SAT ソルバを用いたペアワイズテストケースの生成

Constructing Pairwise Test Cases Using a SAT Solver

難波 亨 † 土屋 達弘 † 菊野 亨 †
Tohru Nanba Tatsuhiro Tsuchiya Tohru Kikuno

1. まえがき

ソフトウェアの不具合，すなわち，フォールトを検出するためのテストは，ソフトウェア開発において必要不可欠であるが，多大な時間とコストを要する．従って，フォールトを検出する能力に優れたテストケースの作成は，非常に重要な課題である．

このような課題を解決する有効な手段の一つとして，ペアワイズテストと呼ばれるソフトウェアテスト手法がある．ペアワイズテストとはソフトウェアの入力パラメータの相互作用に着目し，フォールトを効率よく検出する手法である．具体的には，あらゆる 2 個の入力パラメータについて，とり得る値の組み合わせ全てを網羅するようにテストを行う．この手法の背景にある基本的な根拠は，フォールトの多くは，少数の特定のパラメータ値の組み合わせ (特に 2 個の入力パラメータが多い) によって検出されるということである[1),3)]．

ペアワイズテストの拡張として 2 パラメータではなく $k(=2,3,\dots)$ 個のパラメータについて，パラメータ値の組み合わせを網羅する，$k-way$ テストと呼ばれる手法も存在する．本研究では $k=2$，すなわち，ペアワイズテストのみに議論を限定する．

ここで，ペアワイズテストによって，フォールトを効率よく検出できるとしても，少数のテストケースで実行できなければ，この手法の有効性はなくなってしまう．そのために，いかに少数のテストケースによって，ペアワイズテストを構成するかについて，活発に研究が行われている[5)]．

ペアワイズテストの生成で問題となるのが，実行不可能なパラメータ値の組み合わせ，すなわち，禁則の扱いである．実際のソフトウェアでは，入力の際，同時に選択できないパラメータ値が多数存在する．本研究では，そのような実行不可能なパラメータ値の組み合わせを除外したペアワイズテスト集合を，できるだけ少ないテストケース数で構成する手法を提案する．具体的には，ペアワイズテスト生成を SAT 問題へと変換し，SAT ソルバを用いてテストケース集合を生成する．

ブール式によって禁則を表現することで，SAT を用いて禁則 (制約条件) を考慮したテスト生成が可能となる．これまでにも，禁則が存在する場合に SAT を利用してペアワイズテストを生成する手法が提案されている[4)]．しかし，この従来法では SAT の処理をテスト生成アルゴリズム内部に組み込んでいるため，個別に開発された SAT ソルバを利用できなかった．また，テストケースをグリーディ法に従って一つずつ生成するため，最終的に得られるテスト集合が比較的大きくなるという問題があった．

提案手法では，SAT ソルバを完全にブラックボックスとして用いるため，禁則処理が可能なだけでなく，任意の SAT ソルバを用いる事ができる．また，一度に全体のテスト集合を生成する事で，コンパクトなテストケース集合を得る事が可能となる．

本論文の章構成は以下の通りである．2 章でペアワイズテストについて述べ，3 章で SAT ソルバについて述べる．4 章では，提案手法の説明を行う．5 章では，提案手法を実装したツールの動作結果・考察について述べる．最後に 6 章で研究の結論と今後の展望について述べる．

2. ペアワイズテスト

ペアワイズテストを議論する際に通常想定される以下のようなモデルを仮定する．システムはいくつかの入力からなり，各入力パラメータが定められている．テストケースは，全ての入力パラメータについて，とり得る値を一つ定める事によって得られるベクトルである．このとき，ペアワイズテストとは，どの二つの入力パラメータのとり得るどの値の組み合わせも，少なくとも一つのテストケースに表れるようなテストケース集合と定義できる．

ペアワイズテストについて，表 1 のモデルを例に説明する．“:” の左にある Browser や RAM はパラメータで，“:” の右にある IE や 1G はそれぞれのパラメータがとる値である．この例では，3 値をとるパラメータが 3 個，2 値をとるパラメータが 1 個あり，これを $2^1 3^3$ と表すものとする．テストケースは各パラメータの値の組み合わせなので，総当たりでテストを行う場合，テストケースの総数は $2^1 \times 3^3 = 54$ である．すべての可能なパラメータ値の組み合わせをテストするためには，これだけのテストケースが必要となる．

