VDM++仕様を用いたデシジョンテーブル自動生成ツール VDTableにおける複合条件式への対応

黄 一諾[1,a)]　片山 徹郎[1,b)]

概要：形式手法は，ソフトウェア開発において，仕様を厳密に記述する手段の 1 つである．一方，デシジョンテーブルはソフトウェアの論理の組み合わせを網羅的に表現するテスト技法の 1 つである．本研究室では，形式仕様記述言語 VDM++で記述した仕様書 (VDM++仕様) から，デシジョンテーブルを自動的に生成するツール VDTable を開発した．しかし，既存の VDTable は，論理演算子を結合した複合条件式を処理する時，複合条件式を個々の単純な条件式に分割せずに，記述した条件をそのまま抽出して出力する．そこで本論文では，複合条件式に対応できるように，VDTable を改良する．改良した VDTable は，10 個の論理演算子を結合した複合条件式を記述した VDM++仕様に対して，1.3 秒程度の時間でデシジョンテーブルを自動生成できた．

キーワード：形式手法， VDM++， デシジョンテーブル， VDTable， 複合条件式

Correspondence to Compound Conditional Expressions in Decision Table Automatic Generation Tool VDTable by Using VDM++ Specification

Abstract: Formal methods, in software development, is one of the means to describe the specification strictly. On the other hand, the decision table is one of the testing techniques to exhaustively represent the logical combination included in software. Therefore, our laboratory developed VDTable (VDM Decision Table). VDTable can automatically generate the decision table, from the specification written in specification description language VDM++ (VDM++ specification). However, when VDTable processes a compound conditional expression combining logical operators, it extracts the described condition as it is without dividing the compound conditional expressions into individual simple conditional expressions, and outputs them. This paper improves VDTable so that it can correspond to compound conditional expressions. The improved VDTable can automatically generate a decision table in about 1.3 seconds from the VDM++ specification which has a compound conditional expressions combining 10 logical operators.

Keywords: Formal method， VDM++， Decision table， VDTable， Compound conditional expression

1. はじめに

近年，システムの大規模化，高機能化に伴い，従来の開発手法では，ソフトウェアの品質を維持できなくなってきている．実際に，システムの不具合が経済や生活に与える影響は大きな社会問題となっている [1] ．このような背景から，ソフトウェアの品質がより重要視されるようになった．

一般に，ソフトウェア開発の上流工程の成果物は多くの不具合を含んでいる．その理由の 1 つとして，仕様に曖昧な記述を含んだまま，次の工程へ移っていることが挙げられる．仕様の曖昧性を除去する手段として，仕様を厳密に記述する形式手法がある [2].

一方，ソフトウェア開発の上流工程における設計検討段階において，ソフトウェアの論理を正確に表現する技法の 1 つに，デシジョンテーブル がある [3]．しかし，デシジョンテーブルの作成には，対象のシステムまたは仕様の記述内容を理解することが必要となる．仕様から条件と動作の

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

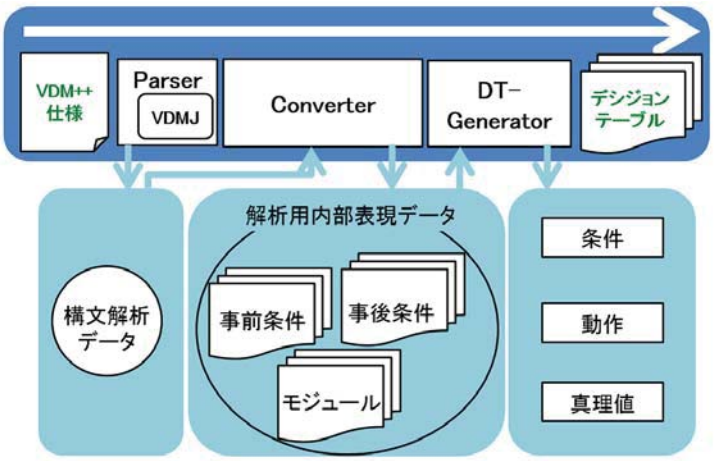

図 1 VDTable の処理の流れ
Fig. 1 The process of VDTable.

