VDM++仕様を対象としたテストケース自動生成ツールBWDMへのドメイン分析テストの適用

平木場 風太[1,a)]　片山 徹郎[1,b)]

概要：形式手法を用いたソフトウェア開発は，仕様の曖昧さを防ぐ．我々は以前に，形式手法 VDM を用いたソフトウェア開発におけるテスト工程の効率化を目的として，テストケース自動生成ツール BWDM を試作した．既存の BWDM の問題点として，複数の変数を含む条件式を持つ仕様に対してテストケース生成ができないという点が挙げられる．本稿では，この問題点を解決するために，ドメイン分析テストのためのテストケース生成手法を提案し，BWDM に適用する拡張を行う．拡張した BWDM に，複数の変数を含む条件式を持つ VDM++仕様を適用した結果，ドメイン分析テストのためのテストケースの生成が可能なことを確認した．このことから，拡張した BWDM は，既存のテストケース生成処理の問題点を解決できたため，BWDM の有用性が向上したと言える．また，VDM++仕様からドメイン分析テストのためのテストケースを自動生成することが可能となったため，テストケース生成効率が向上したと言える．

キーワード：ソフトウェアテスト，境界値分析，ドメイン分析テスト，テストケース自動生成，VDM++

Application of Domain Analysis Testing into BWDM which is a Test Case Generation Tool for the VDM++ Specification

1. はじめに

社会におけるソフトウェアの重要性は高まっており，形式仕様記述言語を用いた仕様記述が重要視されてきている [1]．一方で，作成したソフトウェアにはテストが必要であるが，人手によるテストケースの設計には手間と時間がかかる．そのため，VDM++仕様を対象としたテストケース自動生成を目的としたツール BWDM（Boundary Value/Vienna Develop Method）を立山らが開発した [2]．

既存の BWDM は，VDM++仕様を入力として，境界値分析と記号実行を行い，境界値テストと，if-then-else 式の構造認識に基づいたテストを実施するためのテストケースを自動生成する．既存の BWDM は，1 つの変数のみを含む条件式を持つ仕様の場合に，テストケースを生成できるが，複数の変数を含む条件式を持つ仕様の場合は，テストケースを生成できない．

そこで，本研究では，既存の BWDM における，複数の変数を含む条件式を持つ仕様に対してテストケース生成ができないという問題点の解決を目的として，ドメイン分析テストのためのテストケース生成手法を提案し，BWDM に適用する拡張を行う．ここで，ドメイン分析テスト [3][4] とは，境界値分析において，複数の変数を含む条件式を持つ関数をテストする方法である．

また，複数の変数を含む条件式とは，片方の辺にのみ複数の変数が含まれ，もう片方の辺には定数のみ含まれる条件式を指す．すなわち，両辺に変数を含む条件式を持つ仕様は本研究の対象としない．

本論文の構成は次の通りである．第 2 章では，ドメイン分析テストの定義を示す．第 3 章では，既存の BWDM と問題点を示す．第 4 章では，拡張した BWDM の説明および適用例を示す．第 5 章では，考察を示す．第 6 章では，本研究のまとめと今後の課題を示す．

2. ドメイン分析テスト

本研究における，ドメイン分析テストの定義と，ドメインテストが必要となる仕様の例を，以下で示す．

[1] 宮崎大学
University of Miyazaki
[a)] hirakoba@earth.cs.miyazaki-u.ac.jp
[b)] kat@cs.miyazaki-u.ac.jp

2.1 定義

本研究におけるドメイン分析テストとは，関係性がある複数の変数を同時にテストする方法である [3][4]．ドメインとは，入力するデータの定義域である．ドメイン分析とは，入力する変数と条件式を分析し，ドメインを抽出することである．ドメイン分析テストでは，ドメイン毎に on ポイント，off ポイント，in ポイント，out ポイントと呼ばれる入力値，および，それを基にしたテストケースを作成し，テストする．境界値とは，同値分割した領域の端，あるいは端のどちらか側で最小の増加的距離にある入力値又は出力値である [4]．本研究では，境界値の中でも，条件式を満たす境界値を利用する．これを，TB(True Boundary) と命名する．TB の定義を以下に示す．exp を条件式，$left$ を左辺，int を右辺とする．

- exp が $left = int$ のとき，$TB = int$ とする．
- exp が $left < int$ のとき，$TB = int - 1$ とする．
- exp が $left > int$ のとき，$TB = int + 1$ とする．
- exp が $left <= int$ のとき，$TB = int$ とする．
- exp が $left >= int$ のとき，$TB = int$ とする．

それぞれのポイントの定義を，以下に示す．

- on ポイント：着目条件式の TB である．ドメインを決定づける条件式に付き 1 つ生成する．他の on ポイントと重複してはならない．$targetExp$ を着目条件式とすると，on ポイントは，$targetExp = True$ かつ $left = TB$ となる値でなければならない．
- off ポイント：着目する on ポイントに隣接し，TB でない値である．on ポイントに付き複数（着目条件式に含まれる変数 $* 2$）個存在する．$targetVar$ を着目変数とすると，off ポイントは，$targetVar + 1$ となる値または $targetVar - 1$ となる値でなければならない．
- in ポイント：ドメインを決定づける全ての条件式を満たす値である．ドメインに付き 1 つ生成する．on ポイントや off ポイントと重複してはならない．$onPoints$ を on ポイントの集合とし，$offPoints$ を off ポイントの集合，$inPoint$ を in ポイントとすると，in ポイントは，$(exp1 \wedge exp2 \wedge ... \wedge expN) = True$ かつ $inPoint \notin (onPoints \cup offPoints)$ となる値でなければならない．
- out ポイント：着目条件式のみを満たさない値である．ドメインを決定づける条件式に付き 1 つ生成する．off ポイントと重複してはならない．$outPoint$ を out ポイントとすると，out ポイントは，$(\neg targetExp \wedge exp1 \wedge exp2 \wedge ... \wedge expN) = True$ かつ $outPoint \notin offPoints$ となる値でなければならない．