一方ペアワイズテスト集合は，表 2 にあるような 9 通りのテストケースによって構成する事ができる．これは，テストケース総数のわずか 1/6 である．実際にどの二つのパラメータを選んでも，それらのとり得る値の全ての組み合わせが，9 個のテストケースの中に含まれている事が確認できる．

この例では，ペアワイズカバレッジを満たすには最低でも，$3 \times 3 = 9$ 通りのテストが必要なのは明らかである．従って，図 2 のテスト集合はテストケースが最小のペアワイズテスト集合であると言える．

2.1 パラメータ値の制約

ここで，パラメータ値間での制約について触れる．実際のソフトウェアでは，同時に選択する事ができないパラメータ値の組み合わせが存在する．そのような，実際には実行不可能なパラメータ値の組み合わせがある場合，それらを含まないペアワイズテスト集合を構成する必要がある．

例として，図 1 のモデルで以下の二つの制約があったものとする．

- Browser が Safari だった場合，RAM は 1.5G 以上必要

† 大阪大学 Osaka University

表 1 パラメータと値

Browser:	IE, EireFox, Safari
RAM:	1G, 1.5G, 2G
GRAM:	128M, 256M, 512M
HD:	50G, 100G

表 2 ペアワイズテスト集合

	Browser	RAM	GRAM	HD
1	IE	2G	256M	100G
2	Safari	1G	256M	100G
3	Safari	2G	512M	50G
4	EireFox	1.5G	256M	50G
5	Safari	1.5G	128M	100G
6	IE	1G	128M	50G
7	EireFox	1G	512M	50G
8	EireFox	2G	128M	100G
9	IE	1.5G	512M	100G

- GRAM が 512M だった場合，HD は 100G 以上必要

この二つの制約があった場合，以下の二つのパラメータ値の組み合わせはテストケースに含んではいけない．

- Browser = Safari, RAM = 1G
- GRAM = 512M, HD = 50G

表 3 は二つの組み合わせを除いたペアワイズカバレッジを満たしている．

パラメータ値間での制約が存在すると，カバーする組み合わせが少なくなるので，一見テストケース数が少なくなるように思われるが，ほとんどの場合，図 2，3 を比べるとわかるように，逆にテストケース数が大きくなる．これはテストケース生成の際に，パラメータ値選択の自由度が下がるためである．

このようなパラメータ値間での制約が存在する場合，ペアワイズカバレッジを満たすテストケース集合を生成することは難しい．

問題となるのは制約条件が複数あった場合，それらの組み合わせによって，隠れた制約条件が現れるという事である．例として，上の二つの組み合わせに加え，新たに「RAM = 2G, HD = 100」という制約を加えた場合を考える．つまり，

- Browser = Safari, RAM = 1G
- GRAM = 512M, HD = 50G
- RAM = 2G, HD = 100G

の三つの制約があったと考える．このとき，上の三つの組み合わせに加えて,「GRAM = 512M, RAM = 100G」も禁則になっている．つまり二つの値をとる HD に対して，その値が 50G，100G それぞれに対して制約が存在するため,「GRAM = 512M, RAM = 100G」の組み合わせがテストケースに出現不可能になっている．ペアワイズテスト生成において，与えら制約条件以外の値の組み合わせを全て出現させるようにテストを生成すると，隠れた条件がある場合，出現不可能な組み合わせも網羅してテスト生成を行おうとし，不具合が生じる．上の例では，3 つの制約条件でテストケース生成を行う際に，予め「GRAM = 512M, RAM = 100G」という 4 つ目の隠れた条件も発見しておく必要がある．このように制約条件を指定する場合には，隠れた制約も考慮する必要がある．

表 3 パラメータ値の制約を考慮したペアワイズテスト集合

	Browser	RAM	GRAM	HD
1	Safari	2G	512M	100G
2	IE	2G	512M	50G
3	IE	2G	256M	100G
4	Safari	1.5G	256M	50G
5	EireFox	2G	128M	100G
6	EireFox	1.5G	128M	50G
7	IE	1.5G	128M	100G
8	EireFox	1G	512M	100G
9	EireFox	1G	256M	50G
10	Safari	1G	128M	50G

