2003－CE－69 (5)
2003／5／16

初等アセンブラプログラミングにおける ニアミスプログラムに対するアドバイス提示法

渡辺 博芳　　高井 久美子　　荒井 正之　　武井 恵雄
帝京大学理工学部

本稿では，プログラム動作が正しくないが，完成に比較的近いプログラムに対するアドバイスを計算機システムによって自動的に提示する方法について述べる．そのようなプログラムをニアミスプログラムと呼ぶ．アドバイスを与える際に，最初は簡単なヒントを与え，段階的に詳しいアドバイスを与えるアドバイス提示モデルを採用する．また，提示するアドバイス文を生成する方法として，事例に基づく２つの方法を利用する．(1)過去のニアミスプログラムへのアドバイス文（アドバイス事例）との類似性に基づく方法と(2)正しく動作するプログラムとの差異を用いる方法である．提案した手法に基づいて，COMETⅡ／CASLⅡを教材とした初等アセンブラプログラミングのためのアドバイス文提示システムを開発した．開発したシステムを実際の授業で試験的に利用し，提案手法の有効性が示唆された．

A Method of Giving Advice for Novice Programs Written in Assembly Language

Hiroyoshi Watanabe,　Kumiko Takai,　Masayuki Arai　and　Shigeo Takei
School of Science and Engineering, Teikyo University

This paper describes a method of giving advice for near-miss programs using a case-based reasoning approach in programming education. A near-miss program is a program that does not run properly but is rather close to correct programs. Two case-based methods of generating advice sentences are proposed; one is based on similarity to past advice cases and the other is based on the difference from correct programs. In addition to these, a model of giving step-by-step advice is proposed. Based on these ideas, a case-based advice system for a simple assembly language is implemented. The system had been utilized in actual classes and the results suggested that the case-based advice system was useful.

１．　はじめに

典型的な初等プログラミング演習授業では, 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. このような授業において, 教員は学生へのアドバイス, 学生の提出したプログラムやレポートの評価などを行う必要がある. 特に, 授業日中に全ての学生に題意を満たすプログラムを完成させようとした場合, 教員が行う作業の負荷は非常に大きくなる.

我々は, CASL 及び CASLⅡによる初等アセンブラプログラミングを対象として, 提示した課題に対して学生が作成したプログラムを教員が評価する作業を支援するシステムを開発した[1〜3]. 本システムは, 学生が提出したプログラムに対して, あらかじめ用意したテストデータを用いた動作の評価と, 事例に基づく実現方法の評価を行う. 開発したシステムを実際の授業で使用することで, 我々のアプローチが教員の評価作業負荷の軽減に大きな効果があることを明らかにした[1,2].

次のステップとして, 本研究では, 動作は正しくないが完成に近いプログラム(ニアミスプログラム)に対するアドバイスの支援を目的とする[4].

プログラミング教育を対象として, 学習者の作成したプログラムを解析し, バグレポートやアドバイスを生成する研究は古くから行なわれている[5〜8]. これらの研究では, できるだけ詳細なバグレポートやアドバイス文を生成することを目的としており, プログラミングの概念など深い知識に基づくアプローチがとられている. しかし, 深い知識を計算機で使用できる形式で獲得するには労力が必要であり, いわゆる知識獲得のボトルネックのために, 実用的なシステムを構築するのが困難であった.

ところで, 教員が学生にアドバイスを行う際には, 最初からそのように詳細なアドバイスを提示することはほとんどない. むしろ, 最初はかなり簡単なヒントを提示し, 段階的に詳しくアドバイスを行うことが多い. そこで, 本研究では, 段階的に詳しいアドバイスを提示するモデルを採用する. このようなモデルで提示するアドバイス文は簡潔なものでよいため, アドバイス文の生成に, 深い知識を必要としない. 本研究では, 過去のアドバイス事例との類似性に基づいてアドバイスを行う方法と正しく動作するプログラムとの差異に基づいてアドバイスを行う方法を提案する.

２．アドバイス提示のアプローチ

2.1 段階的なアドバイスの提示

教員が学生にアドバイスを与える場合, 最初に小さなヒントを与え, 段階的に詳細なアドバイスを与えていると思われる. 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. そのような形でアドバイスを与える方が, 最初から詳細なアドバイスを与えるよりも教育的であると考えられる. つまり, 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.