また，それぞれのポイントは，以下のパラメータを持つ．

- 正常系判定値
  - “正常系”とは，ポイントの期待出力がドメインの期待出力と一致する状態のことを言う．
  - “非正常系”とは，ポイントの期待出力がドメインの期待出力と一致しない状態のことを言う．
  - ”正常系判定値”とは，正常系であるかどうかを保持する値である．正常系と非正常系の 2 つの状態を持つ．
- 着目条件式
  on ポイント，off ポイント，out ポイントのみが持つ．どの条件式に着目してポイントを生成したかの情報である．
- 着目変数
  off ポイントのみが持つ．どの変数に着目して，on ポイントに隣接するポイントを生成したかの情報である．

2.2 例

ドメイン分析テストが必要となる仕様の例として，遊園地チケット割引機能をテストすることを考える．遊園地チケット割引機能は，夫婦である夫と妻それぞれの年齢を入力とし，割引価格が適用されるかどうかを判定する関数であり，以下の様なルールを持つ．

A) 以下の条件をすべて満たすとき，遊園地チケットは割引価格となる．
- 夫の年齢と妻の年齢の合計が 50 歳以下である．
- 夫の年齢は 18 歳以上である．
- 妻の年齢は 16 歳以上である．

B) A) でない場合，遊園地チケットは割引価格とならない．

この仕様を VDM++で記述した仕様を，**図 1** に示す．7 行目に，複数の変数を左辺に含む条件式があるため，既存の BWDM ではテストケース生成ができない．また，“割引価格となる”というドメインの各ポイントを生成した例を，**図 2** に示す．ドメインを決定づける 3 つの条件式は，“夫の年齢 + 妻の年齢 <= 50”，“夫の年齢 >= 18”，“妻の年齢 >= 16”である．on ポイントは，“On1”～“On3”の 3 つであり，それぞれ，条件式の境界線上に存在する．off ポイントは，“Off11”～“Off32”の 8 つである．“Off11”～“Off14”は“夫の年齢 + 妻の年齢 <= 50”という条件式に着目した“On1”に隣接する off ポイントであり，4 つ存在する．これは，“夫の年齢”を正負の方向にそれぞれずらした off ポイントが 2 つ存在し，同じように，“妻の年齢”を正負の方向にそれぞれずらした off ポイントが 2 つ存在するためである．しかし，“On2”の off ポイントは“Off21”，“Off22”の 2 つのみである．これは，“夫の年齢”をずらした off ポイントが“妻の年齢 >= 16”となり，TB となるからである．in ポイントは，“In”の 1 つであり，“割引価格となる”ドメインの全ての条件式を満たす値を持つ．out ポイントは，“Out1”～“Out3”の 3 つであり，着目条件式のみを満たさない値を持つ．

3. 既存の BWDM

既存の BWDM は，VDM++仕様を対象としたテスト

```
1 class 遊園地チケット
2
3 functions
4
5 static public 割引判定 : int * int -> seq of char
6   割引判定 (夫の年齢, 妻の年齢) ==
7     if(夫の年齢 + 妻の年齢 <= 50) then
8       if(夫の年齢 >= 18) then
9         if(妻の年齢 >= 16) then
10          “割引価格となる”
11        else
12          “割引価格とならない (妻の年齢 < 16)”
13      else
14        “割引価格とならない (夫の年齢 < 18)”
15    else
16      “割引価格とならない (夫の年齢 + 妻の年齢 > 50)”;
17
18 end 遊園地チケット
```

図 1　ドメインテストが必要となる仕様（遊園地チケット割引機能）
Fig. 1　Specifications that require domain test（amusement park ticket discount function)

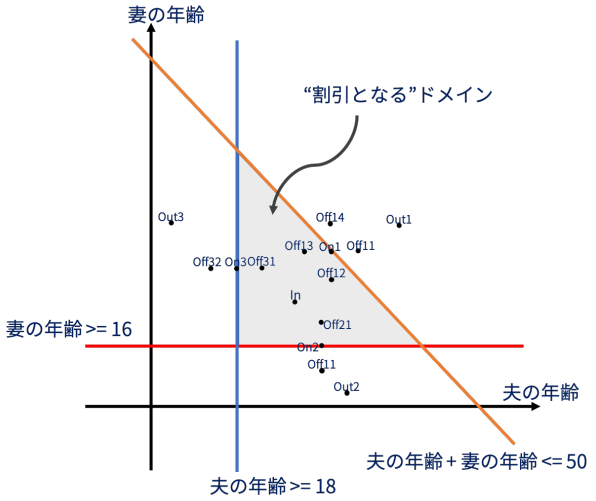

図 2　遊園地チケット割引機能（図 1）の割引になる条件にドメイン分析を適用した例
Fig. 2　Example of applying domain analysis to discount conditions for the amusement ticket discount function(Fig. 1)

ケース生成ツールである．Java と Kotlin で記述された CLI ツールであり，コマンドライン上で利用できる，

VDM++仕様を対象としたテストケースとは，VDM++仕様の関数に対する入力と，期待する出力の組から成る集合である．既存の BWDM では，テキスト形式と CSV 形式で出力できる．生成したテストケースは，VDM++仕様自体のテスト，および，VDM++仕様を基に作成したプログラムのテストに利用することができる．

