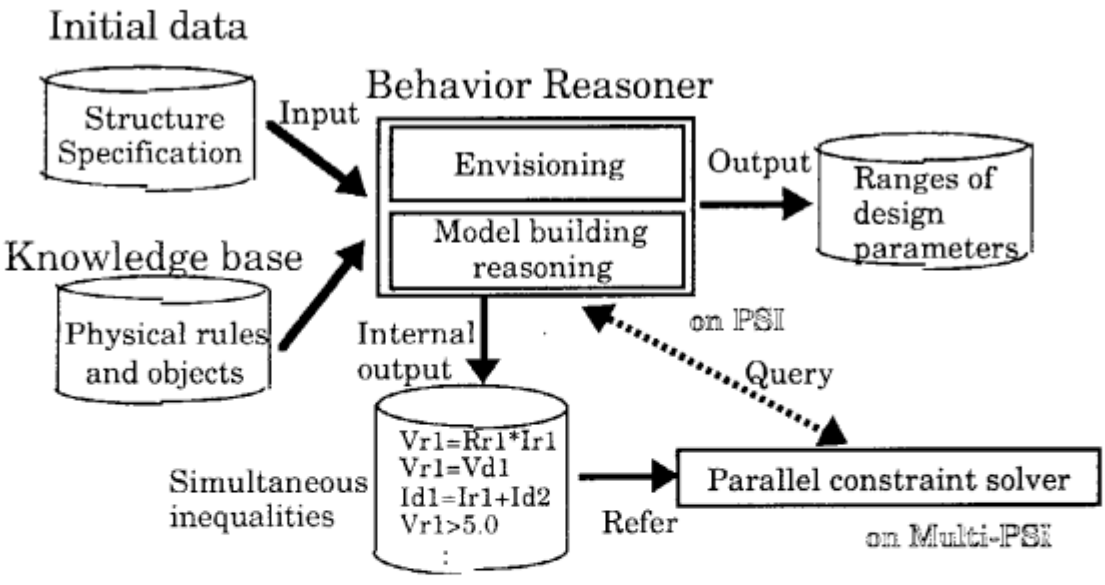


Fig. 2. System organization.

3 System organization

This section describes the system organization of Desq. Figure 2 shows that Desq mainly consists of two subsystems:

(1) Behavior reasoner

This subsystem is based on Qupras. It determines all possible behaviors.

(2) Parallel constraint solver

This subsystem is written in KL1 and can be executed on the parallel inference machines supported by ICOT, which are PIM, Multi-PSI, or Pseudo Multi-PSI.

When the designer specifies initial data, 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. Model building reasoning builds the simultaneous inequalities in the same way that they are built in Qupras. Simultaneous inequalities are passed to the parallel 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 parallel constraint solver. If the conditions are satisfied, the inequalities in the consequences of the physical rules and objects are added to simultaneous inequalities in the parallel constraint solver. Conditions that cannot be evaluated by the parallel constraint solver are hypothesized. After determining the model of the DTL on one state, the behavior reasoner calculates all parameters and predicts the next state. When predicting the next state, the restrictions for constant parameters are passed. The information on variable parameters used to predict the next state are basically passed to the next state.

3.1 Behavior reasoner

3.1.1 Qupras outline

Qupras, a qualitative reasoning system that uses knowledge from physics and engineering textbooks, has the following characteristics:

(1) Qupras has three primitive representations: physical rules (laws of physics), objects, and events.

(2) Qupras 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 need not enter all the equations of the system.

(3) Qupras deals with equations that describe basic laws of physics qualitatively and quantitatively.

(4) Qupras does not require quantity spaces to be given in advance. It finds the quantity spaces itself during reasoning.

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

Qupras is similar to QPT [4] but does not use influence. The representations describing relations of values in Qupras are only equations. Qupras aims to represent laws of physics given in physics textbooks and engineering textbooks.

Laws of physics are generally described in the textbooks not by using influences, but by using equations. Qupras therefore uses only equations.

