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

渡辺博芳† 荒井正之† 武井惠雄†

本論文では，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支援システムのための事例ベース構築法について述べる．事例ベース構築において，評価作業を行う際の事例ベース活用率の向上を目的とする．事例ベース活用率は，教員が行うプログラム評価の全体に対する事例が利用可能な場合の割合 (%) である．本論文では，(1) 事例に含まれるプログラムリストは，対象となるプログラミング言語自身で表現し，事例のインデックスにプログラムリストの一般化情報を反映させること，(2) プログラムリスト一般化においては，(a) 命令の種類，(b) 命令の順序関係，(c) 冗長命令の有無の 3 つの観点を考慮すること，(3) 同一視できるプログラムリストをカバーする一般化レベルをユーザとしての教員がカスタマイズ可能とすることを基本方針とした，3 階層構造を持つインデックスの構築法を提案する．また，CASL，および CASL II を教材とした実際の授業で提出されたプログラムを用いた実験を行い，提案した手法の有効性を示す．

A Method of Constructing Case-bases for Evaluation Assistant Systems for Novice Programs Written in Assembly Language

Hiroyoshi Watanabe,† Masayuki Arai† and Shigeo Takei†

This paper presents a method of indexing cases for case-based evaluation assistant systems for novice programs written in an assembly language. The purpose of the indexing is to improve a ratio of obtaining available cases in case-based evaluation processes compared with flat case-bases. In this paper, we propose a three-level index of evaluation cases following three policies: (1) program lists in evaluation cases should be represented in intact target programming language and indexes to cases should be constructed by using information of generalized program lists in order to expand the variations of program lists covered by one case, (2) three types of variations, that is, (a) variations of used instructions, (b) variations of instructions' order and (c) variations of redundant instructions should be taken into account when program lists are generalized, and (3) the level of the generalization of program lists which covers programs considered as the same should be able to be customized by teachers as users. Retrieval experiments with submitted programs in actual classes demonstrated the effectiveness of the proposed index method.

1. まえがき

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，それ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，すべ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，題意を満たすプログラムを全学生が提出するまで再提出を繰り返させるので，教員は学生数以上のプログラムを評価することになる．したがって，教員が学生のプログラムを評価する作業の負荷は非常に大きくなるため，これらを支援することが望まれる．

また，近年，大学におけるインターネット授業や e ラーニングによる企業研修など，時間と場所を選ばない学習環境が広まりつつあるが，このような学習環境を提供する側の負担は大きくなる傾向がある．そこで，このような新しい学習環境下においても，講師の負担を軽減することは非常に重要である．

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，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),2)]．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，評価事例に基づく実現方法の評価を行う．事例に基づくプログラム評価では，事例ベース推論[3)~6)]のアプローチをとり，あるプログラムの評価を行う際に，過去に評価を行ったプログラムの

† 帝京大学理工学部
School of Science and Engineering, Teikyo University

評価事例を利用して評価を行う．このアプローチは，新しいタイプのプログラムの評価を行うたびに，新しい評価事例を事例ベースに追加することで，システムの評価能力を漸増的に向上できる点に特徴がある．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．一方で，開発したシステムの性能を更に向上させ，より実用的にするためには，事例ベース構築方法を改善する必要があることが分かった[7)~9)]．

本論文では，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システムが利用する事例ベースの構築方法を提案する．事例に基づく評価システムの評価能力，すなわち，あるプログラムを評価する際に事例を利用して評価結果を生成できる可能性は，事例のインデックスや事例検索処理の実装に大きく依存する．そこで，あるプログラム群を評価するとき，プログラム評価の全体に対する事例が利用可能な場合の割合 (%) を事例ベース活用率と呼び，事例ベース構築にあたって事例ベース活用率を高めることを目指す．

以降，2 章では対象とするプログラム評価の問題定義と我々が実現した評価支援システムについて述べ，3 章で事例ベース構築方法を提案する．4 章では提案する事例ベース構築方法のための事例検索処理と事例ベース更新処理を詳述し，5 章以降で実際の授業において提示した問題とそれに対して提出されたプログラムを用いた実験による評価と考察について述べ，まとめる．

2. 初等アセンブラプログラミング評価支援システム

本章では，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の概要を述べる．詳細については文献 1) を参照されたい．

2.1 対象とする評価作業

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などに関する教育的な意図がある．本研究で対象とする 1 つめの評価作業は，学生が作成したプログラムが問題を提示した教員の教育的な意図（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」といい，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かかわらずに，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，より良いプログラムにするためのアドバイスを与えることがある．

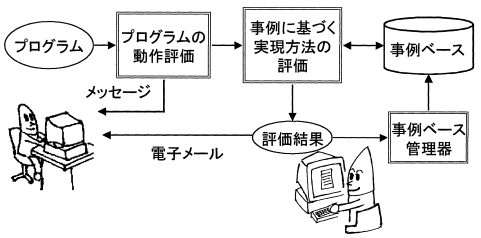

図 1 プログラム評価処理の流れ
Fig. 1 Processes for the evaluation of students' programs.

