2F−8　プログラム理解における論理エラーの分類と意図理解への応用

関本理佳　村山浩　上野晴樹
東京電機大学 理工学部

1 はじめに

現在のプログラミング環境ではプログラム理解の能力を持たないので，プログラムを作成しても文法チェックしかしてくれず，またそれも分かりにくいメッセージであることが多い．まして簡単な論理ミスばかりでなくスペルミスさえも補正してくれない．これらは全てプログラマの負担となっている．人間のチュータがミスを含んだプログラムを読む時は，ミスの原因や訂正の方法を同定することが出来る．そこで我々は，人間のチュータの役割をコンピュータが代行するような教育向き知的プログラミング支援環境の開発を行っている．

本稿では，実験に基づく論理エラーの分類と意図理解への応用について述べる．

2 実験に基づく論理エラーの分類

初心者プログラマの作成したバグを含むプログラムから，バグの生じた原因やプログラマの意図を推論し，訂正のための助言を行うためには，以下のようなことがわかっていなければならない:

■初心者プログラマが一般にどのようなエラーを起こしやすいか．

■論理エラーとユーザの意図との関係．

我々は，これらに付いての一情報を得るために，認知科学的な実験を行った．

この実験より得られた典型的な論理エラーを体系化することにより，プログラマの意図理解が可能になると考える．

2.1 認知科学的実験

プログラミング言語でプログラムを書くときに必要となるプログラミング知識は，プログラミング技法に関する知識とプログラミング言語に関する知識の2つに分類できそうである[1]．そこで,これらに関する誤り知識を探るために以下に示す2つの実験を行った．

2.1.1 プログラミング技法に関する誤り知識を探る実験

目的: プログラマが持っているプログラミング技法に関する知識が誤っているために生じるバグにはどのようなものがあるかを探る．

実験方法: 対象問題を代表的な整列化法である再帰に基づくクイックソートとし，初心者プログラマにクイックソート法による整列の大まかな概念を説明した後にこのプログラム全体を作成してもらった．但し，この実験ではプログラミング言語に関する誤り知識を調べることが目的ではないので，実験の際に被験者から構文等に関する質問があった場合には，マニュアルを見せるなどして質問に答えた．またプログラミング終了後，そのプログラムにエラーがあった場合にはその部分はどのような考えに基づいて書いたのか等の質問に答えてもらい，被験者の意図を探った．

実験結果:

被験者数（人）	78
得られたプログラム数	78
再帰に基づくQuicksortのプログラムと認識できるプログラム数	54
認識できないプログラム数	24

2.1.2 プログラミング言語に関する誤り知識を探る実験

目的: プログラマが持っているプログラミング言語に関する知識が誤っているために生じたバグにはどのようなものがあるかを探る．

実験方法: この実験では，プログラミング言語や低レベルの技法に関する知識に焦点をあて，非常に小さな単位でプログラムの一部分だけを作成してもらった．問題作成の際には，初心者プログラマにアルゴリズムの設計等の負担がなるべくかからないように心がけた．今回は次の3題について実験を行った．

実験1:配列の中央値を求める，実験2:配列内からある値を走査する，実験3:2変数の交換．

実験結果:

	問題1	問題2	問題3
被験者数（人）	36	36	36
得られたプログラム数	36	34	36
Proper プログラム数	26	24	29
Buggy プログラム数	10	10	7

2.2 論理エラーの分類

実験より得られた論理エラーをそのエラー原因により以下のような3種類に分類した．

2.2.1 アルゴリズムに依存するエラー

手続き型プログラムは，”プログラム＝データ構造＋アルゴリズム”と表せることより，プログラムにおいて

Classification of Logical Errors and It's Application to Reasoning User's Intention in Knowledge-Based Program Understanding
Rika SEKIMOTO, Hirosi MURAYAMA and Haruki UENO
Tokyo Denki Univ.