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

The representation of physical rules 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 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.

3.1.2 Behavior reasoner

The behavior reasoner is little different from that of Qupras. It differs in the following three features:

(1) Envisioning

In Qupras, if the conditions of physical rules and objects cannot be evaluated, Qupras asks the user to specify the conditions. It is possible for Desq to continue to reason in such situations by assuming unevaluated conditions.

(2) Calculation of all parameters

After propagation reasoning, the behavior reasoner asks the parallel constraint solver to calculate all parameters including in the circuit.

(3) Propagation of new constraints on constants

There are two types of parameters (quantities): constant and variable. In envisioning, the constraints related to some constant parameters become stronger by hypothesizing some conditions in the definitions of physical rules and objects. The constraints propagate to the subsequent states.

Before the reasoning, all initial relations defined in the initial state are set as known relations, which are used to evaluate the conditions of objects and physical rules. Initial relations are used mainly to set the initial values of the physical variables. If there is no explicit change in an initial relation, the initial relation is held. An example of an explicit change is the prediction of the next value in the prediction reasoning.

Propagation reasoning finds active objects and physical rules whose conditions are satisfied by the known relations. If a contradiction is detected after passing relations of the active objects and physical rules to the parallel constraint solver, the propagation reasoning is stopped. If no condition of physical rules and objects can be evaluated, the reasoning process is split by the envisioning mechanism into two process: one hypothesizing that the condition is satisfied, and the other hypothesizing that it is not.

Prediction reasoning first finds the physical variables changing with time from the known relations that result from the propagation reasoning. Then it searches for the new values or the new intervals of the changing variables at the next specified time or during the next time interval. Desq updates the variables according to the sought values or intervals in the same way they are updated in Qupras. The updated values are used as the initial relations at the beginning of the next propagation reasoning.

3.2 Parallel constraint solver

The parallel constraint solver [19] tests whether the conditions written in the definitions of physical rules and objects are proved by the known relations obtained from active objects, active physical rules, and initial relations. And if conditions of active objects and active physical rules are satisfied, the constraint solver receives those relations and checks their consistency.

To test the conditions in the definitions of objects, physical rules, and events, we want to solve nonlinear simultaneous inequalities. More than one algorithm is used to build the combined constraint solver because we do not know of any single efficient algorithm for nonlinear simultaneous inequalities. We connected the three solvers as shown in Figure 3. The combined constraint solver consists of the

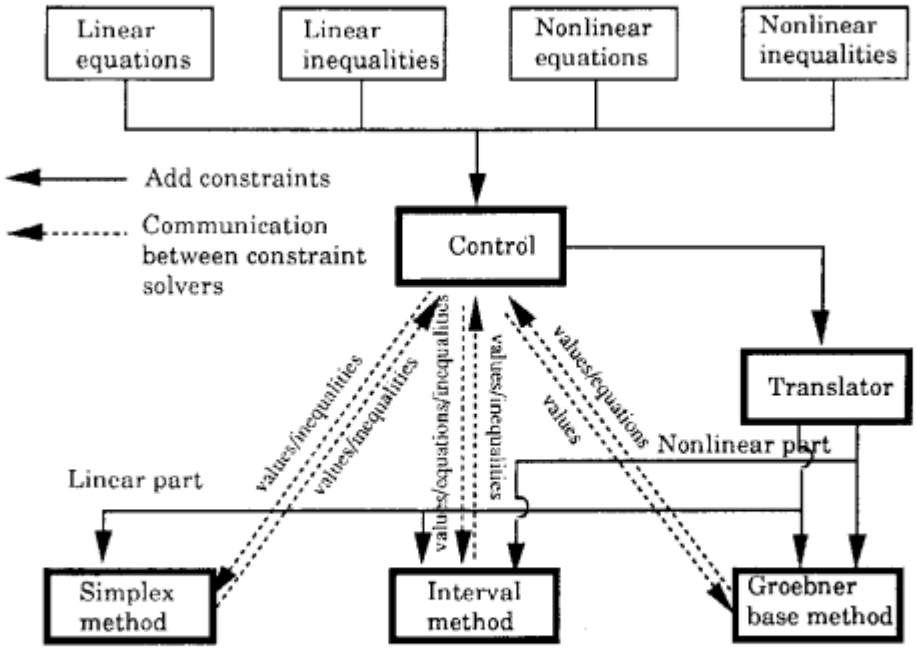


Fig. 3. Parallel Constraint Solver (Consort).

