コンピュータと教育 54－5
（1999．12．6）

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

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の評価作業を支援する方法について述べる．本研究で対象とする評価作業は，評価対象の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判定とそのプログラムに対するアドバイスの作成である．そのようなタスクを計算機で行う方法として，(1) プログラムの動作の評価と (2) 過去の評価事例に基づくプログラムの実現方法の評価を行う手法を提案する．また，実現した評価支援システムにおいては，事例の豊富さに応じて評価モードを選択可能，教員を支援するためのWebベースの使い易いインターフェースなどの特徴を持たせた．本システムを実際の授業で用いて評価を行ったところ，教員のプログラム評価の作業負荷を大幅に減少できることが示された．

A Case-Based Evaluating 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 describes a method of supporting the evaluation of students' programs for the CPU and Assembler course. In the course, COMET and CASL are used as teaching material. The target evaluation tasks are to judge whether a student's program satisfies the requirements of the given problem and to give advice for the student program. We propose a method to perform the evaluation tasks by two processes: (1) evaluating the program's action and (2) evaluating the implementation based on cases. The implemented system has a web-based user interface to support teachers' evaluation work. The system was utilized for an actual class and the results showed that the system reduced the teacher's evaluation work drastically.

1 まえがき

初等プログラミング演習授業において，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．しかし，学生数が多くなると，教員の評価作用は増大する．このようなプログラムやレポートの評価において，総合的な評価は教員が行うべきであるが，比較的低レベルな作業を計算機に行わせることで，評価作業を効率化できるはずである．このような考えに基づき，我々は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を支援するシステムを開発した[1, 2]．このシステムでは，題意を満たすプログラムを解答例プログラムとして用意しておき，学生のプログラムの動作が正しい場合に，解答例プログラムと比較し，解答例プログラムと照合した場合は題意を満たすと判定する．教員が判定を行うのは，照合する解答例プログラムが存在しない場合のみなので，判定作業が軽減されることが期待された．しかし，実際には十分な解答例プログラムをあらかじめ準備するのは困難である．

そこで，本研究では，事例ベース推論[3, 4]のアプローチを導入することで，システムの改善を図る．主な改善点は，(1) 評価事例として，題意を満たすプログラムの他に，題意を満たさないプログラムも保持し，(2) システムが評価できないプログラムを教員が評価した際に，その結果を新たな評価事例として自動的に追加することで，システムの評価能力を高めること，(3) 学生が作成したプログラムが題意を満たしているかどうかの判定に加えて，学生に対するアドバイスの作成も支援すること，(4) 教員用の使い易いインターフェースを実現すること，である．

2 プログラム評価基準と評価処理の流れ

2.1 評価基準

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がある．教員は提出されたプログラムやレポートを分析し，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やレポートの再提出を求める．本研究が対象とする評価作業は，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．

合否判定は以下の2つの基準に基づいて行う．

(a) プログラムの動作が問題の題意を満たしていること．

(b) プログラムの実現方法が問題の題意を満たしていること．

(a) は問題を解くという意味で必須条件である．(b) は上で述べた教育的な意図が達成されたかどうかということに関連する．例えば，出題された問題文に「スタックを用いて」という記述が含まれているとき，教員は，学生がスタックについて学ぶことを意図している．この場合，題意通りに動作するプログラムであっても，スタックが使われていなければ，教員は「実現方法は題意を満たしていない」と判断するであろう．

2.2 評価方針

前節で述べた評価基準をテストするために，プログラム評価処理は，プログラムの動作の評価とプログラムの実現方法の評価の2つのフェーズに分けて行うこととする．

プログラムの動作評価はあらかじめ複数のテストデータを用意しておき，それらのデータに対する動作をテストする．このような評価は，人間よりもむしろ計算機の方が得意であり，完全に自動化可能である．