抽出を手作業で行う場合，手間と時間がかかり，ミスも起きやすい．

そこで，本研究室では，この手間と時間を削減するために VDTable(VDM Decision Table) を開発した [4][5]．VDTable は，形式仕様記述言語 VDM++(Vienna Development Method)[6][7] で記述した仕様書 (VDM++仕様) から，デシジョンテーブルを自動的に生成するツールである．VDTable を用いることによって，VDM++仕様を用いたデシジョンテーブル設計時の作業効率を高めることができる．

しかし， 既存の VDTable は，VDM++仕様に論理演算子 “and，or または not” で結合した複合条件式が存在する場合，複合条件式を個々の単純な条件式に分割せずに，複合条件式をそのまま抽出して出力する．多くの論理演算子を結合した複合条件式を，人間が読むことは困難である．これは，個々の条件の可能なすべての組み合わせに対する動作の確認が困難だからである．すなわち，複合条件式をそのまま抽出して生成したデシジョンテーブルは，有用性が高いとは言えない．

本論文では，VDM++仕様を用いたデシジョンテーブル自動生成ツール VDTable の有用性の向上を目的として，VDTable を改良する．具体的には，論理演算子 “and，or または not” で結合した複合条件式に対応するために，複合条件式を個々の単純な条件式に分割して，分割した単純な条件式を論理演算子によって論理演算し，その結果に基づいてデシジョンテーブルを生成し，GUI 上に提示する．

第 2 章では，既存の VDTable について説明する．そして、第 3 章では，VDTable の改良について説明する．第 4 章は，改良した VDTable を適用した結果について述べる．第 5 章では，改良した VDTable について考察する．最後に，第 6 章では，本研究のまとめと今後の課題を示す．

表 1 条件と動作の抽出規則
Table 1 The extraction process of conditions and actions.

条件抽出パターン	動作抽出パターン
if 条件 then	then 動作 if
elseif 条件 then	then 動作 elseif
	then 動作 else
	else 動作 if
	else 動作 elseif
	else 動作 else
	else 動作 EOF
cases 条件 ->	-> 動作 cases
	others 動作 EOF
let 条件 1 in 条件 2	name = 動作
pre 条件 post	
pre 条件 EOF	
post 条件 EOF	
	EOF (End Of File)

2. 既存の VDTable

VDTable の処理の流れを，**図 1** に示す．VDTable は Parser，Converter，DT-Generator の 3 つの部から成る．

Parser は，VDMJ[8] を利用して，ユーザの指定した VDM++仕様を構文解析し，抽象構文木を生成する．Parser は，各クラス定義ごとに，定義ブロックの型や引数，定義名や式などの解析情報を保持している抽象構文木をもとに，関数定義群および操作定義群から，Converter で入力として用いる構文解析データを出力する．

Converter は，デシジョンテーブルの生成時に必要となる，条件と動作の抽出，および，真理値の作成を容易にするため，Parser が出力した構文解析データを解析用内部表現データに変換する．解析用内部表現データとは，構文解析データをモジュール単位で分割し，定義本体の式を，構文の最小単位であるトークンごとに改行したデータである．

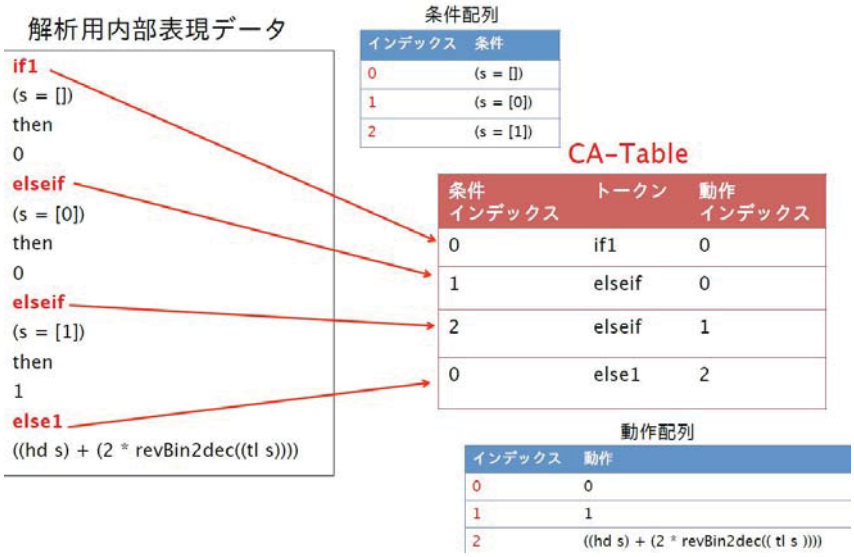

図 2 CA-Table の例
Fig. 2 An example of CA-Table.