3. SAT(充足可能性問題)

SAT(充足可能性問題，satisfiablility problem) とは，ある CNF(乗法標準形) の形式のブール式が与えられたときにそのブール式が真になるような論理変数への値割当が存在するかどうかを判定する問題である．CNF とは変数もしくは変数の論理和からなる式を積だけでつないだ論理式の事である．論理変数またはその否定をリテラルといい，リテラルの論理和を節という．すなわち，CNF とは節の論理積である．具体的な論理式で例を挙げると，

$$x1 \wedge (\neg x1 \vee x2) \wedge (\neg x2 \vee x3)$$

この場合 0 を偽，1 を真としたとき $x1 = 1$，$x2 = 1$，$x3 = 1$ と代入すると上の論理式は真となる．つまり，その論理式を真にするような変数への値の割当が存在するので，充足可能であるといえる．次に下の式を考える．

$$x1 \wedge (\neg x1 \vee x2) \wedge (\neg x2 \vee x3) \wedge \neg x3$$

この場合，$x1$，$x2$，$x3$ にどのような真偽値を割り当てても，論理式全体を真にする事ができない．つまり，充足不可能である．

SAT は NP 完全問題であり，知られているアルゴリズムの時間計算量は，指数オーダーである．しかし，アルゴリズムの発展により，現在では，多くの実用的な問題を高速で解く事が可能となっている．

3.1 SAT ソルバ

SAT 問題を解くツールを SAT ソルバという．SAT ソルバは，毎年国際的なコンペティションが開催されており，日々開発が進んでいる．最新の SAT ソルバは数千万リテラルからなる問題を解く事ができる．

論理式の充足可能性判定 (SAT) は NP 完全問題であるが，SAT ソルバはしばしばこの問題を現実的な時間で決定する事ができる．全ての NP 問題は SAT に還元する事ができるので，SAT ソルバはこのクラスの汎用ツールであると言える．

SAT ソルバは大きく分けて二つの種類に分類する事ができる．一方を完全探索型 (complete search)，もう一方を確率的な局所探索型 (stochastic local search) という．前者は完全 SAT ソルバと呼ばれ，探索域を削りなが虱潰しに解を探索していき，最終的に充足可能か充足不可能かを判断する．後者は確立的 SATSAT ソルバと呼ばれ，確率的な探索をする事で，充足不可能かどうかを判定しない代わりに，素早く充足可能性を見

つける事ができる．具体的な例を以下に挙げる．

- 完全 SAT ソルバ
  –**Minisat**
  –**picoSAT**
- **確率的 SAT ソルバ**
  –**Walksat**

上に述べたように SAT ソルバの能力の向上を受けて，外部 SAT ソルバを効率よく用いることで，決定問題や最適化問題を高速に解く事ができる．

なお，本手法における外部 SAT ソルバには完全 SAT ソルバである Minisat を使用している．Minisat はオープンソースの SAT ソルバで，SAT コンペティションでも優秀な成績を修めている．

4. 提 案 手 法

本章では，SAT ソルバを用いたテストケース生成手法を説明する．提案手法では，ペアワイズテスト集合の構成を SAT 問題として定式化し，SAT ソルバを用いて解を求めるという手法である．

本手法では，[k 個のテストでペアワイズテストが生成できる] かどうかを SAT 問題として定式化し，その最小の k を二分探索で求める．

本手法のおおまかなアルゴリズムは図 1 のようになる．以下に SAT 問題への定式化の手法と，その際に問題となる，隠された制約条件の発見の手法について具体的な説明を行う．

4.1 SAT 問題への定式化

ペアワイズテストの構成問題は，制約充足問題として定式化する事ができる．ここで大きく二つの条件にわけて考える．以下の二つの条件を制約条件として記述する事で，ペアワイズテスト集合を制約充足問題として定式化する事ができる．

- テストケース制約
  テストケースのパラメータにはただ一つの値が選ばれる．
- カバレッジ制約
  テストケース集合においてどの二つのパラメータについても全ての値の組み合わせが出現する．