初等アセンブラプログラミング演習において, 自分たちが学生にアドバイスを与える場面を分析してみると, ３段階程度に分けてアドバイスをすることが多い. 最初に「プログラムリストのどの辺に原因があるか」について簡単なヒントを提示し, 次に「それはどのような種類の原因なのか」についてのヒントを提示する. それでもわからない場合, 更に何段階か分けてアドバイスを続ける場合もあるが, ３回目にはある程度詳細にアドバイスを与えることが多い. そこで, アドバイスシステムでは, ３段階または４段階に分けたアドバイス提示を行うこととした.

Variations

N	Maximum number of levels of advice sentences.
M	Minimum interval time until presenting the next level of advice sentences.
student	Student who requested advice.
ctime	Current time.
as	Advice sentences.
cid	Identification of a case used for generating advice sentences.
si	Status information which includes the following members:
	level The level of advice sentences presented last time.
	cid Identification of a case used for generating advice sentences.
	time Time of presenting advice sentences in ***level***.

Algorithm

1. ***as***,***cid*** ← Advice sentences generation by a sub-system.
2. If ***as*** is not empty then
  2.1 Input ***si*** for ***student***.
  2.2 If ***si.cid*** = ***cid*** then
    2.2.1 If ***si.level*** < ***N*** and ***ctime*** − ***si.time*** > ***M*** then
      2.2.1.1 ***si.level*** = ***si.level*** + 1
      2.2.1.2 ***si.time*** = ***ctime***
      2.2.1.3 Present ***as*** in ***si.level***.
      2.2.1.4 Save ***si***.
  2.3 else
    2.3.1 ***si.level*** = 1
    2.3.2 ***si.cid*** = ***cid***
    2.3.3 ***si.time*** = ***ctime***
    2.3.4 Present ***as*** in ***si.level***.
    2.3.5 Save ***si***.

図１ アドバイス提示モデルのデータ構造とアルゴリズム

2.2 アドバイス文生成の２つの手法

本研究では，２つのアドバイス文生成法を提案し，アドバイス文を生成するケースをできるだけ多くするために，両方の方法を採用する．

１つは，過去のアドバイス事例との類似性に基づいてアドバイスを行う方法であり，「類似性に基づく方法」と呼ぶ．アドバイス事例はアドバイス対象となるニアミスプログラムとそのプログラムに対して提示したアドバイス文の組で構成する．今回，アドバイス対象となるプログラムに類似したプログラムを持つアドバイス事例を検索し，そのアドバイス文を今回の対象プログラムに適合するように簡単な修正を施すことでアドバイス文を生成する．

２つめは，正しく動作するプログラムとの差異を解析してアドバイス文を生成する方法であり，「差異に基づく方法」と呼ぶ．正しく動作するプログラムは，我々が既に実現したプログラム評価支援システムの評価事例ベースに含まれるものが利用可能である．そこで，アドバイス対象となるプログラムに類似するプログラムを評価事例ベースから検索する．アドバイス対象と最も類似する評価事例との差異が採用条件よりも小さい場合に，差異を分析してアドバイス文を生成する．

３．アドバイス提示モデル

学生があるプログラムリストについて最初にアドバイスを要求したとき，バグの場所に関するヒントを提示する．その後，基本的には学生が同じプログラムリストに対してアドバイスを要求するたびに，次のレベルのアドバイス文を提示するが，現在のレベルのアドバイス文が提示してから，ある時間間隔 ***M*** が経過していない場合は次のレベルのアドバイス文は提示されない．

このようにアドバイス文を段階的に詳しく提示するためのアドバイス提示モデルを図１に示す．最初にアドバイス文を生成するサブシステムによってアドバイス文生成を試みる．アドバイス文生成サブシステムは ***N*** レベルのアドバイス文を生成できるものとする．サブシステムがアドバイス文を生成できなかった場合，つまり，***as*** が空のときはアドバイスの提示はできない．

サブシステムがアドバイス文を生成できた場合は，どのレベルのアドバイス文を提示するかを決定するために，学生のステータス情報を読み込む．個々の学生のステータス情報は，現在までに提示されたアドバイスのレベル

(*si.level*), アドバイス生成に利用された事例の識別子(*si.cid*), 現在のレベルのアドバイス文が最初に提示された時刻(*si.time*)から構成される. 未だアドバイス文が提示されていない場合などは，これらの値は全て空となる.