プログラムの実現方法の評価は，評価事例を利用するアプローチ，すなわち，事例ベース推論[3, 4]のアプローチをとる．事例ベース推論は，与えられた問題に類似する過去の問題解決事例を直接利

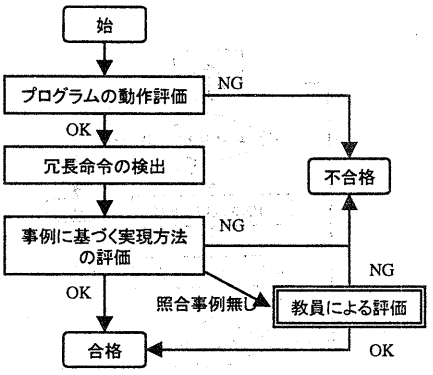


図 1: プログラム評価処理全体の流れ

用して問題解決を行うような推論法である．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こと」である．

事例ベース推論のアプローチをとる利点は2つある．1つはプログラムの評価に必要な知識やヒューリスティクスが少ないことである．プログラムの実現方法の評価を，従来のプログラム理解やプログラム認識のアプローチ [5, 6, 7, 8] で行う場合，非常に多くの評価のための知識・ヒューリスティクスが必要であり，現実的でない．2つめの利点は，事例の追加によってシステムの能力が向上できることである．教員がプログラムの評価を行う度に事例を追加するようにしておけば，それ以降，それと同様なプログラムは半自動的に評価を行うことができる．つまり，システムを使うことによって，システムの能力 (評価可能なプログラムの数) を高めることができる．また，教員が前もって用意した解答例も事例と見なすことができるので，ある時点では，教員が前もって用意した解答例と，それまでに獲得した評価事例が利用可能である．

2.3 評価処理全体の流れ

プログラム評価処理の流れを図1に示す．最初に，プログラムの動作を評価する．ここでアセンブル可能性とテストデータに対する動作をテストし，アセンブルエラーがある場合や正しく動作していないと判定された場合は，システムの出力は不合格となる．プログラムの動作評価の詳細は文献 [1] を参照されたい．

正しく動作すると判定された場合は，次に冗長命令の検出処理を行う．冗長命令とは，その命令がなくともプログラムの動作に影響を与えないような命令 (削除可能な命令) である．冗長命令の検出処理は合否判定には直接関係なく，冗長命令情報は，事例照合処理において用いられる．また，教員が判定を行ったり，アドバイスを記述する際に参考として使用する．冗長命令の検出は，プログラムリストの命令を1つずつ削除してはプログラムの動作を確認することによって行う．

冗長命令の検出処理の後，事例に基づく実現方法の評価を行う．ここで適用可能な事例が存在すれば，その事例の判定結果を適用する．すなわち，事例の判定結果が合格であればシステムの出力も合格となり，事例の判定結果が不合格であればシステムの出力も不合格となる．また，事例のアドバイスに簡単な修正を施し，アドバイスを生成する．適用可能な事例が存在しない場合は，人間である教員が評価を行う．

3 事例に基づく実現方法の評価

3.1 事例の表現

事例ベース推論における事例は，一般に問題記述と解記述，あるいは解法から構成される．本手法において，事例は，プログラムリスト，判定結果（合格，または不合格），アドバイス，事例作成年月日，事例作成者から構成する．これらのうち，プログラムリストが問題記述，判定結果とアドバイスが解記述である．事例の例を図2に示す．

プログラムリストはCASLプログラム，またはCASLプログラムの一般化表現 [1] である．CASLの一般化表現は，汎用レジスタの表現と一般化表現として新しく導入された命令以外はCASLの文法に準拠する．汎用レジスタについては，同じ役割をもつレジスタに対して同じ値を割り当てる．この数はレジスタ番号ではないので，CASLの汎用レジスタは0から4であるが，4を超える値が用いられることもある．また，一般化表現として新しく導入する命令は，同じ処理を表す複数通りの

```
PRG     START
        LD      GRG1, AA
        CPA     GRG1, BB
        J+OOR+  SKIP1
        LD      GRG1, BB
SKIP1   ST      GRG1, CC
        EXIT
AA      DC      1
BB      DC      5
CC      DS      1
        END
        [exchange] AA BB [/exchange]
事例識別子：k001@000002
判定： accept
作成年月日：1999 9 15
作成者： 渡辺博芳
アドバイス：
```