following three parts:

(1) Nonlinear inequality solver based on the interval method [23],

(2) Linear inequality solver based on the Simplex method [7],

(3) Nonlinear simultaneous equation solver based on the Groebner base method [1].

If any one of the three basic constraint solvers finds new results, the results are passed on by the control parts to the other basic constraint solvers. This combined constraint solver can solve broader equations than each individual solver can, but the results of the combined constraint solver are not always valid because each basic constraint solver cannot solve all nonlinear simultaneous inequalities.

The reason we can get quantitative ranges is that the combined constraint solver can process quantities quantitatively as well as qualitatively.

4 Example

We show two examples. One is another DTL circuit example, and the other is a Schmitt trigger circuit.

4.1 DTL circuit

4.1.1 Description of model

We use a DTL circuit identical to that shown in Figure 1. In this example, however, the input voltage and the resistance Rb are indeterminate.

The initial data is shown in Figure 4. The "objects" field specifies components and their classes in the DTL circuit. The "initial_relations" field specifies the relations holding in the initial state. For example, "connect(t2!Rg, t1!D1, t1!D2)" specifies that the terminal t2 of the resistor Rg, the

```
initial_state dtl
 objects
  Rl-resistor ;
  Rg-resistor ;
  Rb-resistor ;
  Tr-transistor ;
  D1-diode ;
  D2-diode2 ;
 initial_relations
  connect(t1!Rl,t1!Rg) ;
  connect(t2!Rg,t1!D1,t1!D2) ;
  connect(t2!D3,t1!Rb,tb!Tr) ;
  connect(t2!Rl,tc!Tr) ;
  resistance@Rl=6000.0 ;
  resistance@Rg=2000.0 ;
  resistance@Rb >= 0.0;
  v@t1!Rl = 5.0 ;
  v@t2!D1 >= 0.0 ;
  v@t2!D1 =< 10.0 ;
  v@te!Tr = 0.0 ;
  v@t2!Rb = 0.0 ;
end.
```

Fig. 4. Initial state for DTL.

terminal t1 of the diode D1, and the terminal t1 of the diode D2 are all connected. The "!" is a symbol specifying a part. The "t2!Rg" expresses the terminal t2, which is one part of Rg. Rg is specified as a resistor in the "objects" definition. The "@" indicates a parameter. The "resistance@Rl" represents the resistance value of Rl. The "resistance@Rl = 6000.0" specifies that Rl is 6000.0 ohms. The resistance Rb is constrained to be positive, and the input voltage, which is the voltage of the terminal t2 in the diode D1, is constrained to be between 0.0 and 10.0 volts. Both values are indeterminate, and Rb is a design parameter.

```
object  terminal:Terminal
 attributes
   v ;
   i ;
end.

object  two_terminal_device:TTD
 parts_of
   t1-terminal ;
   t2-terminal ;
end.

object  diode:Di
 supers
   two_terminal_device;
 attributes
    v ;
    i ;
    resistance-constant ;
 initial_relations
     v@Di=v@t1!Di-v@t2!Di ;
 state  on
  conditions
     v@Di >= 0.7 ;
  relations
     v@Di= 0.7 ;
     i@Di >= 0.0 ;
 state  off
  condition
     v@Di < 0.7 ;
  relations
     resistance@Di=100000.0 ;
     v@Di=resistance@Di*i@Di ;
end.
```

Fig. 5. Definition of a diode.