今回のアドバイス文生成において利用された事例の識別子(*cid*)が前回提示されたアドバイス文の生成で利用された事例の識別子(*si.cid*)と等しければ, アドバイス対象のプログラムはほとんど同じと考えられる．この場合, 先にアドバイス文を提示した時刻(*si.time*)と現在の時刻(*ctime*)の差が *M* より大きければ次のレベルのアドバイス文を提示する.

cid と *si.cid* が等しくない場合は，前回にそのプログラムに対するアドバイス文が提示されていないケースであるので, レベル１のアドバイス文を提示する.

４．類似性に基づく方法

4.1 アドバイス事例

アドバイス事例は, 主にアドバイス対象のプログラムリストと, *N* レベルのアドバイス文から構成する(我々の実装では *N*=3 とした). 以下の問題に対するアドバイス事例に含まれるプログラムリストとアドバイス文の例を図 2 の右側に示す.

問題：N 個の整数データの合計をインデックスレジスタの値を１ずつ減らしながら求めるプログラムを作成しなさい．データの個数はラベル N で示された場所に保存されており, データはラベル DATA 以降に保存されているものとする．また，合計はラベル ANS で示す場所に求めること.

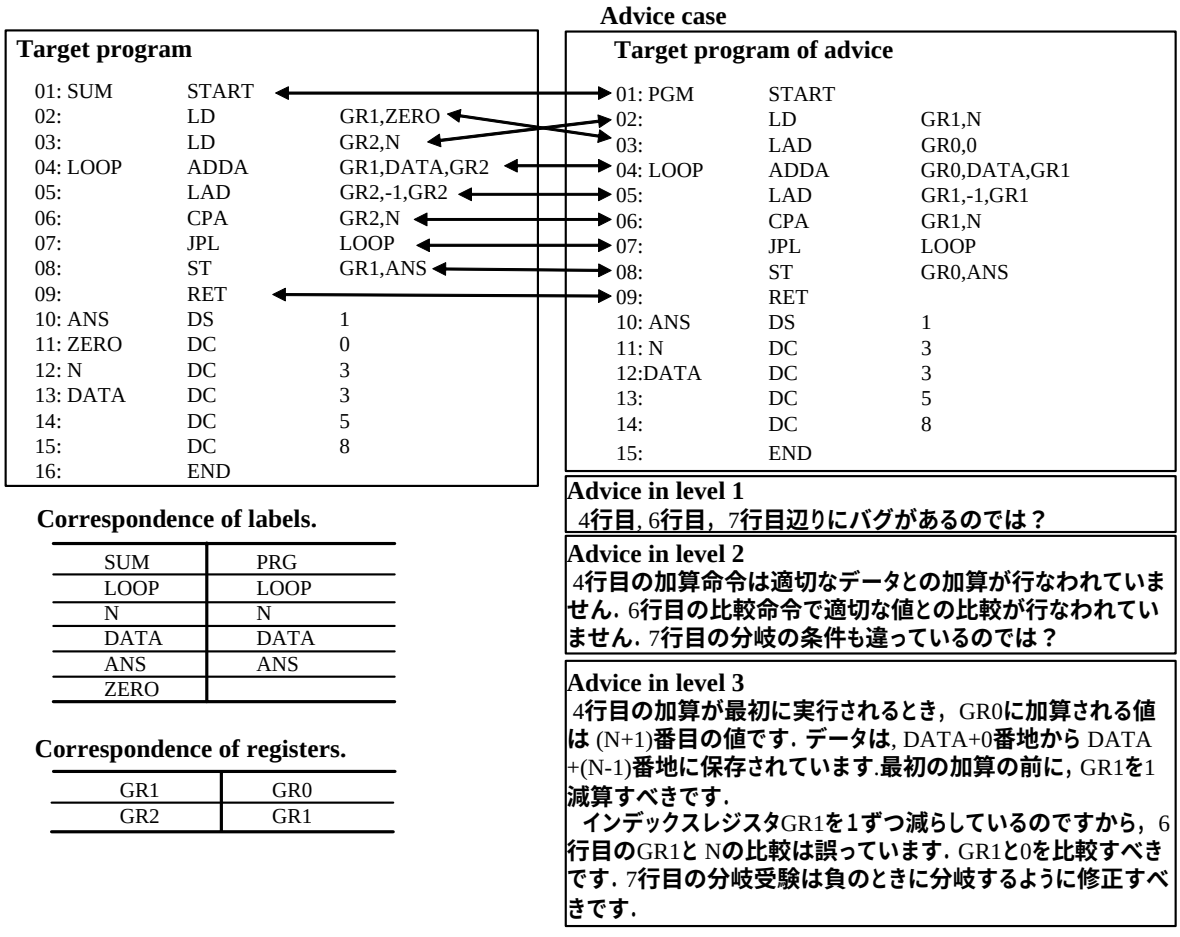

図 2　アドバイス事例とアドバイス対象プログラムの照合例

アドバイス事例は, 評価事例ベースのための事例ベースインデックス[3]とほぼ同じインデックスを付けて管理する.

4.2 アドバイス文生成処理

最初にアドバイス事例ベースのインデックスをたどり, アドバイス対象のニアミスプログラムに最も類似するアドバイス事例を検索する. 検索されたアドバイス事例のプログラムを学生のプログラムと詳細に比較を行い, 適用可能かどうかを調べる. この詳細な比較の処理をプログラム照合と呼ぶ. 検索処理とプログラム照合処理は評価支援システム[1～3]における処理と同じである.

選択された事例がアドバイス対象のプログラムに完全照合した場合, アドバイス事例のアドバイス文に簡単な修正を施して利用する. 完全照合の条件については文献[1,3]を参照されたい. ここでの修正は, プログラム照合処理によって明らかになった対応関係を用いて, ラベル名やレジスタ番号, 行番号を置き換える処理である. 類似事例が得られない場合や選択された事例が完全照合しない場合は, システムによるアドバイス文生成はできない.

