コンピュータと教育 56－2
（2000．5．31）

事例ベース推論によるプログラム評価支援

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，事例ベース推論によって，プログラミング演習授業におけるプログラム評価支援を行う一般的枠組みを提案する．最初に，評価アシスタントとしてのプログラム評価支援システムとその前提となる提出環境を定義した後，事例ベース推論による実現方法について詳述する．事例ベース推論による評価支援システムは，提出されたプログラムの評価を行う際に，過去の事例を検索し，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，その事例の評価結果を評価対象のプログラムに適用する．提案したアプローチの有効性を示すために，CASLによる初等アセンブラ・プログラミングを対象として実現したシステムとその評価試験について述べる．さらに，本研究によって明らかになった，今後取り組むべき課題について考察する．

Case-Based Evaluating Assistant System for Programming Education

Hiroyoshi Watanabe, Masayuki Arai and Shigeo Takei
Dept. Information Sciences
School of Science and Engineering, Teikyo University

This paper presents a method of implementing an evaluating assistant system that supports teachers' evaluation work of students' programs using case-based reasoning. The evaluating assistant system separated from a program submission environment prerequisite for the assistant system is defined, and a general framework of case-based approach for implementing the evaluating assistant system is proposed. The case-based evaluating assistant system compares a program submitted by a student with evaluation cases in the case-base. If some case matches the program, the system applies the evaluation results on the case to the program. The effectiveness of the proposed method and future works are discussed based on a experience of implementing a case-based evaluating assistant system for novice programs written in an assembly language, CASL.

1　まえがき

初等プログラミング演習授業において，学生数が多くなると，学生が作成したプログラムを教員が評価する作業の負荷も増大する．そこで，計算機によってプログラム評価作業を支援することが考えられる．我々は，これまで，CASLによる初等アセンブラ・プログラミングを対象として，学生が作成したプログラムが問題の題意を満たしているかどうかの判定とアドバイスの作成作業を，事例に基づいて支援するシステムを開発した[1～3]．開発したシステムを実際の授業で使用したところ，良好な結果を得ている[4]．本システムで採用した事例ベース推論[5～8]によるアプローチは，アセンブラ・プログラミングのみでなく，他の言語によるプログラミング演習においても適用可能と考えられる．本稿では，本システムの開発経験を踏まえ，事例ベース推論によるプログラム評価支援の一般的な枠組みについて考察する．

2　プログラム評価支援システム

2.1　対象とする評価作業

本論文で対象とするプログラム評価は，提示した課題に対して学生が作成した(提出した)プログラムに対する以下の2つの作業である．

(1) 提出されたプログラムが題意を満たしているかどうかの判定：教員が問題を提示する際には，学生に習得させたい概念等に関する教育的な意図

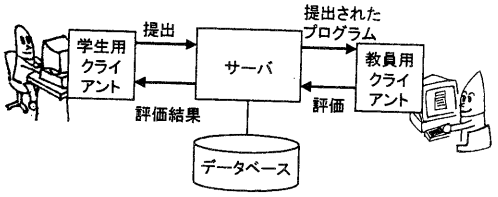

図 1: 前提となる電子的な提出環境

がある．教員は提出されたプログラムを分析し，意図していた概念を学生が習得したかどうかの評価を行い，学生にアドバイスをしたり，場合によってはプログラムの再提出を求める．本研究が対象とする評価作業の1つは，このような，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を判定する作業である．以降では，題意を満たしている場合は「合格」，そうでないときは「不合格」と言い，この題意を満たすかどうかの評価作業を「合否判定」と呼ぶことにする．「不合格」という表現は，現在判定対象となっている，そのプログラムが不合格であることを示し，提示された問題について学生が不合格となるのとは異なることに注意されたい．また，学生がプログラムを提出するのは1度だけでなく，合格に至るまで何度も提出を繰り返すことを想定している．

(2) 提出されたプログラムに対するアドバイスの作成：本研究で対象とする2つめの評価作業は，アドバイス文の作成である．アドバイスは，プログラムが題意を満たすか否かに関わらずに，必要に応じて与える．プログラムが題意を満たさない場合，どのような点が題意を満たしていないかアドバイスが必要である．また，題意を満たす場合でも，よりよいプログラムにするためのアドバイスを与える．