図 2: 事例の例

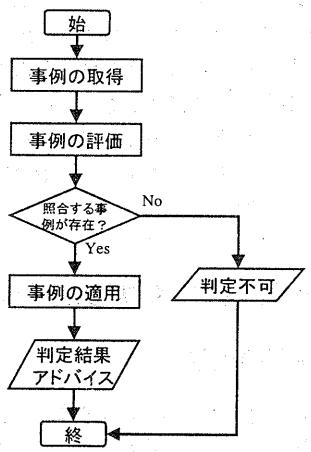


図 3: 事例に基づく実現方法の評価処理の流れ

CASL の記述に照合する表現である．一般化表現の命令と通常の CASL の文法に従うコードは一般化ルール[1]と呼ぶプロダクションルールによって対応付けられる．

3.2 事例に基づく評価処理の流れ

事例に基づくプログラム評価処理の流れを図3に示す．最初に，評価対象のプログラムと同じ問題に解答したプログラムに関する全ての評価事例を取得する．次に，取得した個々の事例のプログラムリストと評価対象のプログラムの照合処理を

[1]文献[1]では照合知識と呼んでいるものに相当する．

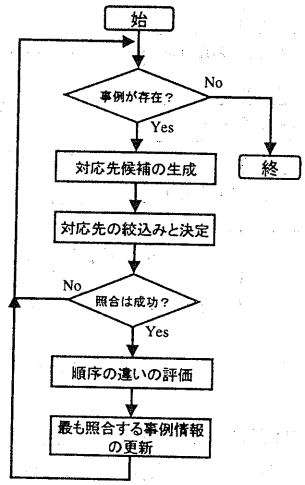


図 4: 事例の評価処理の流れ

行う．照合する事例が存在すれば，その事例を適用し，判定結果とアドバイスを出力する．照合する事例が存在しない場合は判定不可となる．

3.2.1 事例の評価

事例の評価では，事例と評価対象の照合処理を行い，最も照合する事例を選択する．事例と評価対象の照合処理は，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．

事例評価処理の詳細を図4に示す．照合処理は，命令，ラベル，レジスタの対応関係の候補を生成した後，対応関係の候補を他の2つの対応関係の情報を用いて絞り込むことで行う．例えば，ラベルの対応関係とレジスタの対応関係の情報を用いて，命令の対応関係の候補を絞り込むことができる．他の対応関係についても同様である．このようにして，命令，ラベル，レジスタの対応関係に矛盾がなく，かつ評価対象のプログラムで事例の命令に対応先を持たない命令が冗長命令(削除可能な命令)であれば，照合処理は成功である．これらの処理の詳細は，文献[1]を参照されたい．

照合が成功した場合は，事例と評価対象間で対応する命令の順序関係の違いを評価する．命令の順序関係が異なると，与えるアドバイスが異なる

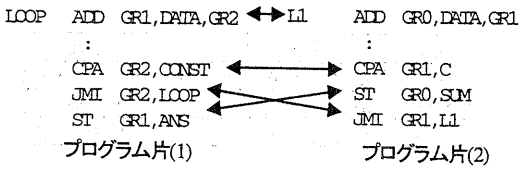


図 5: 対応する命令の順序関係の違い

可能性があるためである．例えば，図5のプログラム片の対応関係において，下の2行の順序関係が異なるが，ST命令をループの中におくか，外におくかの違いによってアドバイスが異なる可能性がある．このような順序関係が異なる部分を切り出す．

