Design Verification of Sequential Control Circuits Based on Theorem-Proving Method

NAOYUKI YAMADA*, YASUHIRO KOBAYASHI*, YOSHIKATSU UEDA**, SATOSHI MATSUDA***, SHOUICHI MUTO*** and JUNICHI YOSHIZAWA***

VSEQ (Verifier for SEQential controller) is a design verification system for sequential control circuits in an electric power substation. Instead of using the prevailing simulation techniques, it verifies the design by using a theorem-proving method, which does not need any test data generation and guarantees reliable verification. Thus VSEQ realizes a theorem-proving method dedicated to the design verification of sequential control circuits.

This paper shows that use of domain characteristics in the formulation of the verification problem and also in control of the proof process leads to a theorem-proving method that is efficient enough for practical design verification.

1. Introduction

Design verification is indispensable to almost all engineering design work and is considered to be a key technique for improving reliability, especially in the fields of power plants and electrical power circuits.

This paper describes a system for design verification of sequential control circuits in electric power substations based on a theorem-proving method. Compared with widely used conventional techniques involving numeric simulation, the application of a theorem-proving method to design verification has two potential advantages: it does not require any test data generation and it gurantees reliable verification. In addition, a theorem-proving method is able to handle the verification of design specifications that are described rather abstractly, with little relationship to their functionality, and to which symbolic simulation [1] is not easily applied. Therefore, this method is considered to be one of the most promising techniques for verification of large and complicated designs.

A theorem-proving method was first applied to design verification by Wagner [2] in the field of logic circuit design. The verification of an 8-bit multiplier using the theorem prover called FOL (First-Order Logic) [3] has revealed the usefulness of this approach. Since then, intensive research has been undertaken in two directions: one looking at the development of interactive systems [4], and the other at automated systems [5], [6]. Although both approaches have contributed to the establishment of a theorem-prover-based verification technique, comparable to a simulation-based one in some cases, there is still an inefficiency problem in their verification process. This stems from the fact that previous approaches rely on general-purpose theorem provers that do not accept domain dependent refinements.

Our approach to design verification through the application of the theorem-proving method is, then, to develop a domain-specific theorem prover. The use of domain knowledge in the formulation of the verification problem and also in the control of the proof process leads to a theorem-proving method that is efficient enough for practical verification. VSEQ (Verifier for SEQuential controller) is a design verification system that uses a theorem prover dedicated to the design verification of sequential control circuits.

Section 2 briefly describes a problem domain and defines a problem to be solved. Section 3 describes the design verification method with emphasis on techniques for improving the efficiency of the theorem-proving method.

2. Domain Characteristics

Figure 1 shows a simplified one-line diagram of an electric power substation that converts high-voltage electric power (275 kV) into low-voltage power (66 kV). The substation is composed of one transformer, circuit breakers (CBs), disconnecting switches (DSs), buses, and cables. In this substation, open/close operations of

*Energy Research Laboratory, Hitachi, Ltd. 1168 Moriyama-cho, Hitachi, Ibaraki 316, Japan.
**Systems Engineering Division, Hitachi, Ltd. 6, Kanda-surugadai 4 chome, Chiyoda-ku, Tokyo 101, Japan.
***Computer & Communication Research Center, The Tokyo Electric Power Co. 1-3 Uchisaiwai-cho, 1 chome, Chiyoda-ku, Tokyo 100, Japan.

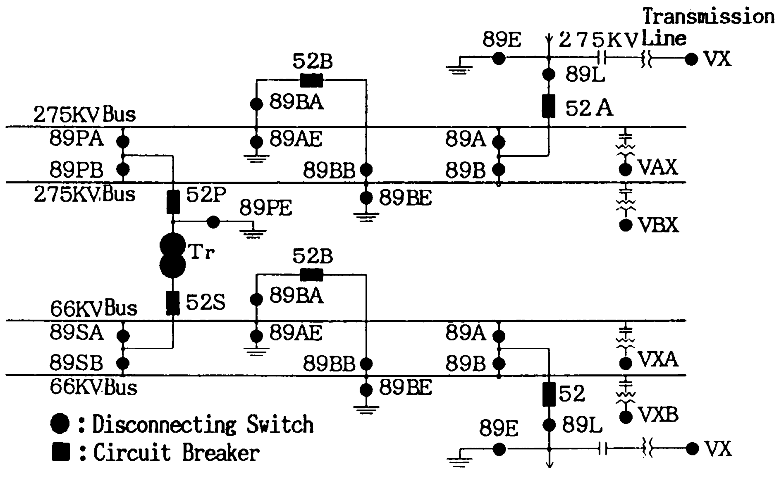

Fig. 1 One-line diagram of electrical power substation.

