1K－5

知的プログラミング環境における学生の状態の理解

矢野正己　上野晴樹

東京電機大学

1．はじめに

教育向きのプログラミング環境においては，プログラミングの教育の支援機能，つまり知的ＣＡＩ機能をもっていた方がよいと思われる．知的ＣＡＩに限らず，プログラミング環境を知的に，つまりシステムが人間教師のように，プログラミングをしている人間の状態を理解し，何をアドバイスすればよいか，何をここではやろうとしているのかということなど，人間教師が教育的観点に立ってやっていることができなければ，知的になったとは言えないであろう．これが，知的ＣＡＩの分野でいう，学生モデルの考え方でもある．ここでは，プログラミング環境における学生モデルについて検討する．プログラミング環境INTELLITUTORは，３つのサブシステムGUIDE，ALPUS，TUTORよりなっている[2]．知的ＣＡＩ機能であるTUTORは，GUIDE（ガイデット・エディタ），ALPUS（プログラム理解）からの情報をもとに，TUTORは学生モデルを構築するが，プログラム理解と学生モデル構築は強い関係にある．プログラム理解は，アルゴリズムの意味表現モデルであるHPG(Hierarchical Procedure Graph)との意味的マッチングによって行なわれるが，学生モデルは，主に，このグラフに付加されているエラーに関する知識と，階層的に分類されているプログラミング知識によって，正しくプログラムされている部分と，そうではない部分とから推論し，プログラミング作業中に構築される．したがって，従来研究開発されてきた知的ＣＡＩとは大きな違いがある．つまりこれまでの知的ＣＡＩは，学生との対話によって，誤り原因を同定していたが，TUTORは学生との対話は最小限で誤りの原因を推定し，学生モデルを構築できるという特徴を持っている．つまり，寡黙なシステムチューターであるにもかかわらず，学生の状態にあわせた教育行動ができるのである．

2．プログラミング環境における学生モデル

プログラミング環境において学生モデルを構築する意義について考えてみる．プログラムというのは，１つのプログラム言語においても，その実現方法はたくさん考えられる．しかし，誰が見てもわかるプログラムというのは，限られてくる．学生は，プログラマとしては初心者であり，プログラミングの経験が少ないために，わかりにくいプログラムを書くことが多い．これは，目的を果たせばいいという観点からは，満足のいくプログラムである．しかし，教育的な観点から見ると，プログラミング経験をいくら積んでいたとしても，わかりにくいプログラムを書くようではいけない．また，プログラミングの経験を積むと共に，教師にアドバイスを求める種類もかわり，アドバイスを示す時期も変わってくる．つまり，プログラムにエラーがないときにも，プログラミングの教育をしなければいけない．こういったことを果たすために，学生の状態を理解する学生モデル構築というが求められている．では，知的プログラミング環境における学生モデルについて検討してみる．

2．1　構文的観点からの学生モデル

構文的観点というのは，プログラムが構文的に正しいかどうかをチェックするときの，教師が描く学生の状態のことをいう．これは，GUIDEの裏で働く機能である．具体的には，学生がプログラミングしているときに，電子マニュアルのどこを参照したのか，PASCAL言語の予約語の適用順序，構文テンプレート内でのプログラミング順序，また構文的エラーの状態，などの情報を抽出する．つまり，学生のプログラミング作業を人間教師があたかも見ているような機能を実現し，その時点で学生の理解をできる限りするものである．これはガイドをしているときに，不必要なメッセージを削除できるなどの可能性がある．このレベルの学生モデルをプログラミング履歴モデルと呼ぶ．

2．2　論理的観点からの学生モデル

論理的に正しいかどうかをチェックしているときに，学生が誤りを起こしていれば，その誤りの原因を同定し，また弱い知識がどこなのかをプログラミング履歴モデルと一緒に推論し，学生モデルとして構築する．これは，階層的に表されるプログラミング知識と，その一部であるアルゴリズムを表すＨＰＧによって可能と考えている．

ここで，プログラミング知識について検討してみると[1]，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とに分類することができる．プログラミング技法に関する知識は，細やかな技法的知識から抽象的な概念的知識までが，いくつかのレベルに階層化されているものと思われる．具体的な細かい技法に関する知識が最下位にあり，抽象的概念知識が最上位にある。これらを最下位から最上位に向けて表すと，次のようになる．

－基本操作要素技法レベル
－基本データ処理技法レベル
－抽象データ処理アルゴリズムレベル
－問題解決概念レベル

最下位の知識つまり基本操作要素技法の知識とは，変数への値の代入，２つの値の比較，条件分岐などの基本的データ操作や制御に関する知識であり，どんな複雑なアルゴリズムでも結局はこれらの基本的操作の組み合せで実現できる．次のレベルの基本データ処理技法の知識とは，基本データ処理

Understanding of Novice Programmer's State on a knowledge Based programming Environment
Masami YANO ,Haruki UENO
Tokyo　Denki Univ.