2.2 前提となるプログラム提出環境

前提となる提出環境は，図1に示すようなコンピュータネットワークでの電子的なプログラム提出である．学生が提出したプログラムはサーバに蓄えられる．教員はサーバにアクセスしてプログラムの評価を行う．評価結果(合否判定とアドバイス)は電子メールなどで学生に送られる．図1のような環境を整えるだけでも，以下のような利点がある．

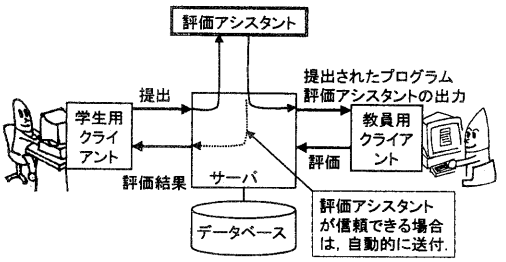

図 2: プログラム評価アシスタントとしての支援システム

・提出者一覧や提出時刻などを簡単にチェック可能．

・提出期限を自動的に設定することが可能．

・提出物や評価結果などが電子的に蓄積可能．

レポートやプログラムの評価を支援するシステムの実現に関するほとんどすべての研究において，評価対象を電子的に提出させている[9〜15]．今後，この分野の研究の活性化とシステムの実用化のためには，これらの研究から課題の提出に共通な部分を抽出して，インフラとしての提出環境を実現し，その提出環境に，個々の研究で独特な部分を追加モジュールとして付加可能な枠組みを実現することが重要となるであろう．

2.3 プログラム評価アシスタントとしての支援システム

図1の環境に，評価アシスタントとしてのプログラム評価支援システムを導入することを考える．これを図2に示す．教員が評価を行う前に，評価アシスタントシステム[1]が予備的な評価を行い，教員はアシスタントシステムの評価結果を参考にしながら，最終的な評価を行う．アシスタントシステムに評価を任せられる場合は，アシスタントシステムの評価結果を直接学生に送ることも考えられる．このような評価アシスタントの存在によって教員の評価作業の負荷軽減が期待できる．

評価アシスタントシステムは，「評価結果」，「その評価結果の根拠」，「確信度」の3つの組を出力する．これらについて以下に説明する．

[1]以降，評価アシスタントとしての評価支援システムであることを強調する際には，評価支援システムを評価アシスタントシステムと呼ぶ．

(1) 評価結果：評価結果は合否判定とアドバイスである．合否判定は,「合格」または「不合格」のいずれかの値をとる．アドバイスは自然言語で記述するので，任意の文字列を値とする．

(2) 評価結果の根拠：評価アシスタントシステムがどのようにしてその評価結果を導いたかを説明するための情報である．その値の形式は，評価アシスタントシステムの実現方法に依存して異なる．例えば，ルールベース推論のアプローチであれば，評価結果を導くために適用されたルール列が値となり，事例ベース推論であれば，適用した事例が値となる．

(3) 確信度：確信度は，Surely，Probably，Unknownのいずれかを値とする．評価結果に十分な確信がある場合は，Surelyであり，評価結果を導けない場合はUnknown，その他の場合は，Probablyである．つまり，Probablyは，十分な確信はないものの，何からの評価結果を導いた場合である．確信度がUnknownの場合は，評価結果と評価結果の根拠は値を持たない．

評価アシスタントシステムが，確信度Surelyで評価結果を出力した場合に限れば，アシスタントシステムの評価能力が教員と同等であることが望まれる．そうであれば，確信度がSurelyの場合は，システムの評価結果を教員がチェックせずに直接学生に送ることができる．

評価アシスタントシステムは，教員の最終的な評価結果を用いて，評価能力を向上させるための学習機能を持つことができる．教員が1つのプログラムを評価する度に，漸増的に学習を行うのが望ましい．

3 事例に基づくプログラム評価

3.1 事例ベース推論とその利点

前節で述べた評価支援システムを実現する手法として，事例ベース推論が有効である．事例ベース推論は新しく与えられた問題を解決するために，過去の事例を直接利用する推論法である[5〜8]．事例に基づくプログラム評価は,「あるプログラムの評価を行う際に，過去の事例を検索し，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，その事例の評価結果を評価対象のプログラムに適用すること」である．