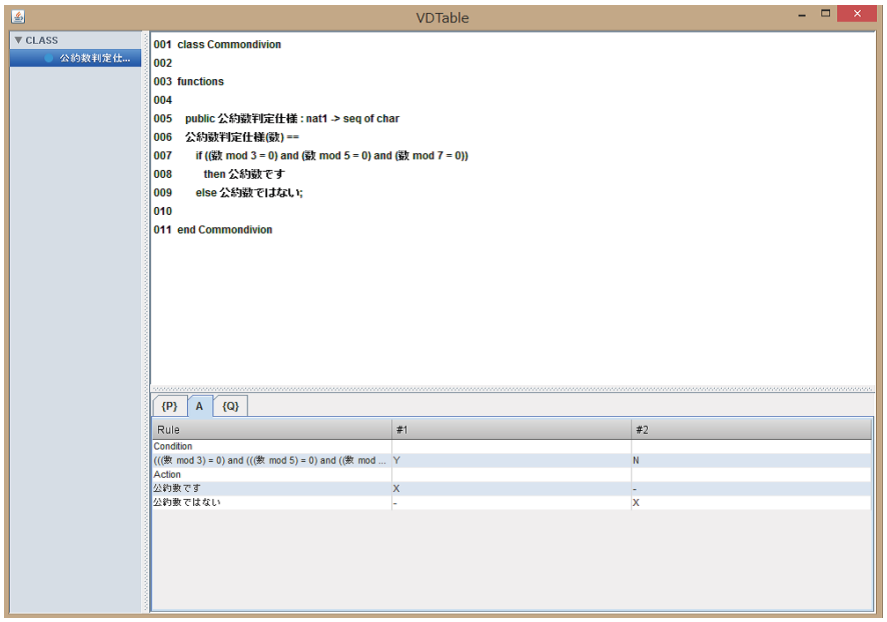

図 3 デシションテーブルの生成例
Fig. 3 Sample of generation of decision table.

DT-Generator は，条件と動作を抽出し真理値を作成することによって，デシジョンテーブルを生成する．**表 1** に， DT-Generator で用いる条件と動作の抽出規則を示す．DT-Generator は，条件と動作の抽出規則を定義している，まず，解析用内部表現データから条件抽出パターンにマッチした条件，および，動作抽出パターンにマッチした動作を抽出し，String 型の配列 (条件配列と動作配列) に，それぞれ格納する．条件と動作を抽出する際に，DT-Generator は CA-Table (Condition Action Table) を作成する．**図 2** に，CA-Table の例を示す．CA-Table とは，条件と動作の対応表である．条件インデックスとトークン，そして，動作インデックスの 3 列の情報を保存する．

次に，DT-Generator は，CA-Table によって真理値を生成する．最後に，DT-Generator は，作成した条件と動作，および，条件と動作の真理値を元に，デシジョンテーブルを生成する．生成したデシジョンテーブルの例を，**図 3** に示す．なお，図 3 は VDTable の出力例であり，生成したデシジョンテーブルはこの下の部分に表示している．

3. VDTable の改良

今回，複合条件式に対応するために，CCE(Clean Compound conditional Experssions) の実装を行った．改良後の VDTable の流れを，**図 4** に示す．CCE の 3 つの機能を，以下に示す．