テストケース数の暫定の最大値 MAX，最小値 MIN，中央値 $k = (MAX + MIN)/2$ を決定．
repeat
- テストケース数 k として CNF 式を生成
- SAT ソルバを用いて充足可能性判定を実施
- if(結果が SAT)
  - $MAX := k$
  - $k := \lfloor (k + MIN)/2 \rfloor$
- else * つまり結果が UNSAT あるいは TIMEOUT
  - $MIN := k$
  - $k := \lfloor (k + MAX)/2 \rfloor$

until 前回に生成したテストケース数と今回生成したテストケース数の差が 1

図 1 テスト集合生成アルゴリズムの概要

ここで，本手法では禁則を考慮したテストケース集合を生成するので厳密には以下のようになる．

- テストケース制約
  テストケースのパラメータにはただ一つの値が選ばれる．
- カバレッジ制約
  テストケース集合において，禁則を除く どの二つのパラメータについても全ての値の組み合わせが出現する．
  いずれのテストケースにも禁則に含まれる値の組み合わせが出現しない．

上で定めた二つの制約から SAT 符号化を行い，外部 SAT ソルバへの入力となる CNF を生成する．

まず，全てのテストケースのパラメータの値に対して，論理変数 $p_{(i,j,k)}$ を用意する．$p_{(i,j,k)}$ が真であるという事は，i 番目のテストケースの j 番目のパラメータの値が k 個目のものである事を示す．ここで，i 番目のパラメータの総数を N_i と表すことにすると，$(k \times \sum_{i=1}^{n} |N_i|)$ 個の変数を用意する事になる．ただし，n をパラメータの総数とする．

あとは，以下のように節を生成すればよい．

- テストケース制約の変換

$$\forall i, j : \bigvee_k p_{(i,j,k)} \tag{1}$$

$$\forall i, j, k, k' (k < k') : \neg p_{(i,j,k)} \vee \neg p_{(i,j,k')} \tag{2}$$

- カバレッジ制約の変換

$$\forall i, i' k, k' (j < j') : \bigvee_i (p_{(i,j,k)} \wedge p_{(i,j',k')}) \tag{3}$$

(1) では，テストケースのパラメータが少なくとも一つ値をとる事を表している．(2) では，テストのパラメータ値が同時に二つの値をとる事がない事を表している．(1),(2) の二つの条件によってあるテストケースのパラメータが常に一つの値を有することを表せる．(3) はいずれかのテストケースにおいて，全てのパラメータの値の組み合わせが存在している事を表している．これはすなわち，ペアワイズテストの条件そのものである．

次に禁則条件を考慮した場合について考える．この CNF 式生成段階において，入力で与えられた禁則条件以外の，隠された禁則条件も考慮する必要がある．ここでは既に分かっているものとして，説明を行う．隠された禁則条件の具体的な発見方法については 4.2 節で説明を行う．禁則条件となる値の組み合わせの集合を S_{const} とする．

- テストケース制約の変換

$$\forall i, : \bigvee_k p_{(i,j,k)} \tag{4}$$

$$\forall i, j, k, k' (k < k') : \neg p_{(i,j,k)} \vee \neg p_{(i,j,k')} \tag{5}$$

- カバレッジ制約の変換

$$\forall j, j' k, k' (j < j') : \bigvee_i (p_{(i,j,k)} \wedge p_{(i,j',k')}) \tag{6}$$

$$\forall j, j', k, k' (j < j') : \bigwedge_i (\neg p_{(i,j,k)} \vee \neg p_{(i,j',k')}) \tag{7}$$

ここで (3) では禁則となるパラメータの組み合わせ S_{const} を除いて，(7) では禁則となるパラメータの組み合わせ S_{const} についてのみ生成する．

ここで (3) や (6) に関しては CNF でないので，そのままの

状態では SAT ソルバへの入力として用いる事ができない．任意の論理式は，二重否定の除去，ド・モルガンの法則，分配法則といった論理的に等価な変換を行うことで，CNF に変換できる．しかし，(3) や (6) をそのような変換だけで CNF に変換すると，節が指数的に増加してしまい，好ましくない．よって新たな論理変数を用意し，節が爆発的に増えず論理的に等価な変換を行った．

論理式 $(A \wedge B) \vee (C \wedge D) \vee (E \wedge F)$ の CNF への変換という具体的な例で示す．

ド・モルガンの法則で等価な変換を行った場合は以下のようになる．

(1) $\underline{(A \wedge B) \vee (C \wedge D)} \vee (E \wedge F) \Leftrightarrow$

(2) $\underline{(A \vee (C \wedge D)) \wedge (B \vee (C \wedge D))} \vee (E \wedge F) \Leftrightarrow$

