ICOT Technical Memorandum: TM-0693

TM-0693

# An Assumption-based Fault Localization Technique for Switching System Diagnosis

by
S. Wada, Y. Koseki & T. Nishida (NEC)

March, 1989



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**

# AN ASSUMPTION-BASED FAULT LOCALIZATION TECHNIQUE FOR SWITCHING SYSTEM DIAGNOSIS

Shin-ichi Wada Yoshiyuki Koseki
C&C Systems Research Laboratories

Tetsuro Nishida
Integrated Switching Development Division

NEC Corporation
4-1-1 Miyazaki, Miyamae-ku, Kawasaki, 213 JAPAN

Abstract

In troubleshooting large scale equipments, fault localization inferences involve indeterminateness and they might lead to erroneous conclusions. To realize indeterminate inferences, an assumption-based fault localization technique has been developed. The technique allows inferences based on indeterminate assumptions. When inconsistencies are discovered, assumptions are revised. The technique also enables flexible diagnosis control, such as reasoning based on anticipated test results without actually carrying out the tests. When the anticipated results become doubtful, it activates the omitted tests. The paper describes the technique and an outline of the troubleshooting expert system for electronic switching systems based on the technique.

## 1 Introduction

Modern electronic equipments, such as ESSs (Electronic Switching Systems), are very large-scale and complex. When troubles occur in such large-scale equipments, many components are considered as candidates of defective components. The key to successful diagnoses is to discriminate defective components from many other components efficiently.

In the discrimination, some inferences are indeterminate or uncertain due to possibilities of exceptions. If a troubleshooter does not deal with uncertainty, it cannot go on with a diagnosis when an erroneous conclusion is reached. Therefore, it is necessary to deal with uncertainty in troubleshooting.

Concerning this, some diagnostic expert systems adopt inference methods based on probability. For example, MYCIN [Shortliffe76] deals with the probabilities by *certainty factors*. However, it cannot deal with the probabilities maintaining consistency. It does not clearly point out indeterminate factors which might lead to erroneous conclusions. Therefore, it cannot appropriately revise the probabilities according to discoveries of inconsistency.

Rather than this, a diagnostic method is desired which clearly specifies indeterminate factors and maintains conclusions derived from them. By specifying indeterminate factors as assumptions, a troubleshooter can point out erroneous factors and revise them when inconsistencies are discovered. It can also correctly revise conclusions deduced from erroneous assumptions. To realize this capability, a fault localization technique based on assumptions has been developed and incorporated in a troubleshooting expert system for ESSs.

In the diagnostic problem domain, diagnostic methods based on truth maintenance techniques have been proposed [de Kleer87; Young87]. They make assumptions for functionalities or violations of the devices, and infer the device behaviors based on these assumptions. On the other hand, the presented technique differs in the objective and usage of assumptions. It uses assumptions for indeterminate factors in inferences, and narrows down suspected components based on them.

Among many possibilities in selecting assumptions to use, the presented technique realizes a control method which makes use of as many indeterminate assumptions as possible without causing inconsistency. By the method, the suspect components are narrowed down to a very few components. In addition, the technique enables reasoning based on anticipated test results without actually carrying out the tests. By introducing the technique, more flexible and efficient diagnosis is achieved.

The rest of this paper presents the motivation for the technique, the basic framework for assumption-based inferences, the control method for assumptions, and an outline of the ESS troubleshooting expert system based on the technique.

## 2 Motivation for Assumption-based Inferences

This section, first, describes the inferences carried out in ESS diagnoses and how the inferences involve

indeterminateness. Next, it describes the motivation for assumption-based inferences.

### 2.1 Heuristic-based Fault Localization

Modern ESSs for telephone exchanges are very large; they are composed of more than 1000 packages. When a fault occurs in them, many components are considered as *suspects* (suspect components). So, a troubleshooter needs to narrow them down to a few field-replaceable-units, or preferably only one. To achieve this, hardware diagnostic functions are built into an ESS. However, they cannot always detect defective components.

On the contrary, human experts employ several kinds of heuristic strategies for efficient troubleshooting. They carry out various tests for ESSs and narrow down suspects according to the test results. Since internal states in ESSs cannot be directly tested, they carry out tests to examine the external behaviors, like output messages, and guess the internal causes in ESSs based on the results.