Figure 5 shows the definition of a diode. Its superobject is a two_terminal_device, so the diode inherits the properties of two_terminal_devices, i.e., it has two parts, both of which are terminals. Each terminal has two attributes: "v" for voltage and "i" for current. The diode has an initial relation, which specifies the voltage difference between its terminals. The diode also has two states: an "on" state where the voltage difference is greater

than 0.7, and an "off" state where the voltage difference is less than 0.7. If the diode is in the "on" state, it behaves like a conductor. In the "off" state, it behaves like a resistor. A transistor is defined like a diode but has three states: "off", "on", and "saturated" (In the example of Figure 1, we used a transistor model with only two states, "off" and "on").

Figure 6 shows the definition of a physical rule, Kirchhoff's law when the t1 terminals of three two_terminal_devices are connected. It is assumed that the current into t1 of a two_terminal_device flows to the terminal t2. In fact, three two_terminal_devices can be connected in eight ways.

```
physics  three_connect_1
 objects
  TTD1 - two_terminal_device ;
  TTD2 - two_terminal_device ;
  TTD3 - two_terminal_device ;
  T1-terminal partname t1 part_of TTD1 ;
  T2-terminal partname t1 part_of TTD2 ;
  T3-terminal partname t1 part_of TTD3 ;
 conditions
  connect(T1,T2,T3);
 relations
  v@T1 = v@T2 ;
  v@T2 = v@T3 ;
  i@T1 + i@T2 + i@T3 = 0 ;
end.
```

Fig. 6. Definition of physical rule.

Table 1. All behaviors of a DTL circuit.

State	Range of input (volts)	Range of resistance value (Ohms)	Output (volts)
1 ON-ON-SAT	1.40081 - 1.5381	486.16 - infinity	0.2
2 ON-ON-ON	1.4 - 1.40081	482.75 - infinity	0.2 - 5.0
3 ON-ON-OFF	0.7 - 1.4	0 - 233,567	4.94
4 ON-OFF-ON	0 - 1.4007	100,000 - infinity	0.842 - 5.0
5 ON-OFF-OFF	0 - 1.4	0 - 233,567	4.94
6 OFF-ON-SAT	1.40081 - 10.0	460.9 - infinity	0.2
7 OFF-ON-ON	1.4 - 10.0	457.8 - 488.53	0.2 - 5.0
8 OFF-ON-OFF	0.7 - 10.0	0 - 484.1	4.94
9 OFF-OFF- *	Conflict		

4.1.2 Results

Table 1 lists all behaviors of the DTL circuit obtained by envisioning. The state column indicates the states of the diode, the diode2, and the transistor. The following columns show the range of the input voltage (volts), the range of the resistance Rb (ohms), and the output voltage (volts). As shown, the envisioning found nine states. Because the input voltage and the resistance Rb were indeterminate, the conditions of the two diodes and the transistor could not be evaluated. So Desq was used to hypothesize both cases and to search all paths. Figure 7 shows the relationship between the resistance and the input voltage. The ranges in Table 1 overlap because the models of the diodes and the transistor are approximate models.