図 2 の例では, アドバイス対象となる学生のプログラムはアドバイス事例に完全照合している. この場合, レベル 1 とレベル 2 のアドバイス文は修正無しで使用する. アドバイス文中に出てくる行番号は, 全て同じ行番号との対応が取れているからである. 一方, レベル 3 のアドバイス文については, 照合情報(図 2 左下の表)を用いて, GR0 を GR1 に, GR1 を GR2 に置き換える修正を施して使用する.

５．差異に基づく方法

5.1 差異に基づくアドバイス文生成の概要

アドバイス文を生成する第 2 の方法は, アドバイス対象のニアミスプログラムと正しいプログラムとの差異に基づく方法である. ここで, アドバイス文生成に利用される正しいプログラムを「ベースプログラム」と呼ぶことにする.

初等アセンブラプログラミングにおけるプログラム間の差異として, ベースプログラムを基準とすると, 以下のようなものが考えられる.

(a)誤命令：対応すると思われる命令の命令コードやオペランドに誤りがある.

(b)命令の不足：必要と考えられる命令が不足している.

(c)過剰な命令：不必要な命令が用いられている.

(d)順序の違い：命令を記述する順序が異なる.

(e)ラベル位置の違い：分岐先に用いられるラベルが付与されている命令が異なる.

以上, 5 種類の差異を扱うものとし, 差異の種類に応じたアドバイス文を生成する.

ベースプログラムは, 我々が既に開発した事例に基づく評価支援システムの評価事例ベースに存在するプログラムを利用する. 評価事例は, 評価事例は問題記述としてプログラムリストを持ち, 解記述として題意を満たしているかどうかの評価結果とアドバイス文を持つ[1,2]. ただし, このアドバイス文は, ニアミスプログラムへのアドバイス文生成には利用しない.

アドバイス文生成処理の全体としては, 評価事例ベースを検索し, 選択された事例を用いてアドバイス文を生成するといった流れになる.

5.2 事例検索とプログラム照合

まず, アドバイス対象のニアミスプログラムに最も類似するプログラムを持つ評価事例を検索する. 次に検索によって得た事例のプログラムリストとアドバイス対象のプログラムリストの詳細な比較(プログラム照合)処理を行う. プログラム照合によって, 対応関係情報と差異情報を生成する.