For example, human experts examine the range where symptoms have been observed and infer suspects based on the result. Consider a simple equipment shown in Fig-1. Suppose that components *C1*, *C2*, *C3* and *C4* have a common structure and function. Also suppose that a fault on a path *A* - *Ci* is detected when signal transmission between *A* and *Ci* is activated. A fault detection causes an error message output which is observed by maintenance personnel. In the equipment, if error messages are observed on multiple paths *A* - *C1*, *A* - *C2* and *A* - *C4*, then a human expert thinks that the defective component should be *A* or *B*.

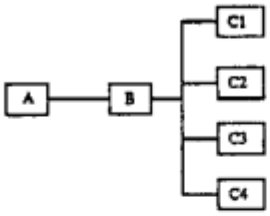



Fig-1: An example equipment

### 2.2 Indeterminateness in Inferences

Heuristic inferences modeled on those of human experts are effective in narrowing down suspects. However, such inferences include some uncertainty or indeterminateness.

For example, consider again an equipment in Fig-1. If multiple error messages are observed only on a specific path *A* - *C3*, then a human expert infers that the cause might be in *C3* rather than in *A* or *B*. This is because a fault in *A* or *B* would also cause error messages on multiple paths *A* - *C1*, *A* - *C2*, and *A* - *C4*. However, this inference is uncertain, because the fault detection might have been accidentally localized. If the observation time were longer, errors might also be observed on the other paths.

Another example is concerned with the interpretation of test results. Consider a test to detect a signal transmission using a signal-monitor device. If a test result is given as *"no signal is detected"*, then it can be heuristically inferred that *"some component for signal generation is malfunctioning"*. However, this inference is indeterminate, because the signal-monitor device or signal monitoring method might have failed while the signal is actually transmitted correctly.

Heuristic inferences like these might lead to incorrect conclusions. In such cases, non-defective units might be concluded as being suspects. Then, the trouble would not disappear when the suspected units are replaced with spares. It is difficult to go on with the diagnosis in such situations. Thus, a problem arises regarding how to revise incorrect conclusions and go on with diagnosis.

### 2.3 Omission of Time-consuming Tests

There is another kind of problem concerning test activation control. There are some tests which are time-consuming or detrimental to the service quality, and whose results can be anticipated. In order to narrow down the suspects and make the diagnoses faster, inferences should be carried out assuming the results without actually carrying out the tests. When the anticipations become doubtful, the omitted tests should be actually carried out.

For example, when an alarm lamp is not on, the following deduction might be possible:

> *"If an alarm lamp is not lighted, and the alarm lamp itself is normal, then suspects are limited to the area ....."*

In order for this deduction to be rigidly possible, a test to check the alarm-lamp normality is necessary. However, the test should be omitted since the result is anticipated to be normal. By assuming the normality of the alarm lamp without actually checking its condition, a diagnosis can be carried out faster. Then, a problem arises regarding the activation control for this kind of tests.

### 2.4 Necessity of Assumption-based Inferences

As described above, ESS diagnoses involve indeterminateness. Human experts know indeterminate factors which might lead to inferential errors, and go on with diagnoses assuming indeterminate factors. When they find inferential errors later, they modify the assumptions and conclusions derived from them. They also omit time-consuming tests and assume anticipated results. In order to realize this human-like method, an

assumption-based inference facility for fault localization was motivated and has been developed.

## 3 Facilities for Assumption-based Inferences

In order to realize diagnoses which involve indeterminateness, facilities for assumption-based inferences are incorporated in the troubleshooting system. This section briefly describes the facilities, whose details are described in section 4 and section 5. The facilities for assumption-based inferences comprise the following two parts:

a. **Basic framework for assumption-based inferences**

- Record relations among assumptions, facts and conclusions
- Maintain conclusion dependency on assumptions

b. **Assumption control unit**

- Determine which assumptions to be believed
- Select omitted tests to be carried out

The first unit records relations among assumptions, facts, and conclusions, separately from the troubleshooting inference mechanism. Its functions are similar to those of ATMS [de Kleer86]. When a new conclusion is made in a troubleshooting inference mechanism, the unit receives a report on the conclusion and antecedents used to derive it. The unit records the relation among assumptions and conclusions. According to the relation, the unit maintains dependency among conclusions and assumptions indicating which set of assumptions have derived conclusions.

