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let in構文への対応

黄一諾 片山徹郎
宮崎大学大学院 工学研究科 工学専攻

1 はじめに

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。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、我々の研究室では、この手間と時間を削減するために VDTable(VDM Decision Table) を提案した [1]。VDTable 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できる。しかし、既存の VDTable は、対応していない VDM++構文が多く存在しているため、適用範囲が狭い。

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 の適用範囲を拡大することを目的として、VDTable を改良する。具体的には、既存の VDTable が対応していない VDM++構文の 1 つである let in 文に対応できるように、VDTable を改良する。

2 既存の VDTable の紹介

VDTable の処理の流れを、図 1 に示す。VDTable は Paser、Converter、DT-Generator の 3 つの部から成る。

Paser は、VDMJ 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 [2]。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。

DT-Generator は、条件と動作の抽出規則を定義しており、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。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。

3 VDTable の改良

今回、let in 文への対応として、Converter と DT-Generator の拡張を行った。具体的には、以下の 3 つの項目を行った。

- let in 文の構文解析データから解析用内部表現データに変換するためのデータ変換規則の提案
- let in 文の条件抽出規則と動作抽出規則の提案
- let in 文の真理値表の生成規則の提案

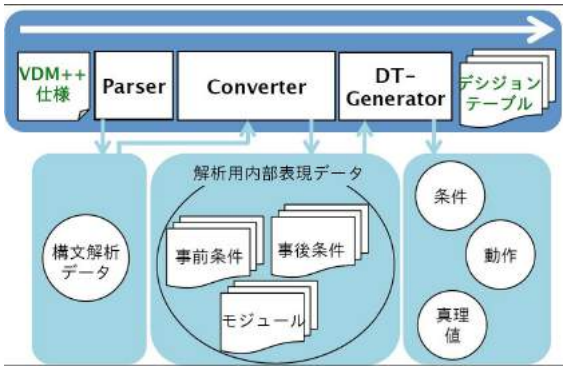

図 1: VDTable の処理の流れ

表 1: let in 文の条件抽出規則と動作抽出規則

条件抽出パターン	動作抽出パターン
let 条件 1 in 条件 2	name 動作

3.1 let in 文の構文解析データから解析用内部表現データに変換するためのデータ変換規則の提案

let in 文の構文解析には、既存の VDTable の Parser をそのまま利用する。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、VDTable の Converter では、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。Converter おいて、let in 文の構文解析データを解析用内部表現データに変換する。この実現のために、let in 文のデータ変換規則を新たに提案する。

- let in 文のデータ変換は、構文解析データからトークン”let”に一致した場合、「let 直後から in の前まで」、「in」、「in の直後から最後まで」の 3 項目を抽出する。
- name の部分を 1 項目として抽出する。

これらは、let in 文の解析用内部表現データとして String 型で保存する。

3.2 let in 文の条件抽出規則と動作抽出規則の提案

let in 文の条件抽出規則と動作抽出規則を新たに提案する。表 1 に、本研究で提案する let in 文の条件抽出規則と動作抽出規則を示す。解析用内部表現データにおいて、”in”の前の行を条件 1 とし、”in”の次の行を条件 2 とし、条件抽出パターンとする。また、”name”の後の部分を動作抽出パターンとする。

3.3 let in 文の真理値表の生成規則の提案

let in 文の真理値表の生成規則を新たに提案する。提案した真理値表の生成手法は、以下の手順である。

1. 真理値表の値を格納する配列を作成
2. 条件を抽出
3. 動作を抽出

表 2: 真理値を格納する String 型二次元配列の作成例

条件 1	-	-
条件 2	-	-
動作	-	-

図 2: 要求額の現金を払い出す仕様の構文解析データ

4. 配列に真理値の値を格納

まず、真理値表の値を格納するため、String 型の二次元配列を用意する。縦の第一列の要素に条件と動作を記述し、横の要素に条件と動作の真理値情報を記述する。配列の作成例を、表 2 に示す。