既存の BWDM は，以下の機能がある．

- 記号実行によるテストケース生成機能
  生成したテストケースは，仕様内の関数の全ての実行フローを網羅できることが期待できる．

```
1 class SampleClass
2
3 functions
4
5 sampleFunction : int*nat*nat − > seq of char
6   sampleFunction(a, b, c) ==
7    if(a < 100) then
8     if(b > 2018) then
9      ”a は 100 未満かつ b は 2018 より大きい”
10    else
11     ”a は 100 未満かつ b は 2018 以下”
12   elseif(c < 12) then
13    ”a は 100 以上かつ c は 12 未満”
14   else
15    ”a は 100 以上かつ c は 12 以上”;
16
17 end SampleClass
```

図 3　入力例：VDM++仕様
Fig. 3　An input example of a VDM++ specification file

```
各引数の境界値
a : 2147483648 2147483647 -2147483648 -2147483649 99 100
b : 4294967295 4294967294 0 -1 2019 2018
c : 4294967295 4294967294 0 -1 11 12

境界値分析によるテストケース（ペアワイズ法適用）
No.1 : 100 -1 11 -> Undefined Action
No.2 : 100 2018 12 -> "aは100以上かつcは12以上"
No.3 : 2147483647 0 4294967295 -> Undefined Action
No.4 : -2147483649 2018 4294967294 -> Undefined Action
No.5 : 99 -1 0 -> Undefined Action
No.6 : 2147483647 4294967294 12 -> "aは100以上かつcは12以上"
No.7 : -2147483648 4294967294 4294967294 -> "aは100未満かつbは2018より大きい"
No.8 : 2147483647 4294967295 0 -> Undefined Action

=======省略=======

No.36 : -2147483649 0 -1 -> Undefined Action
No.37 : 99 4294967295 4294967295 -> Undefined Action
No.38 : 2147483647 -1 -1 -> Undefined Action
No.39 : 2147483647 -1 4294967294 -> Undefined Action
No.40 : 2147483647 2018 4294967295 -> Undefined Action

記号実行によるテストケース
No.1 : 99 2019 1 -> "aは100未満かつbは2018より大きい"
No.2 : 99 2018 1 -> "aは100未満かつbは2018以下"
No.3 : 100 b 11 -> "aは100以上かつcは12未満"
No.4 : 100 b 12 -> "aは100以上かつcは12以上"
```

図 4　出力例：テストケース
Fig. 4　An output example of testcases

- 境界値分析によるテストケース生成機能
  生成したテストケースを用いることによって，仕様内の関数の境界値テストを実施できる．
- ペアワイズ法を用いた，境界値分析によるテストケース生成時の組合せ総数の削減機能 [5]
  境界値分析によってテストケースを生成する際，ペアワイズ法 [6] を用いて，各引数の境界値の組合せ総数を削減したテストスイート（テストケースの集合）を生成できる．

BWDM を使用することにより，VDM++仕様を用いたソフトウェア開発効率を改善できる．以下で，入出力例と，既存の BWDM の問題点を示す．

3.1　入出力例

VDM++仕様ファイルの例を図 **3** に，そのファイルから生成したテストケースの例を図 **4** に，それぞれ示す．また，出力テストケースのフォーマットを，図 **5** に示す．“各引数の境界値”にて，引数 a， b， c それぞれの境界値の

```
各引数の境界値
<引数1>: <入力値1> ... <入力値N>
・
・
・
<引数N>: <入力値1> ... <入力値N>

境界値分析によるテストケース（ペアワイズ法適用）
<ID> : <入力値1> ... <入力値N> -> <期待出力>

記号実行によるテストケース
<ID> : <入力値1> ... <入力値N> -> <期待出力>
```

図 5　既存の BWDM のテストケースの出力フォーマット
Fig. 5　Format of testcase by existed BWDM

集合を出力している．そして，“境界値分析によるテストケース（ペアワイズ法適用）”にて，ペアワイズ法を用いて，各引数の境界値の組合せ総数を削減したテストスイートを出力している．また，”記号実行によるテストケース”にて，関数の全ての実行フローを網羅できるテストスイートを出力している．

3.2　問題点

既存の BWDM が持つ問題の 1 つに，左辺または右辺に複数の変数を含む条件式を持つ仕様に対してテストケース生成ができない点がある．図 1 に示した VDM++仕様には，7 行目の条件式の左辺に複数の変数が含まれているため，この仕様を既存の BWDM に適用しても，テストケース生成ができない．

この理由を，既存の BWDM に図 1 の仕様を入力した場合を例に，以下で説明する．

- 図 1 の 6 行目にて，入力する 2 つの変数は“夫の年齢”と“妻の年齢”と定義している．7 行目の if 条件式は“夫の年齢”と“妻の年齢”の 2 つの変数を利用しているが，既存の BWDM は“夫の年齢+妻の年齢”という 1 つの変数だと解釈する．“夫の年齢+妻の年齢”という引数は仕様に定義されていないため，既存の BWDM はエラーを出力し，動作を停止してしまう．
- 図 1 の 7 行目の“夫の年齢+妻の年齢 <=50”という条件式の TB を求めるには，「制約を満たす入力があるかどうか」という，充足可能性問題（Satisfiable Problem，SAT）を解かなければならない．しかし，既存の BWDM は，充足可能性問題を解くことができない．

本研究では，上記の問題を解決するために，ドメイン分析テストのためのテストケース生成手法を提案し，既存の BWDM に適用することで，BWDM を拡張する．

4.　BWDM の拡張

本研究で拡張した BWDM の処理の流れを，**図 6** に示す．既存の BWDM では，入力した VDM++仕様を構文解析し，その結果を，境界値分析部と記号実行部に渡し，テストケースにおける入力データを生成する．

