コンピュータと教育 59−10
（２００１．３．２３）

CPUとアセンブラ授業におけるプログラム評価とアドバイスの支援

渡辺 博芳　荒井 正之　武井 恵雄

帝京大学 理工学部 情報科学科

本論文では，COMET/CASLを教材としたCPUとアセンブラ授業において，提示した課題に対して学生が作成したプログラムを評価する作業の支援(EA機能)とプログラム作成途中の学生にアドバイスをする作業の支援(TA機能)を統合的に実現する手法を提案する．EA機能については既に事例に基づく評価支援法を提案し，その有効性を実証した．TA機能についても同様なアプローチで実現可能と考えられる．これらを統合的に実現することによって，EA機能とTA機能で利用する情報や部分機能を共有することができる．教員がアドバイスを行う際，最初簡単なヒントを提示し，それでもわからない場合により詳細な解説を行うといった方法がとられることが多い．そこで，TA機能については，アドバイスの詳しさとタイミングを扱えるモデルを提案する．提案手法に関して，具体的な対象タスクを「正しく動作しないプログラムに対するアドバイス」としたシステムの実現について検討を行った．アドバイスが必要な状況は極めて多岐に渡るため，実現システムでは，アドバイスのためのリソースを学生からも収集することを試みた．

An Evaluating and Teaching Assistant System for the Students' Programs of the CPU and Assembler Course

Hiroyoshi Watanabe, Masayuki Arai and Shigeo Takei

Dept. Information Sciences
School of Science and Engineering, Teikyo University

This paper proposes an integrated architecture for evaluating assistant (EA) and teaching assistant (TA) systems in the domain of programming in assembly language CASL. EA supports teachers in their work of evaluating programs submitted by students, that is, judging whether a student's program satisfies the requirements of the given problem and giving advice to the student about the program. On the other hand, TA supports teachers with the work of giving advice to students who are solving a problem, for example, developing algorithms or writing programs. We had already proposed a method of case-based program evaluation for EA and demonstrated its effectiveness. TA should be implemented using almost the same approach. Therefore, we are trying to integrate EA and TA. The integration of EA and TA enables them to share some information and sub-procedures. On implementation of TA, we propose a model of dealing with the degree of detail and timing of advice, because usually teachers give a small hint at first and they give more detailed advice later. We have also been investigating the utility of advice sentences written by students.

1 まえがき

初等プログラミング演習授業は，教員が提示した問題の題意を満たすようなプログラムを学生が作成し，提出されたプログラムやレポートを教員が評価するという形態をとることが多い．このような演習授業において，教員は，(a) 学生からの質問に対する回答，(b) 学生のプログラムやレポートの評価，(c) 理解度や進捗状況の把握を目的とした質問など多くのタスクをこなす必要がある．学生数が多い場合，教員の作業量は膨大になる．これらのタスクで教員が忙殺されると，学生は適切なアドバイスや評価結果を適切なタイミングで受けることができなくなることが懸念される．

本研究の目的は，初等アセンブラプログラミング演習授業において，(1) 教員が学生の作成したプログラムを評価する作業の支援，(2) プログラムを作成段階の学生に対してのアドバイスの支援を行うための統合的なアーキテクチャを提案することである．本稿では，(1) の支援機能を EA(Evaluating Assistant) 機能，(2) の支援機能を TA(Teaching Assistant) 機能と呼ぶことにする．我々は EA 機能については既に事例に基づく実現法を提案し，その有効性を明らかにした [1]．TA 機能についても，同様なアプローチで，実現方法を検討する．

本稿では，EA 機能と TA 機能の統合アーキテクチャについて述べ，TA 機能の実現方法について検討する．TA 機能では，特に「アドバイスの詳しさとタイミングの問題」を扱えるようなモデルを提案する．さらに，学生へのアドバイスのうち，「動作の正しくないプログラムに関するアドバイス」を対象とした実験結果を報告し，課題について考察する．

2 EA と TA の統合アーキテクチャ

2.1 システムの全体像

実現しようとする支援システムの全体像を図 1 に示す．システムはネットワーク環境に実現し，サーバ，学生用クライアント，教員用クライアントから構成する．学生は学生用クライアントから，課題