2.2 プログラム評価処理の流れ

実現したシステムを用いたプログラム評価の流れを**図 1** に示す．学生がプログラムを提出すると，システムは最初にあらかじめ用意された複数組のテストデータを用いてプログラムの動作を評価する．動作が正しくない場合，どのようなデータに対して動作しないかを示し，提出が受理されなかった旨のメッセージを提示する．動作が正しい場合は，事例に基づく実現方法の評価を行う．教員はシステムの出力（評価結果）を編集し，最終的な評価を行う．その結果は自動的に電子メールで学生に通知されるとともに，事例ベース更新のための情報として利用する．

2.3 事例に基づくプログラム評価

2.3.1 事例の表現

評価事例は，問題記述として評価対象となるプログラムリスト，解記述として合否判定結果（合格または不合格）とアドバイス文を持つ．また，事例管理のための情報や事例ベースのインデックスに必要な情報も持たせる．事例の表現方法の詳細については，4.1.4 項で述べる．

2.3.2 事例に基づくプログラム評価処理の流れ

評価処理においては，学生が提出したプログラムに照合するプログラムリストを持つ事例を検索し，それらの中から，学生のプログラムに最も類似する事例を選択する．照合条件を満たす事例が存在すれば，その事例の評価結果（合否判定とアドバイス文）を評価対象に適用する．合否判定結果はそのまま適用する．つまり，事例の合否判定結果が合格であるなら，評価対象も合格となる．また，アドバイス文については，プログラム照合で求めた対応関係情報を利用して，ラベル名，レジスタ番号や行番号の置き換え程度の簡単な修正を施す．

2.3.3 プログラム照合

プログラム照合は，評価対象となる学生のプログラムリストと事例のプログラムリストの比較を行い，事例の利用可能性を検査する処理である．比較の結果，評価対象が事例と照合するか否かを判定し，照合する場合は照合度とプログラムリスト間の対応関係を求める．プログラム照合は，事例の検索と選択処理の中で行う．

アセンブラプログラムは，個々の行が 1 つの命令を表しているので，2 つのプログラムリストにおいて，命令の対応をとることができる．命令の対応関係は，命令コードが等しく，オペランドがそれぞれのプログラムで同じ役割のレジスタやラベルであるという基準で決める．このような基準に基づいて，評価対象のプログラムリストと事例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．

2.3.4 照合度と確信度

評価対象と事例のプログラムの照合度は，大きく「完全照合」と「類似照合」に分けられる．より詳細には，対応する命令の種類，命令の順序関係，冗長命令の差異によって表す．差異については 4.2 節で詳述する．

プログラム照合の結果，以下の条件を満たすとき「評価対象と事例は照合する」という．

- 事例のプログラムのすべての命令が評価対象プログラムの命令に対応先を持つ．なお，評価対象プログラムには，事例のプログラムに対応付けられない命令が存在してもよいが，それらの命令が冗長命令であるものとする．

さらに，以下の条件を満たすとき「完全照合」と呼ぶ．

- 個々の命令が 1 対 1 に対応しており，それらの順序関係も等しいと見なせる．

評価対象と事例が照合するが，完全照合ではないケースを「類似照合」と呼ぶ．

システムが評価結果を教員に提示する際には，学生のプログラムと事例のプログラム照合結果に基づいて評価結果の確信度を出力する．基本的に，完全照合の場合は確信度を Surely とし，類似照合の場合は確信度を Probably とする．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もできる．

3. 事例のインデックス法

3.1 事例インデックスの方針

事例ベース活用率を高めることを目的として，事例ベースにインデックスを付与する．事例ベース活用率を高めるには，過去の評価事例とまったく同じプログラムだけでなく，同様であると見なせる類似のプログラムについても事例を適用する必要がある．学習者が作成したプログラムを知識に基づいて解析してアドバイスを与えるシステム[10)~15)]や知識に基づいてプログラムの評価を支援するシステム[16)]では，プログラムリストを何らかの形で抽象的に表現したものを利用している．プログラムリストを抽象的な表現で記述することによって，同様な複数のプログラムリストを 1 つの抽象的表現で表すことができるからである．このようなプログラムリストの抽象的な表現を「一般化表現」と呼び，具体的なプログラムリストを一般化表現に変換することを「一般化する」という．本手法では，プログラムリストの一般化情報を利用してインデックスを構築することで事例ベース活用率の向上を図る．

インデックス構築においては，以下のような方針をとる．

- 事例のプログラムリストは対象とするプログラミング言語そのままの形式で表現し，インデックスにプログラムリストの一般化情報を反映させる．
- プログラムリストの一般化における 3 つの観点（命令の種類，命令の順序，冗長命令）を反映させる．
- 一般化のレベルに応じた階層構造をとり，同一と見なせるプログラムの範囲に関する条件を教員がカスタマイズ可能とする．