拡張した BWDM では，境界値分析部，記号実行部に加えて，ドメイン分析部を追加した．ドメイン分析部では，2.1 節で記述した，in ポイント，out ポイント，on ポイント，off ポイントを生成する．また，複数の変数が条件式に含まれる VDM++仕様の解析に対応できるように，構文解析部を一部修正した．さらに，ドメインテストに必要な，正常系判定値，着目条件式，そして着目変数の情報をテストケースに含めるために，テストケース生成部において，ドメインテストによるテストケースを出力する際は，正常系判定値，着目条件式，着目変数の情報も出力する処理を追加した．

4.1　構文解析部

BWDM の構文解析は，Nick らの開発した VDM 構文解析ツール VDMJ を用いて実現している [7]．既存の BWDM は，各条件式の左辺または右辺に複数の変数を含む場合，エラーを出力してしまう（3.2 節参照）．したがって，if 条件式の左辺と右辺の式を構文解析し，条件式内の変数を抽出することによって，左辺または右辺に複数の変数が条件式に含まれる VDM++仕様の構文解析を可能とした．

4.2　ドメイン分析部

拡張した BWDM はドメイン分析テストにおける，on ポイント，off ポイント，in ポイント，out ポイントを生成するために，ドメイン分析部を持つ．各ポイントは，テストケースの入力データであり，引数名と値のタプルの配列を保持している．各ポイントの生成については，4.3 節にて説明する．

作成したドメイン分析部のクラス図を，**図 7** に示す．各クラスの詳細を，以下に示す．

- Factor クラスは，引数の情報を持つクラスである．変数名（name）と値（value）を保持する．
- Point クラスは，テストケースの入力データの情報を持つクラスである．テストケース名（name），そのポイントが着目条件式（forcusedConditionalExpression），そのポイントの着目変数（forcusedVariable），そして複数の Factor（factors）を保持する．
- DomainPoints クラスは，ドメインの情報を持つクラスである．期待出力（name），on ポイントの集合（onPoints），off ポイントの集合（offPoints），in ポイント（inPoints），out ポイントの集合（outPoints）を保持する．
- DomainAnalyzer クラスは，ドメイン分析を行うクラスである．ドメインの集合（domains）を保持している．また，on ポイント，off ポイント，in ポイント，out ポイントを生成する機能 (generateXXPoints メソッド)，および，各ドメインの各ポイントを，期待出力生成部に入力できるデータ構造として抽出す

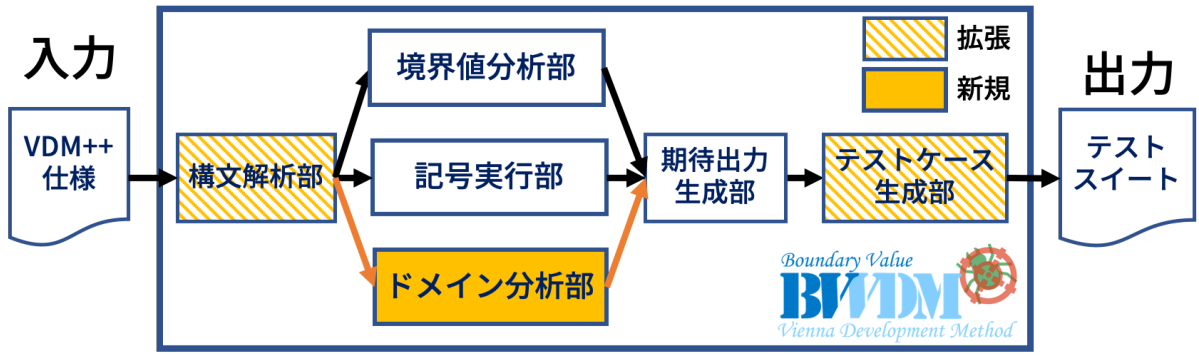

図 6　拡張した BWDM の処理の流れ
Fig. 6　Flow of processing of the extended BWDM

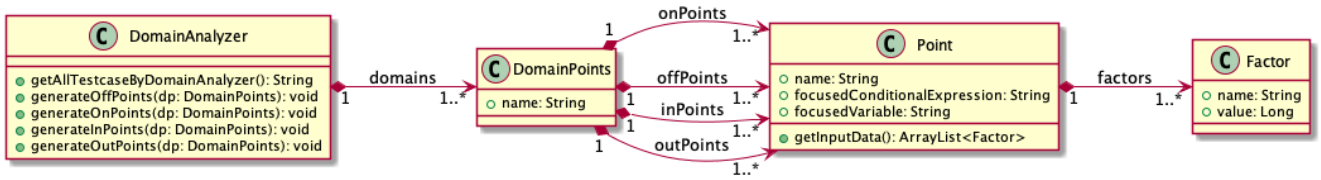

図 7　ドメイン分析部のクラス図
Fig. 7　Class diagram of domain analysis unit

る機能 (getAllTestCaseByDomainAnalyzer メソッド) を持つ．

4.3　各ポイントの生成手法の提案と適用

拡張した BWDM は，入力する仕様のドメイン毎に，on ポイント，off ポイント，in ポイント，out ポイントを生成する．

各ポイントを満たす変数の値を求めるために，SMT ソルバ（Satisfiable Modulo Theories）[8] を利用する．充足可能性問題（SAT）を解くアルゴリズムを実装したソフトウェアを SAT ソルバと呼び，SAT ソルバを算術演算に対応させたソフトウェアを SMT ソルバと呼ぶ．幅広く知られている SMT ソルバの 1 つに，Microsoft Research が開発を進めている Z3[9] がある．Z3 は，C，C++，Java，Python などのプログラミング言語から利用できる API を提供する．拡張した BWDM は，この Z3 を利用し，条件式を満たす入力値を生成する．

