アルゴリズム 16−13
（1990.7.17）

# パッケージ概念を導入した故障診断アルゴリズムの改訂

中川 嘉宏* 栗原 正仁** 大内 東**
（* 北海道工業大学）（** 北海道大学）

Reiter の故障診断理論を拡張した故障診断アルゴリズムと、そのアルゴリズムを組み込んだ診断システム DiaLog による論理回路の故障診断実験結果について述べる。システム構成要素の上位にパッケージの概念を導入すると、従来の素子レベルの故障診断と比較して探索空間の縮小と効果的な枝の刈込みによる効率的な診断が期待でき、大規模な回路の診断が可能になる。実験は組合せ論理回路であるデコーダとやや大規模な順序回路(自動販売機)を例にとり故障診断を行ない、実用的な時間で診断結果を得た。

# A REVISION OF FAULT DIAGNOSIS ALGORITHM BASED ON PACKAGES

Yoshihiro Nakagawa* Masahito Kurihara** & Azuma Ohuchi**

* Dept. of Electrical Engineering, Hokkaido Institute of Technology, Sapporo 006, Japan
** Dept. of Information Engineering, Hokkaido Univ., Sapporo 060, Japan

We present a general fault diagnosis theory which extends the Reiter's component-level theory to the package-level. We implemented it by using the automated reasoning system DiaLog, which we have developing since 1988. We applied it to the diagnoses of logic circuit by supplying the DiaLog with the description of gates, wiring, the axioms of Boolean algebra, etc. The experiments show that the proposed approach is potentially practical for large scale digital circuits.

## 1. はじめに

Reiterは、1987年に経験的知識あるいは特定の診断対象に依存しない故障診断理論を提案した[1]。この方式は診断対象が内包する論理のみに着目して故障原因を推論（深い推論）する。これをシステムの故障診断に適用するには、各システム要素の機能、要素間の接続関係、システムからの観測などを一階述語論理などの言語を用いて論理的に表現し、自動推論 (automated reasoning) プログラムによりそれらの間の論理的矛盾の検査を行なう。そのとき矛盾が検出されればその証明から異常素子の抽出を行なう。

本稿では、はじめに論理回路の故障診断を効率的に行なえるように Reiter の故障診断アルゴリズムを拡張する。具体的には、システム構成要素の上位にパッケージの概念を導入する。これは回路の素子レベルで故障を特定できても、実際の保守の場面ではその素子を含むパッケージ交換となる場合が多いことに着目し、実用性を失うことなしに大規模システムの診断をより効率化させるものである。次に拡張したアルゴリズムに基づいて論理回路の故障診断実験を行なったのでその結果について報告する。

## 2. パッケージ診断

Reiter の理論は、故障診断理論を特定の問題領域だけではなく、一般のシステムに適用することを可能とするために、システムの概念を問題領域とは独立に定義している。本稿で示すパッケージ診断理論においてもその手法を利用するが、Reiterの理論との相違点は、システム構成要素の上位にパッケージが存在することである。

### <2.1> システム

[定義 2.1] システムは (SD, COMPONENTS, PACKAGES) である。ただし、

(1) SD(System Description)はシステムの構成要素や各要素の機能を記述した第一階述語論理式の集合である。システム記述は一般に、要素 c が「異常」であることを意味する 1 引数述語 AB(c) を含む。

(2) COMPONENTS はシステム構成要素を表す定項の集合である。

(3) PACKAGES は COMPONENTS の分割である。その各要素をパッケージと呼ぶ。

例えば COMPONENTS = {A, B, C, D, E, F, G} のとき、{{A, B}, {C, D}, {E, F, G}} は三つのパッケージからなる PACKAGES の例である。

### <2.2> システムの観測

異常性の検知と診断には観測が必要である。

[定義 2.2] システムの観測は第一階述語論理式の集合である。

### <2.3> パッケージ診断

最初に Reiter による診断の定義を紹介する。

[定義 2.3] (SD, COMPONENTS, OBS) に対する要素レベルの診断とは、

$SD \cup OBS \cup \{\neg AB(c) \mid c \in COMPONENTS - \Delta\}$ (1)