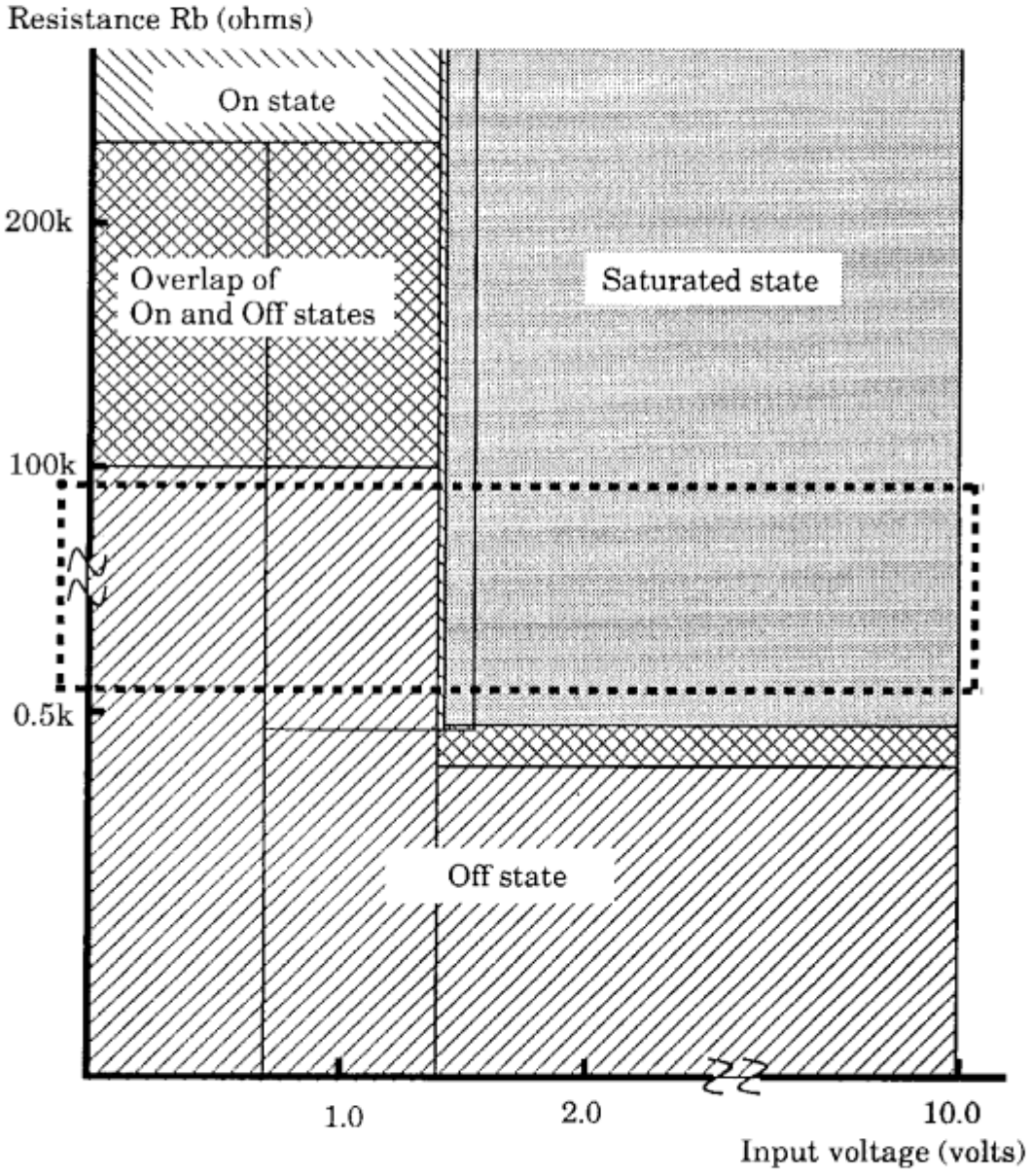
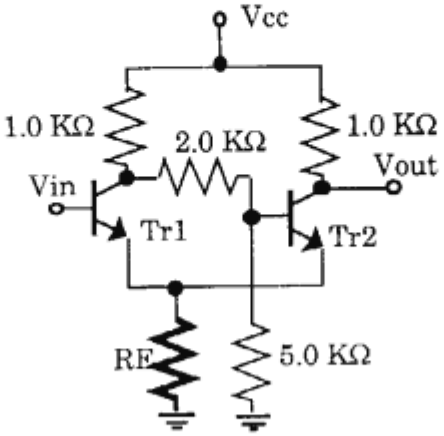
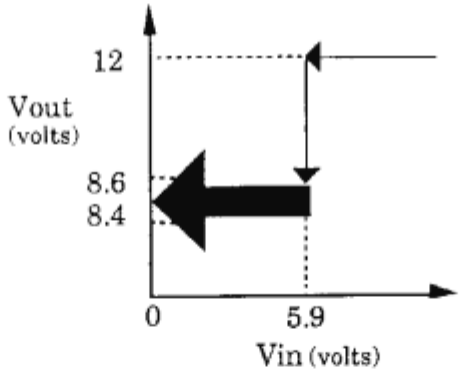


Fig. 7. Relationship between Resistance and Input voltage.

