Java言語、C言語プログラムパターンの翻訳

7C－1

岡山 敬　福田 薫　金子 博　松本 憲幸*
株式会社　東芝**

1. はじめに

ソフトウェア開発において必要不可欠かつ重要な事柄として、プログラムの作成とプログラムの理解がある。プログラムの作成については、高機能エディタや GUI 構築ツールなどを利用することにより効率的に行うことができる。プログラムの理解についても、制御構造や関数の呼出し関係などプログラム全体の構造の把握には、既存の CASE ツールが利用できるが、プログラムの各記述１つ１つを理解するためには、あまり有効ではない。そこで、この理解を支援するために、ソースプログラムの各記述を読み取ってその内容を的確に表現する情報を自動的に付加する方法を考える。

本稿では、Java 言語および C 言語プログラムを解析して各記述パターンが表現している処理や構造に適した内容の日本語コメントを機械的に付加するプログラム翻訳システムの実現方式及び実現の際の問題点と対策およびその効果について述べる。

2. プログラムパターン翻訳システムの構成

プログラムパターン翻訳システムの構成を図１に示す。プログラムを次のようにして翻訳する。

2.1 プログラム翻訳の方法

（１）記述読み込み部

プログラムを１記述ごと読み込み、その記述を表現するリスト形式（原始構文と呼ぶ）に変換する。

（２）パターン正規化部

原始構文を正規化して、より効率的な翻訳が可能な単純化されたリスト形式に変換する。ここまでの処理は、通常コンパイラなどが行う構文解析と同様の処理である([1])。

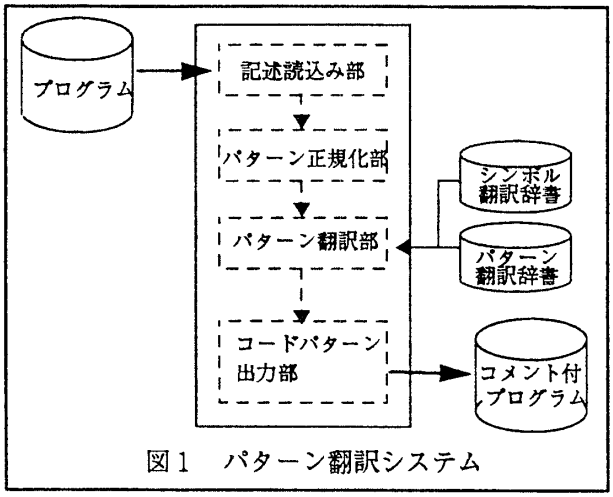

図１　パターン翻訳システム

（３）パターン翻訳部

翻訳規則が定義されているパターン翻訳辞書とプログラムに出現するシンボルに対する翻訳用語が格納されているシンボル翻訳辞書を使用して、パターン翻訳を行う。アルゴリズムは、次のとおりである。

step 1. 記述にマッチする翻訳規則をパターン翻訳辞書から探す。マッチする規則があればその規則を選ぶ。もし、見つからない場合には、終了する。

step 2. 翻訳規則中に変数が含まれている場合は、変数部分の記述に対する定義をシンボル翻訳辞書から探す。定義がある場合には、そのコメントを展開する。もし、見つからない場合には元の記述をそのまま残しておく。

step 3. 展開したコメントを翻訳結果として返す。

（４）コードパターン出力部

プログラムコードと翻訳結果を合わせて出力する。

Translation of Java Language C Language Program Patterns

* *Takashi Okayama, Kaori Fukuda, Hiroshi Kaneko, Noriyoshi Matsumoto*
** TOSHIBA Corporation

2.2 パターン翻訳辞書

パターン翻訳辞書に定義する翻訳規則は、同一記述に対して複数定義可能であり、翻訳時には定義順に適用するため、限定規則から汎用規則という順序で定義する。ある記述に対してどのような翻訳規則を定義するかが非常に重要である。

例１：**open** に関する３つの翻訳規則の定義

定義１：**open(..., 0)**

→" 読み込みモードでファイルオープン"

定義２：**open(..., n) 0<n<16**

→" モード **n** でファイルオープン"

定義３：**open(..., n)**

→" 警告：存在しないモードのオープン"

3. 翻訳コメント定義の実現方式

翻訳コメント定義に関して次の問題が存在する。

（A）プログラムのすべての記述に対してコメントを付加すると逆にプログラムが読みにくくなる。

（B）プログラムの１つの記述中に現われるすべての要素（関数や変数）についてコメントを付加すると逆にプログラムが読みにくくなる。

（A）については、辞書に定義されていないパターンやシンボルの翻訳を付加しないという機構を実現することにより解決する。

（B）については、プログラムの記述形式により付加するコメントを変えることで対応する。

プログラムの記述形式を次の２つに分類した場合、意味のあるコメントとしてそれぞれ次のようなコメントが考えられる。

（１）代入文（記述形式：**X = Y + Z**）

・**X** を求める。

・**Y** と **Z** を加える。

・**Y** から **X** を求める。

・**Y** と **Z** から **X** を求める。

例２：" **X** を求める" を付加する場合

sum = money[i] + pre_sum ;

→" 合計金額を求める"

（２）関数呼出し（記述形式：**X = f(Y)**）

・**X** を求める。

・**f** を実行する。

・**Y** を処理する。

・**Y** から **X** を求める。

・**f** を実行して **X** を求める。

例３：" **Y** を処理する" を付加する場合

status = execlp("print","print",0);

→" 印刷タスクを処理する"

このように同一形式の記述に関しても、実際のプログラム記述を理解するためには、意味のあるコメントとして複数のものが考えられる。これらのコメントの違いは、実際のプログラム記述を理解するために大切な部分が記述のどの部分に存在するかに依存している。これは、パターン翻訳辞書において、関数ごとの複数の限定規則、汎用規則と、同一形式ごとの限定規則、汎用規則を定義しておくことで対応可能である。

4. 効果

プログラムパターン翻訳システムを利用することにより、プログラム記述を効率的に理解することが可能となる。そのため、既存ソフトウェアの保守や改造・移植、流用による類似ソフトウェアの作成などの開発にも有効である。

6. おわりに

Java 言語、**C** 言語のプログラムの各記述に対して日本語のコメントを付加するシステムの実現方式及び実現の際の問題、対策及び効果について述べた。

今後、コメント付加規則及び翻訳辞書を充実させて、さらに高度な翻訳を行えるように改良していく予定である。

参考文献

[1] A.V.Aho and J.D.Ullman : "Principles of Compiler Design", Addison-Wesley, 1977.