各ポイントの生成方法の提案手法を，以下に示す．

4.3.1　on ポイント

on ポイントは，着目条件式の TB である．ドメインを決定づける条件式に付き 1 つ生成し，他の on ポイントと重複してはならない．ドメインに関わる条件式の数だけ以下の手順を繰り返し，各条件式に着目した on ポイントを生成する．

(1) もし，着目条件式の比較演算子が，“>=” または “<=” である場合，“=” に置き換える．“>” である場合，“=” で置き換え，右辺を “+1” する．“<” である場合，“=” で置き換え，左辺を “+1” する．

(2) (1) で修正した条件式と，その他の条件式を Z3 に入力し，解（引数と値のタプルの集合）を求める．解が求まらなかった場合，現在の着目条件式における on ポイントが存在しないため，生成を行わずに次の着目条件式の on ポイントの生成を行う．

(3) 配列 onPoints（4.2 節参照）を参照し，他の on ポイントと値が重なるかどうかを判定する．

(a) 重なっている場合，条件式に，”重なっている変数 != 重なった値 “という条件式を加え，(2) に戻る．

(b) 重なっていない場合，解を基に，Point インスタンスを作り（4.2 節参照），メンバ forcusedConditionalExpression には，着目条件式を格納する．作成したインスタンスを配列 onPoints に格納する．

4.3.2　off ポイント

off ポイントは，着目する on ポイントに隣接し，TB でない値である．on ポイントに付き複数（着目条件式に含まれる変数 * 2）個存在する．配列 onPoints（4.2 節参照）を参照し，on ポイントの数だけ以下の手順を繰り返し，on ポイントに着目した off ポイントを生成する．

(1) 着目する on ポイントのメンバ forcusedConditionalExpression を参照し，着目する on ポイントがどの条件式に着目していたかを保持する．

(2) (1) の条件式から，引数を抽出する．抽出した引数の数だけ以下を繰り返す．

(a) 以下の処理を 2 回繰り返す．1 回目は “N=−1” と

定義し，2 回目は“N=1”と定義する．

(i) 着目する on ポイントインスタンスをコピーする．

(ii) (i) でコピーした Point インスタンスのメンバ配列 factors から，“Factor.name == 引数名”となる Factor インスタンスを検索する．

(iii) (ii) で検索して見つかった Factor インスタンスのメンバ value を“+N”する．

(iv) Point インスタンスのメンバ forcusedVariable に引数名を格納する．

(v) Point インスタンスを配列 offPoints（4.2 節参照）に格納する．

4.3.3 in ポイント

in ポイントは，ドメインを決定づける全ての条件式を満たす値である．ドメインに付き 1 つ生成し，他の on ポイント，off ポイントと重複してはならない．以下の手順を行い，in ポイントを生成する．

(1) もし，条件式の比較演算子が，“>=”である場合，“>”で置き換える．“<=”である場合，“<”で置き換える．これを，すべての条件式に対して行う．

(2) (1) で修正した条件式を Z3 に入力し，解（引数と値のタプルの集合）を求める．解が求まらなかった場合，in ポイントが存在しないため，生成を行わない．

(3) 配列 onPoints と配列 offPoints（4.2 節参照）を参照し，他の on ポイント，off ポイントと値が重なるかどうかを判定する．

(a) 重なっている場合，条件式に，”重なっている変数 != 重なった値“という条件式を加え，(2) に戻る．

(b) 重なっていない場合，解を基に，Point インスタンスを作り（4.2 節参照），配列 inPoints に格納する．

4.3.4 out ポイント

out ポイントは，着目条件式のみを満たさない値である．ドメインを決定づける条件式に付き 1 つ生成し，他の off ポイントと重複してはならない，ドメインに関わる条件式の数だけ以下の手順を繰り返し，各条件式に着目した out ポイントを生成する．

(1) 着目条件式を“!(着目条件式)”に置き換え，否定する．

(2) (1) で否定した条件式と，その他の条件式を Z3 に入力し，解（引数と値のタプルの集合）を求める．解が求まらなかった場合，現在の着目条件式における out ポイントが存在しないため，生成を行わずに次の着目条件式の out ポイントの生成を行う．

(3) 配列 onPoints と配列 offPoints（4.2 節参照）を参照し，他の on ポイント，off ポイントと値が重なるかどうかを判定する．

(a) 重なっている場合，条件式に，”重なっている変数 != 重なった値“という条件式を加え，(2) に戻る．

(b) 重なっていない場合，解を基に，Point インスタンスを作り（4.2 節参照），メンバ forcusedConditionalExpression には，着目条件式を格納する．作成したインスタンスを配列 outPoints に格納する．

```
ドメインテストによるテストケース
- <ドメインの期待出力>
-- Onポイント
<ID> : <入力値1> ... <入力値N>, <正常系かどうか>, <着目する条件式> -> <期待出力>

-- Offポイント
<ID> : <入力値1> ... <入力値N>, <正常系かどうか>, <着目する条件式>, <着目する条件式> -> <期待出力>

-- Inポイント
<ID> : <入力値1> ... <入力値N>, <正常系かどうか> -> <期待出力>

-- Outポイント
<ID> : <入力値1> ... <入力値N>, <正常系かどうか>, <着目する条件式> -> <期待出力>

- <ドメイン名2>
.
.
.
- <ドメイン名N>
```

図 8 ドメイン分析テストのためのテストケースの出力フォーマット
Fig. 8 Format of testcase for domain analysis testing by BWDM