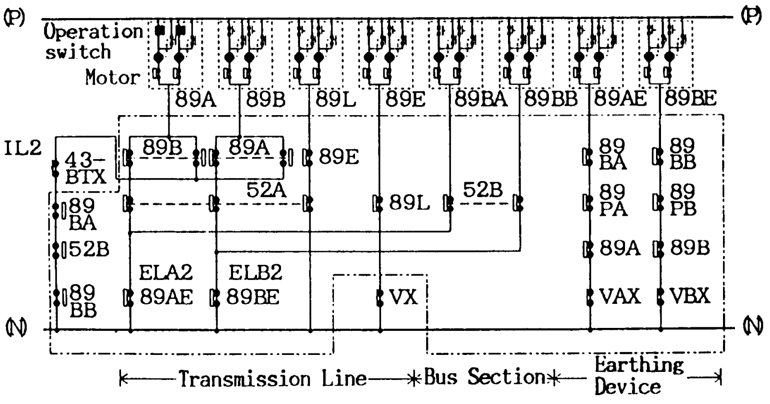

Fig. 2 Example of interlock circuit.

the CBs and DSs are of great importance and no misoperation is permissible, since it would directly produce adverse effects on the transmission system as well as the other substation components. Usually, the operation conditions for these components are well defined. The condition for the DS 89A, for example, is stated as

if the voltages fo both terminals of 89A are equal,
or either terminal of 89A is disconnected,
then 89A is operable.

In order to realize these operation conditions, namely, operation interlocks, sequential control circuits (or interlock circuits) are designed. A part of the interlock circuit for the components in the high-voltage side of Fig. 1 is given in Fig. 2. The Interlocking of each component is implemented by linking its open and close swiches with arbeit contacts and/or break contacts of control relays attached to the other components. Usually, expert designers tend to design compact circuits by making use of commonly usable subcircuits and auxiliary relays. It is therefore not an easy job to figure out the control logic the circuits realize, let alone verify them.

The design verification problem here is to verify the designed sequential control circuits against the design specifications expressed as interlock conditions. From the viewpoint of verification, this problem has the following three characteristics:

(1) Design specifications are the results of interpretation of the circuit description.

In most design verification problems, the design specifications are given as functional descriptions that are defined as behaviours describing the relations between the primary inputs and outputs. However, the design specifications of this problem are not described in such a way. Instead, they are given as the results of interpretation through functional definitions of circuit descriptions that consist of component types, compo-

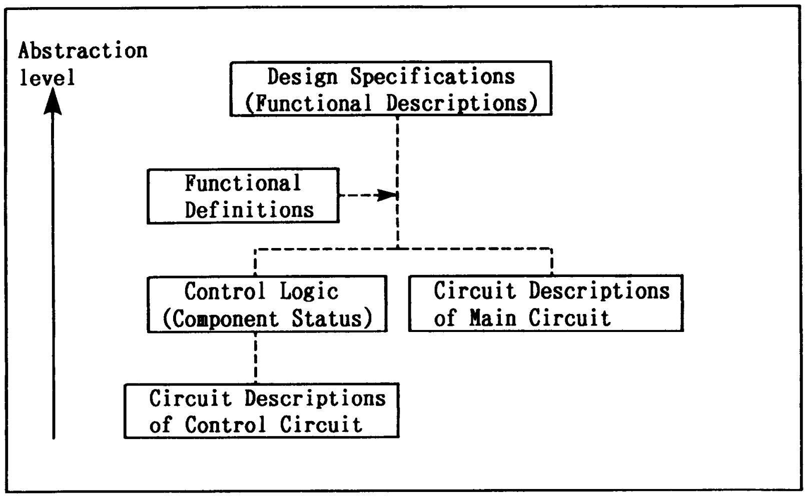
Abstraction level
Design Specifications (Functional Descriptions)
Functional Definitions
Control Logic (Component Status)
Circuit Descriptions of Main Circuit
Circuit Descriptions of Control Circuit

Fig. 3 Relation ship of the information needed for verification.

nent status, and their connectivities. For example, the specification "The voltages of the two terminals A and B are equal" is mapped to the circuit description in which the two terminals A and B are connected through one or more components and the status of the corresponding components are all closed (or working). This characteristics makes it difficult to directly apply the simulation-based verification method, which propagates the signal according to the structure.