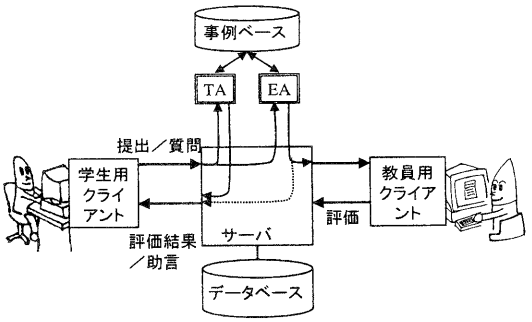

図 1: システムの全体像

プログラムの提出と課題プログラムに対する質問を行うことができる．提出されたプログラム等はサーバのデータベースに蓄える．また，サーバには EA(Evaluating Assistant) 機能と TA(Teaching Assistant) 機能を持たせる．教員は教員用クライアントから，学生の提出状況を閲覧したり，評価やアドバイスを書きこむことができる．

提案するモデルでは，機能としては EA 機能と TA 機能を明確に分離すること，実現方法としてはこれら 2 つの機能を統合的に実現することという方針をとる．このことにより，2 つの機能で利用する情報 (例えば，事例等) や部分機能 (例えば，プログラム照合機能等) を共有できる．また，教員が授業ごとに，両方の機能を用いるケース，どちらか一方の機能のみを用いるケース等を選択することも可能になる．

(1) EA(Evaluating Assistant) 機能

学生が提出したプログラムを教員が評価する作業を支援する機能である．我々は，全員の学生が合格するまでプログラムの提出を繰り返すことを前提にして，以下の具体的な評価作業を対象としている [1]．

- 学生が作成したプログラムが提示した問題の題意を満たしているかどうかの合否判定
- 評価結果に関連するアドバイス (不合格の場合はその理由，合格の場合はよりよいプログラムにするためのアドバイス)．

(2) TA(Teaching Assistant) 機能

学生が課題遂行において「質問」あるいは「アドバイス要求」[1] をした際に，それに対するアドバイ

[1] 本稿では，これらを表すために，以降，「アドバイス要求」という語を用いる．

スを与える作業を支援する機能である．学生からの質問の内容は，作成しようとしているプログラムに関すること以外にも，提示された問題の題意の確認，提示された問題で用いられる概念に関すること，実習環境に関すること等，多岐にわたる．このような全ての質問に対するアドバイスを支援することは困難と考えられるので，アドバイスの対象を限定して検討を行い，徐々にシステムが扱える範囲を広げる方針をとる．当面は，ある程度プログラムを作成できた学生を対象とする．具体的には，動作の正しくないプログラムに対するアドバイスを対象とする．

2.2 EA機能の実現方法

「学生が作成したプログラムは，その動作と実現方法が問題の題意を満たすべきである」を評価基準とし，プログラム評価処理は2つのフェーズに分けて行う．最初に，あらかじめ用意したテストデータを用いて，学生のプログラムの動作を評価する．次に，正しく動作したプログラムを対象として，事例ベース推論によってプログラムの実現方法を評価する．つまり，過去の評価事例を検索し，評価対象のプログラムと同じ実現方法と見なせるプログラムの評価事例が存在すれば，その事例の判定結果とアドバイスを評価対象のプログラムに適用する．方法の詳細は文献[1]を参照されたい．

動作評価の結果，正しく動作しないプログラムについては，正しく動作しないテストデータの組を提示して，「動作が正しくないので，提出が受け付けられなかったこと」を伝える．この時点で，それ以上の助言をしないのは，まずは自分自身で正しく動作しない原因を考えて欲しいからである．従って，EA機能の役割は，「正しく動作しないテストデータの組」を提示するまでとして，学生がそれ以上の助言を必要とする場合には，以下に述べるTAが対処することとする．

2.3 TA機能の実現方法

TAの実現も基本的には，事例ベース推論のアプローチをとるが，実際の事例だけでなく，別の方法で収集したアドバイスリソースを事例の形式にしたものも利用する．また，アドバイスを提示する際に，「アドバイスの詳しさとタイミングの問題」を扱えるような方法をとる．これらについて説明する．