(3) $\underline{((A \vee C) \wedge (A \vee D) \wedge (B \vee C) \wedge (B \vee D))} \vee (E \wedge F) \Leftrightarrow$

(4) $\underline{((A \vee C) \vee (E \wedge F)) \wedge ((A \vee D) \vee (E \wedge F))}$
$\underline{\wedge((B \vee C) \vee (E \wedge F)) \wedge ((B \vee D) \vee (E \wedge F))} \Leftrightarrow$

(5) $\underline{(A \vee C \vee E) \wedge (A \vee C \vee F) \wedge}$
$\underline{(A \vee D \vee E) \wedge (A \vee D \vee F) \wedge}$
$\underline{(B \vee C \vee E) \wedge (B \vee C \vee F) \wedge}$
$\underline{(B \vee D \vee E) \wedge (B \vee D \vee F)}$

本手法で用いた，新たな論理変数を追加する変換を以下に示す．

(1) 新たな変数として，α，β，γ を用意する．

(2) $(A \wedge B) \vee (C \wedge D) \vee (E \wedge F) \Leftrightarrow$

(3) $(\alpha \rightarrow (A \wedge B)) \wedge$
$(\beta \rightarrow (C \wedge D)) \wedge$
$(\gamma \rightarrow (E \wedge F)) \wedge$
$(\alpha \vee \beta \vee \gamma) \Leftrightarrow$

(4) $(\neg\alpha \vee (A \wedge B)) \wedge$
$(\neg\beta \vee (C \wedge D)) \wedge$
$(\neg\gamma \vee (E \wedge F)) \wedge$
$(\alpha \vee \beta \vee \gamma) \Leftrightarrow$

(5) $(\neg\alpha \vee A) \wedge (\neg\,\alpha \vee B) \wedge$
$(\neg\beta \vee C) \wedge (\neg\,\beta \vee D) \wedge$
$(\neg\gamma \vee E) \wedge (\neg\,\gamma \vee F) \wedge$
$(\alpha \vee \beta \vee \gamma)$

二つの変換の手法からわかるように，テストケース数が n 個であったとすると，ド・モルガンの法則で変換を行った場合，2^n 個の節が生成される．対して，論理変数を追加して変換を行った場合は $((2 \times n) + 1)$ 個の節の生成で済む．これにより効率の良い CNF を生成する事ができる．

4.1.1 CNF 式生成の例

上に示す手法を用いて，具体的な入力に対する CNF 式生成の例を示す．入力には表 3 を用い，テストケース数を 2 として生成を行う．上で示したテストケース制約の変換 (4)(5)，カバレッジ制約の変換 (6)(7)，に沿って生成の例を示す．行ごとに一つの節を表す．

簡単の為にここでは入力例である表 3 の隠された禁則条件については考慮しない．

- (4) に該当する節．上の 5 行によって，二つ目のテストケースの全てのパラメータが最低限一つ値を持つ事を表している．

$$p(1,1,1) \vee p(1,1,2) \vee p(1,1,3)$$
$$p(1,2,1) \vee p(1,2,2) \vee p(1,2,3)$$
$$p(1,3,1) \vee p(1,3,2) \vee p(1,3,3)$$
$$p(1,4,1) \vee p(1,4,2)$$
$$p(1,5,1) \vee p(1,5,2)$$
$$p(2,1,1) \vee p(2,1,2) \vee p(2,1,3)$$
$$p(2,2,1) \vee p(2,2,2) \vee p(2,2,3)$$
$$\vdots$$

- (5) に該当する節．上の 3 行によって，一つ目のテストケースのパラメータ [a] が同時に二つ値をとる事はなことを表している．

$$\neg p(1,1,1) \vee \neg p(1,1,2)$$
$$\neg p(1,1,1) \vee \neg p(1,1,3)$$
$$\neg p(1,1,2) \vee \neg p(1,1,3)$$
$$\neg p(1,2,1) \vee \neg p(1,2,2)$$
$$\neg p(1,2,1) \vee \neg p(1,1,3)$$
$$\vdots$$

