機械学習を用いて自然言語仕様書から生成した分類リストを用いた VDM++仕様書生成アプローチの提案

執行　泰弘 a)・片山　徹郎 b)

Proposal of an Approach to Generate VDM++ Specification from Classfied Lists of the Natural Language Specification by Machine Learning

Yasuhiro SHIGYO, Tetsuro KATAYAMA

Abstract

Specifications are generally written in natural language. Natural language contains ambiguity. As a method of writing a specification without ambiguity, VDM which is a formal method exists. Because it is difficult to write specification languages such as VDM++ because they have strict grammars data types and system invariants that are not found in natural language specifications. This study attempts to generate automatically a VDM++ specification from the natural language specification by using machine learning. For automatic generation of VDM++, it is necessary to extract predicates corresponding to the function names and nouns corresponding to variable names from the natural language specification. However,it is difficult to generate a VDM++ specification by using only the extracted nouns and predicates. This paper proposes an approach to generate automatically a VDM++ specification from extracted words list. An identifier is generated from the extracted words, and the VDM++ specification can be generated by converting this identifier into a VDM++ grammar.

Keywords:　Natural language specification, VDM++, Automatic generation, Formal method

1.　はじめに

ソフトウェアのバグが社会にもたらす影響は甚大なものとなっている [1,2]。

ソフトウェアにバグが混入する原因の 1 つとして、上流工程のソフトウェア設計段階において、自然言語を用いることが挙げられる。しかしながら、自然言語は曖昧さを含んでいる。そのため、プログラマが仕様書上の表記を、誤解してしまう可能性がある [3]。プログラマが、本来の仕様書の意図とは異なる実装を行った結果、ソフトウェアにバグが混入してしまう。

この問題を解決する手法の 1 つとして、形式手法(Formal Method)を用いた上流工程でのソフトウェア設計が挙げられる [4]。形式手法を用いた開発では、数理論理学を基盤とした形式仕様記述言語(Formal Specification Language)により、開発対象が持つ特性を仕様として記述する。形式仕様記述言語は数理論理学を基にしているため、自然言語を用いた設計と異なり、定理証明や機械的な検査を用いて、記述した内容が正しいことを数学的に証明することが可能である [5]。したがって、自然言語の持つ曖昧さを排除した、厳密な設計が可能となる。開発現場向けのライトウェアな形式手法として、VDM(Vienna Development Method)が存在する [6]。

また、オブジェクト指向に基づいたモデル化を扱えるように文法を変更した VDM++も存在する [6]。

しかし、VDM++のような形式仕様記述言語は厳密な文法を持ち、自然言語仕様書にはないデータ型やシステムの不変条件などを書くため、記述が困難である。

そこで本研究では、機械学習を用いて、自然言語の仕様書から VDM++仕様書を自動生成することを試みる。

VDM++仕様書の自動生成には、変数名に該当する名詞や関数名に該当する述語を自然言語仕様書から抽出する必要がある。しかし、抽出した名詞や述語を VDM++の文法に合わせて利用することは困難である。これは、抽出した名詞や述語だけでは、厳密な文法を持つ VDM++の構文に対応して仕様書を記述することが困難なためである。

そこで本論文では、機械学習を用いて名詞や述語を VDM++の文法に基づいて分類した、分類リストから VDM++仕様書を自動生成するアプローチを提案する。提案するアプローチでは、分類した語群から識別子を生成し、この識別子を VDM++の構文に従って変換することで、VDM++仕様書を自動生成する。