By looking at Figure 7, a designer can decide the resistance that Rb must have for the DTL circuit to behave as a NOT circuit. The value of Rb should be greater than about 0.5 k ohms and less than about 100 k ohms so that the DTL circuit can output a low voltage (nearly 0 volts) when the input is greater than about 1.5 volts or can output a high voltage (nearly 5 volts) when the input is less than about 1.5 volts. The range is shown by the area enclosed by the dotted lines in Figure 7.



(a) Circuit structure



(b) Specification of behavior

Fig. 8. Inputs for design support of a Schmitt trigger circuit.

4.2 Schmitt trigger

The second example is to determine the value of the resistance RE of a Schmitt trigger circuit shown in Figure 8. This example differs from the previous example in that here the input voltage changes with time.

Figure 8(a) shows the structure. A Schmitt trigger circuit includes positive feedback, and its behavior shows hysteresis. When the input voltage increases, the input voltage change from low-level to high-level output differs from the input voltage change from high-level output to low-level output.

The inputs to determine the value of the resistance RE of a Schmitt trigger circuit are the circuit structure of the Schmitt trigger and the specification of its behavior. The parameter RE in the circuit is indeterminate, and the specification is as shown in Figure 8(b): if the input voltage is greater than 5.9 volts, the output voltage should be 12 volts; if the input voltage is 5.9 volts, the output should be between 8.4 and 12 volts; and so on. The specification shows how the circuit should behave when the input voltage deceases from 12.0 volts to 0.0 volts.

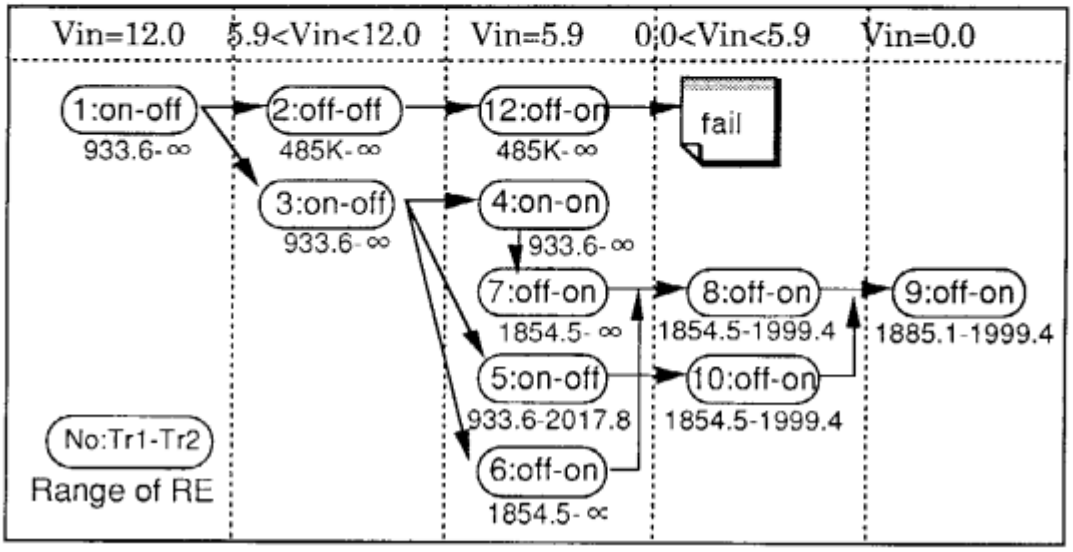


Fig. 9. Determination of a design parameter in a Schmitt trigger.