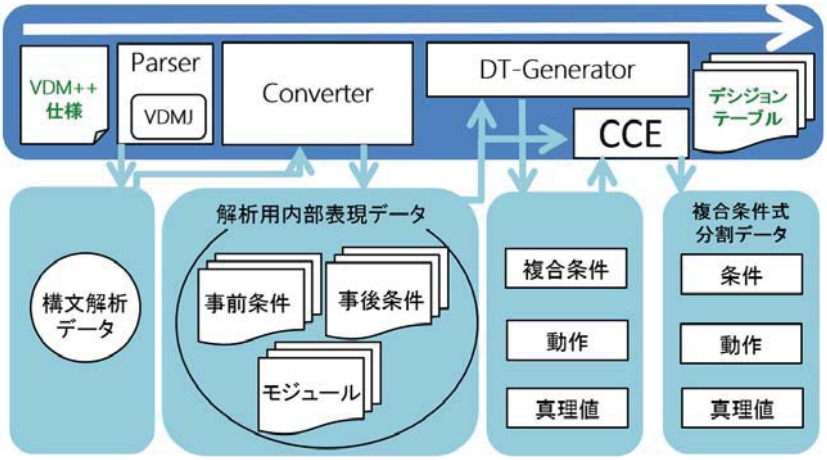

図 4 改良後の VDTable の処理の流れ
Fig. 4 The process of new VDTable.

```
class MyersTriangle
functions
    public 三角形判定 : int * int * int -> bool
    三角形判定 (A,B,C)==
        if (A<B+C and B<A+C and C<A+B)
            then true
            else false;
end MyersTriangle
```

図 5 三角形判定の VDM++仕様
Fig. 5 A VDM++ specification of the Triangle.

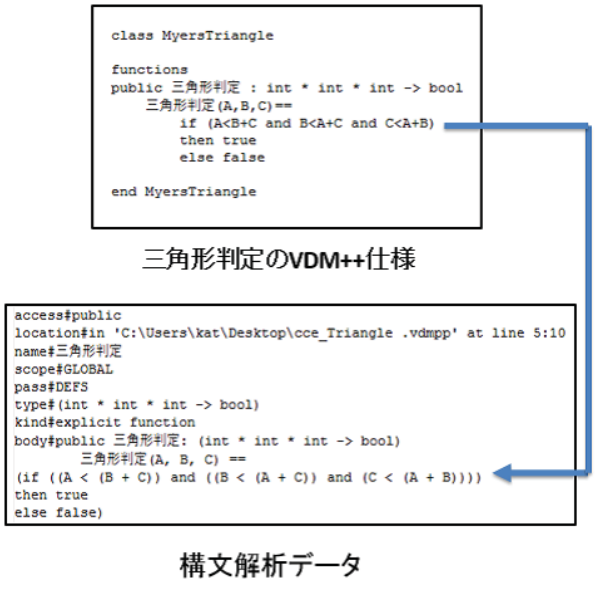

図 6 VDMJ が丸カッコを付ける例
Fig. 6 An example of putting parentheses by VDMJ.

- 複合条件式抽出規則と対応する動作抽出規則の提案
- 複合条件式分割データの生成
- 真理値表の生成手順の提案

3.1 複合条件式抽出規則と対応する動作抽出規則の提案

説明のため，VDM++を用いて記述した三角形判定の仕様を用いる．この記述例を，**図 5** に示す．この仕様は，三角形のそれぞれの辺が他の 2 つの辺の和より小さい関係の時に，三角形と判定する．この判定に複合条件式を用いている．

```
if1
((A < (B + C)) and ((B < (A + C)) and (C < (A + B))))
then
true
else1
false
```

図 7 三角形判定の VDM++仕様の解析用内部表現データ
Fig. 7 The internal expression data for analysis of the Triangle.

```
((A < (B + C)) and ((B < (A + C)) and (C < (A + B))))
true
false
```

図 8 複合条件式と対応する動作の抽出結果
Fig. 8 Result of extraction of compound conditional expression and action.

既存の VDTable の Parser で構文解析する時に，論理演算子“and，or または not”を結合した複合条件式が VDM++仕様に存在する場合，VDMJ は自動的に複合条件式を構成する個々の条件式に対して丸カッコを付ける．VDMJ が複合条件式を分割して丸カッコを付ける例を，**図 6** に示す．また，Converter が出力した解析用内部表現データを，**図 7** に示す．

CCE では，デシジョンテーブルの生成時に必要となる，条件と動作の抽出，および，真理値の作成を容易にするため，Converter が出力した解析用内部表現データと DT-Generator で生成した動作を利用する．

複合条件式の抽出を実現するために，複合条件式抽出規則と対応する動作抽出規則を新たに提案する．

- 解析用内部表現データから，論理演算子“and， or または not”に一致した場合，その行を抽出する．

表 2 単純条件式解析規則
Table 2 Analysis rule for simple conditional expression.