The second unit determines which set of assumptions to be believed in inferences. It also determines the states of conclusions referring to the dependency maintained by the first unit. A troubleshooting inference mechanism goes on inferences believing the assumptions specified by the unit. The troubleshooting inference mechanism also refers to the states of conclusions maintained by the unit. When an inconsistency is discovered, the unit modifies states of the assumptions and conclusions in order to retain consistency. In addition, it can designate omitted tests which are effective in specifying the erroneous assumptions.

## 4 Basic Framework for Assumption-based Inferences

The basic framework records and maintains the relations among assumptions, facts, and conclusions, in accordance with the inferential progress for narrowing down suspects.

Some constructs, such as *assumption*, *justification*, and *nogood*, are introduced, similar to those in ATMS [de Kleer86]. According to newly derived inferences, *justifications* are created which represent the relations among conclusions and its antecedents. An example of a justification is as follows:

```
    fact1,
    fact2,
    assumption1
=> conclusion
```

In a justification, assumptions such as *assumption1* specify indeterminate causes in inferences. For test omission, assumptions also represent anticipated test results whose truth or falsity can be examined.

For discovering and dealing with inconsistencies, a special conclusion, called *nogood*, is introduced. As an example, consider an inconsistent case when the following two conditions hold:

- *"Units {X1, X2, ...} are suspected",*
- *"A trouble does not disappear, after suspected units {X1,X2,...} are replaced with spares."*

The second condition implies that *"units {X1,X2,...} are OK."* This inconsistency is represented in the following justification:

```
    conclusion1 (The suspected units are {X1,X2,...}),
    fact1 (The units {X1,X2,...} are OK.)
=> nogood
```

In the inferences for narrowing down suspects, justifications are created and then a chain of dependency is constructed. Fig-2 shows an example of a chain. It shows the relations among facts (test results), assumptions and conclusions regarding suspects. Dependency among conclusions and assumptions is also maintained. For instance, a conclusion in the right and lower side which means *"the suspected unit is {unitX}"* is based on the assumptions {*asmA*, *asmB*}. The figure also includes two nogoods. The nogood in the right side is based on assumptions {*asmA*, *asmB*, *asmC*} and the other is based on {*asmC*}.

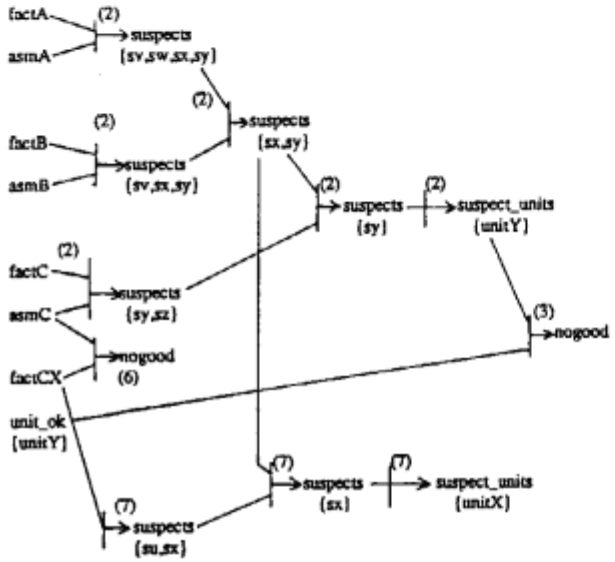
## 5 Assumption Control Method

In the framework described in section 4, there are multiple possibilities in selecting assumptions to be used. This section describes a control method for selecting assumptions. It first describes a condition of maintaining consistency introducing some terminologies. Next, it presents a principle and a method to meet the principle.

### 5.1 Condition of Maintaining Consistency

A word *environment* is introduced as a set of assumptions to be used during the inferential process. When a nogood is discovered, *nogood assumptions* are obtained as a set of assumptions which has derived the nogood. Nogood assumptions can be derived from the dependency record.

An environment is *consistent* if it is not a superset of any set of nogood assumptions; otherwise the environment is *inconsistent*. Inferences based on inconsistent set of assumptions should not be allowed in order to maintain consistency. Thus, a condition of maintaining consistency can be stated as *"inferences should be allowed only in consistent environments."*




factA : The symptom depends on the configuration.
asmA : The symptom is non-intermittent.
factB : The control order is distributed normally.
asmB : The clock works normally.
factC : A signal is not detected by a signal checking test.
asmC : The signal checker device works normally.
factCX: The signal checker device does not work.

