Z言語によるテキストユーティリティの仕様記述について

3H－6

関本 理佳　海尻 賢二

信州大学 工学部
E-mail: rika@cs.shinshu-u.ac.jp

1　はじめに

既存のソフトウェアは、とりあえず動くものの開発ということのみに比重がおかれ、保守のことなど考えずに作られたものが多い。このためドキュメンテーションの不備、プログラムの構造化がされていない、プログラムが理解しにくいなどの問題が生じている。また、ソフトウェア保守にかかる費用は全開発コストの70%を占めるとも言われている。そこで開発コストを削減するためには、この保守作業を支援することが効果的と考えられる。

本研究は、ある程度実用規模のプログラム群を対象としてプログラムの認識を行ない、認識結果として形式仕様を抽出することによりプログラムの理解を支援し、保守作業の効率や信頼性を向上させることを目的とする。具体的には、対象とするアプリケーションドメインを限定し、そのドメイン知識を利用して、既存のソフトウェアからその仕様を生成するリバース・エンジニアリングツールの開発を目指している。

本稿では、ドメインとしてUNIXのテキストユーティリティを対象とし、認識の主要知識であるプログラムプラン、仕様プランについて、そして、これらのプラン知識に基づく仕様生成のプロセスについて述べる。

2　テキストユーティリティと仕様記述

今回対象としたUNIXのテキストユーティリティは、ファイルの内容に対する種々の取り扱いについてのコマンドの集まりである。例えば、与えられたファイルの行数、単語数、文字数を表示するwcコマンドや、ファイルの最初の数行を表示するheadコマンドなどがある。

仕様記述言語には、ドメインの性質等を考慮し、オックスフォード大学で開発された汎用な形式的仕様記述言語であるZ [1, 2] を用いた。Zを用いた仕様化では、仕様化対象のシステムの状態およびシステムに作用するオペレーションを、述語論理と集合論に基づいて定義していく。状態とオペレーションは、スキーマと呼ばれる構造を用いて記述される。

3　プログラム認識に基づく仕様化

プログラム認識による仕様記述生成の流れを図1に示す。本研究ではプログラム認識を、具象プログラムから抽象プログラム、具象仕様、そして抽象仕様を生成するという3つのプロセスに分割して考える。ここでは、具象プログラムからの抽象プログラム化と、具象仕様を生成する仕様化プロセスについて述べる。認識においては、テキストユーティリティの具象プログラム、オンライン・マニュアル、プログラムプラン、そして仕様プランの知識が利用される。

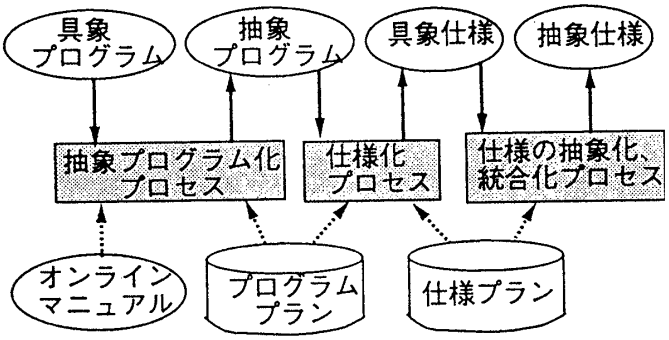

図1　仕様記述生成の流れ

Specification of the textutility using Z
Rika SEKIMOTO , Kenji KAIJIRI
Faculty of Engineering, Shinshu University
500 wakasato,nagano 380, Japan

3.1 プラン知識

・プログラムプラン

プログラムプランは、基本プラン、複合プラン、プロセスプラン、制御プラン、メタプランの5種類よりなる。

認識のベースとなる基本プランには、基本動作に対する動作プランとデータ構造に対するデータプランがあり、形態はテンプレートとして記述される。複合プランは成分として他のプランを含み、プラン名、要素プランリスト、パラメータリスト、テンプレート、スペースタイプの情報を持つ。

・仕様プラン

仕様プランの主要知識は、スキーマを生成するためのテンプレートであるスキーマ名 (schema name)、宣言部 (declaration part)、述語部 (predicate part) についての情報である。

バイト数のカウントを行なうプログラムプラン、仕様プランの知識表現例を、それぞれ図2、図3に示す。

3.2 抽象プログラム化プロセス

(1) フローグラフへの変換

構文解析およびデータフロー解析により具象プログラムを属性付きフローグラフへ変換する。

(2) 機能分割

各コマンドは、オプションの利用等により複数の処理が行なえるようになっている。そこで、パラメータ・スライシングにより、コマンドの機能を分割する。

(3) プログラムプランの認識

機能分割されたプログラムを、関数単位でプログラムプランの複合体として認識する。

3.3 仕様化プロセス

仕様化プロセスでは、抽象プログラム化プロセスからプログラムプラン木を入力として受け取る。

ここではまず、抽象プログラム化プロセスで認識された基本プラン、複合プランと仕様プランのパターンマッチングを行ない、そのプログラムプランに対応するスキーマを生成する。そして、プロセスプランの情報により関数単位の仕様を統合する。これにより機能単位の仕様が生成される。

```
Plan name : byte-count-plan
a kind of      basic
description    byte-count
parameter      ( ?fd , ?byte_count )
template       ( ( fstat(?fd,&stat) ),
                 ( ?byte_count=stat.st_size ) )
```

図2　プログラムプランの知識表現例

```
Plan name : byte-count-spec
a kind of          specification
description        byte-count
schema name        CountBytes
declaration part   ( ( fid? : FID ),
                     ( byte_count! : N ) )
predicate part     ( ( byte_count! = #(fstore(fid?)) ) )
```

図3　仕様プランの知識表現例

4 おわりに

本稿では、対象ドメインをUNIXのテキストユーティリティに限定したプログラム認識を行ない、その結果を形式的仕様記述言語Zを用いて表現するリバース・エンジニアリングの技術について述べた。これにより、プラン知識に基づく形式仕様の生成が、機能単位のレベルにおいては可能となる。

しかし現段階では、仕様記述のスキーマ生成において、仕様プランのテンプレートの情報のみを利用しているため、一つのプラン情報からは一意のスキーマしか生成できず、仕様プランがあまり汎用でないという問題が挙げられる。そこで今後は、仕様記述の分析を進めることによりプラン同士の関係を明確にし、成分として他のプランを含む仕様プランや、抽象度の高いプランの利用を検討していく。また、対話による情報の獲得等も考えている。

参考文献

[1] J. M. Spivey : *The Z Notation -A Reference Manual*, Prentice Hall (1989).

[2] J. B. Wordsworth: *Software Development with Z*, Addison-Wesley(1992).