(2) Design specifications are not directly related to the designed control circuit.

Information concerning this verification is illustrated schematically in Fig. 3. The vertical direction in this figure corresponds to the abstraction level of the information. As mentioned above, the design specifications for control circuits are a kind of functional descirption of the objective system (the main circuit in this case), which can be obtained through the application of a functional definition to circuit descriptions. On the other hand, information about the designed control circuit to be verified specifies the control logic, which prescribes the component status of the objective system. This means that another level of the description exists between the information presented for verification. The representational gap requires a mechanism that reduces the initial verification to one that can be performed between the adjacent levels of descriptions.

(3) The behaviour of both the objective system and control circuit can be treated statically.

Another characteristic of this problem is that the design specifications are concerned only with the static, discrete ON/OFF status of the components of the objective system, and that the control logic of the designed interlock circuit can in turn be determined by the discrete ON/OFF status of its components. This suggests that the logic used for theorem-proving does not require any contrivance to handle dynamic situations.

3. Design Verification system VSEQ

3.1 System Organization

The basic idea of design verification based on the theorem-proving method is that if the proposed design is correct, the design specifications can be logically derived from that design. Usually, this method is applied to verification between adjacent levels of descriptions that are related by some axioms conerned with the abstraction. It is possible to apply a theorem-proving method to descriptions that are not adjacent; however, it causes an unnecessary expansion of the search space and cannot be expected to provide efficient verification.

As shown is Fig. 3, the verification of sequential control circuits in an electric power substation involves three distinct description levels: design specifications, components statuses of the objective system, and circuit descriptions of the designed interlock circuit. However, as shown in Fig. 2, most components of the interlock circuit are derectly related to the statuses of components in the main circuit via their contacts. Therefore, it is straightforward to extract the control logic of the designed interlock circuit from its circuit description.

The overall information flow in VSEQ is shown in Fig. 4. The circuit descriptions of a control circuit consist of component types of the desined sequential control circuit and their connectivities. These data are transferred from the data base of the CAD system. A logic extraction program extracts the control logic from the circuit descriptions of the control circuit, where control logics are groups of components statuses of the main circuit that are realized by the control circuit.

The circuit descriptions of the objective system include the component types and their connectivities and

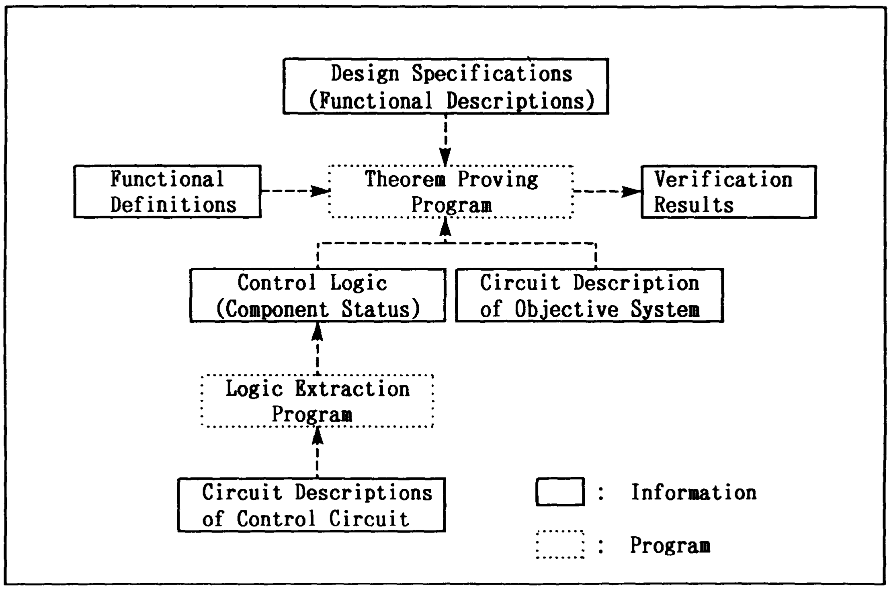
Design Specifications (Functional Descriptions)
Functional Definitions
Theorem Proving Program
Verification Results
Control Logic (Component Status)
Circuit Description of Objective System
Logic Extraction Program
Circuit Descriptions of Control Circuit
: Information
: Program

