2J－9　ミスマッチングをフィードバックする推論方法を取り入れたプログラム理解システムについて

佐藤徳幸†[1]　神保至‡[2]　大熊義嗣¶

†‡情報処理振興事業協会(IPA)技術センター　¶東京電機大学理工学研究科

1　はじめに

我々は、知識ベースに立脚したプログラムの意味理解処理を行ない、C言語プログラムのソースコードをもとに詳細設計仕様書レベルの情報を抽出することを目標として研究・開発を行なっている[1]。

システムに於て、知識ベースを増やせば理解が可能なプログラムパターンが増えるが、マッチングに要する処理時間もそれだけ増大する。しかし人間の場合は、マッチングを取るべきパターンの予測が間違っていた場合に、マッチングに失敗した原因を考え、その結果をフィードバックして次にマッチングを取るべきパターンを選択し推論時間を短縮しているのではないかと考えられる。

本稿では、処理時間の高速化のための、「ミスマッチングをフィードバック」する推論を取り入れたプログラム理解システムについて提案する。

2　推論の方法

広辞苑によると「理解」とは、1）物事の道理をさとり知ること 2）意味をのみこむこと 3）物事がわかることとある。

すなわちこれらを素朴に考えると「こころ」の働きである。よって、「心的活動を理解しようとする試み」である認知科学を基盤にしたアプローチを採用することはコンピュータによって理解させることへの近道であると考えられる。

ゆえに我々はプログラム理解のアプローチとして、認知科学的アプローチを基盤としている。

ところで、「アルゴリズム＋データ構造＝プログラム」とはWirthの著名な書名であるが、この両辺に「意味」という言葉を掛ければ、「プログラムの意味＝アルゴリズムの意味＋データ構造の意味」となる。このことはすなわち、プログラムの意味を導出するためには、アルゴリズムの意味またはデータ構造の意味を出発点にすれば良い事を示していると考えている。

しかし、データ構造から特定の意味を読みとることは典型的なデータ構造が存在しない現状では困難である[2, pp.157]。よって、我々は、アルゴリズムに着目し、あらかじめ蓄えておいたアルゴリズムパターンの知識ベースとマッチングをとりながら同定してゆき、マッチングをとれたものを理解できたものとする。

2.1　プログラムの意味の知識表現

アルゴリズムの知識は、各層に分担記憶し、柔軟な適応力を持たせている。これらは、上位の階層から下位の階層に向けて次のようになる[2, pp.160]。

問題解決概念レベル 代表的問題解決技法の概念や、その抽象的アルゴリズムの概念
例：配列の積や整列化といったような概念や、その抽象的アルゴリズムの情報

抽象データ処理アルゴリズムレベル 代表的データ処理アルゴリズムに関する実現手続きの知識
例：配列の積を求める手続き、各種の整列化の手続きといったもの

基本データ処理技法レベル 基本的データ処理技法に関する知識
例：2つの変数間の値の交換法、個数の計数法、配列の中の最大値の求め方といったもの

基本データ操作レベル 基本的データ操作や制御に関する知識
例：変数への値の代入、2つの値の比較、条件分岐の方法等

以上により、我々はおおむね4水準の階層をなす記憶構造を持ったプログラム知識構造を知識フレームとして実装することを考えている。

2.2　推論の過程

我々は、先に述べたプログラムの意味の知識表現を用いて推論を行なう。具体的には、下位の知識の組合せのマッチングが成功する事によって上位の知識が確かに使われていると推論されるわけである。

A Program Understanding System with feedback miss matching

†SATOU Noriyuki (e-mail: noriyuki@stc.ipa.go.jp)
Software Technology Center,
Information-technology Promotion Agency, Japan
38-1-3 Shiba-kouen Minato-ku TOKYO 105 ,JAPAN
[1](株)日本コンピュータ研究所より出向中
[2](株)三菱総合研究所より出向中