「対応関係情報」はベースプログラムと対処プログラム間の 命令の対応, レジスタの対応, ラベルの対応を示す対応表である. 一方,「差異情報」は, 2.2 で述べた差異の種類ごとに以下のような情報を持つ.

(a)誤命令：「対応するベースプログラムの行番号」と「対象プログラムの行番号」, 及び「誤り位置(命令コード, 第 1～3 オペランドなど)」.

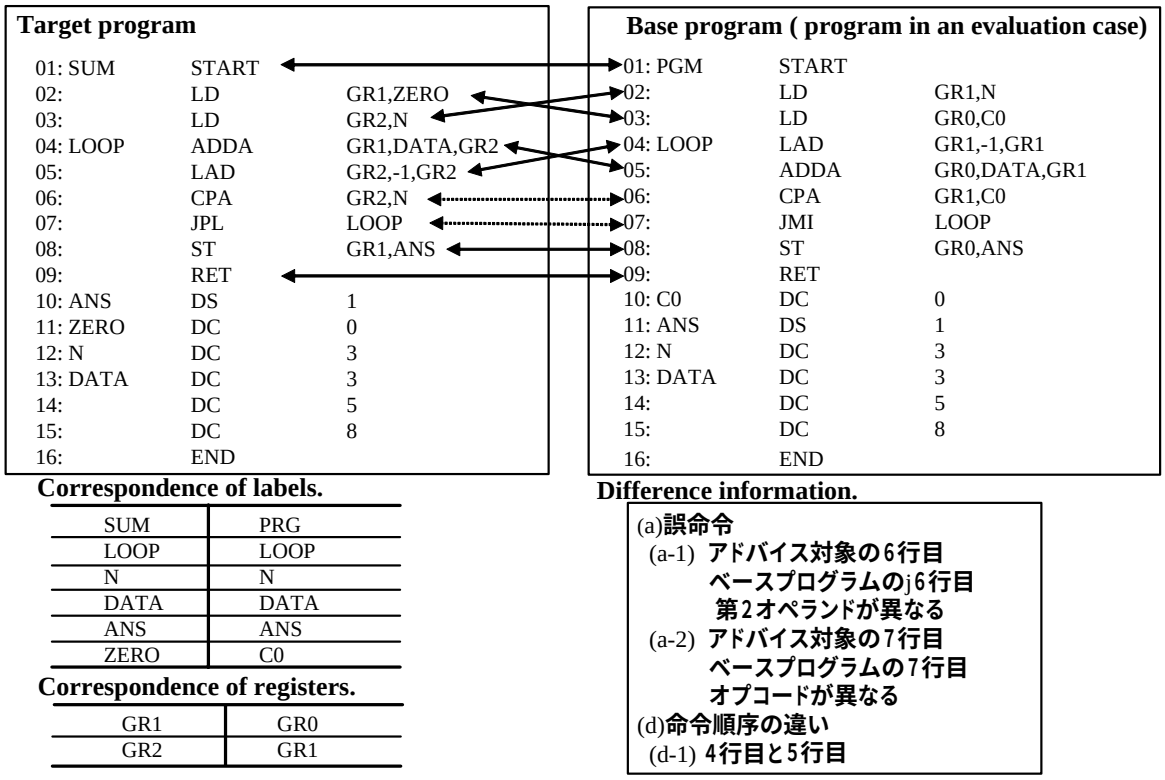

図 3　評価事例とのプログラム照合結果の例

(b)不足命令：対応先の無いベースプログラムの行番号.

(c)過剰命令：対応先の無い対象プログラムの行番号.

(d)順序の違い：順序が異なる範囲の対象プログラムにおける行番号リスト.

(e)ラベル位置の違い：「ベースプログラムの行番号」と「ラベルの違いの種類（どちらか一方にラベルが存在，または，ラベルが不一致)」.

プログラム照合処理は以下の流れで行う.

(1)対応候補の生成：命令コードが等しい命令は対応する可能性があるとして, 命令の対応表の候補として設定する. またその対応が成立する場合に, 対応付けられるレジスタやラベルの対応を, それぞれレジスタの対応表とラベルの対応表の候補とする.

(2)候補との絞込みと対応関係の決定：ベースプログラムと対象プログラムは完全には照合しないはずなので, 対応する個所を最大化するような照合戦略をとらなければならない. そこで, 最も多くの命令の対応を成立させる条件となるレジスタ, またはラベルの対応を優先して決定する.