が無矛盾となる極小集合 $\Delta \subseteq COMPONENTS$ である。

この定義は次のように解釈できる。$\Delta^*$ を診断とする。定義より、$\Delta^*$に属さない要素のみの正常性を仮定すると、システム記述と観測の関係を矛盾なく説明できる。また、論理式の集合は論理的には連言を表わすから、$\Delta^*$を包含するΔに対しても (1)式は無矛盾である。しかし $\Delta^*$の真部分集合 Δ に対しては定義により (1)式は矛盾を含む。また、特にシステム記述と観測との間にはじめから矛盾がないときに限り、$\Delta^* = \{\}$ である。

次に、本稿で新しく提案するパッケージ診断を定義する。以下 $\pi$ は要素 $c \in COMPONENTS$ を引数とし、c が属するパッケージ $P \in PACKAGES$ を値とする関数である。またパッケージの集合 $\Delta p$ 中に現れる要素の集合を $comp(\Delta p) = \{c \mid (\exists P)\ c \in P \in \Delta p\}$、要素の集合 Δ を包含するための最小のパッケージ集合を、$pack(\Delta) = \{\pi(c) \mid c \in \Delta\}$とする。明らかに $\Delta \subseteq comp(pack(\Delta))$、$\Delta p = pack(comp(\Delta p))$ が成り立つ。

[定義2.4] (SD, COMPONENTS, PACKAGES, OBS)に対するパッケージ診断とは、

$SD \cup OBS \cup \{\neg AB(c) \mid c \in COMPONENTS - comp(\Delta p)\}$ (2)

が無矛盾となる極小集合 $\Delta p \subseteq PACKAGES$ である。

(2)式はパッケージ内の素子毎にまとめて正常性（または異常性）を仮定することを意味している。従って、この定義に基づく診断は、Reiterの理論の枠内で診断が

可能である。さらに、診断空間の大幅な縮小から、考察すべき仮定の組合せが少なくなり、明らかに要素レベルの診断より計算時間が短くなる。その反面、この様な直接パッケージ診断は診断結果の理由説明が粗くなる。例えば、単一パッケージ {E, F, G}の故障とのパッケージ診断が得られた場合、E, F, G のいずれかの単一故障なのか、あるいは二重、三重の故障なのかの説明が得られない。特に別のパッケージ診断 {A, B}も得られているときには、いずれがより「もっともらしい」かの判断にとってこの理由説明は重要である。従って、本稿では直接パッケージ診断の効率を活かしながらも、要素レベルの理由説明（パッケージ内の少なくともどの素子が異常であるかの説明）もできるような診断手法を求めることにする。これはReiterの提案した素子レベル診断と、直接パッケージ診断との中間的な診断アプローチである。

[命題2.5] Δが診断ならば、Δp⊆ pack(Δ)を満たすパッケージ診断Δp が存在する。

(証明) Δ' p=pack(Δ)、Δ'=comp(Δ' p)とすると、Δ ⊆Δ' であり、Δ' は (1)式を無矛盾とする。よって、 Δ'p は(2)を無矛盾とする。従って、パッケージ診断Δp ⊆ Δ' p が存在する。□

この命題において、Δp=pack(Δ) になるとは限らない。例えば、PACKAGES={{A, B}, {C, D}}とし、2つの診断Δ1={A}, Δ2={B, C} があるとしよう。Δp1 = pack(Δ1) = {{A, B}}はパッケージ診断である。Δp2 = pack(Δ2) =PACKAGES は極小性を満たさないので、パッケージ診断ではない。ただし、命題2.5で存在が保証されているΔ2に対するパッケージ診断はΔp1である。

[命題2.6] Δp がパッケージ診断ならば、pack(Δ)=Δp を満たす診断Δが存在する。また、pack(Δ') ⊂Δp を満たす診断 Δ' は存在しない。

(証明) Δ*= comp(Δp)とする。Δpは (2)式を無矛盾にするから、Δ*は (1)を無矛盾とする。よって、Δ ⊆Δ* を満たす診断 Δ が存在する。 pack(Δ) ⊆ pack (Δ*) = Δp であるから、pack(Δ') ⊂ Δp を満たす Δ' 診断が存在するとして矛盾を示せばよい。