2.　VDM

形式手法の 1 つに VDM(Vienna Development Method)が

a)　工学専攻機械・情報系コース大学院生

b)　情報システム工学科教授

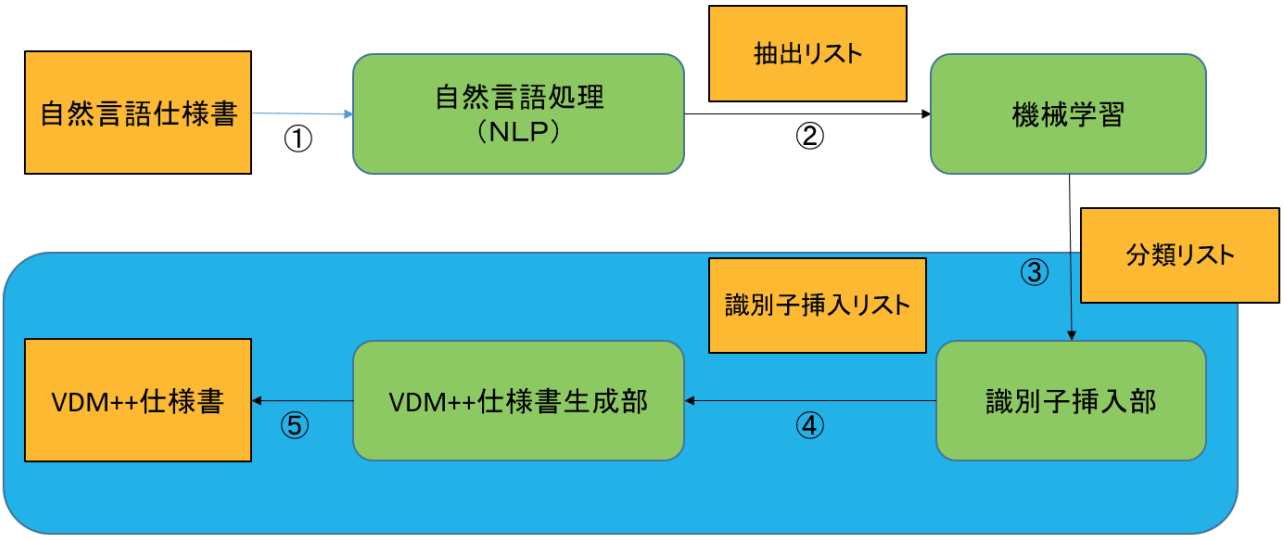


図 1. 提案するアプローチの全体の流れ.

表 1.VDM++の定義.

VDM++のキーワード	要素数	構文
values	2	$1^{st} = 2^{nd}$;
types	2	public $1^{st} = 2^{nd}$;
instance variables	3	1^{st}: 2^{nd}:= 3^{rd};
operations	5 以上	public 1^{st}: 2^{nd} ==>3^{rd}; pre 4^{th}; post 5^{th};

ある [6]。VDM は、1970 年代に IBM のウィーン研究所にて PL/I コンパイラの正しさを検証するために形式手法として開発された。VDM++は、VDM-SL を基にオブジェクト指向拡張した言語であり、現在 VDM の中では主流である [5]。本研究では VDM++仕様書を自動生成する。VDM++は、VDMTools[7]や VDMJ[8]などの支援ツールが揃っており、他の形式手法に比べ仕様の検証がしやすい。

VDM++の定義について説明する。本論文で対象としている VDM++の定義を、表 1 に示す。VDM++のキーワードは、要素数と構文を持つ。構文は、VDM++における記述方法であり、構文内の数字に各要素が対応する。

3. 提案するアプローチ

提案するアプローチは、自然言語仕様書から機械学習を用いて分類した分類リストを基に、VDM++仕様書を自動生成する。図 1 に、提案するアプローチの全体の流れを示す。

1. 自然言語仕様書を基に、機械学習で用いる抽出リストを生成する。抽出リストは、3. 1. 1 節で説明する。
2. 抽出リストを基に、機械学習を用いて分類リストを生成する。分類リストは、3.1.2 節で説明する。
3. 分類リストに識別子を挿入し、識別子挿入リストを生成する。識別子の挿入方法については、3.2.1 節で説明する。
4. 識別子挿入リストから識別子を読み込む。
5. 読み込んだ識別子に応じて、分類リストの要素を VDM++仕様の構文に従って変換することによって、VDM++仕様書を自動生成する。

本論文では、全体のアプローチの内 3~5 について提案する（図 1 下部の青枠）。すなわち、全体のアプローチの 1~2 に相当する、自然言語仕様書から機械学習を用いて生成する分類リストが存在していると仮定する。