Fig. 4 Information flow in VSEQ.

are considered to be correct information in this verification. Functional definitions are group of interpretation rules that relate functions to circuit descriptions and components statuses, and are also treated as correct information in the verification. The design specifications are given by the user at each verification. Eventually, the design verification problem comes down to proving the design specifications against the circuit descriptions, components statuses (i.e. control logic), and functional definitions. These descriptions correspond to the adjacent two levels of design descriptions.

3.2 Control Logic Extraction

As shown in Fig. 2, the operation interlock of each DS (disconnecting switch) is implemented in such a way that the operation switches (ON/OFF) and its driving motors are connected to arbeit and/or break contacts of control relays attached to the other components. Therefore, for a operation switch to be active, all the other contacts should be closed to give a connecting line from the positive pole to the negative pole (for example, from the R-phase to the T-phase in the case of an alternating current). As a result, the control logic can be extracted in the following steps:

(1) Enumerate all possible lines related to the specified operation switch from the positive pole to the negative pole.

(2) Extract the component status that closes the corresponding contacts.

(3) Delete the cases that cannot be realized electrically.

(4) Select the logic concerned with the component status of the objective system (main circuit).

In step (2), the following operational relations between the status of control relays and their contacts are used:

(a) An arbeit cantact is closed when the corresponding relay is closed (working).

(b) A break contact is closed when the corresponding relay is open (not working).

Note that a parallel connection of contacts corresponds to the logical 'OR' and a serial connection, to the logical 'AND'. The arbeit and break contacs of the same control relay cannot be closed simultaneously. These cases are deleted in step (3).

The extracted control logic for DS 89A in Fig. 2 is expressed in predicate calculus form as follows:

```
(OR (AND (STATUS 89B OFF) (STATSU 52A OFF)
            (STATUS 89AE OFF))
     (AND (STATUS 89B ON) (STATUS 89BA ON)
            (STATUS 52B ON) (STATUS 89BB ON)))
```

The first element of the OR expression corresponds to the logic of the second vertical line from the left in Fig. 2, and the second element to the first vertical line.

3.3 Theorems and Axioms

VSEQ uses the theorem-proving method in first-order predicate calculus. Therefore, all the information needed for verification is represented in predicate calculus form. In line with the discussion above, the information used for verification includes design specifications, circuit descriptions, functional definitions, and control logic.

Design specifications are represented by introducing appropriate predicates. The specification (an operable

```
(COMP-TYPE  89A  DS)   ; DS : Disconnecting Switch
(COMP-TYPE  52B  CB)   ; CB : Circuit Breaker
(CONNECTED (PORT 89A 1) (BUS 275-1))
(MULTI-CONNECTED (PORT 52A 2)(PORT 89A 2)(PORT 89B 1))
```

Fig. 5 Examples of circuit descriptions.

```
(IF (AND (COMP-TYPE $X $P)                     ; If the component is closed,
         (NORESISTANCEP $P)                    ; then the voltages of both
         (STATUS $X close))                    ; terminals are equal.
    (V-EQUIVALENT (PORT $X 1)(PORT $X 2)))

(IF (AND (COMP-TYPE $X $P)                     ; If both terminals of a component
         (NORESISTANCEP $P)                    ; constitute a closed loop,
         (HAS-NR-LOOP (PORT $X 1)(PORT $X 2))) ; then their voltages are equal.
    (V-EQUIVALENT (PORT $X 1)(PORT $X 2)))

(IF (CONNECTED (PORT $X $P)(PORT $Y $Q))       ; If two terminals are connected
    (HAS-NR-LOOP (PORT $X $P)(PORT $Y $Q)))    ; then they constitute a closed loop.

(IF (AND (CONNECTED (PORT $X $P)(PORT $Y $Q))  ; If terminal p of x is connected
         (COMP-TYPE $Z $R)                     ; to terminal 2 of z and z is closed
         (STATUS $Z close)                     ; and terminal 1 of z has a closed
         (NORESISTANCEP $R)                    ; loop to terminal q of y then both
         (HAS-NR-LOOP (PORT $Z 2)(PORT $Y $Q))); terminals p of x and q of y
    (HAS-NR-LOOP (PORT $X $P)(PORT $Y $Q)))    ; constitute a closed loop.
```

Fig. 6 Examples of function definitions.

condition, in this case) for the DS 89A shown in section 2, for example, is represented as,

(OR (V-EQUIVALENT (PORT 89A 1) (PORT 89A 2))
  (ONE-TERMINAL-OPEN 89A)).