Fig-2: Justification chain

## 5.2 Principles

Considering how assumptions should be used in troubleshooting, the principles in controlling assumptions are as follows:

**(1) Utilize as many assumptions as possible:**
Utilization of assumptions are effective in narrowing suspects. Then, a troubleshooter should use as many assumptions as possible without causing inconsistency. This principle implies that all assumptions should be used, when no inconsistency is discovered.

**(2) Utilize multiple environments simultaneously:**

When a nogood is produced based on more than one assumptions, there are multiple candidate environments (sets of assumptions) to resolve the inconsistency. For example, consider that a set of nogood assumptions is obtained due to a nogood discovery:

Nogood: $\{asmA,\ asmB,\ asmC\}$

Consider that the following environment has been believed and needs to be revised:

Environment: $\{asmA,\ asmB,\ asmC,\ asmX,\ ...\}$

Then, three environments are candidates to resolve the inconsistency:

Candidates of new environments:

$\{asmA,\ asmB,\ asmX,\ ...\}$
$\{asmA,\ asmC,\ asmX,\ ...\}$
$\{asmB,\ asmC,\ asmX,\ ...\}$

There seems to be no reasonable way to determine only one of these, especially when the assumptions have the same plausibility. For this reason, multiple environments should be used simultaneously, and inferences should be allowed in any of these environments.

**(3) Activate omitted tests:**
For quick diagnosis, some tests, like time-consuming tests, are omitted and the anticipated results are assumed. However, when multiple environments are possible, some of the omitted tests might be effective in narrowing the multiple possibilities. These tests should be actually carried out so that the multiple possibilities of environments could be narrowed by the truth or falseness of the anticipated results.

## 5.3 Method for Selecting Assumptions

This section describes a control method which realizes the principles (1) and (2) in section 5.2. For representing the control method, a set of maximal and consistent environments, called *MCES (Maximal Consistent Environment Set)*, is introduced:

MCES: *MCES (Maximal Consistent Environment Set)* is a set of environments which satisfy the following:

1. The environment is consistent.
2. The environment is maximal. That is, it is not a subset of any consistent environments.

A control method which realizes the principles 1. and 2. in section 5.2 is to use the environments in the MCES simultaneously. This is because the MCES

consists of all the maximal environments which include as many assumptions as possible retaining consistency.

The MCES should be updated due to nogood discoveries. Initially, if no nogoods are discovered, the MCES consists of an environment containing all assumptions. When a nogood is discovered, it is revised so that it can retain consistency and maximality.

For example, consider that the MCES is initially as the following:

MCES[0]:
$\{asmA,\ asmB,\ asmC,\ asmX,...\}$

Consider that, due to a nogood discovery, the nogood assumptions are obtained:

Nogood assumptions: $\{asmA,\ asmB,\ asmC\}$

Then, the MCES is revised as the following:

MCES[1]:
$\{\{asmA,\ asmB,\ asmX,\ ...\},$
$\{asmA,\ asmC,\ asmX,\ ...\},$
$\{asmB,\ asmC,\ asmX,\ ...\}\}$

Consider that another set of nogood assumptions is obtained:

Nogood assumptions: $\{asmB,\ asmC\}$

Then, the MCES is revised as the following:

MCES[2]:
$\{\{asmA,\ asmB,\ asmX,\ ...\},$
$\{asmA,\ asmC,\ asmX,\ ...\}\}$

### 5.4 Activation Control for Omitted Tests

Based on the principle (3) in 5.2, an omitted test should be carried out, if it is effective in reducing the number of environments in MCES. When the MCES is updated due to a nogood, tests which satisfy the following conditions should be carried out:

1. The test has been omitted.
2. The assumption for anticipated result of the test is contained in some environments in the MCES, and is not contained in some other environments in the MCES.

The second condition implies that some environments in the MCES can be eliminated from the MCES, based on the truth or falsity of the anticipated result.

Based on the actual test result, the MCES should be revised as the following:

When the anticipated test result is validated:
The revised MCES should consists of the environments which are in the former MCES and which contain the assumption for anticipated test result.

When the anticipated test result turns out to be false:
The MCES should consists of the environments which are in the former MCES and which do not contain the assumption for anticipated test result.