Desq determines the ranges of the resistance RE satisfying this specification. Figure 9 shows the process for determining the range. Desq finds the behavior of the Schmitt trigger satisfying the specification, and at the same time it determines the ranges of the resistance satisfying the specification.

The constraint solver determines that when the input voltage is 12 volts, there is no conflict only when the two transistors in the Schmitt trigger are in the "on and off" states: with other combinations, there is conflict. The numbers below the circled state combinations in Figure 9 show the range of RE. For example, in this "on and off" state, the resistance must be greater than 933.6 ohms. There are two possibilities from this state. One is "off and off" when the input voltage is between 5.9 volts and 12.0 volts. But this state finally cannot satisfy the given specification. The other state, which is "on and off", becomes three states when the input voltage is 5.9 volts. Positive feedback instantly changes the "on and on" state to the "off and on" state. And the "off and on" state when the input voltage is 5.9 volts changes to the "off and on" state when the input voltage is between 0.0 and 5.9 volts. And when the input voltage is 0.0 volts, it remains in the same "off and on" state. The range of RE is reduced in finding the behavior satisfying the specification. In the final state, the range of RE is from 1885.1 to 1999.4 ohms, so we can find that the Schmitt trigger can behave satisfying the given specification if RE is between 1885.1 and 1999.4 ohms.

5. Related Work

Desq does not suggest structures of devices like the methods described in Refs. 11 and 26 do. Rather, it suggests the ranges of design parameters for preferable behaviors. This suggestion is also useful, though because determining values of design parameters is one of the important steps of design [3].

This approach may be regarded as an application of constraint satisfaction problem solving. There are several papers dealing with constraint satisfaction problem solving that use electronic circuits as examples [24], [6], [9]. Sussman and Steele's system cannot suggest ranges for design parameters because their system uses only equations. Heintze, Michaylov, and Stuckey's work using CLP(R) to design electronic circuits is the most similar to Desq, but Desq differs from their work in the following points:

(1) For Desq, knowledge on objects and laws of physics is more declarative.

(2) Desq can design ranges of design parameters (of devices) that change with time.

(3) Desq can deal with nonlinear inequalities and can in some cases solve nonlinear inequalities.

Mozetic and Holzbaur use numerical and qualitative models, and in their view, our approach uses numerical models rather than qualitative models. But if a constraint solver is used to solve inequalities, it is possible to use both numerical and qualitative calculations.

6. Conclusion

We have described a method of using qualitative reasoning to suggest ranges for design parameters and have implemented the method in Desq. The ranges obtained are quantitative, because our system deals with quantities quantitatively as well as qualitatively. In an example using a DTL circuit, Desq suggested that the range of a resistance (Rb in Figure 1) should be greater than about 0.5 kilo ohms and less than about 100 kilo ohms to work the DTL circuit as a NOT circuit. If the designer needs a more detailed design - for example, to minimize the response time by performing numerical calculation - he need not calculate outside the range. This can save on the calculation cost, which is greater for direct numerical calculation (outside range). In the other example of the Schmitt trigger, when the Desq input was the structure of the Schmitt trigger and a specification of how the circuit should behave, Desq found that the design parameter, resistance RE, should be between 1885.1 and 1999.4 ohms to satisfy the specification.

There are some possibilities, however, for which Desq cannot suggest valid ranges or the best ranges for design parameters. This is because

(1) Consort has a limited ability to solve nonlinear inequalities

Desq may suggest invalid or weak ranges because Consort cannot perfectly solve nonlinear inequalities. But almost all results can be obtained by performing more detailed analysis using numerical analysis systems - SPICE, for example.

(2) Inexact definitions are used

It may be difficult to describe the definitions of physical rules and objects. This is because inexact results may be obtained from inexact definitions.

We are investigating a feature to modularize the definition of components by building the definition of an object from the definitions of components which are parts of the object. In an experiment in which we made the definition of a Schmitt trigger from its components [22], the CPU time for simulation was only about one a tenth of that required for simulation using the definitions of the components of the Schmitt trigger circuit. The modularizing feature is expected to improve the performance of Desq.