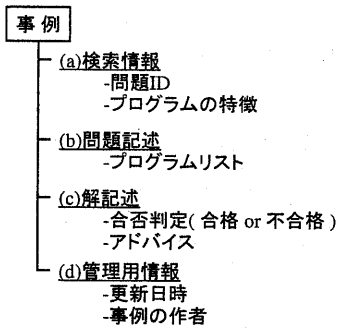

図 3: 事例の表現

事例ベース推論が評価アシスタントとしての支援システムの実現に有効な理由を以下に述べる．

・評価に事例を直接利用するので，従来の知識ベースのアプローチ[16〜21]に比べて，ヒューリスティクスが大幅に少ない．

・評価結果の根拠として，評価に用いた事例を示せばよいので，特別な説明機能が不要である．

・与えられた問題と事例との照合の度合いを基にして，確信度を決定できる．

・評価アシスタントシステムの学習機能を事例の追加によって，容易に実現できる．

・教員自身の学習も効率化できる．つまり，学生の作成したプログラムを評価する過程で，教員自身も学習している．事例ベース推論を用いれば，教員が過去に評価したことのあるプログラムは評価アシスタントシステムが評価可能である．過去の事例ではカバーできないプログラムが確信度Unknownになるので，教員が見たことのないタイプのプログラムを容易に弁別できる．

3.2 事例の表現

事例ベース推論において，事例は一般に問題記述と解記述から成り，これに検索のための検索情報を付加する．すなわち，評価支援システムが用いる事例は図3に示すように，(a) 検索情報，(b) 問題記述，(c) 解記述，(d) 管理情報から構成する．

(a) 検索のための情報としては，どの問題に対し

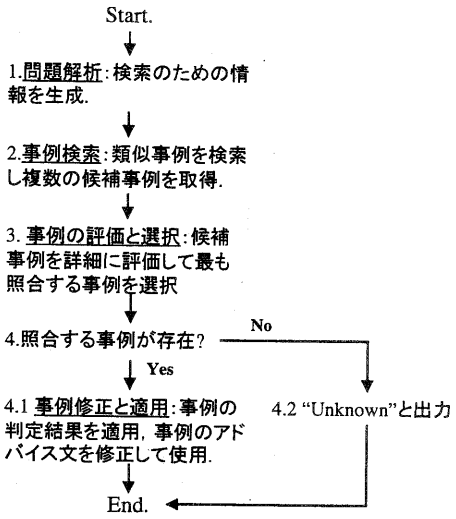

図 4: 事例に基づく評価処理の流れ

て作成したプログラムかを示すための問題識別子とプログラムの特徴を持たせる．プログラムの特徴として，どのような情報を持たせるかは，対象とする言語に依存する．例えば，C言語ならば，if文の数，while文の数などが考えられる．

(b) 問題記述はプログラムリストである．同じ実現方法によるプログラムでも，プログラムリストレベルでは多くのバリエーションが存在するため，プログラムリストは，標準化した形式[18, 22]や一般化した形式[1]で記述することが望ましい．プログラムリストの形式は当然，対象とする言語依存となる．

(c) 解記述として評価結果(合否判定とアドバイス；2.3参照)を持つ．

(d) 管理情報として，事例が追加，または内容の更新が行われた日時と追加や更新を行った教員の名前を持たせる．

3.3 事例に基づく評価処理の流れ

事例ベース推論によるプログラム評価処理の流れを図4に示す．

(1) 問題解析：提出されたプログラムリストから，検索のための情報を生成する．

(2) 事例検索：問題解析で生成した情報を用いて事例を検索する．最も単純な検索方法は，特別な情報を用いずに，対象問題識別子が同一の事例をすべて取得する方法であるが，一般に事例の評価プロセスは計算量が多くなるので，この段階で，与えられたプログラムと照合しない事例を刈り込むことが望ましい．

(3) 事例の評価と選択：事例の評価は，提出されたプログラムと事例のプログラムリストが同じ実現方法かどうかを調べるために，それらのプログラムの照合を行う．候補となるすべての事例との照合を行い，最も照合度の高い事例を選択する．完全に照合すれば，事例の判定とアドバイスがほとんどそのまま利用可能であり，確信度はSurelyとなる．完全照合ではないが，ある一定の照合条件を満たす場合は，確信度をProbablyとして，事例の判定とアドバイスを利用する．

照合の条件や照合プロセスは対象とする言語に依存する．つまり，対象とする言語ごとに，まず，完全照合の条件(確信度Surelyで事例の評価結果が適用可能な条件)と照合の条件(確信度Probablyで事例の評価結果が適用可能な条件)を定義し，その上で，それらの条件を検査するためのプログラム照合アルゴリズムを設計する必要がある．ある言語に関するプログラム評価支援システムを実現する場合に，この部分が最も重要なポイントとなる．