(1) アドバイスリソースを事例形式で表現．

事例ベース推論は，過去の問題解決事例を事例ベースに保持し，新しく与えられた問題をそれに類似する過去の事例に基づいて解決する推論法である．問題解決の結果を新事例として事例ベースに追加することで，システムの問題解決能力が向上するという特徴がある．しかし，TAが対象とする状況は，EAに比較して非常に多様性があるために，実際の事例だけでカバーできる範囲は極端に限定されることが予想される．そこで，実際に行ったアドバイスを事例として獲得する以外にも，アドバイスリソースを収集する必要がある．本稿では，アドバイスを提示するために用いる情報や知識をアドバイスリソース[2]と呼ぶ．

事例の獲得以外の方法で収集したリソースも事例と同じ形式で保持することとする．一般的に，事例は「問題記述」と「解記述」から構成される．TAにおいては，質問文，または質問項目，デバッグ対象となるプログラムリスト等が問題記述に相当する．また，文章や図などで表現されたアドバイスが解記述である．アドバイスリソースはこのような事例の形式で表現する．また，本稿では，実事例かどうかに関係なく，アドバイスリソースを事例の形式で表現したものは全て事例と呼ぶ．

(2) アドバイスの詳しさとタイミングの戦略

学生にアドバイスを行う場合に，アドバイスの詳しさとタイミングをどうするかという問題がある．例えば，学生が正しく動作しないプログラムの原因がわからないときに，教員は最初，「この辺がいけないんじゃない?」といった感じで簡単なヒントを提示し，それでも分からない場合には，徐々に詳しいアドバイスを提示することが多い．アドバイスを受けた学生が最初の簡単なヒントで問題解決が行えるなら，詳細な説明を提示された後に比較して，より能動的に問題解決をする経験が積めること，学習対象について自信を持てることなどの効果が期待される．そこで，アドバイスを提示する際に，その詳しさとタイミングの問題を扱

[2]例えば，正しく動作しないプログラムに対するアドバイス文や，そのプログラムに類似する正しく動作するプログラムの評価事例等がアドバイスリソースとして利用可能である．

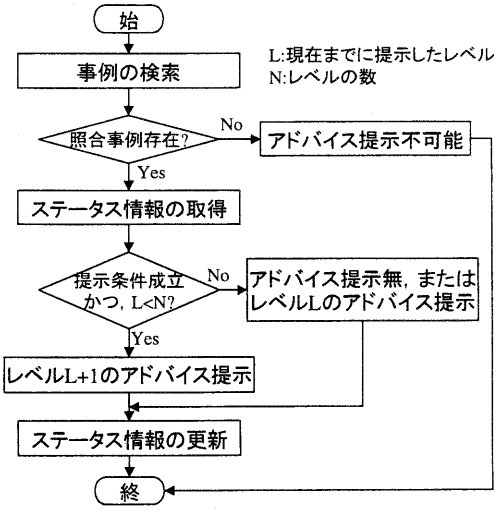

図 2: アドバイス提示の基本モデルにおける処理の流れ

えるような提示方法のモデルを提案する.

2.4 アドバイス提示の基本モデル

(1) アドバイス事例の表現とステータス情報

前節で述べたアドバイスの詳しさとタイミングを扱うために，アドバイス事例の解記述には，アドバイスの詳しさに応じていくつかのレベルに分けてアドバイス内容を記述しておく．レベルの数をNとする．また，アドバイスを受ける学生ごとに，以下のような情報を含むステータス情報を保存しておく．

・ある「アドバイス要求」に関して，これまで提示したアドバイスのレベル(図2のL).

・次のレベルのアドバイスを提示するか否かの条件を検査するための情報(例えば，現在のレベルのアドバイスを最初に与えた時刻等).

(2) アドバイス提示処理

アドバイス提示時には，図2のような処理を行う．学生からアドバイス要求がなされると，学生からの，質問文や質問項目，またはデバッグ対象となるプログラムリストに照合するアドバイス事例を検索する．照合する事例が存在しなければ，TAがアドバイスを提示することはできない．照合する事例が存在する場合は，アドバイスを受ける学生ごとのステータス情報を取得し，次のレベルのアドバイス提示条件を調査する．条件を満足していれば，次のレベルのアドバイスを提示する．条件を満足していなければ，アドバイスを提示しないか，同じレベルのアドバイスを提示する.