はデータ構造が対象世界のモデルであるのに対し，アルゴリズムはそのモデルに対する操作のシーケンスであると考えることができる．より具体的に言えば，データ構造を初期状態から目標状態まで更新していく操作のシーケンスがアルゴリズムである．つまりアルゴリズムは複数のデータ処理のチャンク（まとまり）の組合せになる[2]．そこで，アルゴリズムに依存するエラーを更に以下のように分類した．

まず，各変数に対する処理が適切な位置で行われていないものと，適切な位置で行われているのだが処理が誤っているものに分類．前者は処理の順序が異なっているもの，必要な処理が欠如しているもの，余分な処理が存在するもの等のエラーである．後者は各処理で単独で起こるものと，１つのエラー原因によって複数の処理に影響が生じてしまうものに分けられる．

2.2.2 技法に依存するエラー

技法に依存するエラーは，特定の問題解決に限らず，一般の問題において初心者がおこしやすいエラーの技法であり，以下のようなエラーが存在する．

技法に依存するエラー
- ２変数の交換に関するエラー
- 初期設定に関するエラー
- 配列の定義範囲に関するエラー

2.2.3 言語に依存するエラー

言語に依存するエラーは，一般的な構文に付いての論理ミスやケアレスミス等であり，以下のようなエラーが存在する．

言語に依存するエラー
- 代入文
- 条件文 —— if文
- 繰り返し文
  - repeat文
  - while文
  - for文

3 意図理解への応用

分類した論理エラーを，我々が開発しているプログラム理解システム ALPUS（Algorithm-Based Program Understanding System）に応用した．システムへの組み込みについて報告する前に，まず ALPUSにおけるプログラム理解の考え方について述べる．

3.1 ALPUS におけるプログラム理解の考え方

ALPUS におけるプログラム理解は，アルゴリズムの知識を中心としているが，基本的には認知科学的な考え方に基づいている．ここでいう認知科学的とは，人間のやり方を参考にしてモデル化するという意味である．人間が複雑なアルゴリズムに基づくプログラムを理解する場合には，いくつかの証拠から仮説としてのアルゴリズムを頭の中に描き，次にそのアルゴリズムをテンプレートとして，プログラムコードをテンプレートマッチングによって理解しようとするものと考えられる[2]．

我々はこのような観点からアルゴリズム，プログラミング技法，論理エラーに関する知識を HPG（Hierchical Procedure Graph）と呼ぶ階層的手続きグラフで表現した．プログラム理解は，この知識をテンプレートとしてプログラムコードとのテンプレートマッチングによって行われる．

3.2 論理エラーに関する知識の組み込み

ALPUS がミスを含むプログラムからプログラマの意図を理解するためには，エラーに関する知識を持つ必要がある．そこで，分類した論理エラーをそれぞれ以下のように ALPUS に組み込んだ．

3.2.1 アルゴリズムに依存するエラー

このエラーは対象となる問題に依存する．各変数に対する処理が適切な位置で行われていないものは，その問題固有の HPGと構造が異なるので，これによりエラーを発見する．適切な位置で行われているのだが処理が誤っているものは，HPG の関連箇所にバギーパターンとして管理する．

3.2.2 技法，言語に依存するエラー

技法，言語に依存するエラーは，対象問題に依存しない汎用的なエラー知識としてテンプレート化する．以下にその具体例を示す．

□技法に依存するエラー：２変数（Ａ，Ｂ）の交換

```
Proper      work:=A; A:=B; B:=work;
Buggy       A:=B; B:=A;
```

□言語に依存するエラー：代入文（ＸにＹを代入）

```
Proper      X:=Y;
Buggy       Y:=X;
```

4 まとめ

論理エラーの分類を行い，この知識を ALPUSに組み込むことにより，誤りを含むプログラムからそのエラー原因やプログラマの意図を推論し，訂正のための助言が行えるようになった．

今後は ALPUS の意図理解能力の向上を目指し，より適切な訂正の助言が行えるように，エラーの体系化を進めていく予定である．

＜参考文献＞

[1] 上野晴樹：知的プログラミング環境－プログラム理解を中心に－．情報処理，Vol.28，No.10，pp.1280-1296，1987

[2] 上野晴樹：プログラム理解システムALPUSの考え方と方法．人工知能学会研究会資料，SIG-KBS-8903-3