(4) 事例の適用：照合する事例が存在する場合は，最も照合度の高い事例の評価結果を提出されたプログラムに適用する．「評価結果を適用する」とは以下の2つの処理を示す．

(a) 事例の合否判定を提出されたプログラムの判定結果とする．

(b) 事例のアドバイス文を必要に応じて修正して，提出されたプログラムに対するアドバイスとして利用する．例えば，事例のアドバイス文中の変数名や行数をそれらに対応する変数名や行数に書き換える．このような修正は，事例照合時に生成された照合情報を用いて行うことができる．

照合する事例が存在しない場合は，確信度をUnknownとする．

3.4 事例ベースの更新と管理

事例ベース管理では，評価支援システムの能力を向上させるために新しい事例を追加する．つまり，

確信度がSurelyでないプログラムについて，そのプログラムリストと教員の評価結果の組を新事例として事例ベースに追加する．確信度がSurelyであったプログラムに対しても，評価支援システムの評価結果と教員の評価結果が異なる場合は，新事例を追加する．

また，同じプログラムに対する教員の評価結果は必ずしも一定ではないことを考慮した事例ベース管理も必要となる．つまり，教員は人間であるので，ミスをすることもあろうし，教員の評価基準が状況や時間経過とともに変化することも考えられる．これについては5章で詳述する．

4 初等アセンブラ・プログラミングを対象とした実現システム

4.1 システムの概要

実現したシステムは学生用クライアント，教員用クライアント，サーバから構成される．

(1) 学生用クライアント：学生用クライアントは，CASLプログラムの編集，アセンブル，シミュレート機能を持つシミュレータ[23]にプログラム提出機能と，プログラムの動作評価機能を追加したものである．これは，Windowsアプリケーションとして実現した．

(2) 教員用クライアント：教員用クライアントは，WWWブラウザでパスワードによってセキュリティのかかったページにアクセスすることで実行する，従って，教員は，Webベースの使い易いインターフェースで，提出状況の閲覧機能，判定入力支援機能，提出されたプログラムの閲覧機能を実行できる．これらの機能は，サーバにhttpのCGIプログラムとして実現した．

(3) サーバ：サーバは，問題情報，事例，一般化ルール等のプログラム評価に必要な知識，提出されたプログラム，学生の提出状況データを保持し，クライアントにおける操作のログを記録する．事例に基づくプログラム評価処理はサーバで行う．

なお，現バージョンでは，提出環境と評価アシスタントとしての支援システムを明確に分離していない．

4.2 ドメイン依存部分の実現方法

本節では，CASLという言語を対象とした本システムに依存する部分について述べる．

(1) プログラムの動作評価：アセンブラ言語で記述されたプログラムは，シミュレータで動作させることで，プログラムの動作が題意を満たしているかどうかを比較的容易に検査できる．そこで，事例ベース推論によるプログラム評価を行う前に，プログラムの動作評価を行うこととした．プログラムの動作評価は学生用クライアントで行う．プログラムの動作が正しくない場合は，その旨のメッセージを提示し，動作が正しいプログラムのみを事例ベース推論によるプログラム評価の対象とした．

(2) 事例の表現：事例の表現は3.2節で述べた形式に従うが，検索のための情報としてのプログラムの特徴は持たない．また，プログラムリストは，CASLによる表記か，またはCASLを一般化した形式とする．CASLの一般化形式は独自に定義した[1]．

(3) 事例検索：本システムにおける事例検索は同じ問題に対する事例をすべて取得する処理である．すなわち，候補事例の刈り込みは行っていない．

(4) 事例評価(プログラムの照合)：プログラムの照合は事例のプログラムリストと評価対象のプログラムリストの間で，命令，ラベル，レジスタの矛盾のない対応をとる処理である[3]．照合の結果，以下の条件1を満たす場合は，「評価対象と事例は照合する」といい，特に，条件2を満たす場合は，「完全照合」という．

- 条件1 事例のプログラムの全ての命令が評価対象プログラムの命令に対応先を持ち，事例のプログラムの命令に対応付けられない評価対象プログラムの命令が冗長命令である．
- 条件2 事例と評価対象プログラムの個々の命令が1対1に対応しており，それらの順序関係の違いが些細なものである．