- (6) に該当する節．上の 1 行によって，一つ目あるいは二つ目のテストケースにおいてパラメータ a の値 0 とパラメータ b の値 0 の組み合わせが出現する事を表している．ただし，この段落では節となっていないので，上で説明した変数を追加する方法で変換を行う．

$$(p(1,1,1) \vee p(1,2,1)) \wedge (p(2,1,1) \vee p(1,2,1))$$
$$(p(1,1,1) \vee p(1,2,2)) \wedge (p(2,1,1) \vee p(1,2,2))$$
$$(p(1,1,1) \vee p(1,2,3)) \wedge (p(2,1,1) \vee p(1,2,3))$$
$$(p(1,1,1) \vee p(1,3,1)) \wedge (p(2,1,1) \vee p(1,3,1))$$
$$\vdots$$

- (7) に該当する節．上の 2 行によって，いずれのテストケースにおいてもパラメータ a の値 0 とパラメータ c の値 0 の組み合わせが出現しない事を表し，同様に，下の 2 行によって，いずれのテストケースにおいてもパラメータ b の値 1 とパラメータ c の値 1 の組み合わせが出現しない事を表している．

$$\neg p(1,1,1) \vee \neg p(1,3,1)$$
$$\neg p(2,1,1) \vee \neg p(2,3,1)$$
$$\neg p(1,2,2) \vee \neg p(1,3,2)$$
$$\neg p(2,2,2) \vee \neg p(1,3,2)$$

4.2 隠された制約条件 (禁則条件)

SAT 符号化を行う際に問題となるのが，隠された制約条件の存在である．上で述べた SAT 符号化において，禁則条件を考慮してカバレッジ制約の変換を行っている．すなわち SAT 符号化を行う際には，予め全ての禁則条件を見つけておく必要がある．ここでは複数の制約条件から隠された制約条件を発見する手法について説明する．

アルゴリズムは図 2 のようになる．

以下に隠された禁則条件を発見する手法を示す．図 3 に示す入力が与えられたとする．このとき，二つの禁則条件から

明示的な禁則に表れるパラメータ集合 S を確定
集合 S を全パラメータとするモデルに対して数学的手法でペアワイズテスト集合 P を生成
集合 P のそれぞれのテストケースに対して以下の作業を実行
- 充足可能性判定により禁則が含まれるか判定
- if(結果が UNSAT)
  - テストケース中の全ての値の組み合わについて総当たりで充足可能性判定による禁則の判定を行い，既知の禁則以外の禁則があるかどうか確認

図 2 禁則条件生成アルゴリズム

(a,b) = (0,1) という値の組み合わせも禁則条件となっている．

まず，パラメータ集合 S を決定する．明示的な禁則からパラメータ集合 $S = \{a, b, c\}$ となる．

次に数学的手法でペアワイズテスト集合 P を生成する．生成したテストケース集合が表 4 となる．

ここから一つ一つのテストケースに対して，SAT を利用して隠された禁則が含まれているか確認を行っていく．まず，1 番目 (a,b,c) = (0,0,0) については，以下の 2 式の積の充足可能性を判定すればよい．

$$\bigvee_k p_{(j,k)} \wedge \bigwedge_{((j,k),(j',k'))\in\{((a,0),(c,0)),((b,1),(c,1))\}} (\neg p_{(j,k)} \wedge \neg p_{(j',k')})$$

$$p_{(1,0)} \wedge p_{(2,0)} \wedge p_{(3,0)}$$

ここで，$p_{(j,k)}$ はこのテストケースの j 番目のパラメータが k 番目の値を有していることを表す．この例では，UNSAT となり，禁則は含まれていることが分かる．

UNSAT となった場合，隠された禁則がないか，値のペアを総当たりで確認する．例えば，$(a, c) = (0, 0)$ について確認する場合は，以下の論理式の積を充足可能性判定すればよい．

$$\bigvee_k p_{(j,k)} \wedge \bigwedge_{((j,k),(j',k'))\in\{((a,0),(c,0)),((b,1),(c,1))\}} (\neg p_{(j,k)} \wedge \neg p_{(j',k')})$$

$$p_{(1,0)} \wedge p_{(3,0)}$$

この場合，UNSAT となるが，これは既知の禁則条件である．

2 行目 (a,b,c) = (1,1,1) について充足可能性判定を行うと，再び UNSAT となる．総当たりで確認を行うと，(b,c) = (1,1) の組み合わせで UNSAT となり，これは既知の禁則条件である．