こうして、最も上位まで成長した木をそのプログラムの持つ意味として採用する[3]。

3 ミスマッチングをフィードバックする推論

マッチングによるパターン同定によって意味を理解してゆく限りは、その理解対象とする幅は知識量に依存する。しかし、単純な悉皆検索を行なっていては、推論が実用時間内に終了しない事が十分に考えられる。

そこで、我々はミスマッチングをフィードバックし、推論速度を向上させる事を考えた。フィードバックの種類は次の2点を考えている。

- 関数名・変数名の利用
- バックトラックの効率化

3.1 関数名・変数名の利用

実際にプログラムをコーディングする場面を考えてみると、関数名や変数名は機械的に命名するのではなく、その機能を説明するような名前を命名される事が多いようである。よって、関数名や変数名の持つ意味といったものは、アルゴリズムの仮説を立てる有効な手がかりとなると考えられる。

そのための一つの方法として、あらかじめ関数名・変数名辞書用意しておき、それを利用してアルゴリズムの仮説を立てる方法が考えられる。しかし、それらの命名規則はコーディングを行なったチーム毎にまとまりが見られる程度で、理解対象とするすべてプログラムに適応可能な規則は発見できそうにない。

そこで、これらをプログラム毎に学習しフィードバックするシステムを提案する。

目的は、関数名・変数名をキーにして、マッチする事があり得ないと思われるようなパターンマッチングを避ける事をである。

これは、“ruleモジュール” および “instanceモジュール” によって構成する。ruleモジュールは命名例の一般的な知識を完全では無いまでも蓄えておく。instanceモジュールはruleモジュールによって補い切れなかったミスマッチを学習してゆく。

このことにより、ヒトにおけるセマンティックスフォーカシングの学習を真似ができると考えている。

3.2 バックトラックの効率化

例として、図1のプログラムの意味を考えてみる。

このプログラムは非常に簡単な例ではあるが、問題点を示唆している。

このプログラムは、5行目6行目9行目からなる「数値の入力を求め、その数値を出力する」パターンと、5行目および7行目から9行目からなる「数値の入力を求め、その総計と個数を算出する」パターンとの合成と考えられる。

```
1   void sum_average()
2   {
3      int x, sum, n, ave;
4      sum = n = 0;
5      while (scanf("%d", &x) != EOF) {
6        printf("%d\n", x);
7        sum += x;
8        n++;
9      }
10     if (n) {
11        ave = saum / n;
12        printf("Average is %d\n", ave);
13     }
12   }
```

図 1: 例題プログラム

しかし、7行目まで読んだ限りでは、「数値の入力を求め、その数値を出力する」パターンと「数値の入力を求め、その総計を求める」パターンとマッチングが成功してしまい、バックトラックが発生する。

一般的なパターンマッチング法ではバックトラックにより不弁別的に正しい意味解釈を検索するが、ヒトの場合は既に行なった理解結果に手を加えて正しい理解を行なっていると考えられる。

そこで、7行目8行目の解釈を切り離し、とりあえず走査を進め、11行目理解に取り掛かった時に、sumをnで割っている事から、7行目8行目は総計と個数を数え上げているとの意味処理結果を得る。

4 おわりに

ミスマッチングをフィードバックする推論方法を取り入れたプログラム理解システムについて述べた。

現在、プロトタイプシステムを実装作業中であり、αバージョンを1995年10月を目処に作成する予定である。

参考文献

[1] 佐藤徳幸, 神保至, 中島歩. “認知科学的アプローチに基づくプログラム理解システムについて”. 情報処理学会第49回全国大会, 5巻, pp. 261–262, Sept 1994.

[2] 大須賀節男, 上野 晴樹等. “知的プログラミング”. オーム社, 1993.

[3] 佐藤徳幸, 神保至, 中島歩. “トップダウン推論とボトムアップ推論を組み合わせたプログラム理解手法の有用性”. 電子情報通信学会総合大会, No. D-677, Mar 1995.