Referring to the example MCES[2] in section 5.3, if $asmB$ is the anticipated result of omitted test, this test

| Phase | MCES | Nogood assumptions | Suspected units |
|---|---|---|---|
| **(1)Initial state:** All the assumptions are allowed to be used in inferences. Among them, *asmC* is an anticipated result for an omitted test. | {{asmA,asmB,asmC}} | | |
| **(2)Narrowing suspects:** Several facts (test results) are obtained by carrying out tests, and suspects are narrowed down using assumptions. This way, a conclusion is reached that the *unitY* is defective. | | | {unitY} |
| **(3)Unit replacement & contradiction:** *UnitY* is replaced; however, the trouble does not disappear. This causes a contradiction. | | {asmA,asmB,asmC} | |
| **(4)Contradiction resolution:** The MCES is revised according to the nogood assumptions {*asmA, asmB, asmC*}. | {{asmA,asmB},{asmB,asmC}, {asmA,asmC}} | | |
| **(5)Carrying out an omitted test:** Because *asmC* is an anticipated result for an omitted test, the test is carried out. | | | |
| **(6)Finding an assumption error:** The test result gives *factCX*, which is contrary to *asmC*. This produces a nogood, and the MCES is revised. Environments, including *asmC*, are removed from the MCES, and only an environment {*asmA, asmB*} remains in the MCES. | {{asmA,asmB}} | {asmC} & {asmA,asmB,asmC} | |
| **(7)Another suspect:** Based on *factCX*, new inferences are made, and another unit *(unitX)* is considered to be defective. | | | {unitX} |

Fig-3: Diagnosis example

is effective and should be carried out. On the other hand, even if $asmA$ is the anticipated result of omitted test, it is not effective and should not be carried out. If $asmB$ is validated by an actual test, then the MCES[2] can be reduced to the following:

MCES[3a]:
$\{\{asmA,\ asmB,\ asmX,\ ...\}\}$

If $asmB$ turns out to be false, then the MCES[2] can be reduced to the following:

MCES[3b]:
$\{\{asmA,\ asmC,\ asmX,\ ...\}\}$

### 5.5 Example

This section presents a simplified example of diagnosis process. The example refers to the reasoning chain in Fig-2. The diagnosis progress is shown in Fig-3. It includes explanations for diagnosis phase, the MCES, nogood assumptions, and suspect units. As shown in the figure, when a nogood is discovered, a test which has been omitted is activated and suspects are revised. In this way, flexible diagnosis is achieved.

## 6 ESS Troubleshooting System Outline

Based on the fault localization technique, a prototype of a troubleshooting expert system for ESSs has been developed on a PSI (Prolog machine) [Taki84]. It has been developed by incorporating the assumption-based inference facilities into a heuristic-based troubleshooting expert system without such facilities [Koseki87]. This section describes the incorporation. Also, it describes other features of the troubleshooting expert system.

### 6.1 Incorporation of Assumption-based Inference Facilities

The facilities for assumption-based inferences described above are incorporated in the troubleshooting expert system for ESSs. The facilities carry out recording inferential relations and maintaining consistency. The troubleshooter carries out the other tasks involved in troubleshooting, such as selecting tests, carrying out tests, and inferring suspects.

The control method for diagnosis is shown in Fig-4. The left part of the diagram shows the control for the case when no inconsistency is discovered. The system starts a diagnosis by symptom input. It analyzes the symptom and generates suspected components. The system narrows down the suspected components by carrying out appropriate tests and by omitting tests. The system recommends that the maintenance technician replace suspect units. The system continues this process as long as no inconsistency is discovered.

The right part of the diagram shows a control for the cases when inconsistencies (nogoods) are discovered. In such cases, the system updates the MCES according to nogood assumptions and it changes states of conclusions in order to maintain consistency. Then, it carries out omitted tests which are effective in reducing the size of MCES, if such tests exist.

The knowledge base has been modified for incorporating the assumption-based inference facilities. The knowledge base for only usual cases has been modified