3.1 データ構造

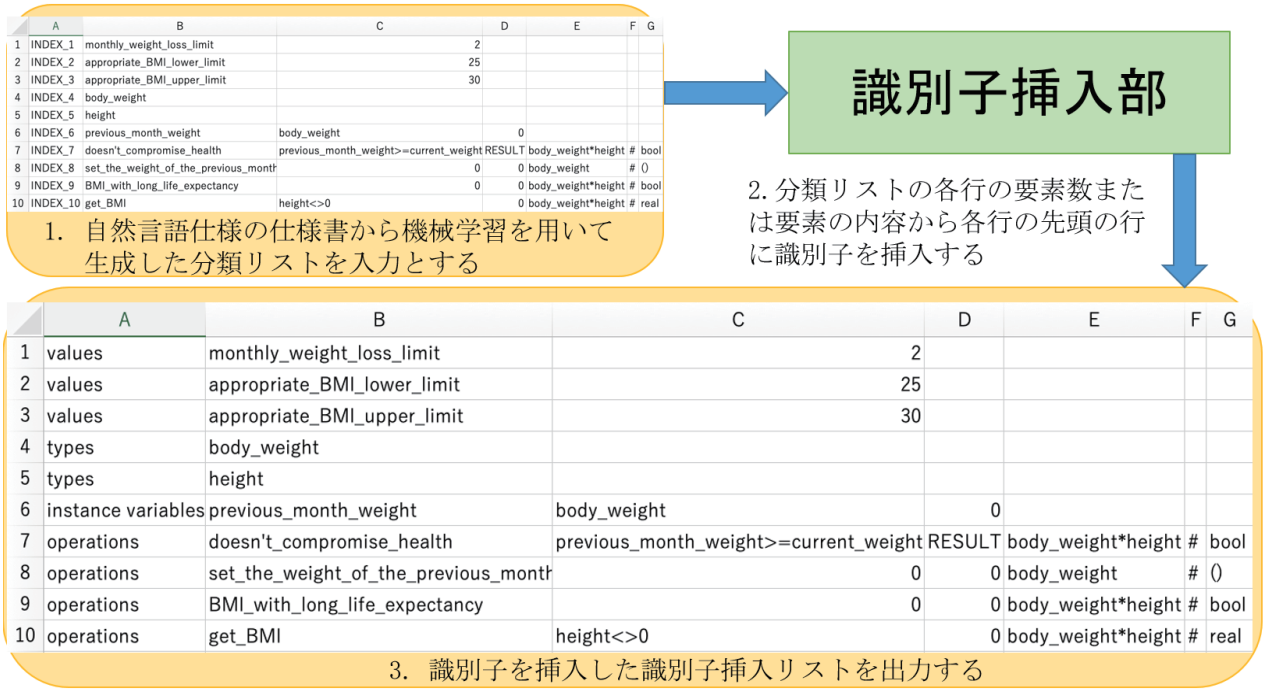
提案するアプローチは、分類リストと識別子、抽出リストの 3 つのデータ構造を用いる。3 つのデータ構造を、それぞれ以下で定義する。

3.1.1 抽出リスト

抽出リストは、自然言語仕様書から文および文節を抽出し、抽出した文および文節とそれらから得られる語の修飾関係から成るリストである。修飾関係は、自然言語仕様書に対して形態素解析を行い、形態素解析から得られる関係を基に抽出リストに追加する。抽出リストは、機械学習の入力として用いる。

3.1.2 分類リスト

分類リストは、1 つの変数または関数についての要素を行ごとにまとめたリストである。図 2 の左上に、分類リストの例を示す。分類リストの各列について説明する。1 列目（列 A）は一時的な ID であり、後に識別子を挿入する。



	A	B	C	D	E	F	G
1	values	monthly_weight_loss_limit	2				
2	values	appropriate_BMI_lower_limit	25				
3	values	appropriate_BMI_upper_limit	30				
4	types	body_weight					
5	types	height					
6	instance variables	previous_month_weight	body_weight	0			
7	operations	doesn't_compromise_health	previous_month_weight>=current_weight	RESULT	body_weight*height	#	bool
8	operations	set_the_weight_of_the_previous_month	0	0	body_weight	#	()
9	operations	BMI_with_long_life_expectancy	0	0	body_weight*height	#	bool
10	operations	get_BMI	height<>0	0	body_weight*height	#	real

図 2.　識別子挿入アルゴリズムの流れ.