最も照合度の高い事例が条件2を満たす場合は，確信度をSurelyとし，条件1を満たすが，条件2を満たさない場合は，確信度をProbablyとする．条件1を満たす事例が存在しない場合，確信度をUnknownとする．

(5) 事例修正：事例の修正は3.3節で述べた通り，アドバイス文の軽微な修正のみを行う．具体的に

は，対応するラベル名およびレジスタ名の置き換えと「行目」というキーワードに着目した行番号の置き換えである．

(6) 事例ベース更新：確信度がSurelyでない場合，または確信度がSurelyで教員の判定が異なる場合に新しい事例を追加する．

(7) 自動モードと手動モード：システムがSurelyで評価可能な場合は自動的に結果を学生に通知するモード(自動モード)とすべてのプログラムを教員も評価するモード(手動モード)を設けた[3]．

4.3 実現システムの評価試験

4.3.1 評価試験の条件

本学2年生を対象とした1999年度の2クラスの演習授業(仮にA組，B組と呼ぶ)において，実現したシステムを実際に使用した．授業を履修した学生の人数は，A組が79人，B組が73人である．全員が合格となるまで，プログラムの再提出を行うので，プログラムの提出数は，学生数よりも多くなる．本システムを使用した際に出題した問題を以下に示す．

- P1:2つの値の大きい方を求める(B組)
- P2:N個のデータの加算(B組)
- P3:N個のデータの最大値を求める(B組)
- P4:Nビットの循環右シフト(B組)
- P5:Nビットの循環左シフト(A組)
- P6:スタックを使って()の対応検査(B組)
- P7:スタックを使って文字列反転(A組)

各問題について，以下の値を集計する．

(a) 総提出数

(b) 動作の正しいプログラム数

(c) 教員の評価を必要とするプログラム数の割合(%)

(d) 教員が行う必要のある評価作業の割合(%)：プログラムの評価作業が「動作の評価」,「実現方法の評価」,「アドバイスの作成」の3つのタスクから成ると仮定し，教員が行う必要のあるタスク数で評価．(c)と(d)の集計方法の詳細は文献[4]において報告した．

(e) 事例ベース活用率(%)：動作の正しいプログラム(事例に基づく評価の対象となったプログラム)のうち，システムが事例を適用して評価結果を得たプログラムの割合.(Unknownでなかったケースの割合)

(f) 判定精度(%)：動作の正しいプログラムでシステムが評価を行った(Unknownでなかった)ケースのうち，システムと教員の判定結果が等しいケースの割合．(文献[4]とは分母が異なることに注意)

(g) システムが生成したアドバイス文をそのまま使用できた割合(%)：動作の正しいプログラムでシステムが評価を行ったケースのうち，システムの生成したアドバイス文を教員が書き換えること無しに使用できたケースの割合．(「特にアドバイス無し」も含む)

表 1: 各問題に対する評価結果

	P1	P2	P3	P4	P5	P6	P7
(a)	119	119	140	156	167	157	136
(b)	75	80	96	102	82	78	79
(c)	11	10	17	22	23	26	27
(d)	7.3	11	14	22	20	21	26
(e)	93	94	80	71	55	59	66
(f)	100	100	97	100	100	100	100
(g)	93	69	88	67	84	91	46

(a) 総提出数，(b) 動作の正しいプログラム数，(c) 教員の評価を必要とするプログラム数の割合(%)，(d) 教員が行う必要のある評価作業の割合(%)，(e) 事例ベース活用率(%)，(f) 判定精度(%)，(g) システムが生成したアドバイス文をそのまま使用できた割合(%)

4.3.2 評価試験の結果と考察

前節で述べた値の集計結果を表1に示す．

(1) 教員の作業軽減効果：表1の(c)(d)に着目すると，簡単な問題(P1やP2)で10%程度に，やや複雑な問題(P4～P7)でも20～30%程度に軽減されている．このことから，入門編のアセンブラプログラミング教育で用いる程度の問題において，本システムによる教員の評価作業軽減の効果は顕著である．

(2) 事例ベース活用率：表1の(e)に着目すると，簡単な問題(P1やP2)で90%以上，やや複雑な問題(P4～P7)でも50%以上の活用率である．

(3) 判定精度：表1の(f)はほぼ100%であり，確信度がSurelyの場合はP3も100%となるので，十分な精度を達成していると言える．

(4) システムの生成したアドバイスをそのまま

利用できた割合：表1の(g)に着目すると，P7では低い値となっているものの，全体としては比較的高い利用率である．アドバイスが判定精度に比較して低い値となるのは，主に2つの原因がある．以下にその原因と今後の対策法を示す．