(3) アドバイス提示条件

アドバイスを提示するタイミングに関する戦略としては以下のようなものが考えられる.

(a) 問題演習開始から，ある程度の時間(n分間) はアドバイスを一切提示しない.

(b) 基本的には，アドバイス要求があるごとに，より詳細なアドバイスを提示するが，現在のレベルのアドバイスが最初に提示された時刻から，ある程度の時間(m分間) は次のレベルのアドバイスを提示しない.

3 動作の正しくないプログラムに対するTA機能

3.1 実現の方針

前章で述べた手法の有効性を検証するために，動作の正しくないプログラムに対するアドバイスを対象として，具体的なシステムを実現する．アドバイスは「アドバイス文の提示」により行うこととし，当面は，図や音声は用いない.

動作の正しくないプログラムは，動作の正しいプログラムに比較して非常の多くのバリエーションを持つ．そこで，できるだけ多くのアドバイスリソースを収集するために，以下のような方針をとる.

・アドバイス文を余力のある学生からも収集する.

・類似性に基づく方法と差異に基づく方法を実現する.

(1) 余力のある学生からのアドバイス文を収集

ウェブ等を利用して，動作の正しくないプログラムのリストを公開しておき，教員と余力のある学生がそれに対するアドバイス文を登録できるような枠組みを用意する．課題を早く達成した学生に，アドバイス文の登録を促すことで，アドバイスリソースを収集することを目指す．さらに，他人のプログラムを理解し，アドバイスをすることで理解が一層深まるといった意味で，アドバイス

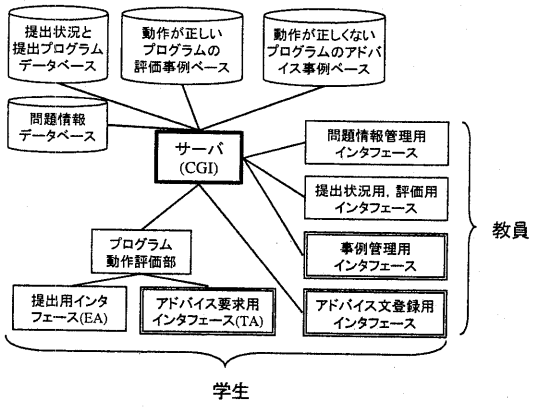

図 3: 実現するシステムの構成

文を登録する学生にも教育的効果が期待できる．

学生からもアドバイス文を収集することで，あるプログラムに対するアドバイスが複数組存在する可能性が生じる．

(2) 類似性に基づく方法と差異に基づく方法

アドバイス文を生成する1つめの方法は，アドバイス事例を利用することである．すなわち，教員や余力のある学生が登録したアドバイス文をアドバイス事例として保存しておき，現在のアドバイス要求の内容に照合する類似のアドバイス事例を利用する．これが類似性に基づく方法である．

一方，動作の正しいプログラムに対する評価事例を利用してアドバイス文を生成することも考えられる．すなわち，動作の正しいプログラムと比較した結果，ごく一部の差異を除いて照合するような評価事例が存在すれば，その差異の部分が正しく動作しない原因であると考えられる．これが，評価事例との差異情報に基づく方法である．

3.2 システム構成

実現しようとするシステム全体の構成を図3に示す．インタフェースのうち，二重線で描かれている部分以外はEA機能に関連して既に実現されている．EAとTAの主な処理機能はサーバ上にCGIプログラムとして実現し，クライアント上にはインタフェースのみを実現する．つまり，サーバには各インタフェースに対応した処理を行うCGIプログラム群を配置する．ただし，プログラムの動作が正しいかどうかを評価する処理は，負荷分散のために学生用のクライアント上に実現する．

(1) アドバイス要求機能

アドバイス要求の対象として，正しく動作しないプログラムを設定して「アドバイス要求」ボタンをクリックすることで，アドバイスが提示される．

学生用クライアントシステムは，まず，対象となるプログラムの動作を評価し，アセンブルエラーがある場合と正しく動作する場合はその旨のメッセージを提示して処理を終了する．正しく動作しない場合には，対象となるプログラムをサーバに送信する．サーバは次節以降で述べる方法によってアドバイス文を生成し，クライアントがそれを提示する．