命題2.5より、Δ' p ⊆ pack(Δ') ⊂ Δp を満たすパッケージ診断Δ' p が存在する。これは、Δpの極小性に矛盾する。□

[定理2.7] Δp がパッケージ診断であることと、Δp が集合 pack(Δ1),...,pack(Δn)のうち極小なものに一致することは同値である。ただし、 Δ1, ..,Δn は (SD, COMPONENTS, OBS) に対するすべての診断である。

(証明) Δp がパッケージ診断ならば、命題2.6より pack(Δ)=Δp を満たす診断 Δ が存在し、 pack(Δ')⊂ pack(Δ) を満たす診断Δ' は存在しない。よって、pack(Δ)は極小である。

逆に、Δp がpack(Δ1),...,pack(Δn)の極小のもの、例えば pack(Δ)に一致するとき、命題2.5より

Δ'p ⊆ pack(Δ)=Δp を満たすパッケージ診断Δ'p が存在する。もし、Δ' p ⊂ pack(Δ)=Δp ならば、命題2.6より、 pack (Δ')=Δ'p ⊂ pack(Δ) を満たす診断Δ' が存在し、 pack(Δ)の極小性に反する。よって、Δ' p= pack(Δ) = Δp、すなわち Δp はパッケージ診断である。□

ここで、Δp=pack(Δ) のとき、 Δ を Δp の根拠という。

### <2.4> パッケージ診断の計算

パッケージ診断の計算は、基本的には Reiter の方法に基づいて要素レベルの診断を計算し、定理 2.7 によりその極小なものを求める考え方で行なう。ただし、この仮定において明らかに極小性を満たさない解の候補は刈込みにより探索の範囲から除外される。Reiter の方法はコンフリクト集合の計算とHS-木の生成から成っている。コンフリクト集合とは、

SD ∪ OBS ∪ {¬AB(c) | c ∈ S} (3)

が矛盾となるような集合 S である。HS-木の終端（葉）ノードのラベルが √、非終端ノードのラベルがコンフリクト集合であり、枝のラベルがシステム構成要素であるような木である[1]。

以下にパッケージ診断アルゴリズムを示す。ただし、H(n) は、ルートからノードnまでのパスについているラベルの集合である。また、 HP(n) = pack(H(n)) である。また、ここで使用する定理証明器は矛盾性を検出するために用いるもので、矛盾ならばコンフリクト集合をかえし、無矛盾ならば √ を返す。

Step1 システム構成要素がすべて正常であると仮定して定理証明器を呼び出す。もし √ が返ればパッケージ診断は { } であり、終了。コンフリクト集合が返れ

ば、そのコンフリクト集合を HS-木 の根のラベルとする。（ラベルがコンフリクト集合でないノードを「閉じられたノード」と呼ぶ。）

Step2　すべてのノードが閉じられていれば終了。このとき ✓ でラベル付けされたノード n の HP(n) のうち、極小のものがパッケージ診断となる。

Step3　まだ閉じられていないノード n を任意に選ぶ。n がコンフリクト集合 S によってラベル付けされているとする。それぞれの $\sigma \in S$ について、$\sigma$ をラベルとする枝を介して子ノード $n_\sigma$ を生成する。COMPONENTS - H($n_\sigma$) の要素を正常と仮定して定理証明器を呼び出し、それが返した値（✓またはコンフリクト集合）を $n_\sigma$ のラベルとする。Step2 へ進む。

Step3 において $n_\sigma$ のラベルを計算するときに、すでに求められているコンフリクト集合 S のうちで、$S \cap H(n_\sigma) = \phi$ を満たすものがあれば、定理証明器を呼び出さずに $n_\sigma$ のラベルを S としてよい。

このアルゴリズムによって生成される HS-木は冗長な部分があるので、Reiter は診断の極小性に基づく以下の三つの刈り込みを行なって効率を上げている。

(1) ✓がついた他のノードと H(n) が同じか、大きくなるようなノード n は閉じる。

(2)すでに生成されたノードと同じ H(n) となるノード n は閉じる。

(3) ノード n のラベル S の真部分集合 S' がノード n' ラベルであるとき、各要素 $\alpha \in S-S'$ に対し $\alpha$ をラベルとして n から出ている枝をカットする。