7. Acknowledgements

This research was supported by the ICOT (Institute of New Generation Computer Technology). We wish to express our thanks to Dr. Fuchi, Director of the ICOT Research Center, who provided us with the opportunity of performing this research in the Fifth Generation Computer Systems Project. We also wish to thank Dr. Nitta, Mr. Ichiyoshi, and Mr. Sakane (Current address: Nippon Steel Corporation) in the seventh research laboratory for their many comments and discussions regarding this study, and Dr. Aiba, Mr. Kawagishi, Mr. Menjyu, and Mr. Terasaki in the fourth research laboratory of the ICOT for allowing us to use their GDCC and Simplex programs and for helping us to implement our parallel constraint solver. And we wish to thank Prof. Nishida of Kyoto University for his discussions, Mr. Kawaguti, Miss Toki, and Miss Isokawa of Hitachi Information Systems for their help in the implementation of our system, and Mr. Masuda and Mr. Yokomizo of Hitachi Central Research Laboratory for their suggestions on using qualitative reasoning to design electric circuits.

References

[1] Aiba, A., Sakai, K., Sato Y., and Hawley, D. J.: Constraint Logic Programming Language CAL, pp. 263-276, Proc. of FGCS, ICOT, Tokyo, 1988.

[2] Bobrow, D. G.: Special Volume on Qualitative Reasoning about Physical Systems, Artificial Intelligence, 24, 1984.

[3] Chandrasekaran, B.: Design Problem Solving: A Task Analysis, AI Magazine, pp. 59-71, 1990.

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

[5] Hawley, D. J.: The Concurrent Constraint Language GDCC and Its Parallel Constraint Solver, Proc. of KL1 Programming Workshop '91, pp. 155-165, ICOT, Tokyo, 1991.

[6] Heintze, N., Michaylov, S., and Stuckey P.: CLP(R) and some Electrical Engineering Problems, Proc. of the Fourth International Conference of Logical Programming, pp. 675-703, 1986.

[7] Konno, H.: Linear Programming, Nikka-Gikken, 1987 (in Japanese).

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

[9] Mozetic, I. and Holzbaur, C. : Integrating Numerical and Qualitative Models within Constraint Logic Programming, Proc. of the 1991 International Symposium on Logic Programming, pp. 678-693, 1991.

[10] Mizoguchi, R.: Foundation of expert systems, Expert system - theory and application, Nikkei-McGraw-Hill, pp. 15, 1987 (in Japanese).

[11] Murthy, S. and Addanki, S.: PROMPT : An Innovative Design Tool, Proc. of AAAI-87, pp. 637-642, 1987.

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

[13] Nishida, T.: 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).

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

[15] Ohki, M. and Furukawa, K.: Toward Qualitative Reasoning, Proc. of Symposium of Japan Recognition Soc., 1986, or ICOT-TR 221, 1986.

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

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

[18] Ohki, M., Oohira, E., Shinjo, H., and Abe, M.: Range Determination of Design Parameters by Qualitative Reasoning and its Application to Electric Circuits, FGCS'92, pp. 1022-1029, 1992.

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

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

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

[22] Shinjo, H., Ohki, M., Oohira, E., and Abe, M.: Acquisition of hierarchical knowledge from deep knowledge in Qualitative Reasoning, ICOT-TR, (in Japanese), 1992 (to appear).

[23] Simmons, S.: Commonsense Arithmetic Reasoning, Proc. of AAAI-86, pp. 118-128, 1986.

[24] Sussman, G. and Steele, G. : CONSTRAINTS - A Language for Expressing Almost-Hierachical Descriptions, Artificial Intelligence, 14, pp. 1-39, 1980.

[25] 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).

[26] Williams B. C.: Interaction-based Invention: Designing Novel Devices from First Principles, Proc. of AAAI-90, pp. 349-356, 1990.