(2) アドバイス文登録機能

教員，または余力のある学生がアドバイス文を登録する機能である．アドバイス文入力用インタフェースには，問題文と行番号付きのプログラムリストが表示されており，アドバイスの詳しさのレベルN個のアドバイス文入力欄を設ける．

アドバイス文入力用インタフェースには2つの方法でアクセスできるようにする．1つは，動作の正しくないプログラムの一覧から，プログラムを選択する方法である．プログラムの一覧にはプログラムIDと現在登録されているアドバイスの数を提示する．アドバイス数が0のプログラムに対するアドバイス文登録を促すことで，アドバイス可能なプログラム数を増やすことが可能である．なお，動作の正しくないプログラムは，次のような場合にそのプログラムに照合するプログラムがプログラム一覧になければ，一覧に追加される．

・学生が動作の正しくないプログラムを提出した時点．

・学生がアドバイス要求を行った時点．

アドバイス文入力用インタフェースにアクセスする2つめの方法は，プログラムのソースリストを利用した検索を行い，検索の結果，得られたプログラムに対するアドバイス文を登録する方法である．これは，自分で動作の正しくないプログラムに対する解決策を見出した場合に，そのプログラムについてのアドバイス文を入力する際に有効である．

(3) 事例管理機能

事例の削除や，評価事例中の評価結果とアドバイス文，アドバイス事例中のアドバイス文を修正するための編集機能である．

EA機能に関する，他の機能については文献[1]を参照されたい．

3.3 アドバイス提示処理

(1) アドバイス事例

事例は問題記述と解記述から構成する．動作の正しくないプログラムに対するアドバイスのためのアドバイス事例の問題記述は，対象となるプログラムリストである．これはEA機能のための評価事例の問題記述と同じである．一方，解記述は，N段階に分けられたアドバイス文である．**3.1**節で述べた通り，アドバイス文は複数組存在する可能性があるので，アドバイス事例の解記述にはN段階に分けられたアドバイス文を複数組，保持可能とする．また，各組を識別するためのアドバイスIDを割り当てる．

これらの事例を文献[2]で提案した事例ベース構造で保持しておく[3]．

(2) アドバイス提示戦略と学生のステータス情報

アドバイス提示において，**2.4**節(3)で述べたような戦略のうち，(b)の条件のみを調べることとする．**2.4**節(3)の(a)について考慮しないのは，動作の正しくないプログラムに対するアドバイスが必要になるのは，課題を与えられてから十分時間が経過した後と考えられるからである．(a)のような条件が重要になるのは，問題解決の基本方針を与えるようなアドバイスに関してである．

前述の(b)の条件を調査するために，以下のような，アドバイス要求に関するステータス情報を，アドバイスを受ける学生ごとに保持しておく．

(a) 対象となるプログラムID
(b) 現在のアドバイスのレベル（図2のL）
(c) 現在のレベルのアドバイスを最初に提示した時刻
(d) 既に提示したアドバイスID

アドバイスの提示処理の流れは，ほぼ図2に示した通りである．ただし，提示条件が成立しない場合の処理は，レベルLで未だ提示していなアドバイスがあれば，それを提示する．すなわち，提示条件が成立していなくとも，同一レベルの新しいアドバイスを受けられるケースもある．

アドバイスの提示においては，それまで提示したアドバイス文(下位のレベルのアドバイス文)も併せて提示する．

[3]ただし，動作の正しくないプログラムについては，冗長命令情報を用いることができない．

4 実験と考察

4.1 実験条件

前章で述べたシステムのうち，「評価事例との差異に基づくアドバイス文生成機能」と「事例管理機能」を除いた部分について実験システムを構築した．2000年度の2回の授業において，本システムを用いて，学生が登録したアドバイス文が利用可能かどうかを調べる実験を行った．