(3)差異の同定：まず，ベースプログラムと対象プログラムで対応先を持たない命令を, それぞれ「不足命令」,「過剰命令」として拾い上げる. 次にそれらの比較を行い, 一部修正により対応すると判断できるものを「誤命令」と判定し,「不足命令」と「過剰命令」からは削除する．最後に，命令の対応情報を用いて,「命令の順序の違い」と「ラベル位置の違い」を同定する.

図 3 にアドバイス対象とベースプログラムの照合結果の例を示す．(2)の対応関係の決定までの処理で, 図 3 中の実践で示される命令の対応と左下のラベルとレジスタの対応表が生成される．次に(3)の差異の同定によって，まず, ベースプログラムの 6 行目と 7 行目を「不足命令」として，アドバイス対象の 6 行目と 7

行目を「過剰命令」として拾い出す．それらを比較することで，図 3 中の破線の対応関係を「誤命令」として同定し,「過剰命令」「不足命令」の差異は空になる．2 行目と 3 行目の順序の違いは些細であるので差異とはしないが，4 行目と 5 行目を「順序の違い」として同定する．

5.3 アドバイス文生成処理

プログラム照合で生成した「対応関係情報」と「差異情報」を用いて，詳細さのレベルに応じたアドバイス文を生成する. 最も簡単なヒント(レベル 1)では，差異の種類に関わらず，誤りのあると思われる行番号を提示し，レベル 2 以上では差異の種類に応じた文章を生成する．複数の種類の差異が存在する場合は, それら全てに対するアドバイス文を提示する．

最も簡単なヒント(レベル 1 のアドバイス)を提示するためには，まず，差異の存在する対象プログラムにおける行番号を求める. 不足命令については，1 つ前の命令が対応する対象プログラム内の行番号を用いる．次に， 求めた行番号の列に連続する部分が含まれている場合は，その部分を「x 行目～y 行目」という表現に変換する．

レベル 2 以降は差異の種類に応じた文章を生成する. 生成方法の詳細の説明は省略するが, 図 3 の例では以下のようなアドバイス文が生成される．

(レベル 1)　4 行目～7 行目辺りに原因があるのでは？

(レベル 2)　4 行目と 5 行目の命令の順序をチェックしましょう．

6 行目の命令に誤りがあるのでは？

7 行目の命令に誤りがあるのでは？

(レベル 3)　4 行目と 5 行目の命令の順序は，5 行目 4 行目の順序がよいのでは？

6 行目の命令のオペランドをチェックしましょう．

7 行目の命令の命令コードをチェックしましょう．

(レベル 4)　6 行目の命令のオペランドは N がよいのでは？

7 行目の命令の命令コードは JMI がよいのでは？

このように,「誤命令」に関する差異については 4 レベルのアドバイス文を生成するが, その他の差異については 3 レベルのアドバイス文を生成する．

6．実現したアドバイスシステム

これまで述べた手法によってアドバイスを提示するサブシステムを既に実現しているプログラム評価支援システムに付加した. システム全体構成を図 4 に示す．

学生が使用するクライアントシステムのプログラム提出用インターフェースに,「提出ボタン」に加えて,「アドバイス要求ボタン」を配置した．このボタンを押すことで, 学生のプログラムがサーバに送られ, サーバに設置されているアドバイス文生成システムによって処理される．

アドバイス文登録用インターフェースは, 正しく動作しないプログラムに対してアドバイス文を入力するために用いる. 正しく動作しないプログラムは, 学生によるアドバイス要求時やプログラムの提出時にシステムが収集する．アドバイス文が入力されたプログラムはアドバイス事例として利用可能になる. このインターフェースを学生にも公開することで, 余力のある学生にアドバイス事例作成を補助してもらうことも考えられるが, 今年度の授業では教員スタッフがアドバイス文を入力した．