次に、条件抽出規則に従って、各条件を抽出する。そして、動作抽出規則に従って、name の部分を動作として抽出する。

最後に、真理値の値を格納する。今回提案する真理値表の生成手法では、let in 文を「let 条件 1 in 条件 2」と記述した時に、「条件 1 と条件 2 の両方が真の場合は ("Y"を格納する)、動作が成立する ("X"を格納する)」とし、「条件 1 が真で ("Y"を格納する)、条件 2 が偽の場合は ("N"を格納する)、動作が成立しない ("-"を格納する)」とする。この理由は、VDM 仕様における let in 文の動きは、条件 1 の部分を条件 2 に代入となるからである。すなわち、条件 2 に対して、条件 1 は常に真となる。よって、let in 文の真理値表の生成の際には、条件 1 が常に真であるとし、条件 2 の真偽状況のみによって、動作の成立を判断する。

DT-Generator は、抽出した条件と動作、および、生成した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。

4 適用例

適用例に用いる VDM++仕様は、ATM の預金の引き出しについて VDM++を用いて記述した仕様の一部であり、let in 文を利用して、ATM のユーザに対して要求額の現金を払い出す仕様の部分である。この仕様を Parser で構文解析した結果を、図 2 に示す。

図 2 の構文解析データから、仕様中の let in 文の内容を解析用内部表現データに変換した結果を、図 3 に示す。条件 1 と条件 2、また、動作を含んだ let in 文の内容を String 型で保存している。let in 文を解析用内部表現データに正しく変換していることが分かる。

図 4 に、適用例の VDM++仕様を、改良した VDTable に適用した結果を示す。図 4 のデシジョンテーブルより、図 3 から、トークン"in"の前の行を条件 1、"in"の次の行を条件 2 として正しく抽出することが分かる。また、図 4 から、"name"の後の部分を動作として正しく抽出することが分かる。

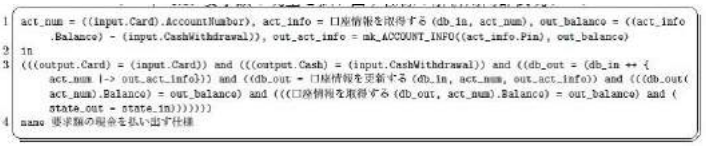

図 3: 要求額の現金を払い出す仕様の解析用内部表現データ

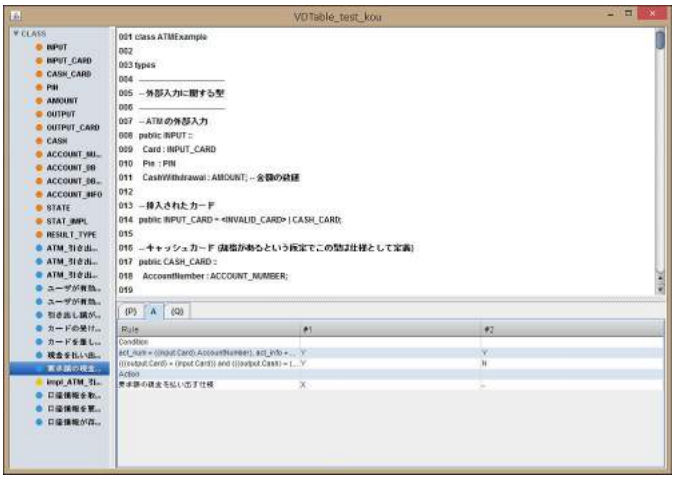

図 4: 要求額の現金を払い出す仕様を改良した VDTable に適用した結果

最後に、図 4 のデシジョンテーブルの真理値表より、デシジョンテーブルが条件 1 と条件 2 の両方が真の場合は ("Y"を格納した)、動作が成立し ("X"を格納した)、条件 1 が真で ("Y"を格納した)、条件 2 が偽の場合は ("N"を格納した)、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。

5 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 の適用範囲を拡大することを目的として、VDTable を改良した。具体的には、let in 文を構文解析データに変換するためのデータ変換規則の提案、let in 文の条件と動作の抽出規則の提案、および、let in 文の真理値の生成規則の提案を行った。

改良した VDTable を、let in 文を含んだ ATM の預金の引き出しの VDM++仕様に適用して、デシジョンテーブルを正しく生成することを確認した。VDTable が、let in 文を含む VDM++仕様から条件と動作を正しく抽出していること、また、真理値表を適切に生成することが可能となった。以上のことから、今回改良した VDTable の適用範囲が拡大し、実用性が向上したと言える。

以下に、今後の課題を示す。

- 適用範囲の拡張
- 複合条件式への対処
- 可読性の向上
- 複数の構文を含む関数への対応

参考文献

[1] 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).

[2] VDMJ. http://sourceforge.net/projects/vdmj/.