パッケージ診断ではさらに極小性による以下の刈り込みを行なうことができる。

(4)ノード n が ✓ によってラベル付けされていて、ルートから n へのパス上にないノード n' が HP(n) ⊆ HP(n')を満たすとき、n' の親ノードと n' を結ぶ枝をカットする。

図1 に HS-木生成の例を示す。最初にコンフリクト集合 {A, B, C, F, G} が得られたとしこれを根のラベルとする。根の第一子ノードのラベルとしてコンフリクト集合 {C, F, G} が得られたとする。刈り込み(3)によりラベル A, B の枝をカットする。第三子ノードにおいてコンフリクト集合 (D, F, H) が得られたとする。第四子ノードにおいて ✓ が得られたとする。第五子ノードと

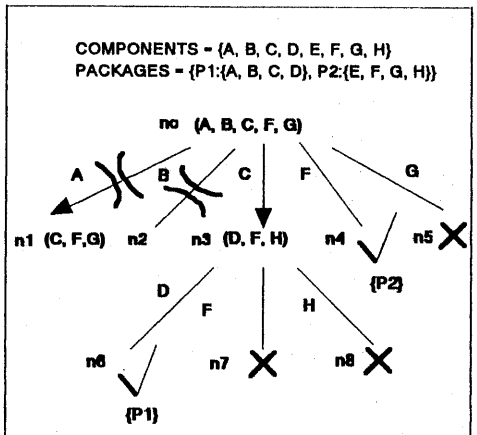

Fig.1 Generating HS-tree by package diagnosis algorithm

根を接なぐ枝は刈込み(4)によりカットされる。診断 {F} を根拠にパッケージ診断 {{E, F, G, H}} が得られる。次に、{ D, F, H } とラベル付けされたノード( n3) から D, F, H とラベル付けされる枝をのばす。D に隣接するノード n6 においては ✓ が得られた場合は、診断 { C, D } を根拠にパッケージ診断 {{A, B, C, D}} が得られる。ノード n7 は刈り込み (1)により閉じられる。ノード n8 は刈り込み (4)のパッケージ診断の極小性から閉じられる( Reiter の診断ではこのカットは行なわれないことに注意 )。

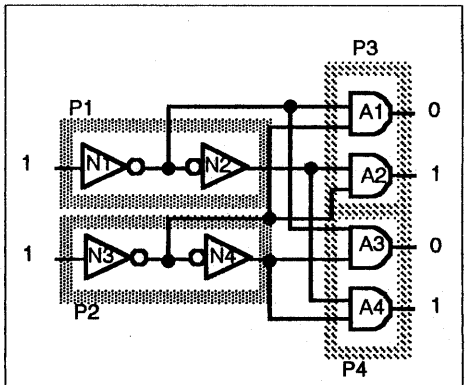

Fig.2 An example circuit - 2-inputs decoder

### 3. 知識表現と診断実験

はじめに、図2のデコータ回路を例にとりパッケージ

診断による故障検出を行なう上で考慮すべき知識表現について述べる。次に故障診断の具体例としてデコーダ及び自動販売機について実験結果を示す。なお、使用した定理証明器は著者らが 1984年より開発、調整を行なっている Thinker を用いた[2],[3]。また、Thinker を制御する診断システムを DiaLog と呼ぶ。

```
SYSTEM "2-INPUTS-DECODER"
PACKAGES
 P1 NOTG (N1 N2)
 P2 NOTG (N3 N4)
 P3 ANDG (A1 A2)
 P4 ANDG (A3 A4)
GENERAL
C1  {-ANDG(?x) AB(?x)
      EQUAL (out(?x),and(in1(?x),in2(?x)))}
C2  {-NOTG(?x) AB(?x) EQUAL(out(?x),not(in1(?x)))}
C3  {-EQUAL(1,0)}
C4  {-EQUAL(out(?x),1)-EQUAL(out(?x),0)}
DEMODULATORS
M0  {EQUAL(out(S1),1)}
M1  {EQUAL(out(S2),1)}
M2  {EQUAL(out(A1),0)}
M3  {EQUAL(out(A2),1)}
M4  {EQUAL(out(A3),0)}
M5  {EQUAL(out(A4),1)}
C10  {EQUAL(in1(N1),out(S1))}
C11  {EQUAL(in1(N3),out(S2))}
C12  {EQUAL(in1(N2),out(N1))}
C13  {EQUAL(in1(A1),out(N1))}
C14  {EQUAL(in1(A3),out(N1))}
C15  {EQUAL(in1(N4),out(N3))}
C16  {EQUAL(in2(A1),out(N3))}
C17  {EQUAL(in2(A2),out(N3))}
C18  {EQUAL(in1(A2),out(N2))}
C19  {EQUAL(in1(A4),out(N2))}
C20  {EQUAL(in2(A3),out(N4))}
C21  {EQUAL(in2(A4),out(N4))}
C50  {EQUAL(and(?x,?x),?x)}
C51  {EQUAL(and(1,?x),?x)}
C52  {EQUAL(and(0,?x),0)}
C53  {EQUAL(and(?x,?y),and(?y,?x),t)}
C54  {EQUAL(or(?x,?x),?x)}
C55  {EQUAL(or(0,?x),?x)}
C56  {EQUAL(or(1,?x),1)}
C57  {EQUAL(or(?x,?y),or(?y,?x),t)}
C58  {EQUAL(not(1),0)}
C59  {EQUAL(not(0),1)}
PARAMODULATORS
C90  {EQUAL(out(?x),1)EQUAL(out(?x),0)}
SUBSUMER
C999 {EQUAL(?x,?x)}
END
```