3 行目 (a,b,c) = (2,2,-) で充足可能性判定を行うと，SAT となる．4 行目以降も同様の処理を行っていく．もしいずれかの行で UNSAT となり，総当たりで確認を行った際に UNSAT となったパラメータの値の組が既知のものではなかった場合，それは隠された禁則条件であることがわかる．

チェックに利用したテストケースはペアワイズテストを構成するので，この方法ですべてのパラメータ値のペアについてチェックできる．

5. 実験・評価

5.1 実験方法

4 章で提案したアルゴリズムを実装したツールを作成した．このツールを用いてテストケース数で評価を行う．二分探索の

a:	0, 1, 2
b:	0, 1, 2
c:	0, 1
d:	0, 1
禁則条件	
(a,c) = (0,0)	
(b,c) = (1,1)	

図 3 入力の例

表 4 数学的手法で生成されたペアワイズテスト集合

a	b	c
0	0	0
1	1	1
2	2	-
0	1	-
1	2	0
2	0	1
0	2	1
1	0	-
2	1	0

際のタイムアウトまでの時間を二種類用意し，同じ入力で二度実験を行った．実装したツールの性能の比較対象のツールとして PICT[2] を利用する．比較対象として PICT を選んだ理由は，PICT はパラメータ値の制約を考慮する事ができ，テストケース数も他のツールより比較的小さいからである．

表 5 は作成したテストモデルと，それについての各ツールの実行結果である．表の Model は入力パラメータとその値である．No.Cons は除外するペアの数，つまり禁則条件の数となっている．ここで SAT-TOOL1 は SAT ソルバの実行時のタイムアウトまでの時間を 60 秒，SAT-TOOL2 では 300 秒としたものである．

表 5 のモデルでは，パラメータの値が小さくなるほど，パラメータ数が多くなっている．また，制約の数はパラメータ数以下としている．表 5 での入力モデルは以下のような性質を持っている．

- パラメータの種類は 2~5 種類
- パラメータのとる値は 2~9
- パラメータのとる値が少ないほど，パラメータ数が増加
- 2~4 の値をとるパラメータを含む (8 番を除く)
- 制約の数はパラメータ数の半分

5.2 実験結果・考察

実験結果より以下の事がわかった．

- テストケース生成結果を全体的に見ると，提案手法を用いて生成されたテストケース数が少なくなっている場合が多い．これは提案手法が最適解を出そうとする手法であり，比較に用いたツールである PICT がグリーディな手法である事に因るものであると考えられる．しかし入力モデルによっては，PICT の方が良い解を生成している．
- 表 5 の 3，15 番のモデルに関しては，PICT のテストケース数が最も少なくなっている．8 番のモデルに関して提案手法を用いて，結果を生成する事ができなかった．本来

表 5 テスト結果

	Model	No.Cons	SAT-TOOL1	SAT-TOOL2	PICT
1	$2^{20}5^5$	13	33	**27**	42
2	$2^{50}3^{30}$	40	**20**	**20**	37
3	$3^{20}6^{15}$	18	87	86	**80**
4	$4^{10}7^3$	7	50	**50**	63
5	$2^{20}5^{10}8^5$	18	105	**99**	103
6	$3^{15}5^57^3$	12	**55**	**55**	67
7	$2^{30}3^{20}4^{10}$	30	**27**	**27**	40
8	$7^{10}8^{10}9^{10}$	15	—	—	**175**
9	$2^53^44^35^2$	7	**25**	**25**	32
10	$2^{25}4^{10}7^28^1$	19	**56**	**56**	72
11	$3^{15}4^{10}7^58^2$	16	95	**92**	95
12	$4^{10}5^58^29^1$	9	**75**	**75**	89
13	$3^54^36^28^19^1$	6	**72**	**72**	75
14	$2^{30}3^{20}4^{10}5^86^3$	34	**49**	**49**	63
15	$2^{10}3^{10}4^{10}5^89^5$	22	140	137	**127**
16	$2^{20}4^{15}5^47^38^2$	22	80	**77**	92

PICT はグリーディな手法でテストケースを生成するため，本手法のアルゴリズムでは，少なくとも PICT と同じ大きさの解が求められるはずである．つまり外部 SAT ソルバの実行がその大きさに近づく前にタイムアウトしてしまっていると考えられる．3，8，15 番の入力モデルから判断すると，ある一定以上の値をとるパラメータが大量にある場合に対して SAT ソルバの求解速度が落ちる事がわかる．