The circuit descriptions are given by the types of each component involved and their connectivities. These are represented by using the predicates 'COMP-TYPE', 'CONNECTED', and 'MULTI-CONNECTED'. Some examples from the circuit in Fig. 1 are shown in Fig. 5.

As mentioned earlier, functional definitions are necessary to relate functions to circuit descriptions and component status. In fact, a group of functional definitions is provided that correspends to each predicate appearing in the design specifications. The representation of these definitions is more flexible than that of cirtuit descriptions. It is desirable to adopt expressibe definitions; however, this requires rather a complex control in the proof procedure. The proper method of representation should be determined in light of the trade-off between expressiveness and efficiency. Some of the definitions for the predicate 'V-EQUIVALENT' in the above design specifications are shown in Fig. 6. These are first converted into a conjunctive normal form and supplied to the theorem prover.

In principle, these definitions should be provided by the user. If the application domain is specified, however, it is possible to prepare the necessary definition beforehand. Currently, VSEQ has nearly 70 definitions inherent in the verification of sequential control circuits of electric power substations.

3.4 Theorem-Proving Program

VSEQ embodies a resolution-based theorem prover with a connection graph method [7, 8, 9] as its main control strategy. A connection graph method is selected for two reasons. First, it offers such an efficient procedure that in each proof step, the applicability of resolution of each resolvent is inherited from its parent clauses, and no further searching for it is needed. Second, its control mechanism is so simple that it is easy to augment with other methods that make the proof process efficient. A proof procedure and its refinements are described in detail by Yamada *et al.* [9].

Although the theorem-proving method with the connection graph method is efficient, there are still inefficiencies in its control when it is applied to practical verification problems. Such a theorem-proving method can be made more efficient by introducing domain knowledge, as well as a formulation of the problem, into its control.

Generally, there are two approaches to realizing an efficient theorem-proving procedure: one is to restrict the search space to a small one by decomposing the original search space, using domain knowledge; the other is to develop control strategies that eliminate unnecessary resolution steps. VSEQ takes both approaches, as we explain below.

3.4.1 Decomposition of Search Space

Design specifications for operation interlock oftern involve more than one condition combined with the logical 'OR'. In these cases, the designed interlock circuit embodies more than one possible configuration, producing the corresponding logical 'OR' in the extracted control logic. As shown in the previous section, the design specification of 89A, for example, is described with two conditions; one requires "V-equivalent" and the other, "One-terminal-open". If the designed sequential control circuit is correct, these conditions are satisfied by the control logic extracted from the circuit. The corresponding control logic is shown at the end of section 3.3. In this case, the condition "V-equivalent" is satisfied by the second OR-element and "One-terminal-open" by the first OR-element. In real verification, however, the correspondence between the specification and the control logic cannot be used beforehand. Therefore, the search space of the theorem-proving problem involves all these specifications and control logic.

Suppose for simplicity that two literals denoted a and b constitute an OR expression in the design specifications, and that the extracted control logic is composed of two clauses C and D, each of which is expressed in the form of the logical 'AND' of more than one literal (C=(AND c1 c2···c*m*), D=(AND d1 d2···d*n*)), where each c*i* and d*j* correspond to a component status of the objective system. Then, the search space except for the circuit description and functional definitions consists of (NOT a), (NOT b), (OR C D), or equivalently,

(NOT a), (NOT b), (OR c1 d1), (OR c1 d2),···(OR c1 d*n*),
...
(OR c*m* d1), (OR c*m* d1), (OR c*m* d2), ···(OR c*m* d*n*).

In this search space, a deduction from (NOT a) to (OR (NOT c1) (NOT c2)···(NOT c*m*)) does not yield a contradiction, but some combination of d*j* ($j=1, n$), even when the specification a is satisfied by the control logic C. This implies that the refutation process would be rather complicated.