Fig.3 System Description for the 2-inputs decoder

### <3.1> COMPONENTS

故障診断の対象となる素子の集合で、図2 の回路では、COMPONENTS = {N1, N2, N3, N4, A1, A2, A3, A4}である。

### <3.2> システム記述 SD

SD としては、素子動作、回路接続情報、ブール代数の公理、等式の公理、2値性の記述及び素子記号の種別を一階述語論理の節形式で記述する。図2の回路の SD を図3に示した。図3で、節 C1, C2 の記号 ANDG(x), NOTG(x) はそれぞれ、素子 x (変数) が AND ゲート、NOTゲートであることを示す述語、AB(x) は素子 x が異常(abnormal), EQUAL(x, y) は x と y が等しいことを意味する述語である。(EQUAL は Thinker の組込み述語、AB は DiaLog が認識する特別な述語である)。C1 の表現は、「x が AND ゲートで、かつ、x が異常でなければ、x の出力はその 2 つの入力値の and 関数の値に等しい」という意味を表している。

### <3.3> システム観測 OBS

OBS を等式で表現する。図3の場合、M0,...,M5 である。節 M3 が回路の異常を示す。回路が正常に動作している場合、与えられた入力データでは ANDゲート A2 の出力はゼロとなる。また、M0, M1 に現れる記号 S1, S2 は仮想の信号源素子を意味する。

### <3.4> PACKAGES

PACKAGES は COMPONENTS の分割である。図2の回路では、

PACKAGE = {{N1, N2}, {N3, N4}, {A1, A2}, {A3, A4}}

となる。DiaLog では図3に示すように "P1 NOTG ( N1 N2 )" のように記述し、NOTゲート N1, N2 がパッケージ名 P1 に含まれていることを表す。

```
          DiaLog for Macintosh II Ver.1.0
    Use the HELP command to display the list of all
 the DiaLog commands.

  DIALOG-TP NIL
  CONFLICT-Set: (N3 N2 N1 A2)
  (REFUTE) took 477 ticks (7.950 seconds) to run.

  DIALOG-TP (N3)
  CONFLICT-Set: (N4 N2 N1 A4 A2)
  (REFUTE) took 505 ticks (8.417 seconds) to run.

  DIALOG-TP (N2)
  CONFLICT-Set: (N3 A2)
  (REFUTE) took 357 ticks (5.950 seconds) to run.

  PRUNING TREE:: (n0)-N1-
  PRUNING TREE:: (n0)-N2-(n2)-N3-
  PRUNING TREE:: (n0)-N2-(n2)-A2-
  DIALOG-TP (A2)
  CHECKED
  (REFUTE) took 794 ticks (13.233 seconds) to run.

  DIALOG-TP (N4 N3)
  CHECKED
  (REFUTE) took 973 ticks (16.217 seconds) to run.
   Of that, 210 ticks (3.500 seconds) was spent in
GC.

  DIAGNOSES PACKAGES::
  (P3) BASED-ON: (A2)
  (P2) BASED-ON: (N4 N3)
  ?
```

Fig.4 A diagnosis conserning the 2-inputs decoder.