4.4 テストケースの生成

既存の BWDM は，期待出力生成部において，入力データを基に期待出力を生成する機能を持つ．DomainAnalyzer クラス（4.2 節参照）の getAllTestCaseByDomainAnalyzer メソッドを呼び出すことにより，4.3 節で生成した各ドメインの各ポイントを，期待出力生成部に入力できるデータ構造として抽出することができる．そのため，既存の BWDM の期待出力生成部をそのまま用いて，期待出力生成とテストケース生成は可能である．拡張した BWDM では，各ドメインの各ポイントのテストケースを出力する．

しかし，既存の BWDM のテストケース生成部には，ドメインテストに必要な，正常系判定値，着目条件式，そして着目変数の情報を出力テストケースに加える処理が存在しない．これに対応するため，テストケース生成部において，ドメインテストによるテストケースを出力する際，正常系判定値，着目条件式，着目変数の情報についても出力する処理を追加した．着目条件式の出力には，Point クラスのメンバ forcusedConditionalExpression を参照する．off ポイントの場合，着目変数の出力を行う．出力には，Point クラスのメンバ forcusedVariable を参照する．正常系判定値は，出力するテストケースの期待出力と，ドメインの期待出力（DomainPoints.name）を比較して判断する．等しければ，“正常系”と出力する．等しくなければ“非正常系”と出力する．

出力テストケースのフォーマットを，**図 8** に示す．< と > で囲まれている部分を，それぞれの情報で置き換える．

4.5 出力結果

拡張した BWDM に，遊園地チケット割引機能（図 1）を入力として適用した結果の一部を，**図 9** に示す．図 9 では，“割引価格となる”期待出力以外の期待出力を持つド

```
ドメインテストによるテストケース
- "割引価格となる"
-- Onポイント
No.1 : 20 16, 正常系, 妻の年齢>=16 -> "割引価格となる"
No.2 : 18 18, 正常系, 夫の年齢>=18 -> "割引価格となる"
No.3 : 32 18, 正常系, 夫の年齢+妻の年齢<=50 -> "割引価格となる"

-- Offポイント
No.1 : 20 17, 正常系, 妻の年齢>=16, 妻の年齢 -> "割引価格となる"
No.2 : 20 15, 非正常系, 妻の年齢>=16, 妻の年齢 -> "割引価格とならない(妻の年齢 < 16)"
No.3 : 32 19, 非正常系, 夫の年齢+妻の年齢<=50, 妻の年齢 -> "割引価格とならない(夫の年齢 + 妻の年齢 > 50)"
No.4 : 32 17, 正常系, 夫の年齢+妻の年齢<=50, 妻の年齢 -> "割引価格となる"
No.5 : 33 18, 非正常系, 夫の年齢+妻の年齢<=50, 夫の年齢 -> "割引価格とならない(夫の年齢 + 妻の年齢 > 50)"
No.6 : 31 18, 正常系, 夫の年齢+妻の年齢<=50, 夫の年齢 -> "割引価格となる"
No.7 : 19 18, 正常系, 夫の年齢>=18 -> "割引価格となる"
No.8 : 17 18, 非正常系, 夫の年齢>=18 -> "割引価格とならない(夫の年齢 < 18)"

-- Inポイント
No.1 : 20 18, 正常系 -> "割引価格となる"

-- Outポイント
No.1 : 20 0, 非正常系, 妻の年齢>=16 -> "割引価格とならない(妻の年齢 < 16)"
No.2 : 20 33, 非正常系, 夫の年齢+妻の年齢<=50 -> "割引価格とならない(夫の年齢 + 妻の年齢 > 50)"
No.3 : 0 18, 非正常系, 夫の年齢>=18 -> "割引価格とならない(夫の年齢 < 18)"

- "割引価格とならない（妻の年齢<16）"
-- Onポイント
No.1 : 20 15, 正常系, 妻の年齢>=16 -> "割引価格とならない（妻の年齢<16）"
No.2 : 18 10, 正常系, 夫の年齢>=18 -> "割引価格とならない（妻の年齢<16）"
No.3 : 40 10, 正常系, 夫の年齢 + 妻の年齢 <= 50 -> "割引価格とならない（妻の年齢<16）"

-- Offポイント
=======省略=======
```

図 9 遊園地チケット割引機能（図 1）のテストケースの一部
Fig. 9 A part of testcases for the amusement ticket discount function(Fig. 1)

メイン（“割引価格とならない（妻の年齢 < 16）”，“割引価格とならない（夫の年齢 < 18）”，“割引価格とならない（夫の年齢 + 妻の年齢 > 50）”）のテストケースは省略してある．

“割引価格となる”ドメインの 3 つの on ポイントは，ドメインを決定づける 3 つの条件式（“夫の年齢 + 妻の年齢 <= 50”，“夫の年齢 >= 18”，“妻の年齢 >= 16”）にそれぞれ着目し，着目条件式の TB が入力となっており，かつ，期待出力と正常系判定値（ポイントの期待出力とドメインの期待出力が一致するかどうか）が適切であることが確認できる．off ポイントは，各 on ポイントに隣接しており，TB ではない値が入力となっており，かつ，着目変数，期待出力，正常系判定値が適切であることが確認できる．in ポイントは，期待出力とドメインの期待出力が一致しており，各条件式の TB でない値が入力となっており，かつ，正常系であることが確認できる．out ポイントは，関係する 3 つの条件式に着目し，着目条件式のみを否定する TB でない値が入力となっており，かつ，期待出力と正常系判定値が適切であることが確認できる．

また，図 9 では省略している，“割引価格となる”期待出力以外の期待出力を持つドメイン（“割引価格とならない（妻の年齢 < 16）”，“割引価格とならない（夫の年齢 < 18）”，“割引価格とならない（夫の年齢 + 妻の年齢 > 50）”）に対しても，テストケースが適切に生成できていることを確認した．

