データ型に着目した VDM-SL 仕様の記述変形に関する考察

2012SE052 池場俊貴
指導教員：張漢明

1　はじめに

ソフトウェア開発において，プログラムの安全性や信頼性を確認するために形式手法は有効な方法である．プログラム作成のおいて，形式手法を使うことで，数学的に厳密に意味付けられた言語（形式仕様記述言語）を用いて，システムの要求を正しく実現することが期待できる．

VDM の仕様を記述する過程で，記述の書き換えは不可欠な作業であり，以下ような問題がある．仕様の記述を書き始めるとき問題の本質は最初明らかではない．記述を進める過程で問題の本質が明らかになり，より適切な記述を書き換える必要がある．また手作業による書き換え作業は労力がかかる．書き換え作業を自動化するとより効率的に進めることができる．

書き換えの例としては，識別子を変更する，同値な論理式の交換する，論理式を述語化する，データ型構造の変更などがある．本研究では VDM-SL のデータ型構造に着目する．本研究の目的は仕様の変形の自動化を支援するためにデータ型を変形するさいの変形規則を提示することである．

2　背景技術

背景技術として，形式手法と VDM-SL についての概要を述べる．

2.1　形式手法

形式手法は、数学的に厳密に意味付けられた言語（「形式仕様記述言語」と呼ぶ）を用いて情報システムの要求や設計等を記述し，情報システムがユーザの要求等を満たしているかなど論理的に推論するための仕組みを提供する手法である．1960 年代の後半から 1970 年代にわたって盛んに研究されたプログラムの検証理論やその基に，30 年以上の長い歴史を有しており、ヨーロッパを中心に基礎研究や実用化が進められている．

2.2　VDM-SL

VDM は「形式仕様記述」と呼ばれる手法の代表的なツールである．形式仕様記述は，システム状態のデータ構造やそれを操作するメソッドの機能等に関する仕様を明示化，検証する手法である．幅広い記述スタイルを持っている言語．実際の開発で使われている．VDM-SL は VDM 記述の方法の一つである．VDM-SL は，手続き的な記述に比べて宣言的な記述の抽象度が高いので，問題の本質に注力することができる．

3　研究のアプローチ

本研究では以下の観点から研究を進めた．

- VDM-SL の合成型の構成子と演算子の定義から合成型間の関係について分析
- 変形可能な合成型間の変形規則を提示

VDM-SL のデータ構造は，2 つのグループに分けられ，

- 個々の要素の集まりの集合型，列型，写像型
- データの内部構造の組型，レコード型

この 2 つのグループに分割して，グループ内での変形について研究をおこなった．

4　合成型間の関係についての分析

それぞれのデータ型構造の特徴を参考にして，変形が可能かどうか，演算子などの変換規則が表せるのかどうかを検討する．そして得られた結果を表にしてまとめる．

4.1　集まり

集合型，列型，写像型の特徴を以下に示す．

- 集合型は要素の集まりで，要素に重複はない
- 列型は要素に順番があり，同じ要素の重複がある
- 写像型はそれぞれの要素に対応する値があり，要素に重複はない

集合型から列型への変形は可能．列型から写像型へは，写像型の要素に列型の順番にあてることで変形出来る．集合型から写像型へは，集合型→列型→写像型と間接的に表すことが出来る．

列型から集合へは，列型の順番が失われてしまうので完全には変換できない．写像型から列型へは，写像型の要素が自然数の時のみ，その要素を順番の番号として変形できる.. 写像から集合型へは，写像型→列型→集合型と表すことができ，写像型→列型，列型→集合型の制約条件がかかる．

4.2　データの内部構造

組型，レコード型の特徴を以下に示す．

- 組型は　組のリストを順番で表す
- レコード型は構成要素を直接選択でき，識別子のタグ名で記述する

組型からレコード型への変形は，新たにタグ名をつけることで，表すことが出来る．レコード型から組型へは，その逆で変形する際にタグ名が失われる．

4.3 変形

得られた結果を表にしてまとめる．●は変形ができる．▲は変形するさいに情報の一部が失われる．白抜きの○，△は間接的に判断できる．本研究では，表の●にあたる集合型から列型，列型から写像型，組型からレコード型，レコード型から組型の 4 つの変形を検討する．