1つの事例の照合処理が成功に終わると，それまでの照合処理で最も照合する事例と照合度を比較する．より照合度が高いと判断される場合は，最も照合する事例を置き換える．照合度は，順序関係の異なる部分が少ないほど高く，順序の異なる部分が同程度であれば事例に対応先を持たない命令の数が少ないほど高い．どちらも0であれば，完全照合となる．

また，事例ではプログラムリストを一般化表現で記述している場合があるが，照合度が同程度の場合は，より特殊な(一般化の度合いが低い)方を優先する．一般化の度合いは，一般化表現の命令数で表す．すなわち，一般化表現の命令数の多い方が一般化の度合いが高いとして評価を行う．

3.2.2 事例の適用

評価対象に照合する事例が存在する場合は，その事例の判定結果を評価対象プログラムの判定結果とする．また，事例のアドバイス文を以下のように修正して評価対象プログラムに対するアドバイスとする．

・第x行，x行目という表現のxという数字を事例のプログラムリストのx行目の命令に対応する評価対象プログラムの命令の行に修正する．

・事例のプログラムのレジスタ名，ラベル名を表す文字列はそれに対応する評価対象プログラムのレジスタ名，ラベル名に置き換える．

システムは，以上のようにして求めた判定結果，アドバイスの他に，適用した事例の識別子，その事例との照合における順序関係の違い情報，冗長命令情報を出力する．

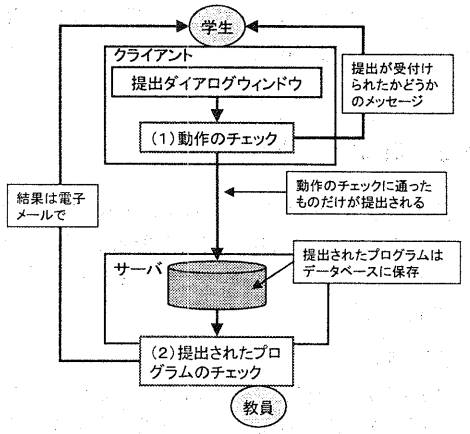


図 6: 学生向けのシステムの説明図

3.3 事例の追加

完全照合の事例が存在しなかった場合には，教員が評価を行った結果を新事例として追加する．

4 実現したシステム

4.1 システムの概要

図6は実現したシステムの概要を学生に説明するために用いている図である．つまり，図6は表向きの（学生の側から見た）システム像を表している．この図を用いて，実現したシステムにおいてプログラムが評価される過程を説明する．

(1) 学生は，学生用クライアントシステムの提出ダイアログウィンドウからプログラムを提出する．

(2) 学生用クライアントシステムは提出されたプログラムの動作の評価を行う．正しく動作していると判定されたプログラムだけが「提出受付」となる．ここで提出が受け付けられたか，動作が正しくなかったかによって，その旨のメッセージを提示する．

(3) 提出されたプログラムはサーバに保存され，サーバにおいて実現方法の評価を行う．評価結果に基づいて，判定結果(合格，または再提出)とアドバイスを電子メールで送付する．なお，学生に

は，実現方法の評価はシステムが行うのか，教員が行うのかは知らせていない．

4.2 評価モード

実際の授業の多様なニーズに応えるために，このシステムの判定結果の利用形態として，3つのモードを定義した．以下にそれら3モードについて説明する．

(1) 自動モード

プログラムが提出された時点で，事例との照合を行い，事例と完全に照合した場，事例から生成された判定結果とアドバイスを自動的に学生に通知するモードである．事例と完全に照合しない場合のい，教員が判定作業を行うので，教員の作業負荷の軽減の効果が大きい．一方で，システムにより自動的に判定されるプログラムと教員が判定するプログラムでは，判定結果の通知の時間差が大きいという欠点がある．従って，自動モードは，簡単な問題や穴埋め問題など，解答のバリエーションが少ない場合や，事例が豊富にある場合に用いると効果的である．