したがって，拡張した BWDM は，既存の BWDM の問題点である，条件式内に複数の変数がある VDM++仕様を解析できること，かつ，ドメインテストによるテストケース生成が適切にできることを確認できた．

5. 考察

本研究では，既存の BWDM における，複数の変数を含む条件式を持つ仕様に対してテストケース生成ができないという問題点の解決を目的として，ドメイン分析テストのためのテストケース生成手法を提案し，BWDM に適用する拡張を行った．この章では，人手によるドメイン分析テストのためのテストケース作成との比較検証，関連研究との比較，および，拡張した BWDM の問題点を示す．

表 1 テストケース作成に要した時間の比較
Table 1 The time to generate testcases

被験者 or 拡張した BWDM	時間
被験者 A	8m 16s
被験者 B	10m 23s
被験者 C	30m
被験者 D	24m 04s
被験者（平均）	18m 10s
BWDM	15s

5.1 人手によるドメイン分析テストのためのテストケース作成との比較検証

人手によるドメイン分析テストのためのテストケース作成と，拡張した BWDM によるドメイン分析テストのためのテストケース生成で，作成（生成）に要した時間の比較検証を行った．その結果を，**表 1** に示す．

対象とした VDM++仕様は，2 章で用いた図 1 である．“割引価格となる”を期待出力に持つドメインに対するドメイン分析テストのためのテストケースを作成する時間を計測した．生成するテストケースとしては，以下を基準とした．

(1) on ポイント，off ポイント，in ポイント，out ポイントを記述する
(2) on ポイント，off ポイント，out ポイントには，着目条件式も記述する
(3) off ポイントには，着目変数も記述する
(4) 各ポイントには，期待出力と正常系であるかどうかも記述する．

検証に参加したメンバーは本研究室の大学院生 3 人と学部 4 年生 1 人であり，普段からソースコードの読み書きを行い，基本的なプログラミングの知識を有している．VDM++の文法の知識を持たない者も含まれるが，今回の検証に必要な文法は，事前に他の VDM++の例を用いてレクチャーした．また，ドメイン分析テストのためのテストケース生成についても，事前に他の VDM++仕様とテストケースの例を用いてレクチャーした．

人手による検証では，図 1 を印刷した紙を渡し，仕様を確認後，テストケースを書き始めてから，テストケースを記述し終えるのに要した時間を計測した．入力データと戻り値の組み合わせが不正確な場合，間違いを指摘し，被験者が正しい組み合わせを記述した時点で時間計測終了とした．また，制限時間を 30 分とし，制限時間を超えた場合，その場で時間計測終了とした．

拡張した BWDM による検証では，コマンドライン上での命令操作で，拡張した BWDM によるテストケース生成を行うのに要した時間を計測した．また，実験に用いたコンピュータは，OS:macOS 10.14.5，CPU:2.3GHz Intel Core i5，メモリ:16GB である．

なお，Java の System.nanoTime[10] メソッドを用いて，命令操作を省いた純粋なテストケース生成処理に BWDM が要した時間を計測した結果，1.25 秒であった．

人手による作成と比較した結果，平均で 18 分程の時間短縮を確認できた．対象にした VDM++仕様には，VDM++独特の文法等は含まれないため，VDM++に対する慣れなどの影響は無視できるものと思われる．また，人手によるテストケース生成の場合，ヒューマンエラーも見られた．具体的には，off ポイントの記述時に，条件式の解釈を間違え，誤った期待出力を記述してしまった．（例：入力（17, 20）の期待出力を“遊園地チケットは割引価格とならない．(妻の年齢 < 16)”と記述した．）仕様の規模が拡大すると，人手とコンピュータとの処理効率の差に加えて，ヒューマンエラーの有無などにより，テストケース生成に要する時間の差は更に拡大していくと思われる．

5.2 関連研究

ドメイン分析に基づいたテストケースを自動生成する手法としては，丹野らの研究 [11] がある．この手法では，変数同士に依存関係がある場合でも，制約ソルバ [8] を用いることで，依存関係を考慮した境界値等のテストケースを網羅的に生成する．入力として，ソフトウェアの設計情報をモデル化した設計モデルと呼ばれるテキストを入力する必要がある．設計モデルは [11] で定義されている．

設計モデルは仕様書を基に，人手で記述する必要があるため，設計モデルの作成に時間と手間がかかってしまうという問題がある．これに対して，拡張した BWDM は，VDM++仕様を基に，自動でテストケースを生成できるため，設計モデルを作成する必要がないという利点がある．しかし，仕様書が VDM++で記述されておらず，自然言語で記述されている場合，VDM++仕様で記述しなければ，拡張した BWDM を使うことができない．そのため，自然言語で記述された仕様書からドメイン分析テストのためのテストケースを作成する場合，設計モデルを記述するか，VDM++仕様を記述するかで対応が分かれることとなる．なお，設計モデルの記述と VDM++仕様の記述に必要な要素がほぼ同じなため，記述量はほとんど変わらないと考える．設計モデルは丹野らの手法でしか利用できないが，VDM++仕様は BWDM 以外にも，VDMTools[12] や Overture IDE[13] などの支援ツールが揃っており，仕様の検証や記述を行いやすいという利点がある．