処理パターン	単純条件式解析規則
(条件 1 “and, or または not” 条件 2 “and, or または not” ... ... “and, or または not” 条件 N)	最初の “(” から “and, or または not” まで “and, or または not” から “and, or または not” まで “and, or または not” から 最後の “)” まで

```
((A < (B + C))
and
((B < (A + C))
and
(C < (A + B))))
true
false
```

図 9 複合条件式分割データ
Fig. 9 Compound condition expression division data.

表 3 真理値の作成例
Table 3 The examples of generating truth values.

((A < (B + C))	1	0	1	0	1	0	1	0
((B < (A + C))	1	1	0	0	1	1	0	0
(C < (A + B))))	1	1	1	1	0	0	0	0
true	1	0	0	0	0	0	0	0
false	0	1	1	1	1	1	1	1

- 表 1 に示した，既存の動作抽出規則によって，動作を抽出する．

抽出したデータは，String 型で保存する．図 7 から抽出した結果を，**図 8** に示す．

3.2 複合条件式分割データの生成

デシションテーブルを生成するためには，複合条件式を解析して，個々の単純な条件式に分割する必要がある．そこで，単純条件式解析規則を新たに提案する．**表 2** に，本研究で提案する単純条件式解析規則を示す．

単純条件式解析規則は，図 8 から抽出した複合条件式からトークン “and , or または not” この 3 つ演算子に一致した場合，「最初の “(” から “and, or または not” まで」,「“and, or または not” から “and, or または not” まで」または「“and, or または not” から 最後の “)” まで」を条件とし抽出する．

そして，真理値を計算するために，各論理演算子を抽出する．これらは，複合条件式分割データとして String 型で保存する．CCE で生成した複合条件式分割データの例を，**図 9** に示す．

3.3 真理値表の生成手順の提案

真理値表の生成手順を提案する．真理値表の生成手順を，以下に示す．複合条件式分割データの第一行から (2*演算子の数+2) 行までの奇数行を抽出して，条件として格納する．複合条件式分割データの (2*演算子の数+2) 行から最後の一行まで抽出して，動作として格納する．最後に，複合条件式に関する真理値表を作成する真理値表を生成した結果を，**表 3** に示す．

GUI に表示する時，条件部の提示では，真理値表の値が 1 の場合は，条件が真とし（“Y” を出力する），0 の場合は，条件が偽とする（“N” を出力する）．動作部の提示では，論理演算によって，出力が 1 の場合は，動作が成立する (“X” を出力する) とし，出力が 0 の場合は，動作が成立しない (“-” を出力する) とする．デシジョンテーブルを生成した結果を，**図 10** に示す．

Rule	#1	#2	#3	#4	#5	#6	#7	#8
Condition								
((A < (B + C))	Y	N	Y	N	Y	N	Y	N
((B < (A + C))	Y	Y	N	N	Y	Y	N	N
(C < (A + B))))	Y	Y	Y	Y	N	N	N	N
Action								
true	X	-	-	-	-	-	-	-
false	-	X	X	X	X	X	X	X

図 10 三角形判定デシションテーブルの生成結果
Fig. 10 Result of generation of decision table for the Triangle.

```
class Sample
functions
    public 日数表示 : nat1 * nat1 -> nat1
    日数表示 (年, 月) ==
        if (月=2)then
            うるう年判定仕様 (年)
        elseif ((月=4) or (月=6) or (月=9) or (月=11))then
            30
        else
            31
    pre 月 <= 12
    post 31 <= RESULT;
end Sample
```

図 11 日数表示の VDM++仕様
Fig. 11 A VDM++ specification to display the number of days.

4. 適用例

今回改良した VDTable が正しく動作することを，適用例を用いて確認する．適用例に用いる VDM++仕様を，**図 11** に示す．この仕様は，VDM++を用いて記述した，日数表示である．入力した年の数と月の数によって，その月の最大日数を表示する．この日数表示は，複合条件式を

Rule	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14	#15	#16
Condition																
((月 = 4)	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
((月 = 6)	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N
((月 = 9)	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
(月 = 11))))	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
Action																
うるう年判定仕様(年)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X
30	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	-
31	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X