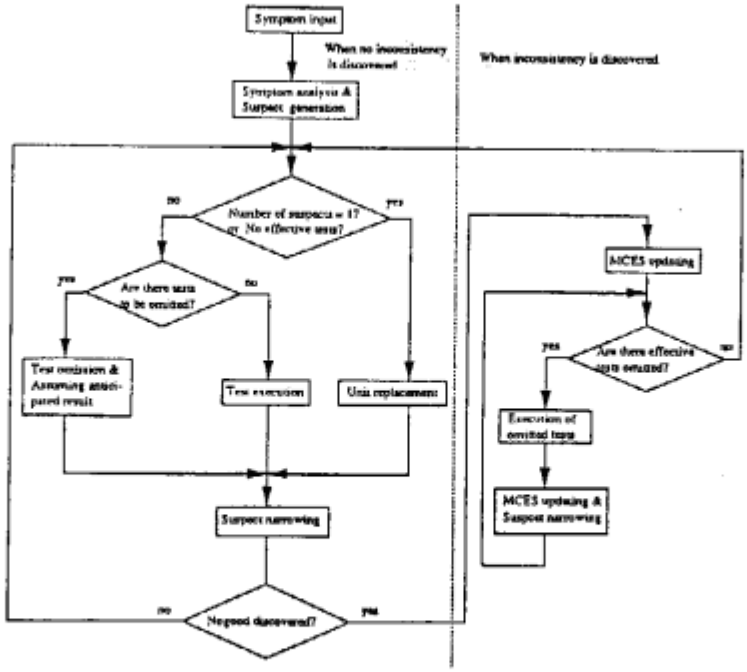



Fig-4: A diagnosis control flow

into a more accurate version which clearly distinguishes uncertainty in the indeterminate inferences. The modification has been naturally achieved by adding assumptions to specify uncertain factors and descriptions of tests and suspect narrowing method in exceptional cases.

### 6.2 Features

This section briefly describes features of the troubleshooting system for ESSs, whose details are described in [Wada88].

**Abstract-signal-flow description:**
The system includes structural knowledge regarding the target ESS, in a form called *abstract-signal-flow* model. Fig-5 shows its extract. Developing and modifying knowledge base in this style are far easier than developing and modifying knowledge base comprising empirical symptom-cause associating rules, like MYCIN [ShortlifTe76]. Such development and modification are also easier than fully describing logic-gate level circuit descriptions. Using this method, the entire ESS structure was successfully described.

**Network-based knowledge representation:**
Several kinds of knowledge, regarding symptom, tests and structure, are represented in a unified network by utilizing the object-oriented facilities in *ESP (Extended Self-contained Prolog)* [Chikayama84]. Based on the representation method, the knowledge base is kept compact and modifiable.

**Flexible man-machine interaction:**
The system allows flexibility in man-machine interaction. For instance, the system can hypothetically narrow down suspects based on user-specified test results. This facility helps the user understand the effectiveness of tests. It is easily implemented in the framework for assumption-based inferences.

**User friendly interface:**
The system provides a user friendly interface. An example of a user-interface screen is shown in Fig-6. It shows current suspects, location of the physical units, abstract-signal-flow diagrams, a justification chain diagram, and so on.