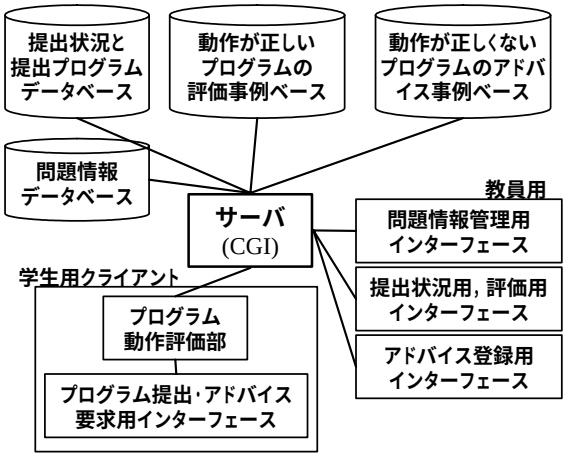

図 4　システム構成

表１　アドバイスが提示された割合

	事例有	事例無	合計
a 類似に基づく方法で提示	72	0	72
b 差異に基づく方法で提示	68	61	129
c アドバイス要求総数	226	260	486
提示率((a+b)/c×100)(%)	61.9	23.5	41.4

事例有：過去の評価事例が存在した 11 問.
事例無：出題時に評価事例ベースが空であった 17 問

表 2　アンケート結果

選択肢	人数
多くが役に立った	11
役に立ったこともあった	27
ほとんど役に立たなかった	11
アドバイス要求ボタンを使用しなかった	59

７．授業での試験利用

本学 2 年生を対象とした演習授業の 2 つのクラス(履修者は 91 人と 90 人)で, 本システムを試験的に使用した．出題した課題は延べ 28 問である．そのうち，11 問は過去に出題したことがあり, 今回の出題時に過去の評価事例ベースが存在していた．さらにその 11 問中 8 問については, 教員が正しく動作しないプログラムについてアドバイス文を入力しており, アドバイス事例ベースも存在した.

授業全体で 440 件のアドバイス要求があり，そのうち，アドバイスが提示されたのは 183 件であった. アドバイス要求の数が最も多かった問題では, 90 人の履修者中 18 人から全部で 61 件のアドバイス要求があった．表 1 にアドバイスが提示された割合を示す. 全体ではアドバイス要求の約 40%でアドバイスが提示されたが，評価事例が存在した場合は 60%以上でアドバイスが提示された.

アドバイス要求ボタンによるアドバイスが役に立ったかについてアンケートをとった結果を表 2 に示す. 履修者全員からの回答を回収できなかったが, 回答者の半数以上はアドバイス要求ボタンを使用していない. アドバイス要求ボタンを利用した学生の多くが「多くが役に立った」「役に立ったこともある」と回答しており，本システムの有用性が示唆された.

８．おわりに

初等アセンブラプログラミングを対象として, 動作は正しくないが完成に近いプログラム(ニアミスプログラム)に対するアドバイスを自動的に提示する方法を提案した. 提案した手法に基づいてアドバイスシステムを実現し, 実際の授業で試験的に利用した結果, その有効性が示唆された.

謝辞　本研究の一部は，科学研究費補助金 No. 14580428 の補助による.

参考文献

[1] 渡辺博芳，荒井正之，武井惠雄：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，情処論, Vol.42, No.1, pp.99 ～ 109, 2001.

[2] Watanabe,H., Arai,M. and Takei,S. : Case-Based Evaluation of Novice Programs, Proc. of AI-ED 2001, San Antonio pp.55 ～ 64, 2001.

[3] 渡辺博芳，荒井正之，武井惠雄：初等アセンブラプログラミング評価支援のための事例ベース構築法，情処論，Vol.44, No.2, pp.496～506 (2003).

[4] 渡辺博芳，荒井正之，武井惠雄：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, 情処研報, Vol.2001-CE-59, pp.69 ～ 74, 2001.

[5] A. Adam and J. P. Laurent, LAURA, A System to Debug Student Programs, *Artificial Intelligence*, **15**, pp.75～122, 1980.

[6] W. L. Johnson, Understanding and Debugging Novice Programs, *Artificial Intelligence*, **41**, pp.51～97, 1990.

[7] H. Ueno, Concepts and Methodologies for Knowledge-Based Program Understanding - The ALPUS's Approach, *IEICE Trans. Inf. & Syst.*, **E78-D**, No.2, pp.1108～1117, 1995.

[8] 海尻賢二:ゴール／プランに基づく初心者プログラムの認識システム，信学論, Vol.J78-D-II, No.2, pp.321～332, 1995.