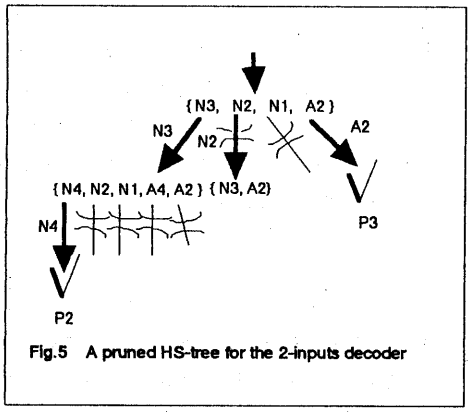


Fig.5 A pruned HS-tree for the 2-inputs decoder

### <3.5> デコーダ診断実験結果

図4に診断の実行時の様子を示した。この実験に使用した DiaLog 及び Thinker は Macintosh II の Common Lisp でインプリメントされている。

はじめに、全ての素子が正常と仮定したとき、コンフリクト集合 {N3, N2, N1, A2} が得られ、その後、N3 を異常素子と仮定するとさらにコンフリクト集合 {N4, N2, N1, A4, A2} が得られた。最終的に素子 A2 を根拠にパッケージ単一故障 P3 と素子二重故障 {N4, N3} を根拠にパッケージ単一故障 P2 が得られた。

図5にこの診断結果を HS-木で表したものを示し、表1にパッケージ診断を行なわない場合との比較を示す。

表1からわかるように、パッケージ・レベル診断は素子レベル診断と比較して、定理証明器 Thinker の呼出回数は2回少なく、診断時間は 27 秒短縮され、効率が向上していることがわかる。

Macintosh-II (MC68020,15MHz) Thinker (Common Lisp)

| Package-level diagnosis | TP Calls | Time(SEC.) |
|---|---|---|
| {$P_3$: A2}, {$P_2$: N4, N3} | 5 | 51.4 |
| Component-level diagnosis | TP Calls | Time(SEC.) |
| {A2}, {N4, N3}, {A4, N3} | 7 | 78.4 |

Table 1 Package-level diagnosis for the 2-inputs decoder

### <3.6> 自動販売機のパッケージ診断

次に、やや大規模な論理回路の診断実験として、自動販売機のパッケージ診断の例を述べる。この機械は入力として 10円、 50円および 100円の 3 種類の硬貨を受付け、出力として定価 80円の品物(例えば缶ジュース)と釣銭をだす。

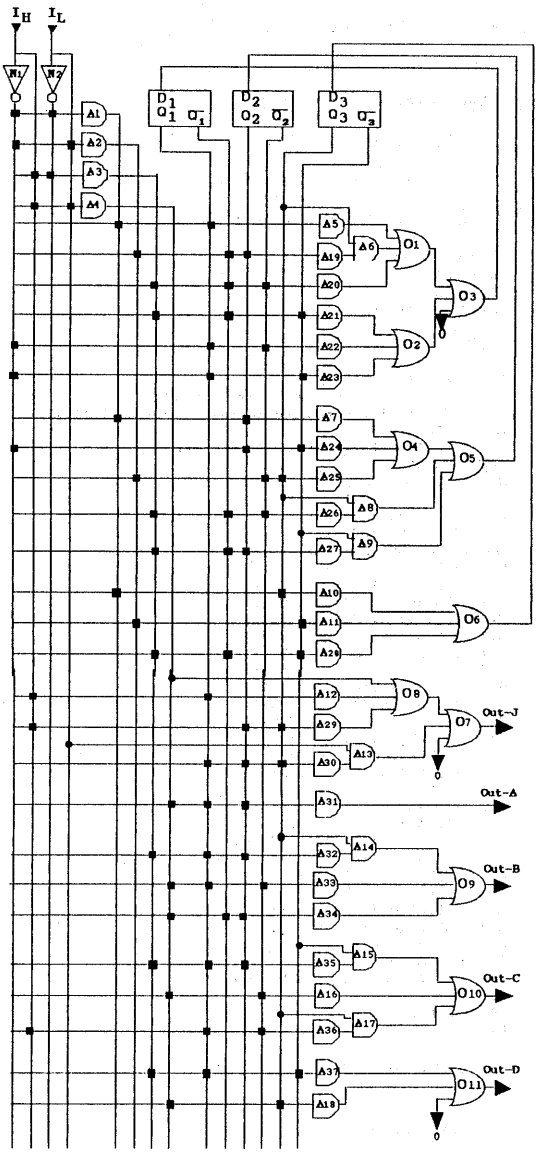


Fig.6 A sequential circuit for diagnosis - automatic vending machine.