1つめの原因は実現したアドバイス文の修正機能が貧弱であり，適切な修正が行えていないことである．実現したシステムでは，単に文字列としての置き換え操作をしているために，適切に修正できないケースがある．形態素レベルでの置き換え操作を行うことで対処したい．

第2の原因は，同じプログラムに対しても，教員はアドバイス文の表現をよりよく変えることがあることである．実現したシステムでは，合否判定結果のみに着目して新事例を追加していたが，アドバイス文が修正された場合にも新事例を追加することで，多少改善されると考えられる．

5 検討すべき課題

本章では，本研究により明らかになった，今後取り組むべき課題について述べる．(1)と(2)は事例に基づくプログラム評価に限らず，プログラムやレポート評価支援の研究全体に関すること，(3)と(4)は事例に基づくプログラム評価の研究に関することである．

(1) インフラとしてのレポート提出環境の整備: 事例に基づくプログラム評価に限定されたことではないが，先にも述べように共通のプラットフォームを実現することが重要と言える．つまり，これまでのプログラムやレポートの評価支援や演習支援に関する研究成果[1〜4,9〜22]を踏まえて，インフラとしての提出環境を実現し，それを共通のプラットフォームとする．その提出環境に，個々の研究で独特な部分を追加モジュールとして付加可能な枠組みを実現することが重要と思われる．インフラとしての提出環境が実現されていれば，その部分を作らずに評価支援の研究開発が行える．また，多くの研究者や教員の間で提出環境を共有すれば，個々の研究で独特な部分（追加モジュール部分）の相互利用も促進される．

(2) 考慮すべきこと：評価支援システムにおいて，以下のことを考慮する必要がある．

・教員も人間であるのでミスを犯す可能性があることを前提とする．

・教員の判断基準は，個々の教員で異なる可能性もあるし，同じ1人の教員でも，様々な要因によって変化する可能性があることを前提とする．

特に後者について，出題された文章表現の範囲では同じ問題であっても，その演習において教員が学生に習得して欲しいと考えるポイントが異なれば，評価基準も異なることは十分あり得る．また，評価作業を行う上で，教員自身も学習をしていると捉えると，教員自身が学習した結果が評価基準に影響を与える可能性もある．このような評価支援システムを取り巻く条件の変化に柔軟なシステムを実現することが重要である．

(3) プログラム照合条件と照合処理の検討：プログラムの照合処理は，2つのプログラムが同じかどうかを判断するために行う．同じプログラムでもいくつものバリエーションが存在するため，バリエーションを除去するための工夫は古くから行われている[15〜22,24]．ここで，「2つのプログラムが同じかどうかの判断」の条件は，目的とするタスクによって異なることに注意する必要がある．これまでの研究の多くは，「プログラムのバグの所在を指摘する」ことを目的とする場合のバリエーションの除去であり，「プログラムの合否判定やアドバイス」を目的とする場合は今後さらに検討が必要である．例えば，文献[22]では，プログラムのバグ診断におけるバリエーション除去の方法の1つに，PASCALのレコード型で書かれている部分を単純な型に書きかえる処理をあげているが，合否判定やプログラムに対するアドバイスを目的とする場合，レコード型を使ったプログラムと使っていないプログラムは区別する方がよいと考えられる．

(4) 事例ベースの管理・洗練化法の検討：新事例を追加するだけでなく，新しく教員の評価結果を得る度に，事例ベースを再構成するような，より高度な事例ベースの管理，あるいは洗練化方法を検討する必要がある．特に，同様な複数の事例を一般化することでまとめておければ，システムの性能を落とさずに事例数を削減し，処理の効率化が図れる．また，(2)で述べたように，教員の判断基準の変化によって，事例ベースに矛盾が生じた

場合に，それに対処することも重要である．事例ベースの洗練化の方法には，事例の一般化と特殊化や事例の忘却[25]が有効であると考えられる．

6 むすび

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の合否判定とアドバイスの作成を支援するシステムを実現した経験に基づいて，より一般的な事例ベース推論によるプログラム評価支援システムの枠組みを提案した．また，その一般的枠組みと実現したシステムとの関係について述べることで，提案した一般的な枠組みの有効性を示唆した．最後に，実用的かつ柔軟なプログラム評価支援システムを実現するための課題について述べた．今後，これらの課題について様々な角度から取り組むことが望まれる．

参考文献

[1] 渡辺博芳，荒井正之，武井恵雄：CPUとアセンブラ授業のための合否判定支援システム，情報処理学会研究報告，コンピュータと教育，Vol. 98, No. 48, pp.61 – 68，1998.