(1) 出題した問題

本システム使用時に提示した問題は以下の2問である．

P1：2文字分のデータ(2語)を1語に詰め込む．
P2：スタックを用いて文字列データ逆順にする．ただし，結果は1語に2文字詰め込む．

(2) アドバイスの詳しさのレベル

アドバイスのレベル数N=3とし，概ね以下のような区分とした．

第1レベル：エラーのある場所．
第2レベル：エラーの原因(何が悪いかをおおまかにアドバイス)
第3レベル：エラーの回避方法(どう直せばよいかをアドバイス)

(3) アドバイス文評価方法

第1，第2，第3レベルのアドバイス文をそれぞれ評価し，1組のアドバイスを総合的に評価を行った．評価は以下の4段階評価とした．

A：よいアドバイス．
B：使用可能なアドバイス．
C：使用したくないアドバイス．(例えば，内容的には適切だが，表現がわかりにくいケース等)
D：使用不可能なアドバイス．(例えば，内容的に適切でない，正しく動くプログラムの提示になっている，いたずら等)

表 1: アドバイスの登録状況

問題	P1	P2
システムが収集した動作の正しくないプログラム数	10	33
アドバイス文が登録されたプログラムの数	3	1
登録されたアドバイスの件数 3レベルのうち1レベルのみのものも含む	15	1
全レベルが入力されたアドバイス数	10	1

表 2: P1に対するアドバイス文の評価結果

アドバイスのレベル	L1	L2	L3	総合
(イ)全体の平均点	2.3	1.5	1.6	1.9
(ロ)よいアドバイス数	8	4	2	4
(ハ)使用可能なアドバイス数	10	6	7	6
(ニ)全アドバイス数	11	13	11	10

(ロ) 平均点が2点を超えるアドバイスの数.
(ハ) 平均点が 1.6点を超えるアドバイスの数 (3人の評価がB,B,C 以上のもの)

以上の評価を，3人の教職員が別々に行って，結果を集計した.

4.2 実験結果と考察

(1) アドバイス文の登録状況

アドバイス文の登録状況を表1に示す．授業の履修者は78人であり，簡単な問題であるP1では，全体の1割以上の学生がアドバイス文を登録したが，難しい問題であるP2ではアドバイス文登録者はたった1名であった．難しい問題においては，他人のプログラムに対してアドバイス文を登録する余裕がある学生がほとんどいないことがわかる．

P1では動作の正しくない10プログラムのうち，3プログラムのみにアドバイス文が登録された．これは，動作の正しくないプログラム一覧の上から3つである．すなわち，余裕のある学生にアドバイスを登録してもらう形態では，動作の正しくないプログラム一覧に早い時間に掲載されたプログラムに，アドバイス文登録が集中することがわかる．

(2) アドバイス文の評価結果

P1について，評価のAを3点，Bを2点，Cを1点，Dを0点として，3人の教職員の評価結果を平均した得点を表2に示す．P2については1組のアドバイスが登録されいたが，第1から第3レベル，総合評価とも，3人の教職員全員がDの評価であった．(内容が適切でなかった．) なお，登録されたアドバイスには，正しいプログラムリストの掲載やいたずらは存在しなかった．

表2によると，総合評価で使用可能と判断されたアドバイス数は全体の6割である．提案した枠組みで学生が登録したアドバイスをそのまま使用するには，十分な品質とは言えない．第1レベルのアドバイス(エラーのある場所)に関しては十分な品質であると思われるので，第2，第3レベルのアドバイスの品質向上が望まれる．

4.3 今後の課題

前節で述べた実験結果から，提案する枠組みを有効に機能させるための課題として，以下があげられる．

(a) アドバイスリソースの収集

実験結果から，特に，難しい問題に関しては，もともと学生にアドバイスリソースを求めるのが困難かもしれない．それをどのように補完するかが課題となる．解決策として，評価事例を用いたアドバイス文生成機能の実現や，教員や人間のTAによるアドバイス文登録等が考えられる．難しい問題は，アドバイスが必要なプログラムリストのバリエーションが多いので，前者によってカバーできる範囲は限定されると思われる．また，後者では教員等の作業負荷の増加を抑制するような工夫が必要である．

(b) アドバイス事例の品質確保

余力のある学生にアドバイスリソースを求める場合には，アドバイス事例の品質確保が問題となる．解決策としては，教員等による保守管理が考えられるが，教員の作業負荷を増大させないような工夫が必要である．

さらに，提案するような枠組みが実現できたとして，その効果を評価する方法についても検討する必要がある．つまり，我々が既に実現した事例に基づく評価支援システムは，以下のような理由で，有効に機能し，その有効性も確認できた．

・教員がプログラムを評価をする作業は，もともとオンラインで行う方が都合が良いため，評

価事例の獲得とその事例を用いた評価支援が自然な形で行えた.