図 12 日数表示のデシションテーブルの生成結果
Fig. 12 Result of generation of decision table of the VDM++ specification to display the number of days.

```
class Commondivion
functions
   public 公約数判定仕様 : nat1 -> seq of char
   公約数判定仕様 (数) ==
         if(数 mod 1 = 0) and (数 mod 2 = 0)
         and (数 mod 3 = 0)and(数 mod 4 = 0)
         and (数 mod 5 = 0) and (数 mod 6 = 0)
         and(数 mod 7 = 0) and (数 mod 8 = 0)
         and (数 mod 9 = 0)and(数 mod 10 = 0)
         and (数 mod 11 = 0)
      then
         公約数です
      else
     公約数ではない;
end Commondivion
```

図 13 公約数判定の VDM++仕様
Fig. 13 A VDM++ specification of the common division.

用いている．VDTable が自動生成したデシジョンテーブルを，**図 12** に示す．図 12 のデシジョンテーブルの真理値表より，論理演算によって，デシジョンテーブルが条件 1 と条件 2 と条件 3 と条件 4 のいずれかが真の場合 (“Y” を格納した) は，動作が成立し (“X” を格納した)，それ以外の場合は，動作が成立していない (“-” を格納する)．このことから，真理値表を正しく生成していることが分かる．

以上から，今回改良した VDTable に，日数表示の VDM++仕様を適用した結果，改良した VDTable が正しく動作することを確認した．

5. 考察

本研究では，VDM++仕様を用いたデシジョンテーブル自動生成ツール VDTable の有用性の向上を目的として，VDTable を改良した．

本提案手法を実現するため，我々は CCE を実装した．実装した CCE では，VDM++仕様で記述した複合条件式を個々の単純な条件式に分割する．分割した単純な条件式によるデシジョンテーブルを生成して，GUI 上に提示する．

本章では，提案手法について考察する．

5.1 実用性の考察

改良した VDTable の実用性を，以下の 3 つの複合条件式を含んでいる VDM++仕様を用いて確認する．

(1) 三角形判定
(2) 日数表示
(3) 公約数判定

具体的には，各 VDM++仕様に対し，改良した VDTable と，人手による複合条件式を分割してデシジョンテーブルを作成すること，それぞれの完成までに要した時間を計測し，比較する実験を行う．

なお，三角形判定の仕様 は 3 章で用いた VDM++仕様を用いる．日数表示は，4 章で適用例に用いた VDM++仕様を用いる．公約数判定では，**図 13** に示す VDM++仕様を用いる．

実験の結果を，**表 4** に示す．改良した VDTable が，デシジョンテーブルを生成する時間は，10 個の論理演算子を結合した複合条件式を記述した公約数判定の VDM++仕様に対して，1.3 秒程度の時間であり，人手で行うより速い．

すなわち，改良した VDTable の実用性を確認できた．

5.2 関連研究

デシジョンテーブルの生成を支援するツールとして，CEGTest[9] がある．CEGTest は，ユーザが作成した原因結果グラフから，デシジョンテーブルを自動生成する．原因結果グラフの作成は，ユーザが仕様書から人手で作成しなければならない．そのため，デシジョンテーブルを作成する際に，形式仕様記述内容の理解，および，条件と動作の抽出に手間と時間がかかり，入力ミスも起きやすい．これに対して，VDTable は，人手で仕様書から原因結果グラフを作成する必要がなく，仕様書からデシジョンテーブルを自動生成できる．

他に，形式仕様を入力としたテストツールとして TOBIAS[10] が存在する．TOBIAS は、テストパターンを入力として，テストケースを自動生成する．テストパターンとは，ツールが出力するテストケースの組合せ規則を，

表 4 計測時間 (sec)
Table 4 Measurement time(sec).