図6で IH, IL は硬貨検出器からのコード入力端子を表し、IH, IL = 00, 01, 10, 11 はそれぞれ、入力無し、10 円、50 円及び 100 円の入力を表す。また、Out-J はジュース出力信号、Out-A,...,Out-D は釣銭とすべき 10 円硬貨の枚数の 2進コード出力端子である。ここで、回路が異常な動作をしている状況を、「 50 円硬貨を 2度

| Functions | Packages | Assignments |
|---|---|---|
| HEX INVERTERS | $P_1$ | $N_1, N_2$ |
| QUADRUPLE | $P_2$ | $A_1, A_2, A_3, A_4$ |
| 2-INPUT AND | $P_3$ | $A_5, A_6, A_{10}, A_{11}$ |
| GATES | $P_4$ | $A_7, A_8, A_9$ |
| | $P_5$ | $A_{12}, A_{13}, A_{14}$ |
| | $P_6$ | $A_{15}, A_{16}, A_{17} A_{18}$ |
| TRIPLE 3-INPUT | $P_7$ | $A_{21}, A_{22}, A_{23}$ |
| AND GATES | $P_8$ | $A_{29}, A_{30}, A_{31}$ |
| | $P_9$ | $A_{32}, A_{33}, A_{34}$ |
| | $P_{10}$ | $A_{35}, A_{36}, A_{37}$ |
| | $P_{11}$ | $A_{19}, A_{20}$ |
| | $P_{12}$ | $A_{24}, A_{25}$ |
| | $P_{13}$ | $A_{26}, A_{27}, A_{28}$ |
| TRIPLE 3-INPUT | $P_{14}$ | $O_1, O_2, O_3$ |
| OR GATES | $P_{15}$ | $O_4, O_5, O_6$ |
| | $P_{16}$ | $O_7, O_8$ |
| | $P_{17}$ | $O_9, O_{10}, O_{11}$ |
| 4-BIT D-LATCHES | $P_{18}$ | $D_1, D_2, D_3$ |

**Table 2. Components assignment to packages.**

投入したとき、ジュースは得られたが、釣銭が 10円のみであった」とした。この状況を観測とし、 14個の等式で記述した。次に、回路素子が、それぞれどのパッケージに属するかを表2 のように決めた。なお、診断のために最終的に用意した節の総数は、98 個となった。

| SingleFault | Rationale | Double-Fault | Rational |
|---|---|---|---|
| {$P_{13}$} | {$A_{28}$} | {$P_1, P_3$} | {$N_2, A_6$} |
| {$P_{15}$} | {$O_6$} | {$P_1, P_7$} | {$N_2, A_{23}$} |
| | TP Calls 37 | {$P_1, P_{11}$} | {$N_2, A_{20}$} |
| | Time(min.)24.8 | {$P_1, P_{14}$} | {$N_2, O_3$} |
| {$P_6$} | {$A_{17}, A_{18}$} | {$P_2, P_3$} | {$A_3, A_6$} |
| {$P_{10}$} | {$A_{36}, A_{37}$} | {$P_2, P_4$} | {$A_4, A_9$} |
| {$P_{17}$} | {$O_{10}, O_{11}$} | {$P_2, P_7$} | {$A_3, A_{23}$} |
| {$P_{18}$} | {$D_{3\text{-}q}, D_{3\text{-}nq}$} | {$P_2, P_{11}$} | {$A_3, A_{20}$} |
| | | {$P_2, P_{12}$} | {$A_4, A_{25}$} |
| | | {$P_2, P_{14}$} | {$A_3, O_3$} |
| Apollo DN3500 (MC68030 25-MHz 8Mb) | | | TP Calls 95 Time(min.) 94.7 |