However, from Fig. 2 it is easy to find that each clause of the control logic excludes the others, since they are supported by different contacts of the same control relay. In light of this exclusiveness, which is represented as (OR (NOT C) (NOT D)), the search space given above can be decomposed into

(NOT a), (NOT b), c1, c2,···c*m*
(OR (NOT d1) (NOT d2)···(NOT d*n*))

and (NOT a), (NOT b). (OR (NOT c1) (NOT c2) ···(NOT c*m*)),
d1, d2,···d*n*.

In other words, verification by theorem-proving in this case can be divided into two cases, in each of which a straight forward deduction from one literal in the specifications can be expected to generate a refutation. The possibility of the decomposition is tested before adding the control logic to the initial connection graph by the following procedures:

(1) Convert the control logic into a disjuctive normal form.

(2) Check the exclusiveness of each pair of AND-elements in the converted form.

It is easy to prove that like the splitting rule of Davis and Putnam [10], the unsatisfiability of the search space does not change through this decomposition based on the model theory.

3.4.2 Control Strategy

Although the connection graph method does not require an search at each resolution step by inheriting resolution links from parent clauses, as mentioned above, a mechanism that selects the resolution link should be supplied. A link selection in the resolution procedure based on the connection graph method is realized by the following three steps:

(1) Selection of a clause

(2) Selection of a literal in the selected clause

(3) Selection of a resolution link in the selected literal.

VSEQ adopts a linear deduction method for the selection of the clause. That is, the proof procedure started from a negation of the theorem to be proved, the resolvent becomes a parent clause of the next resolution, and so on. However, the option of selecting a literal and a resolution link at each resolution step remains. Reflecting the characteristics of design verification, VSEQ implements the predicate-ordering and axiom-ordering mechanisms for literal and a resolution link selection, respectively.

(i) Predicate-ordering mechanism

Generally speaking, two types of clause are used in verification problems. One is a ground clause and the other is a clause containing variables. Design specifications, circuit descriptions, and control logic are represented by ground clauses, whereas function definitions are represented by clauses with variables. A theorem-proving process in design verification starts from the negation of each design specification and tries to find a contradiction in the proposed design (the con-

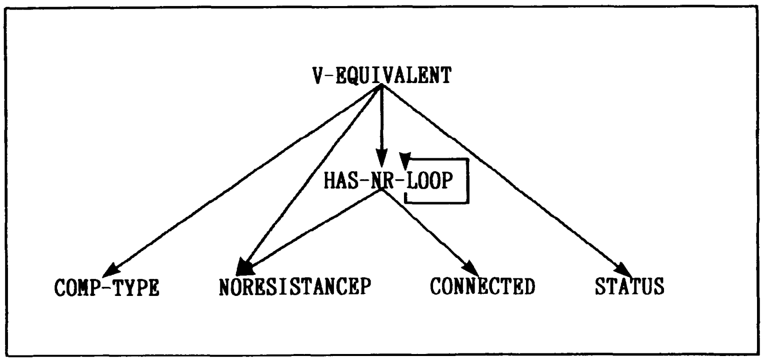

Fig. 7 Example of predicate tree.

trol logic in this case) by using functional definitions. Accordingly, the resolvents using functional definitions contain variables. On the other hand, an investigation of the termination procedure indicated that if the design to be verified is correct, that is, if the corresponding theorem is provable, the ground clauses contribute to terminating some branches of the proof process. Furthermore, if the design contains any errors, that is, if the corresponding theorem is not provable, ground clauses do much to reject some of the branches. Therefore, irrespective of success or failure, the theorem-proving process is made efficient by preferentially selecting the resolution links that connect to ground clauses.

The priority mentioned above can be determined by inspecting the relation between literals (or equivalently predicates) in the clauses of functional definitions. Each definition consists of positive and negative literals, if converted to a conjunctive normal form. Thus, it is possible to link the predicate of a positive literal to the predicate of a negative one in the same clause. Applying this process to all the definitions, we obtain tree structures that are denoted as predicate trees. The predicate tree obtained from the functional definitions given in Fig. 6 is shown in Fig. 7. In this figure, one arc that starts from and ends at the same predicate corresponds to a recursive definition, like the last one in Fig. 6. For each leaf predicate in this predicate tree, if the theorem is provable, the corresponding ground literal should exist and vice versa. In VSEQ, the above-mentioned predicate trees are constructed before starting the proof procedure, and are used to prioritize literals during literal selection in the proof procedure.

