論理型仕様の理解支援のための一手法

3H－2

山本 修　小野 康一　深澤 良彰

早稲田大学理工学部

1 はじめに

ソフトェア開発においては、ソフトウェアを厳密に定義することが必要である。そのための形式的仕様記述言語として論理型言語がある。論理型言語は、正当性の証明に適しているなどの優れた性質を持つ。しかしその反面、理解性・読解性に必ずしも優れているとは言えない。理解のしにくさの原因として、仕様中のある記述がソフトウェアにおける何を表現しているのかを理解することの難しさが挙げられる。

求められるソフトウェアを仕様として定義する場合、定型的なデータ構造やその操作を陰に用いることが多いと考えられる。ここでの定型とは、テーブルやレコードなどのデータ構造やそれに対する操作である。これらの定型は、特定のアプリケーションによらず、普遍的に現れる。

このようなよく使われる定型を“概念”と呼ぶことにする。概念はデータ構造とそれに対する操作から定義されるものとする。概念が仕様記述言語によって記述されている場合、読解者はどのような概念が存在するかを記述から推測する必要がある。これは多大の労力を要する。

そこで、一階述語論理に基づいた仕様に現れる概念を、定理証明技法を用いることにより仕様の記述から推測し、読解者に提示する手法を提案する。

2 本手法の目的

本手法の対象とする仕様は、変数の宣言部と操作の記述部からなるものとする。そのような仕様の一部を図1に示す。この仕様には図2に示すようなテーブルが表現されている考えられる。そして操作(A)にこのテーブルに対するソートが記述されていると考えられる。

しかし仕様の記述からただちにこれらを理解することは難しい。この仕様を理解するには、関数群{name, english,specific}がテーブルという概念の構成要素であり、(A)の操作がテーブルを、 english の値を key としてソートしていることを推測しなければならない。

そこで、(A)の操作について図3のような情報を提示することができれば、読解者はこの操作の意味をただちに理解することが出来る。

この例に限らず、テーブルなどの概念はソフトウェア全般に現れる。従ってこのような概念に関する情報を提示することは、仕様の理解支援として有効である。この情報を提示し、理解支援とするのが本手法の目的である。

変数宣言の記述部

```
NAME       : P(strings)        ;
NUM    == : 1..300            ;
name       : NUM --> NAME      ;
english    : NUM -->1..100     ;
specific   : NUM -->1..100     ;
                  :
                  :
```

操作の記述部

```
sort_table ( ) ==

 ∀ i:1..dom(english)-1 hold (english'(i) < english'(i+1))
& ∀ i:1..dom(english) hold (∃ j:1..dom(english) hold(
                    & name'(i) = name(j)
                    & english'(i)  = english(j)
                    & specific'(i) = specific(j) ))      - (A)
                  :
                  :
```

図 1 仕様の記述の一部

num	name	english	specific
1	○○○	79	73
2	×××	・・・	・・・
⋮	⋮	⋮	⋮

図2: 図1の仕様に現れるテーブル

概念: テーブル　- 要素　name, english, specific
操作: ソート　- key　english の値
　　- 範囲　論理式全体

図3 図1の(A)の記述に対する情報提示

3 概念の推測の手法

概念に関する情報を提示するために、仕様中にどのような概念が存在するかの推測を行なう。このために、まず概念の候補を選択する。そして次の段階でそれらの候補を検証することにより概念の存在を推測する。

A Technique for Understanding Logical Specifications
Osamu Yamamoto , Kouichi Ono
and Yoshiaki Fukazawa
Department of Electrical Engineering,
School of Science & Engineering, Waseda University
3-4-1, Ookubo, Shinjuku-ku, Tokyo 169

3.1 概念のデータ構造からの推測

この段階ではまず、仕様中に存在すると推測される概念の候補を選択する。この選択を仕様の変数宣言部にデータ構造推測テンプレートを適用することにより行なう。このテンプレートには仕様記述言語の プリミティブ を用いて概念のデータ構造を表現する際に成立する条件が記述されている。そして個々の仕様に特有のデータ構造を表す変数名群を代入するスロットを持つ。このテンプレートに適合するプリミティブ群が存在した場合その概念が存在すると推測し、そのプリミティブ群を選択する。式1はレコードのデータ構造についてのテンプレートであり、定義域が同一である関数群を選択する。

$$\forall n(dom(@FUNC_n) = dom(@FUNC_1)) \qquad (1)$$

図1の変数宣言部の関数群 {name, english, specific} は 定義域が同一であり、テーブルという概念のデータ構造を表現していると推測することができる。そこでこれらをテーブルのデータ構造を表現するプリミティブの候補として選択する。