**Table 3. Package diagnosis for the automatic vending machine.**

デコーダの例と同様 Thinker で実験を行なった。初めにサイズ 40 のコンフリクト集合が得られた。この集合をもとにパッケージ単一故障と二重故障を求めた結果を表3 に示す。根拠のサイズが 1 のパッケージ単一故障が 2個 (診断時間、約25分)求まった。Thinker の呼出回数は 37回であった。さらに、呼出回数 95回(診断時間、約 1 時間半)で根拠のサイズが 2のパッケージ単一故障が 4個と二重故障が 10個得られた。

次に、表4 に素子レベルの根拠を求めない直接パッケージ診断の結果を示す。診断としては表3 と同一となった。しかし、Thinker の呼出回数が大幅に減少しているにもかかわらず診断時間は回数に比例して減少していない。これは、各パッケージ内の素子毎にまとめて異常を仮定することは、回路的にはそれらを同時に開放除去したことに相当し、入出力関係からきまる推論の制約条件がゆるくなることを意味する。このため推論時の場合分けが多くなり Thinker の矛盾検出時間が増大したことが原因である。

また、従来の Reiter の理論による素子レベル診断では、二重故障の診断結果を得るためには、Thinker の呼出回数は最悪の場合約 400回以上と見積もれるが、途中まで実験を行なって経過をみたところ大幅なカットも現れなかった。仮にカットにより Thinker の呼出しが半分になったとしても時間的に現実的な診断となり得ない。著者らが提案したパッケージ・レベル診断では、パッケージの二重故障まで約 2時間の診断時間で求めることができた。比較的大規模な論理回路であっても、本方式の診断システムが有効であることがわかる。

| パッケージ単一故障 | TP Calls | Time(min.) |
|---|---|---|
| {$P_6$}, {$P_{10}$}, {$P_{13}$}, {$P_{15}$}, {$P_{17}$}, {$P_{18}$} | 14 | 4.3 |
| パッケージ二重故障 | TP Calls | Time(min.) |
| {$P_1, P_3$}, {$P_1, P_7$}, {$P_1, P_{11}$}, {$P_1, P_{14}$}, {$P_2, P_9$ } {$P_2, P_4$}, {$P_2, P_7$}, {$P_2, P_{11}$}, {$P_2, P_{12}$}, {$P_2, P_{14}$} | 27 | 78.5 |

**Table 4. Direct package diagnosis without the basis.**

## 4. むすび

著者らはすでに、 Reiter の故障理論に基づいた小規模な論理回路の故障診断システムの実現について発表している[3]。本研究では、パッケージの概念を導入して Reiter の理論を拡張し、探索空間の縮小と効果的な枝の刈込みによる大規模論理回路診断へのアプローチを提案した。

Reiter の理論は、原理的に任意のシステムの故障診断に適用できる非常に一般的な理論であり、将来の高度な故障診断システムの統一的な構成に明確な見通しを与えている。システムの異常検知という観点からは、システムのモデル (知識表現) と観測に基づいて自動的に異常を検知し、その異常であるという理由 (証明) に基

づいて、論理的かつ効率的に異常原因を推論している。本稿の成果は、Reiter の理論の実際的な応用可能性を拡大した新しい成果であり、論理回路の分野に限らず、故障診断一般について今後の研究に有効に寄与するものと考える。

## 参考文献


[1] R.Reiter: A theory of diagnosis from first principles, Aritifitial Intelligence,32,(1987)

[2] 栗原、大内、加地: Automated Reasoning に基づくシステム問題へのアプローチ, 電学論 C, 107, 141, 昭(62-2)

[3] 中川, 北谷, 栗原, 加地: 論理回路の故障診断における自動推論システムの実験とヒューリスティクス, 電学論 C, 109, 731, 平(1-10)

[4] 安居院、内藤:論理回路の故障診断, 電子科学シリーズ68, 1978

[5] R.Davis. Diagnostic Reasoning Based on Structure and Behavior, Aritifitial Intelligence,24,(1984)

[6] L.Wos et.al.: Automated reasoning-introduction and applications, (1984) Prentice-Hall: 川越、他訳: コンピュータによる推論技法(昭64) マグロウヒル