(ii) Axiom-ordering mechanism

VSEQ applies axiom ordering for the selection of the resoution link. The fact that a literal has more than one resolution link indicates that more than one definition has the same literal as its concluding parts. Therefore, by giving a priority to each of those definitions, an appropriate resolution link is selected. When a recursive definition is used, this ordering mechanism has the most effect. Axiom ordering corresponds to the processing mechanism of PROLOG, which tries to select clauses in the order in which they are input.

In addition to the mechanism mentioned above, VSEQ implements the following mechanisms, like other general-purpose theorem provers:

(a) A mechanism for checking infinite loops

(b) A mechanism for checking tautologies

(c) A mechanism for evaluating attached procedures.

3.5 Verification Example

VSEQ is implemented in VOS 3LISP (Common Lisp) and is currently being run on the HITAC M-series computer.

It has been applied to the verification of several sequential control circuits (interlock circuits) of an electric power substation like the one shown in Fig. 1. The sequential control circuit for the operation interlock of the component in Fig. 1 was three times as large as that shown in Fig. 2. It involved switches, relays, and their contacts, and the number of components was around 150. In this circuit, a total of 21 design specifications were specified, one for each disconnecting switch. VSEQ verified all of these specifications automatically.

One of the more complicated verifications was the one referred to in section 3.3. In this case, there were 76 clauses with 181 literals and 579 resolution links in the initial connection graph. VSEQ verified this case with only 59 proof steps within 30 seconds. An investigation of the proof history revealed that every unnecessary proof step was pruned in the first stage by the mechanism described in the previous section. Thus the verification process was found to be comparable in efficiency to that of a human expert.

By inserting intentional design errors, we affirmed that VSEQ could detect the following errors:

(1) Design errors due to the misuse and/or incorrect connection of components.

(2) Design errors due to roundabout circuits.

Through these applications, VSEQ was confirmed to be satisfactory for practical use.

4. Conclusions

A system called VSEQ that is based on a theorem-proving method has been developed for verifying the design of sequential control circuits. The main features of this system are as follows:

(1) It formulates the design verification of sequence control circuits as a theorem-proving problem based on a description of the objective system.

(2) It embodies efficient theorem-proving by decomposing the search space and by providing an efficient control scheme using domain knowledge.

Currently VSEQ is applicable to the design verification of sequential control circuits with contacts when they can be treated statically. It is now being extended to hanhdle dynamic information.

Through the application of VSEQ to a realistic problem it was seen that verification efficiency is improved by using knowledge about the domain characteristics in the verification formulation and also in the control of the proof process. The development of such a domain-specific theorem prover is a key to realizing a powerful design verification tool in other engineering domains.

Acknowledgements

We are deeply indebted to Mr. T. Kawai and Mr. K. Motomitsu of Kokubu Works, Hitachi, Ltd. and Ms. J. Ito of Systems Engineering Division, Hitachi, Ltd. for their useful discussions.

References

1. Barroe, H. G. VERIFY: A Program for Proving the Correctness of Digital Hardware Designs, *Artif. Intell.* **24**, p. 437 (1984).
2. Wagner, T. J. Hardware Verification, Ph. D. Dissertation, CSD, Stanford Univ., No. STAN-CS-77-632 (1977).
3. Wayhraugh, R. Prolegomena to a Theory of Mechanized Formal Reasoning, *Artif. Intell.* **13**, p. 133 (1980).
4. Herbert, J. The Application of Formal Specification and Verification to a Hardware Design, Proc. of CHDL-85, p. 434 (1985).
5. Wojcik, A. S. Formal Design Verification of Digital System, Proc. 20th DA Conf., p. 228 (1983).
6. Haynes, L. Logic Design Verification Using Static Analysis, Ph. D. Dissertation, Univ. of Illinois at Urbana-Champaign (1983).
7. Kowalski, R. A Proof Procedure Using Connection Graphs, JACM 22 (1975).
8. Siekmann J. *et al.* Paramodulated Connection Graphs, Acta Informatica, 13, p. 67 (1980).
9. Yamada N. *et al.* A Theorem Proving System for Logic Design Verification, Journal of Information Processing, 11, p. 92 (1988).
10. Davis, M. and Putnam, H. A Computing Procedure for Qualification Theory, JACM 7 (1960).

(Received August 28, 1989; revised July 2, 1990)