(2) 手動モード

システムの判定結果を基に，最後は必ず，教員が判定を行うモードである．提出されたプログラムに照合する事例が存在する場合は，照合した事例に対する判定結果を参照できることで，教員の作業負荷は軽減される．ただし，教員は提出を受け付けた全てのプログラムを処理をするので，作業負荷軽減の効果は自動モードほどではない．一方で，判定結果の通知の時間差の問題はない．

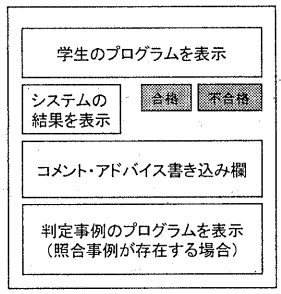
従って，解答のバリエーションが多い場合や，判定のための知識，事例が豊富でない場合にはこのモードにするのがよいと思われる．ある特定のプログラムについてのみ，時間的に早く合格が通知された場合などに，早く合格通知が届くプログラムの方がよいプログラムであるという誤解を学生に与える懸念があるためである．

(3) 動作のみ評価モード

プログラムの動作評価の結果，動作が正しければ，合格とするモードである．実現方法を問わないような題意の問題に対して用いるとよい．

学籍番号	状況	判定待ち	バージョン	提出時刻
984301	判定中	入力待ち	1, 2, 3	99/5/24/15:30
984302	合格		1, 2	99/5/24/14:30
984304			1	99/5/24/15:20
:	:	:	:	:

(a) 提出状況閲覧ページの概要



(b) 判定入力支援ページの概要

図 7: 教員用クライアントのイメージ図

4.3 システム構成

実現したシステムはサーバ，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，2モードのシミュレート機能を持つシミュレータ WCASL[9] 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では，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，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．提出状況の閲覧ページは図7(a) のようなページであり，入力待ちをクリックすると同図 (b) のような判定入力支援ページにリンクする．図7(a) のバージョンリストには，動作の正しくないプログラムも含め，学生が提出したプログラムのバージョン番号が提示されており，番号をクリックするとそのプログラムを閲覧することができる．

(3) サーバ

サーバは，問題情報，事例，一般化ルールなど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．また，事例に基づくプログラムの実現方法の評価や教員用クライアントのための機能は，以下のようなCGIプログラム群として実現した．

(a) 提出受付用CGI ： 提出を受付け，提出状況データを更新する．自動モードの場合は，事例に基づく実現方法の評価機能を実行し，判定可能なら，自動的に電子メールを送信する．

(b) 判定入力支援用CGI ： 事例に基づく実現方法の評価機能を実行し，システムの判定結果を提示すると共に，判定結果とアドバイスを入力するためのフォーマットを提供する．

(c) 判定入力確認用CGI ： (b) で入力された判定結果とアドバイスについて，それでよいかの確認をとる．

(d) 結果送付及び事例登録CGI ： 判定結果とアドバイスを電子メールで送信する．また，事例登録の必要がある場合に，提出されたプログラム，教員の入力した判定結果，およびアドバイスを新事例として追加する．

(e) 提出状況閲覧用CGI ： 上で述べた提出状況一覧を提示する．

(f) 提出プログラム閲覧用CGI ： 個々のプログラムを提示する．

5 実験と考察

5.1 実験条件

本システムを実際の授業に用いて評価を行った．その授業での学生数は79人であり，Nビットの左循環シフトを行うプログラムを作成する問題を提示した．本システムを用いるにあたり，2個の合格の事例[2]を用意し，評価支援は手動モードで行った．

5.2 実験結果と考察

システムの判定結果と教員の判定結果の比較を表1に示す．合格，不合格の()内は，事例が完全照合をしたケースの数を表す．評価の過程で，1個の不合格事例と37個の合格事例が追加された．