2 列目 (列 B) は VDM++仕様書に記述する変数名または関数名である。3 列目 (列 C) 以降は、以下に示す対象によって構成する要素が異なる。

- 変数の場合、列 C の要素は実数値から成る。
- インスタンス変数の場合、列 C 以降は型名、初期値から成る。初期値は存在しない場合がある。
- 関数の場合、列 C 以降は事前条件，事後条件、関数の引数、区切り文字「#」、返り値から成る。

3.1.3 識別子

識別子は、分類リスト内の要素の数および要素の内容に基づいて生成する。生成した識別子は、3.2.1 節に示す方法で分類リストに挿入する。図 2 の下に、識別子を挿入したリストの例を示す。識別子は、表 1 の VDM++のキーワードに対応する。

分類リストの各識別子の挿入条件は、次の通りである。

- キーワード「values」は、分類リストの 2 番目以降の列の要素数が 2 であり、かつ、3 番目の要素が値である場合である。
- キーワード「types」は、分類リストの 2 番目以降の列の要素数が 2 であり、かつ、3 番目の要素が型定義である場合、もしくは、要素の数が 1 であり、かつ、3 列目の要素は実数である場合である。
- キーワード「instance variables」は、分類リストの 2 番目以降の列の要素数が 3 である場合である。
- キーワード「operations」は、分類リストの 2 番目以降の列の要素数が 5 である場合である。

3.2 提案アルゴリズム

本論文で提案する VDM++仕様書を生成するアルゴリズムは、識別子挿入アルゴリズムと VDM++仕様書変換アルゴリズムの 2 つのアルゴリズムから成る。

3.2.1 識別子挿入アルゴリズム

識別子挿入アルゴリズムの流れを、図 2 に示す。 識別子挿入アルゴリズムは、3.1.3 節で説明した挿入条件に従って、分類リストの 1 列目 (列 A) の一時的な ID に、分類リストの各行に対応した識別子を挿入する。識別子を挿入した分類リストを識別子挿入リストと呼ぶ。

3.2.2 VDM++仕様書変換アルゴリズム

VDM++仕様書変換アルゴリズムの流れを、図 3 に示す。VDM++仕様書変換アルゴリズムは、識別子挿入リストに基づいて VDM++仕様を記述する。

VDM++仕様書の記述方法について説明する。まず、生成する VDM++仕様書の初期ファイルとなる仕様書を事前に準備する。この初期ファイルを、図 4 に示す。これは、VDMTool Box[7]において、クラス名を入力すると初期に生成される VDM++仕様書を参考にして作成した。図 4 における VDM++仕様書の各行について、以下に説明する。なお、図 4 では、クラス名に「template」を入力として渡したものを表示している。

- VDM++仕様の最初の行に、「class <class-name>」を記述する。ここで、「<class-name>」には、自然言語仕様書から取得したクラス名を記述する。
- 「class <class-name>」の以降の行に、VDM++の各キーワードを順に記述する。