[2] Watanabe,H., Arai,M. and Takei,S. : Automated Evaluation of Novice Programs Written in Assembly Language, Proc. of ICCE99, Vol.2, pp.165–168，1999.

[3] 渡辺博芳，荒井正之，武井恵雄：CPUとアセンブラ授業のための事例に基づくプログラム評価支援システム，情報処理学会研究報告，コンピュータと教育，Vol.99, No.54，pp.33 – 40，1999.

[4] 渡辺博芳，荒井正之，武井恵雄：CPUとアセンブラ授業のための事例に基づくプログラム評価支援システム －授業での実用に基づく評価－，情報処理学会第60回全国大会講演論文集，pp.4-439 – 440,5M-9，2000.

[5] 奥田健三，山崎勝弘：事例ベース形推論とその応用例，情報処理，Vol.31, No.2, pp.244–254,1990.

[6] 特集「事例ベース推論」，人工知能学会誌，Vol.7, No.4, pp.558 – 607, 1992.

[7] Kolodner,J. : Case-Based Reasoning, Morgan Kaufmann Publishers,Inc., 1993.

[8] Leake,D. ed, : Case-Based Reasoning : Experiences, Lessons and Future Directions, AAAI Press / MIT Press, 1996.

[9] Konishi,T., Suyama,A. and Itoh,Y.: Evaluation of Novice Programs Based on Teacher's Intention, Proc. of ICCE95, pp. 557–566, 1995.

[10] 藤原祥隆，松西年春，岡田信一郎，大鎌広，後藤寛幸，黒丸鉄男：プログラミング演習支援のための階層分散処理システムの設計と評価，電子情報通信学会論文誌，Vol.J78-D-II, No.11, pp. 1701–1709, 1995.

[11] 吉野孝，宗森純，伊藤士郎，長澤庸二：教育用プラットフォームDEMPO II の開発とプログラミング演習への適用，情報処理学会論文誌，Vol.37, No.5, pp. 891–901, 1996.

[12] 服部徳秀，石井直宏：プログラミング演習の評価サポートシステムの構築，教育システム情報学会誌，Vol.14, No.1, pp. 21–28, 1997.

[13] 関本理佳，海尻賢二，山形昌也：ネットワークを利用したレポート受付・評価支援システムの実現，教育システム情報学会誌，Vol.14, No.5, pp. 217–222, 1998.

[14] 曽 超：演習評価サポートシステムOPECについて，情報処理学会第60回全国大会講演論文集，pp.4-435 – 436,5M-7, 2000.

[15] 福田勇一：プログラム構造比較システム，情報処理学会第45回全国大会講演論文集，pp.1-5 – 6, 5X-3, 1992.

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

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

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

[19] 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.

[20] 海尻賢二 ：2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，電子情報通信学会論文誌，Vol.J78-DII, No.2, pp.321 – 332, 1995.

[21] Kim,S. and Kim,J.H.: Algorithm Recognition for Programming Turoring Based on Flow Graph Parsing, Applied Intelligence, Vol. 6, Iss. 2, pp. 153–164, 1996.

[22] Ueno,H.: A Program Normalization to Improve Flexibility of Knowledge-based Program Understander, IEICE Trans. Inf. & Syst., Vol.E81-D, No.12, pp.1323 – 1329, 1998.

[23] 渡辺博芳 ： WCASL, CASL & COMET Simulator for Windows, http://www.ics.teikyo-u.ac.jp/wcasl/.

[24] 服部徳秀，石井直宏：ソースコードのバリエーション除去システム，電子情報通信学会論文誌，Vol.J80-DI, No.1, pp.50 –59, 1997.

[25] Watanabe,H., Okuda,K. and Yamazaki,K.: Methods for Adapting Case-bases to Environments, IEICE Trans. Inf. & Syst., Vol.E82-D, No.10, pp.1393 – 1400, 1999.