表 1 集まりの変形

元＼先	集合	列	写像	組	レコード
集合	−	●	○	−	−
列	▲	−	●	−	−
写像	△	▲	−	−	−
組	−	−	−	−	●
レコード	−	−	−	●	−

5 変形規則

分析の結果から，変形可能な合成型の構成子，演算子の変形を考える．おこなった変形規則の一部を示す．組型→レコード型を例にとる．組型からレコード型への変換を s() とする．対象が変数の場合はそのまま表示する．書き換えには構成子の識別子 A1，A2 に適切な名前が必要になる．

```
識別子の対応表 table
        1 →  タグ名 1
        2 →  タグ名 2
-構成子
  T = A1 * A2 * ···* An →
                レコード名::タグ名 1 : A1
                           タグ名 2 : A2
   mk_(a1, a2, ..., an) →
              mk_レコード名 (a1, a2, ..., an)
-演算子
選択     t.#n            s(t).table(n)
相当     t1 = t2         s(t1) = s(t2)
不当     t1 <> t2        s(t1) <> s(t2)
```

6 考察

事例として参加登録システムを作成し，データの合成型間の変形の有効性を検討する．また合成型の変形の有効性を考察する．

6.1 事例：参加登録システム

参加登録システムとはイベントなどの参加登録者の情報を管理するシステムである．事例として，参加登録リストから名前を検索し，その人のメールアドレスを返す関数を作成する．組型で書かれた記述を，レコード型に変形することを考えた．参加登録リストの情報としては，名前，メールアドレス，電話番号があり，組型のとしての組 Info にメールアドレスと電話番号を入れ，名前から Info への写像を作る．

```
Info = Email * Address * Telephone;
RegisterBook = map Name to Info;
```

レコード型にはそれぞれの要素にタグ名が必要なので，組型の順番に対してタグ名の対応表を示す．

```
1 -> メール
2 -> 番号
```

変形規則と対応表を用いてメールと番号の組 Info をレコード型に変形できる．

```
Info :: メール : Email
        番号 : Telephone;
RegisterBook = map Name to Info;
```

参加登録リストから名前を検索し，その人のメールアドレスを返す関数を作成する．参加リストから 1 名の情報を取り出し，名前が正しくなければその人を除いて別の 1 名の情報を取り出す．名前が同じ場合にはその人のメールアドレスを返す．

```
FindEmail : Name * RegisterBook -> Email
FindEmail(name, book) ==
       let mk_(email, -, -) = book(name) in email
       pre name in set dom book;
Findemail : Name * RegisterBook -> Email
Findemail(メール, book) ==
                book(Name).メール
pre name in set dom book;
```

このように変形規則を使ってデータ型間の変形を示すことができた．レコード型に書き換えることで，データの内部構造をより容易に扱うことができる．

6.2 書き換えを行う利点

集合型で記述されたデータに順番が必要になったとき，列型への変形が有効である．写像型や列型で記述されたデータに重複をせずにまとめたいとき，集合型への変形が有効である．また，組型で記述されたデータに対し，内部構造を扱いたい時や，型に名前をつけて分かりやすくしたい場合はレコード型に変形するのが有効である．

7 おわりに

本研究では VDM − SL のデータ構造の変換に着目し，データ構造に対し分析を行い，変換規則を提示することを行った．記述の変形の自動化の支援を提示することにより，プログラムの書き換え作業の効率化ができる．今後の課題としては，VDM-SL の記述の変換の自動化を行うことを考える．

参考文献

[1] 荒木啓二郎，張漢明：『プログラム仕様記述論』．オーム．社,2014.

[2] 岩村園子:『やさしく学べる離散数学』．共立出版, 2013.

[3] 玉井哲雄:『ソフトウェア工学の基礎』．岩波書店, 2005.