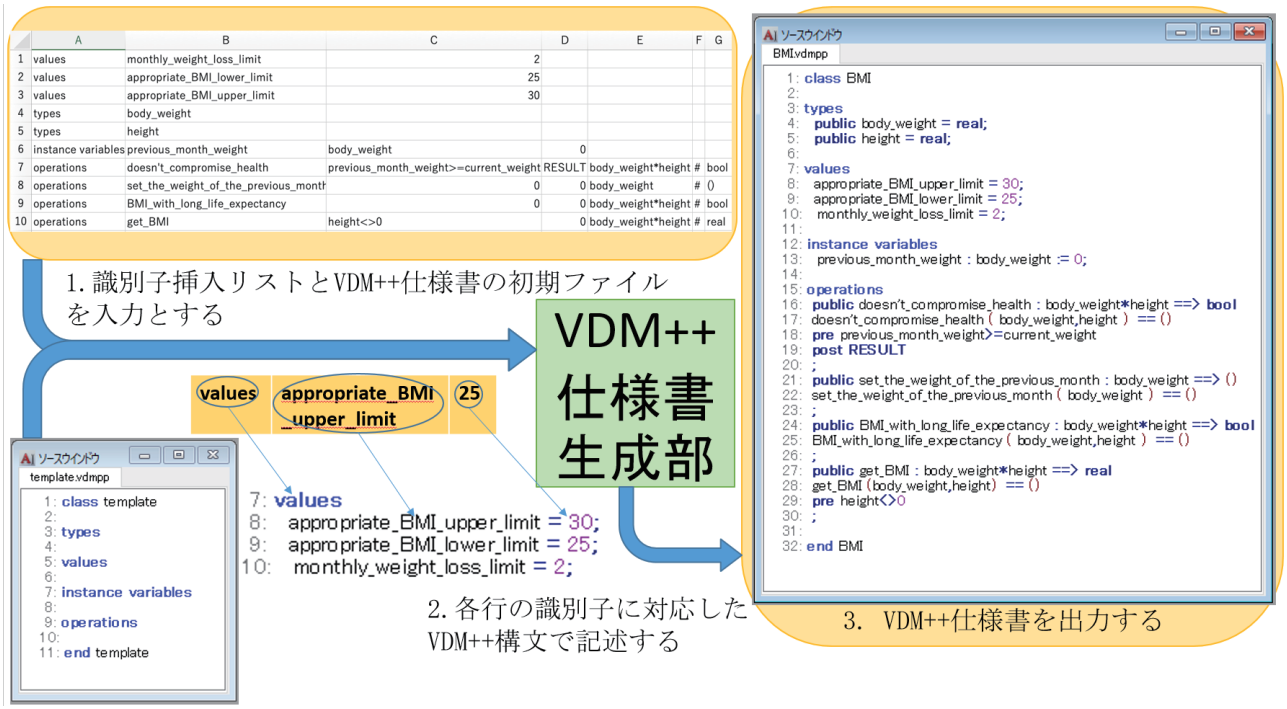


図 3. VDM++仕様書変換アルゴリズムの流れ.

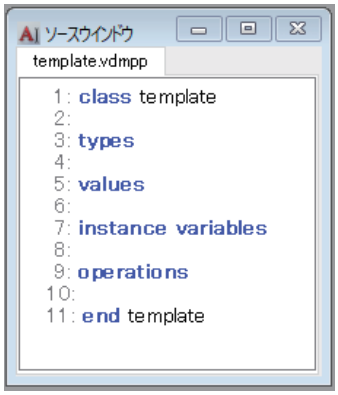


図 4. VDM++仕様書の初期ファイル.

- VDM++仕様書の最後の行に、「end <class-name>」を記述する。

次に、表 1 に示した構文に従って、識別子挿入リストの要素を、VDM++の各キーワードの箇所に記述する。識別子挿入リストの 2 列目以降の各要素を、キーワードの構文に従って各キーワードに続く行に配置する。

例として、図 2 の識別子挿入リストの 3 行目を用いて説明する。まず、1 列目(列 A)の識別子は「values」を読み取る。次に表 1 に示すキーワード「values」に対応する構文「1st = 2nd;」に従って識別子挿入リストの 2 列目(列 B)の「appropriate_BMI_upper_limit」、および 3 列目(列 C)の「30」を配置する。最後に、VDM++仕様書の「values」の次の行に識別子に対応した文「appropriate_BMI_upper = 30;」を記述する。

4. 適用例および考察

本論文では、提案するアプローチによって VDM++仕様書が生成できることを確認するために、挿入器と変換器の 2 つのツールを試作した。挿入器は識別子挿入アルゴリズムを、変換器は VDM++仕様書変換アルゴリズムを、それぞれ実装したものである。分類リストを挿入器に適用し、識別子挿入リストを変換器に適用した。挿入器の入力を図 2 の左上に、挿入器の出力を図 2 の下に、それぞれ示す。

