6D−5　ソフトウェア故障診断ＤＳにおける制御表破壊発見方式

田添　清　島田 茂夫　飯田 敏幸
ＮＴＴ情報通信処理研究所

1．はじめに

ソフトウェアの障害解析はメモリダンプ（ＭＤ）を利用してソフトウェアがアクセスする多種／多量の制御表の内容を追跡・検証することにより行われる。アドレス計算の誤り等で制御表が破壊されていた場合には破壊の発見に多くの時間を要する。破壊は障害の中でまれであるが、この発見を支援する効果は大きい。この様な考えに基づき制御表の破壊を発見する方法を提案し、評価したので報告する。

2．破壊発見方式の位置付け

ソフトウェアの障害箇所切りわけを行うエキスパートシステム(ＥＳ)の構築を容易にするドメインシェル(ＤＳ)としてソフトウェア故障診断ドメインシェル(ＦＩＮＤＳ)[1,2]を開発した。ＦＩＮＤＳにソフトウェアトラブルの診断知識をフローチャート形式で入力してＥＳを構築する。この診断知識は制御表が破壊されていないことを前提としているため、制御表の破壊に対しては、ポインタによる連鎖関係の破壊等を考慮した知識を整理し投入する等の対策が必要となってくる。しかし、制御表の構造的制約（制御表破壊発見の知識）が診断知識とは独立であるので、図１に示すように制御表破壊発見の知識は診断知識とは切り離してＤＳ機能の一部とすることができると考えられる。

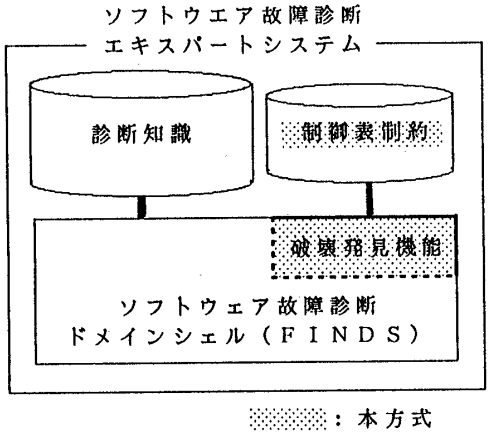


図１　破壊発見方式のドメインシェルにおける位置付け

3．制御表破壊の発見方式

(1)方式の概要

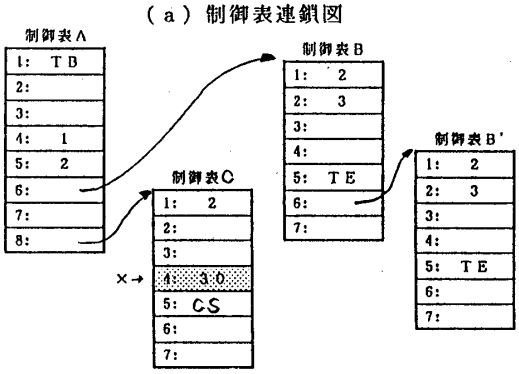
本方式は制御表の各フィールドが制約（値が限定されていること。次節で説明）を持つという性質を利用し、制約とフィールドの値を照合して制御表が破壊されているかどうかを発見するものである。この手順の例を図２を用いて説明する。

各制御表はポインタにより連鎖されており、この連鎖を順に辿りながら各制御表のフィールドを検証する。具体的には以下の手順で検証が進められる。

①制御表Ａのフィールド１／４／５の検証
②制御表Ｂのフィールド１／２／５の検証
③制御表Ｂ'のフィールド１／２／５の検証
④制御表Ｃのフィールド１／４／５の検証

制御表Ａ、Ｂ、Ｂ'の検証が済み制御表Ｃの検証に移った時、制御表Ｃのフィールド４の値が”１０－２０”でないので、「制御表Ｃのフィールド４が破壊されている」あるいは「制御表Ｃに至る経路に破壊があった」ことがわかる。この情報は障害解析の専門家に示され、後の詳細解析のデータとなる。

（ａ）制御表連鎖図



（ｂ）制御表の制約

制御表Ａ	制御表Ｂ、Ｂ'	制御表Ｃ
1: CHAR TB	1: 定数　2	1: 定数　2
4: 定数　1	2: 範囲　0-5	4: 範囲10-20
5: 定数　2	5: CHAR TE	5: CHAR CS
6: ポインタ	6: ポインタ	
8: ポインタ		

図２　本方式による破壊発見例

(2)制約の種類

制約は大きく分類すると以下の二つになる。

①制御表自身で閉じた制約条件（一次制約）
・定数属性：フィールド値は定数である
・離散値属性：フィールド値は離散値をもつ
・範囲属性：フィールド値はある範囲の数である
・ＣＨＡＲ属性：フィールド値は文字属性をもつ
・ｎバイト境界属性：制御表の先頭アドレスがｎバイト境界をもつ
・相関属性：フィールド値が他のフィールド値と相関関係をもつ