3.2 操作による概念の推測

この段階では、前の段階で選択した概念の候補について、それらが仕様中に存在するかの判断をおこなう。このために、候補が存在する場合に仕様中に現れると考えられる操作の論理式を、推測したデータ構造を用いて生成する。そして式2の定理証明を行なう。

$$\forall x\ \exists y\ (S \rightarrow T) \qquad (2)$$

S : 仕様の操作を表す論理式
T : 候補となる概念を表す論理式

この証明が成功する場合、論理式 S で表された操作中に T が表現する概念が含まれていると推測する。そして、ある概念に対する操作であると推測される論理式が仕様中に複数存在する場合、その概念が存在すると判断する。そして概念に対する各々の操作について、概念のデータ構造を表現するプリミティブ、その操作に対する情報を提示する。

この論理式の生成には操作推測テンプレートを用いる。このテンプレートには概念に対する操作を表す論理式が記述されている。そして、データ構造の推測により選択したプリミティブを代入するスロットと、操作の記述中に現れる変数名・定数名群を代入するスロットを持つ。

$$\begin{aligned} & \bigwedge_n \quad (@F'_n = @F_n + \{@KEY \mapsto @VAR_n\}) \\ \& \ & \bigwedge_n \quad (if\ dom(@F_n) = not(const) \\ & \qquad \{\text{“}\&\text{”} ran(@F'_n) = ran(@F_n) + \{@VAR_n\}\} \\ & \qquad else TRUE\ ) \end{aligned} \qquad (3)$$

式3は関数を構成要素に持つレコードへのデータの代入を表すテンプレートであり、$@F_n$ がプリミティブ を代入するスロット、$@key$,$@VAR_n$ が定数・変数を代入するスロットである。これらに変数を代入することにより定理証明に用いる論理式を生成する。この際に概念の存在が判断された場合に提示する情報を保持しておく。

定理証明を行なう前に、文字列検索により、操作を表す論理式が選択したプリミティブを表す文字列を全て含んでいるかを確かめる。全てのプリミティブを含んでいない操作については概念に対する操作が存在しないとして証明を行なわない。これにより、推測したデータ構造を含まない論理式に対する定理証明を行なうという無駄を省くことができる。

4 論理式に部分的に現れる概念の推測

論理式の操作の記述を表す論理式中に部分的に概念が現れる場合、前述の定理証明では概念の存在を推測できない場合がある。式4が成り立つ場合、式5の証明は失敗し、論理式に部分的に現れる概念とそれに対する操作を推測することができない。

$$\forall x\ \exists y\ (D \rightarrow T) \qquad (4)$$

D : 推測するべき概念が存在する論理式
T : 候補となる概念を表す論理式

$$\forall x\ \exists y\ (\ (A\&B)\ |\ (C\&D)\ \rightarrow\ T) \qquad (5)$$

そこで、論理結合子によって結合されている部分的な論理式中に概念が存在するかそれぞれ個別に推測する。しかし、全称束縛、存在束縛されている論理式から一部分のみを取りだし、それを全称束縛、存在束縛すると論理式の意味を変えてしまう。

そこで概念に対する操作の存在を推測する部分と、テンプレートから生成した論理式を置換する。そしてこの論理式が元の操作を表す論理式から導けるか式6の定理証明により確かめる。

$$\forall x\ \exists y\ (\ (A\&B)\ |\ (C\&D) \rightarrow (A\&B)\ |\ T\ ) \qquad (6)$$

このために、仕様の操作を表す論理式を標準形に変形する。そして論理和によって結合された全ての部分的な論理式について概念の推測を行なう。

証明が成功した場合、テンプレートから生成した論理式を埋め込んだ部分にその概念が存在すると推測する。

5 今後の課題

現在のところ、推測することが可能な概念はファイル、テーブル、レコード等の普遍的なものに限られている。今後、アプリケーションに依存したテンプレートを定義しするなどして、推測できる概念をさらに拡張することを考えている。