出力の各行に対して、識別子を正しく挿入できていることを確認する。1 行目から 3 行目は、初期値を持つ変数を表す。要素の数は 2 であり、3 番目の列(列 C)は数値であるため、挿入器は識別子「values」を挿入する。4、5 行目は、型のみを定義する変数を表す。要素の数は 1 であるため、挿入器は識別子「types」を挿入する。6 行目はインスタンス変数を表す。要素の数が 5 以上であるため、挿入器は識別子「operations」を挿入する。よって、挿入器は入力の分類リストにおける各行に関して、正しい識別子を挿入できる。識別子挿入アルゴリズムが識別子を正しく挿入できることが確認できる。したがって、挿入器は、識別子挿入リストを正しく生成できる。

次に、変換器の入力を図 3 の左上に示し、変換器の出力を図 3 の右に示す。なお、変換器の入力(図 3 の左上)は、挿入器の出力(図 2 の下)と同じである。表 1 に示した構文に基づいて、変換器が VDM++仕様書を正しく生成することを確認する。変換器は、表 1 のキーワードに従って VDM++仕様書を記述する。

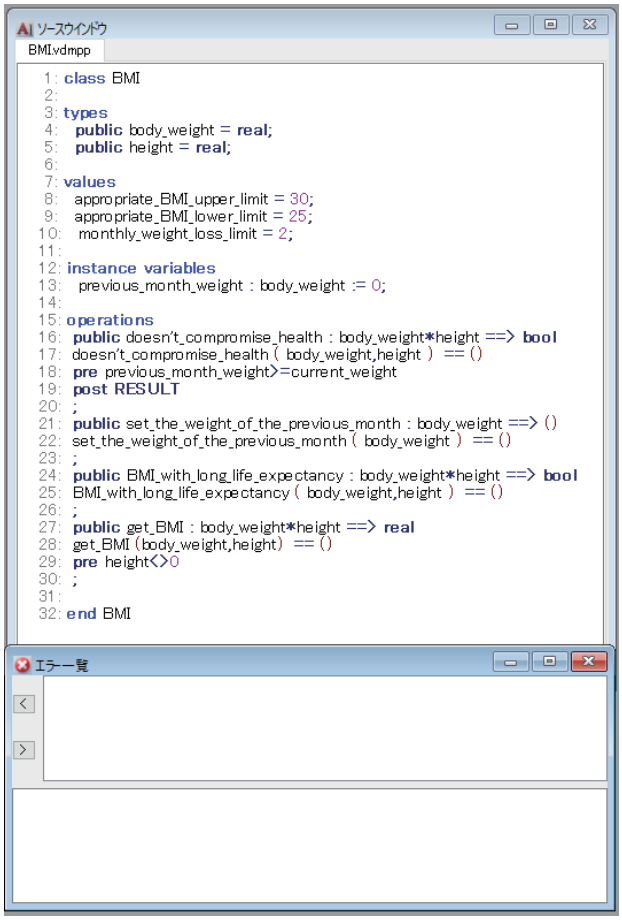


図 5.　VDM++ Toolbox による出力結果.

例えば、入力の 4、5 行目の 1 列目(列 A)の識別子は「types」であるため、変換器は表 1 のキーワード「types」の構文に従って、VDM++仕様書の types の次の行に識別子に対応した文を記述する。入力の 6 行目の 1 列目(列 A)の識別子は「instance variables」であるため、変換器は表 1 のキーワード「instance variables」の構文に従って、VDM++仕様書の instance variables の次の行に識別子に対応した文を記述する。入力の 8 行目から 10 行目の 1 列目(列 A)の識別子は「operations」であるため、変換器は表 1 のキーワード「operations」の構文に従って VDM++仕様書の operations の次の行に識別子に対応した文を記述する。12 行目の 1 列目(列 A)の識別子は「instance variables」であり、表 1 のキーワード「instance variables」の構文に従って、識別子に対応した文を記述する。したがって、変換器は、表 1 のキーワードの構文に従って、対応する文を記述した VDM++仕様書を生成できる。

さらに、変換器が生成する VDM++仕様書が、VDM++の構文に従っていることを確認する。生成した VDM++仕様書を VDM++ Toolbox に記述した際の画面を、図 5 に示す。構文およびタイプチェッカーを備えた VDM++ Toolbox は、生成した VDM++仕様書に対して警告を表示しなかった。このことから、変換器は、正しい構文の VDM++仕様書を生成できる。