技法に関する知識で，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などが含まれる．経験を経たプログラマは，これらの手法の組み合せでプログラミングすると思われるから，プログラムが比較的整然としており理解しやすい．その次のレベルの抽象データ処理アルゴリズムレベルの知識とは，配列の積を求める手続き，各種の整列化の手続きなどのような，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をいう．このレベルの表現をＨＰＧによって行なっている．ＨＰＧはアルゴリズムを構造化し，それをプロセスのシーケンスとして表している．最上位の，問題解決概念技法の知識は，配列の積や整列化のような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情報である．また，このレベルの知識の特徴は，宣言的であり，またある概念を内部に含むようなより大きい単位の概念を存在し，それらの間の関係が属性情報として管理されていることである．あるレベルの技法でその上位レベルの技法が表現でき，逆にそのレベルの技法はすぐ下位のレベルの技法によって表現できる．

一方，プログラミング言語に関する知識は，２段の階層構造をしていると思われる．すなわち，いろいろなプログラミング言語に共通の言語要素である基本的プログラム文に関する概念的知識が下位にあり，上位には個別のプログラミング言語に関する概念的知識が存在する．基本的プログラム文に関する概念的知識とは，VAR，TYPE，ARRAY，INTEGERなどの変数宣言文，代入，算術演算，比較演算，論理演算，インデキシング，型変換などのデータ操作文，およびIF文，WHILE文，REPEAT文，CASE文，FOR文，PROCEDURE文，などの制御文などに関する概念的知識である．これらは各プログラミング言語からの独立の手続き型プログラミング言語に共通の基本言語要素であり，多少形を変えて各プログラミング言語の中に採り入れられている．これらの情報と関連した情報，すなわち，構文規則，意味規則，および使い方や使用例などのヒューリスティックスも関連知識として教師は体系的に記憶しているものと思われる．この知識構造をつかって，学生の状態を理解し，またアドバイスを生成するものと考えられる．

このプログラミング知識の階層的表現によって，学生が犯した誤りのレベルが推定できる．

左側の走査プロセスで，演算子を「＋」とすべきところを，「－」とプログラムを書いたとする．そのとき，このエラーは，変数への値の代入に関する知識について誤ったこととして，チュータは判断する．つまり，基本操作要素技法のレベルでの誤りと推定する．そして，プログラミング履歴モデルを用いてさらに推論すると，学生は，エディタの編集機能を用いて，下のステートメントからコピーしてきたとわかったら，これは単なる編集ミスで，誤りのレベルでも最下位のものとして推論し，助言もエラー箇所を示すだけとし，左側からの走査プロセスでの役割を説明しなくてもよいことになる．また，基準値設定プロセスで，配列の名前を書かなければいけないのに，欠如していて，走査プロセスでの，インデックスの指す配列要素と，基準値の比較の二値比較において，A[RIGHTINDEX]>A[BASE]のようなものであれば，基準値設定プロセスと走査プロセスについて独立に説明するよりも，おそらくその学生は，基準値設定プロセスでの役割をなんとなく理解していたのだが，はっきりとは理解していなかった．しかし，これはクイックソート法ではとても重要な知識であり，これがわかっていなければ，クイックソートをわかったとはいえない．つまり，誤った知識は，下位の知識レベルであるにも関わらず，これはアルゴリズムレベルでの誤りで，詳しく説明をすることを要すると推定しなければいけない．これは，ＨＰＧの基準値設定プロセスと走査プロセスで，このような典型的なエラーパターンと，そのときの誤り原因を包括的なものとして付加しておくことによってできる．しかし，これでは，あくまでも包括的な誤り原因であり，個人的誤り原因の推定まで達していない．これも先ほども述べたが，プログラミング履歴モデルとあわせて学生の状態を推定することによって，個人的学生モデルの構築が可能であろうと考えている．このレベルでの学生モデルをＨＰＧに基づく学生モデル呼ぶことにする．

このアプローチは，学生の思考を探り，学生の意図を推定していることにもなる．表層的な誤り原因同定ではなく，より深層的な誤り原因をしようと試みている．つまり，学生が考えてきた過程を推定することによって，その学生に適した助言，教授が可能となり，学生が求めんとしていることに異ならないプログラムに関する教育的支援ができると我々は考えている．

３．　おわりに

プログラミング履歴モデル，ＨＰＧに基づくモデルというプログラム教育支援用のプログラミング環境における学生モデルを提案し，学生の状態を理解し，学生の要求に応じた支援ができる枠組を提案した．

現在までに，プログラム理解された結果を基に，ALPUSからの情報だけで表層的レベルの学生モデルを構築できようになり，それを基に，助言文生成，また学生からの質問に対しても学生のレベルに応じた助言ができるようになった．[3]

そして，教材知識についてGUIDE，ALPUS，TUTORから多目的に使えるような知識表現について対象モデルの考えに基づいて検討しており，またプログラミング履歴モデル，ＨＰＧに基づく学生モデルが構築できる枠組をプログラミング書法なども含めプログラミング教育全体から検討し，設計している．

謝辞

本研究に協力していただいた石田剛司，酒匂英彰両氏，上野研究室の皆さんに感謝します．

参考文献

[1]上野　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，vol28,no10,pp1280-1296,1287

[2]Ueno.H INTELLITUTOR:A Knowledge Based Intelligent Programming Environment for Novice Programmers,proc COMPCON89 Spring,pp390-395,1989

[3]矢野，上野，知的プログラミング支援環境における知的ＣＡＩ－学生モデルと教材知識についての考察－ AI89-16,1989