VDM++仕様	人手による複合条件式分割したデシジョンテーブルの作成	改良した VDTable
三角形判定	128	0.523
日数表示	237	0.563
公約数判定	982	1.342

ユーザが正規表現を用いて定義したものである．しかし，このツールは，デシジョンテーブルの生成には対応していない．これに対して，VDTable では，形式仕様の記述内容を理解する必要なく，VDM++で記述した仕様をツールの入力として指定することにより，デシジョンテーブルを自動で得ることができる．

6. おわりに

本研究では，VDM++仕様を用いたデシジョンテーブル自動生成ツール VDTable の有用性の向上を目的として，VDTable を改良した．

具体的には，VDTable に CCE を新たに実装した．実装した CCE では，VDM++形式仕様で記述した複合条件式を個々の単純な条件式に分割して，自動生成したデシジョンテーブルを GUI 上に提示する．改良した VDTable に，複合条件式を含んだ VDM++仕様を適用して，デシジョンテーブルを正しく生成できることを確認した．すなわち，改良した VDTable が VDM++仕様から条件と動作を正しく抽出することを確認した．また，デシジョンテーブルにおいて，条件と動作の真理値を正しく生成していることを確認した．さらに，改良した VDTable は，10 個の論理演算子を結合した複合条件式を記述した VDM++仕様に対して，1.3 秒程度の時間でデシジョンテーブルを自動生成できた．

以上のことから，今回の改良によって，VDTable の有用性が向上したと言える．

以下に，今後の課題を挙げる．

- 可読性の向上
  すべての真理値の組み合わせを生成した場合，適用する VDM++仕様の条件の数によっては，生成するデシジョンテーブルが指数関数的に大きくなり，デシジョンテーブルの可読性が下がる．今後は，デシジョンテーブルを生成する時に，すべての真理値の組み合わせを生成するかどうかをユーザに問い合せる機能，もしくは，ユーザが選択した部分だけを表示する機能の追加を考えている．
- デシジョンテーブル自動生成ツールの実用性向上
  現状，VDTable は，for や while などの各構文および再帰呼出しに対応しておらず，適用可能な形式仕様の範囲に制限がある．そのため，形式仕様を用いた条件および動作の抽出は，if-then-else 構文，cases 構文と let in 構文，および，事前条件式と事後条件式のみを対象としている．今後は，適用可能な形式仕様の範囲を広げるため，実装したツールの機能拡張を行う必要がある．
- 大規模なシステムを対象とした形式仕様への適用
  現在までに，VDTable に適用した仕様は，条件の真理値の組み合わせが最大で 2^{13}(8192) 通りのものである．この仕様に対して，VDTable は，0.7 秒程度の時間でデシジョンテーブルを自動生成できた．より大規模で複雑な仕様に対し，VDTable がどの程度の時間でデシジョンテーブルを生成できるのか，時間を計測し，評価する必要がある．
- テストデータの自動生成
  形式仕様を用いたテスト設計時の作業効率をより向上するため，ツールが自動生成するデシジョンテーブルに基づいたテストデータを，形式仕様から自動生成することを考えている．

参考文献

[1] 日経コンピュータ. システム障害はなぜ二度起きたか みずほ, 12 年の教訓. 日経 BP 社, 2011.
[2] 中島震. ソフトウェア工学の道具としての形式手法. Technical Report NII-2007-007J, National Institute of Standards and Technology, 2007.
[3] 吉村鉄太郎. Decision Table. 情報処理学会誌, Vol.5, No.5, pp.268-276, 1964.
[4] 西川拳太, 他. 形式仕様を用いたデシジョンテーブル生成手法の提案. ソフトウェアエンジニアリングシンポジウム 2014 論文集. pp39-44, 2014.
[5] Y. Huang, T. Katayama, Y. Kita, H. Yamaba, and N. Okazaki. *Improvement of Decision Table Automatic Generation Tool VDTable for let in Statement*, Proc. 2017 Int'l Conf. on Artificial Life and Robotics (ICAROB 2017), pp.271-274, 2017.
[6] J. Fitzgerald, Peter G. Larsen, P. Mukherjee, N. Plat, and M. Verhoef. Vali-dated Designs for Object-oriented Systems. Springer-Verlag, 2005.
[7] 栗田太郎, 荒木啓二郎. モデル規範型形式手法 VDM と仕様記述言語 VDM++: 高信頼性システムの開発に向けて. 日本信頼性学会, Vol. 31, No. 6, pp. 394-403, 2009.
[8] VDMJ. http://sourceforge.net/projects/vdmj/ (accessed May 17, 2017).
[9] CEGTest. http://softest.jp/tools/CEGTest/ (accessed May 17, 2017).
[10] Y. Ledru, L. du Bousquet, O. Maury, and P. Bontron. *Filtering TOBIAS combinatorial test suites*, Fundamental Approaches to Software Engineering, pp.281-294, 2004.