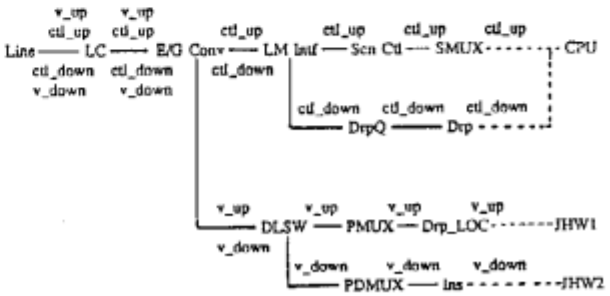



Fig-5: An abstract-signal-flow model

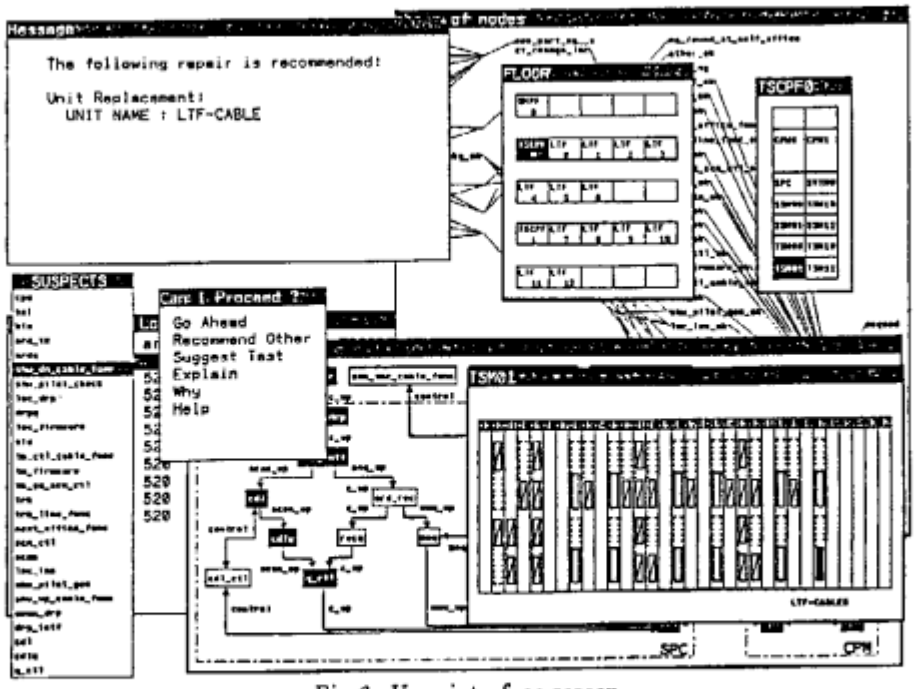



Fig-6: User interface screen

## 7 Concluding Remarks

This paper has presented an assumption-based fault localization technique and a troubleshooting expert system for ESSs. The technique enables effective diagnostic inferences which involve uncertainty. The described control realizes default inferences for narrowing down suspects and revises assumptions due to inconsistencies. The method also enables flexible diagnosis control, like omission of time-consuming tests.

By incorporating the assumption-based facility into a troubleshooting system, the system has become able to continue diagnosis when the trouble still remains after suspected units have been replaced. Modifying the knowledge base was smoothly achieved, mainly by adding assumptions to specify uncertain factors. The assumption-based technique, together with the compact knowledge representation, makes troubleshooting possible for large-scale ESSs. While the presented technique is used in a heuristic-based system, it can also be applied to the system in *model-based* diagnostic methods, such as [Genesereth84; Davis84].


## Acknowledgments

The authors express deep appreciation to Yasufumi Souda, Satoshi Goto, Yoshihiro Nagai and Hajimu Mori for continuous encouragement and support. The authors thank Toru Hisada, Katsuki Satozaki, Nobuyasu Wakasugi, Mitsugu Oishi, Hideharu Kijima and other members for developing the expert system. This research was carried out as a part of the Fifth Generation Computer Project of Japan.


## References


[Chikayama84] T. Chikayama, S. Takagi, and K. Takei, "ESP - An Object Oriented Logic Programming Language," *ICOT Technical Report TM-0075*, Institute for New Generation Computer Technology, 1984.

[Davis84] R. Davis, "Diagnostic Reasoning Based on Structure and Behavior," *Artificial Intelligence 24*, 1984, pp. 347-410.

[de Kleer86] J. de Kleer, "An Assumption-based TMS," *Artificial Intelligence 28*, 1986, pp.127-162.

[de Kleer87] J. de Kleer and B. C. Williams, "Diagnosing Multiple Faults," *Artificial Intelligence 32*, 1987, pp.97-130.

[Genesereth84] M. R. Genesereth, "The Use of Design Descriptions in Automated Diagnosis," *Artificial Intelligence 24*, 1984, pp. 411-436.

[Koseki87] Y. Koseki, S. Wada, T. Nishida, and H. Mori, "SHOOTX: A Multiple Knowledge Based Diagnosis Expert System for NEAX61 ESS," *Proc. of the International Switching Symposium*, March 1987, pp.78-82.

[Shortliffe76] E. J. Shortliffe, *Computer Based Medical Consultations: MYCIN*, Elsevier, New York, 1976.

[Taki84] K. Taki, M. Yokota, A. Yamamoto, H. Nishikawa, S. Uchida, H. Nakashima and A. Mitsuishi, "Hardware Design and Implementation of the Personal Sequential Inference Machine (PSI)," *Proc. of the International Conference on Fifth Generation Computer Systems 1984*, November 1984, pp.398-409.

[Young87] M. Young, "A Framework for Describing Troubleshooting Behavior Using Default Reasoning and Functional Abstraction," *Proc. of the Third Conference on Artificial Intelligence Applications*, February 1987, pp.260-265.

[Wada88] S. Wada, Y. Koseki, and T. Nishida, "A Human Strategy-based Troubleshooting Expert System for Switching Systems," *Proc. of the International Conference on Fifth Generation Computer Systems 1988*, November 1988, pp.1291-1298.