・教員は提出されたプログラムを全て評価する必要があるため,全体の作業量が明確であった.

それに比較して,アドバイス支援では以下のことが言える.

・アドバイス作業は,もともとオフラインで行う方が都合が良いため,自然な事例の獲得が困難である.

・教員が行わなければならないアドバイス作業量が明確でない.

これらを考慮して,今後,検討を進めたい.

5 関連研究

プログラムリストを解析・診断してバグを指摘したり,アドバイスを行う試みは多い[3〜7].しかし,知識に基づく手法では,知識ベースの構築負荷が大きく,多くの課題に対応することが困難である.それに対して我々は事例に基づくアプローチをとり,拡張性と実用性を高めている.

一方,高橋らは,プログラミング学習において学習者間の知識を共有することを目的として,プログラミングノートから生成した考察文データベースと,既学習者が登録したエラーメッセージごとの助言文データベースを学習者が利用できるような枠組みを構築した[8].彼らのエラーメッセージごとの助言文データベースは,課題に依存しないために再利用率が高く,コンパイラのメッセージに戸惑う初学者には有効であると考えられる.

本研究では,上の2つのアプローチの両方を用いる.つまり,(a)プログラムリストを正しく動作するプログラムとの比較により,アドバイスを生成する方法と(b)余力のある学生等にアドバイスリソースを求める方法である.さらに,上の関連研究に比較して,詳しさとタイミングを考慮したアドバイス提示を行う点が特徴的である.

6 むすび

学生が作成したプログラムの評価支援(EA)に加えて,プログラム作成途中の学生に対するアドバイス支援(TA)を行うための統合的手法を提案した.また,「動作の正しくないプログラムに対するアドバイス」を対象として,部分的に実現したシステムを用いて行った,予備的な実験の結果からは,アドバイスリソースをどのように収集し,その品質をどのように確保するかが課題となることが明らかになった.

今後は,これらの検討とともに,評価事例を用いたアドバイス文生成法(差異に基づく方法)について具体的な検討を行いたい.

謝辞 実際の授業での実験および学生が記述したアドバイス文の評価にご協力頂いた本学技術職員高井久美子さんに感謝する.なお,本研究の一部は科学研究費補助金(12780293),人工知能研究振興財団の援助による.

参考文献

[1] 渡辺博芳,荒井正之,武井恵雄:事例に基づく初等アセンブラプログラミング評価支援システム,情報処理学会論文誌, Vol.42, No.1, pp.99 – 109, 2001.

[2] 渡辺博芳,荒井正之,武井恵雄:初等アセンブラプログラミングを対象とした事例に基づくプログラム評価のための事例ベース構築法,情報処理学会第61回全国大会, 3P-6, 2000.

[3] Adam,A. and Laurent,J.P.: LAURA, A System to Debug Student Programs, Artificial Intelligence, Vol.15, pp. 75–122, 1980.

[4] Murray,W.R.: Automatic Program Debugging for Intelligent Turoring Systems, Computional Intelligence, Vol.3, pp. 1–16, 1987.

[5] Johnson,W.L.: Understanding and Debugging Novice Programs, Artificial Intelligence, Vol.41, pp. 51–97, 1990.

[6] Schorsch,T.: CAP: An Automated Self-Assessment Tool to Check Pascal Programs for Syntax, Logic and Style Errors, Proc. of SIGCSE95, pp.168 – 172,1995.

[7] Ueno,H.: Concepts and Methodologies for Knowledge-Based Program Understanding – The ALPUS's Approach, IEICE TRANS. INF & SYST, Vol.E78-D, No.2, pp. 1108–1117, 1995.

[8] 高橋参吉,松永公廣:知識共有を促進したプログラミング学習システムの構築,教育工学関連学協会連合第6回全国大会講演論文集, Vol.2, pp. 227–228, 2000.