表 1: 教員の判定とシステムの判定の比較

	教員の判定		
システムの判定	合格	不合格	合計
合格(完全照合)	45(44)	0(0)	45(44)
不合格(完全照合)	0(0)	0(0)	0(0)
判定不可	36	1	37
小計	81	1	82
動作評価により不合格	0	153	153
合計	81	154	235

(1) システムの判定の精度

システムは判定不可の37ケースを除く198ケースで判定を行った．それらの判定結果は全て教員の判定結果と一致しており，システムの判定精度は100%となった．また，照合する事例が存在した際に，システムが生成したアドバイスをそのまま利用できたケースは38ケース(うち4ケースはアドバイス特になし)であった．

(2) 教員の評価作業の負荷軽減

教員の判定作業の負荷がどれだけ軽減されるかについては，精密に評価することはできないが，大まかに評価を行ってみる．

まず，教員が評価を行う必要のあるプログラム数で評価する．完全照合する事例が存在する場合はシステムの判定は信頼できるので，教員は評価を行わずに済むと考えると，教員が評価する必要があったプログラムの数は，判定不可の37個と完全照合しなかった1個の38個となる．235個のプログラムが提出されたので，システムを用いない場合，235個の評価を行う必要があると考えると，教員の評価するプログラムは約16%に減少した．ただし，システムを用いて提出するということで，教員に直接提出する場合よりも，気軽に提出する傾向がある．例えば，自分でプログラムの動作を確認せずに，とりあえず提出してみるという学生もいると考えられる．そこで，プログラム総数を動作が正しいものに限定すると82個であり，この場合は，約46%に減少したことになる．

次にプログラムの評価作業が(a)動作の評価，(b)

[2]事例のプログラムリストは一般化した．

実現方法の評価，(c) アドバイスの作成[3]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する．全体のタスク数は，(a) が235，(b) が82，(c) が235で合計552である．教員が行った評価タスクは (b) が38，(c) が44で合計82である．従って，教員の評価タスク数は約15%に減少した．判定対象を動作が正しいものに限定すると，タスク総数は82プログラム×3つの作業であり，この場合でも約3分の1に減少したと言える．

作業負荷軽減に関して，他に以下のような効果もある．

・学生が提出する全てのプログラムについて動作の評価は自動化されるため，この作業は100%軽減されるし，見落としもなくなる．

・Webページで簡単に学生のプログラムにアクセスでき，教員の評価が必要なプログラムがマークアップされるので，評価対象にアクセスする作業負荷が軽減される．

・入力した判定やアドバイスが自動的に電子メールで送付されるので，学生に結果やアドバイスを通知する作業が軽減される．

以上のような考察から，教員の評価作業負荷軽減に関して，本システムの効果は極めて大きいと言える．

6 むすび

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について述べた．システムがプログラムを自動的に評価する方法として，同様なプログラムの評価事例に基づいて合否判定とそのアドバイス文を生成するアプローチをとった．また，システムの判定結果に基づいて教員が判定やアドバイスを行うための使い易いインターフェースも実現した．

実際の授業で本システムを使用した結果，教員のプログラム評価作業の負荷を大幅に減少できることがわかった．今後，より多くの授業で使用することによって，本システムの詳細な評価を行う予定である．さらに，問題情報作成用エディタの開発，プログラムリストを自動的に一般化する方法の検討などが今後の課題となる．

[3]特にアドバイスがない場合でも「特にない」とアドバイスを作成したと捉える．

謝辞 実現システムについてご意見を頂き，また，実際の授業での実験にご協力頂いた本学技術職員高井久美子さん，システム開発に熱心に協力された本学卒研生 矢古宇努君，半田博美君に感謝する．

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研究会，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] Kolodner,J. : Case-Based Reasoning, Morgan Kaufmann Publishers,Inc., 1993.

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

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

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

[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] 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，信学論，Vol.J78-DII，No.2, pp.321 – 332, 1995.

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