- 二分探索における外部 SAT ソルバを実行した際の，タイムアウトまでの設定時間に関するテストケース数を比較してみると，自明な事ではあるが，タイムアウトまでの時間を長く設定したものの方が小さいテストケース集合を生成している．設定時間を長くすればするほど良い結果が出るのは明らかである．しかし，二分探索が結果となる解に近づいてくると，充足可能あるいは充足不可能かどうかを判定する事が難しくなり，終了間際の 3，4 回の呼び出しでは，ほとんどタイムアウトとなる．
  表 5 の 2，4，6，7，9，10，12，13，14 番と多くのモデルにおいて生成されるテストケース数が，タイムアウトの時間に関わらず，同じ値を示している．現実的な時間でのテストケース生成には，タイムアウト時間の設定が必要不可欠である．例えばテスト生成時間に強い制約がない場合，タイムアウトの設定時間を長く取り，多少テストケース生成に時間がかかっても，なるべく少ないテストケース集合を目指す必要があると考えられる．
- 評価の際，実行時間については触れなかった．PICT はグリーディな手法を用いており，最適解を求めるわけではない．一方本手法では，アルゴリズムとしてテストケースの最小値を求める事を目標とし，実際はタイムアウトを用いて妥協している．よって本手法を用いたツールの実行時間んは PICT に比べるとかなり大きい．大きな入力モデルに対しては，生成する CNF 式が大きくなり，その生成時間や外部 SAT ソルバへのデータの受け渡しにも時間がかかっていると考えられる．

6. 結　　論

本研究では，パラメータ値の制約を考慮したペアワイズテスト集合の構成手法を提案し，その手法の評価を行った．提案した手法は，ペアワイズテストを SAT 問題として定式化し，外部 SAT ソルバを用いて計算を行う事で，パラメータ値の制約を満たしつつテストケースが少ないペアワイズ集合を生成する．

実験では，提案手法によって，全ての例ではないが，いくつかの入力モデルにおいて，現存のツールよりテストケース数を少なくする事ができた．しかし，テスト集合を得る事ができなかった入力モデルも存在する．これは手法の一部を見直し，大きな入力モデルに対して特別な操作を行うなど改善の余地がある．

本研究はペアワイズテストの構成問題を SAT 問題へと変換を行う事で，外部 SAT ソルバを汎用的なツールとして利用している．3.1 節でも述べたように，SAT ソルバは毎年性能を競う大会が開かれるなど，その性能は向上し続けている．つまり性能が高い SAT ソルバや，あるいはペアワイズテスト生成に適した SAT ソルバを用いる事で，今回の実験より良い結果を期待する事ができる．

本研究では，ペアワイズテスト，すなわち二つのパラメータ間の値をカバーするテストのみについて議論してきた．今後の研究としては，任意の $k(1 \leq k \leq n)$ 個のパラメータ間の値をカバーする k-way テストに対する拡張が考えられる．また，制約条件に関しても，本研究では，あるパラメータの組み合わせをテストケースから排除することについてのみ提案を行なったが，様々な制約条件を扱えるようにする事も必要である．

参 考 文 献

1) K. Bell and M. Vouk. On effectiveness of pairwise methodology for testing network-centric software. In *Information and Communications Technology, 2005. Enabling Technologies for the New Knowledge Society: ITI 3rd International Conference on*, pp. 221–235, 2005.
2) J. Czerwonka. Pairwise testing in real world: Practi-

cal extensions to test case generators. In *Proceedings of 24th Pacific Northwest Software Quality Conference*. Citeseer, 2006.
3) N. Kobayashi, T. Tsuchiya, and T. Kikuno. Non-specification-based approaches to logic testing for software. *Information and Software Technology*, 44(2):113–121, 2002.
4) Myra B. Cohen, Matthew B Dwyer. Constructing International Test Suites for Highly-Configurable System in the Presence of Constraints:A Greedy Approach. *IEEE Transaction on Software Engineering*, pp. 633–650, 2008.
5) 土屋, 菊野. ペアワイズテスト ーソフトウェアテストの効率化を求めてー. 電子情報通信学会論文誌, j90-D(10):2663–2674, Oct. 2007.