5.3 拡張した BWDM の問題点

拡張した BWDM の問題点を以下に示す．

- 複数ドメインのテストケース生成を行った際にテストケースが重複してしまう可能性がある
  複数ドメインに対して，ドメイン分析テストのためのテストケース生成を行った場合，各ドメインの各ポイントを生成するため，他のドメインのテストケースと内容が重複する可能性がある．重複したテストを再度行うことはテスト効率の低下につながる．このため，重複するテストケースを排除できる機能を BWDM に実装することによって，BWDM の有用性が更に向上すると考える．
- 条件式の両辺に変数がある条件式の解析ができない
  条件式中の両辺に変数がある場合，拡張した BWDM はドメイン分析を行うことができないため，テストケースの入力データ生成を行えない．このため，そのような条件式を，片方の辺にのみ変数が存在する条件式に変換する処理を BWDM に実装することによって，BWDM の適用範囲の更なる拡大を見込める．
- ドメイン分析の結果が分かりにくい
  拡張した BWDM は，ドメイン分析の結果を図 9 のように出力するため，結果が適切なのかが分かりにくい．例えば，図 2 のように可視化することで，人間が，ドメインに対するテストが十分に行えているかの判断や，正しい条件が定義できているかの判断を支援できると考える．したがって，可視化機能を BWDM に実装することによって，テスト効率の向上を見込むことができる．ただし，引数が 3 つ以上になった場合の可視化が困難であることも考えられる．
- BWDM の適用範囲が狭い
  拡張した BWDM が静的解析可能である型は，VDM++の基本型の中でも，整数を表す int 型，0 を含む自然数を表す nat 型，0 を含まない自然数を表す nat1 型のみである．実数を表す real 型や有理数を表す rat 型，複数の型から構成される合成型 (集合型，写像型など) には対応していない．そのため，それらを用いた仕様からテストケースを生成できない．それらの未対応の型への対応は，VDM++仕様を静的解析する際に型情報を読み込み，境界値分析時に，読み込んだ型情報から境界値の生成処理を BWDM に追加することによって，対応可能と考える．また，操作定義，型定義などの定義，事前条件や事後条件などの構文にも未対応である．これらの記述からテストケースを生成する手法を考案し，BWDM に実装することで，BWDM の有用性，および，VDM++仕様を基にしたテストケース生成に対する有用性がさらに向上すると考える．

6. おわりに

本研究では，既存の BWDM における，複数の変数を含む条件式を持つ仕様に対してテストケース生成ができないという問題点の解決を目的として，ドメイン分析テストのためのテストケース生成手法を提案し，BWDM に適用する拡張を行った．

まず，複数の変数が条件式に含まれていても解析できるように，構文解析部を拡張した．次に，ドメイン分析部を追加し，on ポイント，off ポイント，in ポイント，out ポイントを生成できるようにした．また，SMT ソルバを用いて，充足可能性問題を解けるようにした．さらに，テストケースに，正常系判定値，着目条件式，そして着目変数の情報を含めるために，テストケース生成部を拡張した．

今回の拡張により，既存の BWDM では生成できなかった，複数の変数を含む条件式を持つ VDM++仕様の構文解析と，ドメイン分析テストのためのテストケース生成ができるようになったことを確認した．また，20 行ほどの VDM++仕様に対して，ドメイン分析テストのためのテストケース生成に要した時間を人手と比較検証した結果，18 分程の時間短縮を確認できた．

以上により，本研究で行った拡張によって，既存の BWDM の問題点を解決し，BWDM の有用性が向上したと言える．また，VDM++仕様からドメイン分析テストのためのテストケースを自動生成することが可能となったため，テストケース生成効率が向上したと言える．

以下に，BWDM の今後の課題を示す．

- 複数ドメインのテストケース生成を行う際の重複の排除
- 両辺に変数がある条件式の解析
- ドメイン分析結果の可視化
- BWDM の適用範囲拡大

参考文献

[1] IPA 独立行政法人情報処理推進機構: なぜ形式手法か (online), 入手先 <https://sec.ipa.go.jp/users/seminar/seminar_tokyo_20130918-1.pdf> (2019.5.16)
[2] 立山博基，片山徹郎: VDM++仕様を用いたテストケース自動生成ツール BWDM における if-then-else 式の構造認識手法の提案, ソフトウェアエンジニアリングシンポジウム, pp.130-137 (2017).
[3] Lee Copeland, 雅彦宗: はじめて学ぶソフトウェアのテスト技法., 日経 BP 社 (2005).
[4] JSTQB 技術委員会: ソフトウェアテスト標準用語集 (online), 入手先 <http://jstqb.jp/syllabus.html> (2019.5.20).
[5] Futa Hirakoba, Tetsuro Katayama, et al.: Application of Pairwise Testing to Test Cases by Boundary Value Analysis in BWDM, Proceedings of International Conference on Artificial Life and Robotics, 2019, pp.161-164.
[6] D. Richard Kuhn, Dolores R. Wallace and Albert, M. Gallo: Software fault interactions and implications for software testing, IEEE Transactions on Software Engineering, Vol.30, pp.418-421 (2004).
[7] Nick Battle: GitHub - nickbattle/vdmj (online), 入手先 <https://github.com/nickbattle/vdmj> (2019.5.16).
[8] 宋剛秀，田村直之: SAT ソルバーの最新動向と利用技術 (online), 入手先 <https://jssst-ppl.org/workshop/2017/slides/ppl2017_c4_soh.pdf> (2019.5.16)
[9] GitHub - Z3Prover/z3 (online), 入手先 <https://github.com/Z3Prover/z3> (2019.5.16).
[10] Sun Microsystems: System(Java Platform SE8) (online), 入手先 <https://docs.oracle.com/javase/jp/8/docs/api/java/lang/System.html> (2019.5.17).
[11] 丹野治門，張暁晶: ドメインテスト技法に基づく網羅的なテストデータ自動生成手法の提案, 情報処理学会研究報告, Vol.2014-SE-186, No.6 pp.1-8 (2014).
[12] vdmtools: GitHub - vdmtools/vdmtools (online), 入手先 <https://github.com/vdmtools/vdmtools> (2019.5.17).
[13] The Overture Project (online), 入手先 <http://overturetool.org/> (2019.5.17).