以上より、提案した 2 つのアルゴリズムに基づいて試作した挿入器と変換器を組み合わせることによって、分類リストから VDM++仕様書を生成できる。したがって、提案アプローチによって、分類リストから VDM++仕様書を自動的に生成できるといえる。

5.　おわりに

本論文では、分類リストからVDM++仕様書を自動的に生成するアプローチを提案した。制約として、自然言語仕様書から機械学習により生成する分類リストが存在すると仮定した。提案したアプローチに基づいて、識別子挿入アルゴリズムとVDM++仕様書変換アルゴリズムの2つのアルゴリズムを提案した。提案した2つのアルゴリズムに従って、挿入器と変換器の2つのツールを試作した。適用例を用いて、分類リストからVDM++仕様書を自動生成できることが確認できた。

今後の課題を、以下に示す。

- 自然言語仕様書から機械学習を用いて分類リストを生成できていない

自然言語仕様書から機械学習を用いて分類リストを生成するためには、自然言語仕様書に対して NLP(自然言語処理)を行い抽出リストを生成しなければならない。

- VDM++の他の構文に対応していない

提案したアプローチでは、VDM++の基本的な型にしか対応することができていない。これにより、未対応の型定義を持つVDM++仕様書を生成することができず厳密な仕様書を作成することができなくなるため、対応する必要があると考える。この課題を解決するためには、提案したアプローチの対応する識別子を追加し、追加した識別子に対応する VDM++の構文を定義しなければならない。

- 分類リストを生成できていない

提案したアプローチは、分類リストが存在することを前提として提案しているが、現状では分類リストを生成できていない。

以上の課題のうち、自然言語仕様書から機械学習を用いて分類リストを生成できていないという課題について詳しく述べる。本論文で提案したアプローチは、自然言語仕様書から機械学習を用いて抽出リストから分類リストを生成する。抽出リストは、分類リストにおいて変数名および関数名に対応する語を抽出していること、および、抽出した語が持つ修飾関係を抽出していることが、分類リストを生成するために必要である。

そのために、変数名および関数名となり得る語の修飾関係を抽出できるように、自然言語仕様書から抽出リストの修飾関係となり得る文および文節を抽出する必要がある。

よって、自然言語仕様書に対して形態素解析および係り

受け解析を行い、1つの語および他の語から得られる修飾関係を明確にする必要がある。そして、明確にした修飾関係を分類リストが得るために、抽出リストは語ごとに修飾関係を取得できるようにしておかなければならない。抽出リストで語ごとの修飾関係を明確にすることができると、分類リストの生成が可能であると考える。

現段階では、分類リストを出力するためには、まず、各識別子ごとに識別子を挿入するために必要な要素と要素の内容を抽出することに特化した学習モデルを事前に作る必要がある。その後、識別子に必要な要素の情報を、抽出リストの修飾関係から取得する必要があると考える。

これら全体の構想を実装することで、自然言語仕様書から分類リストを生成することが可能になると考える。

参考文献

1) E. Marcus and H. Stern: “Blueprints for high availability”, John Willey and Sons, 2000.
2) National Institute of Standards and Technology (NIST), Department of Commerce: “Software errors cost U.S. economy $59.5 billion annually”, NIST news release 2002-10, 2002.
3) IPA 独立行政法人情報処理推進機構:なぜ形式手法か, https://www.ipa.go.jp/files/000005424.pdf, Accessed: 2020/02/13)
4) 荒木啓二郎, 張漢明. プログラム仕様記述論. オーム社, 2002.
5) M. Fähndrich, M. Barnett, and F. Logozzo: ”Embedded contract languages.” In the 2010 ACM Symposium on Applied Computing (SAC 2010), pp. 2103–2110. ACM, 2010.
6) Andrews, D. J., Larsen, P. G., Hansen, B. S., Brunn, and others: Vienna Development Method – Specification Language – Part 1: Base Language. ISO/IEC 13817-1 ,1996
7) VDMTools. http://fmvdm.org/vdmtools/index.html, Accessed: 2020-1-18.
8) VDMJ. https://github.com/nickbattle/vdmj, Accessed: 2020/2/11.