An Erroneous Table Detection in Software-trouble Diagnosis
Kiyoshi TAZOE, Shigeo SHIMADA, Toshiyuki IIDA
NTT Communications and Information Processing Laboratories

②制御表連鎖による制約条件（二次制約）
・ポインタ属性：フィールドの値は決まった種類の制御表を指す
・連鎖形式属性：制御表の連鎖形式が特殊な形式を持つ（ポイントされた制御表にポイント元制御表を指すポインタがある）

一次制約に関してはフィールドの値との照合のみでそのフィールドの値が破壊されたかどうかの検証が可能である。

二次制約についてはポイント先の制御表の制約を照合することで破壊されているかどうかの検証が可能である。

一次制約と二次制約を組み合わせて照合箇所を増加させ、破壊検出の可能性を高めることができる。

(3)インプリメンテーション

構成を図３に示す。制約取得部により制御表定義情報から制約を、制御表取得部によりＭＤから制御表の値を得る。照合部はこれらを照合し、フィールドが制約を満たすかどうかを検証し結果を表示する。

また、照合部は制御表連鎖に従って、制約取得部と制御表取得部を制御する。

検証の終了条件は以下のとおりである。
・検証済み制御表をポインタが示す時
・検証中の制御表をポインタが示す時
（検証中の制御表は他の制約で検証されるため）
・フィールドが制御表連鎖の終了を示す時

また、実行時間を削減するため以下の２点を考慮した。

①　探索範囲の限定

制御表の連鎖に従って制御表を検証すると全ての制御表について調べることになり実用的でない。そこでＥＳの診断知識が参照する制御表についてのみ検証することとした。ＥＳの診断知識はシステムのエラーコードを基に起動されるので、最も疑わしいのは診断知識が参照する制御表と考えられるからである。

②　一次制約照合の優先

一次制約の検証は自制御表内だけで検証可能である。一方、二次制約の検証は通信回線を通してＭＤから制御表の値を受信する必要がある。そこで一次制約を二次制約よりも優先して検証を行うことで検出時間の短縮を図った。

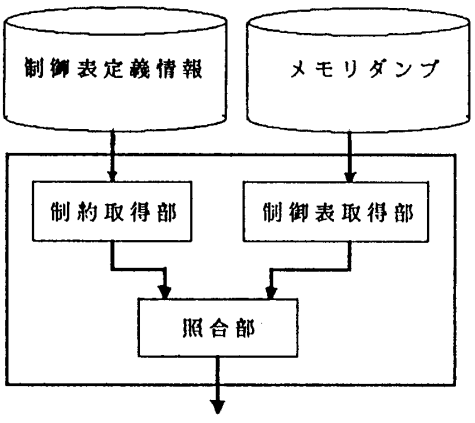


図３　破壊発見方式の構成

４．評価及び考察

制御表を実際のＭＤ上にランダムに1357個設定し、それぞれについて本方式を適用して検出能力を調べた。その結果、次のようなことがわかった。

①CHAR属性かつ定数属性の制約を持つフィールドの破壊検出率はほぼ100％である。

"ＴＢ"という制約はCHAR属性と定数属性の二つの制約を同時にもつため、ほぼ100％の破壊検出率が得られると考えられる。

②離散値制約として"０"を含まないフィールドの破壊検出率が９０％程度と高い。

離散値属性の"０"を含んだフィールドは破壊検出率が10％程度と低い。これは初期設定時に０クリアしているため、制約の"０"と一致し、誤りにも関わらず「誤りとは言えない」と判定したためである。

③制約条件を重ねることで破壊検出率（総合破壊検出率）は向上する。

従って、部分的に破壊検出率が低くとも制御表の連鎖に従って検証を進めていくことにより、破壊が発見できる可能性は高くなる。

また、検出位置によりある程度、破壊位置の範囲も特定できる。

表１：試験結果

制　約	破壊検出率(％)
TB	100
0 4 8 12 16 24	10.4
16 18 20 25 32 40	94.1
0 2 3	9.4
0 1 4 5	9.6
nバイト境界(n=4)	43.6
総合破壊検出率	100

５．終りに

ソフトウエアの制御表破壊発見方式を提案した。制約によっては100％近い破壊検出率が得られた。破壊検出率が低い場合でも、制御表の連鎖を辿り、制約検証を重ねることで破壊が発見できる可能性が高くなり、障害解析者への大きな支援となる。本方式は制御表構造に固有な制約に基づいているので、ソフトウェア故障診断ドメインシェルの機能として取り込むことができる。今後、検出率を向上させるための制約の調査、大量の制約条件を効率良く入力する方式、また破壊箇所の特定法について検討する。

参考文献

[1] Iida and Tachibana, "Expert System for Analyzing Software Trouble in Remote Sites," PPCC-3, 1989.
[2] 飯田："ｿﾌﾄｳｴｱ故障診断ﾄﾞﾒｲﾝｼｪﾙ"，信学会秋期全国大会、1989.