3.1.1 プログラムリストの一般化情報の反映

事例のインデックスにプログラムリストの一般化表現に関する情報を利用し，事例に含めるプログラムリスト自体は対象とするプログラミング言語によって表現する．事例ベースの活用率を高める方法として，事例に含めるプログラムリスト自体を一般化表現で記述する方法が考えられる．事実，我々が実現したシステムの最初のバージョン[1)]ではこの方針を採用した．しか

し，これには以下のような問題があることが分かった．

- 事例のプログラムリストは教員が最終的な評価を行う際に参照するので，教員が一般化表現という特別な表現法を習得する必要がある．
- 同じ評価結果を適用可能なプログラムのバリエーションの範囲は教員の題意に依存するので，事例に含まれるプログラムリストをどの程度まで一般化しておくかをシステムが自動的に決定することが困難である．

そこで，事例のプログラムは対象とする言語そのままの形式で表現し，インデックスにプログラムリストの一般化情報を反映させることとした．

3.1.2 一般化における 3 つの観点

プログラムリストの一般化においては，次の 3 つの問題を解決しておかなければならない．

(1) 命令の種類に関する一般化：1 つの操作が 2 種類以上の命令によって実現可能なことがある．たとえば,「あるレジスタに定数を設定する操作」は，CASL では，LD（load）命令と LEA（load effective address）命令の両方で実現できる．このような使用する命令の違いによるバリエーションをカバーしなければならない．これは，命令の一般化表現と一般化ルールを用いて行うことができる．CASL に対する一般化表現と一般化ルールは，我々が独自に定義した[1),2)]．

(2) 命令の順序に関する一般化：同一の命令群を用いたプログラムでも，個々の命令の記述順序が異なる場合がある．たとえば，ある命令がループの中にある場合と外にある場合で，評価結果やアドバイスが異なる可能性がある．このような命令の順序関係の違いをカバーしなければならない．ただし,「順序の違いは些細である」と見なせる場合は順序関係は同一として扱う．CASL においては，以下のような場合は,「順序の違いは些細である」とした．

- 順序が異なる範囲に含まれる命令の命令コードがすべて等しい．
- 順序が異なる範囲に含まれる命令が指標レジスタ修飾をともなわない，レジスタへの値の設定か，レジスタ主記憶間のデータの転送に関する命令（CASL では LEA，LD，ST）のみから構成される．

(3) 冗長命令の有無に関する一般化：ある命令の有無にかかわらず，プログラムの動作が同一である場合，その「命令は冗長である」という．そのような冗長命令の有無に関するバリエーションをカバーしなければならない．冗長命令が含まれることで，可読性が悪くなる場合と，逆に良くなる場合もある．そこで，冗長命令の有無で評価結果やアドバイスが異なる可能性がある．プログラムリストの一般化の観点では，冗長命令を削除し，プログラムの行数が少ないほど，一般化階層では上位に位置付けられることとする．これは，機械学習における選択的一般化規則のうちの条件削除規則[17)]と同様である．

3.1.3 一般化レベルのカスタマイズ

プログラムを評価する観点からどの範囲のプログラムを同一と見なしてよいかは，問題を出題した教員の題意によって異なる．たとえば，CASL において比較命令を CPA と CPL のどちらを使ってもよいとする場合と，CPA と CPL の使い分けを習得させたい場合で同一と見なせるプログラムのバリエーションの範囲は異なる．これは使われている命令の種類に関する例であるが，命令の順序関係や冗長命令の有無に関しても，差異を問題視しない場合とその差異が評価において重要な場合とがある．これらの基準は教員の教育的意図に基づくので，あらかじめシステムに組み込むことはできない．そこで，システムのユーザである教員がカスタマイズ可能とすることが望ましい．

プログラムリストの一般化表現がどの程度一般化されているかを「一般化レベル」と呼ぶ．同一と見なせるプログラムのバリエーションが多いほど，一般化レベルが高い．上で述べたことは，事例ベース構築の立場からは，プログラムリストの一般化表現の一般化レベルをユーザである教員がカスタマイズ可能にしたいということである．

本手法では，前項で述べた 3 つの観点のうち,「命令の種類」に関して同一と見なせるプログラムリストの一般化レベルについては，教員がカスタマイズ可能とする.「命令の種類」に関しては一般化ルールを適用することで一般化を行うので，適用する一般化ルールの部分集合を定義することで，一般化レベルを決定できる．一方,「命令の順序関係」や「冗長命令」に関する一般化では，順序の異なり方や冗長命令のタイプのパターンを分析し，それを判別することは容易ではない．そこで，一般化レベルのカスタマイズは,「命令の種類」に関する一般化のみを対象とし，他の 2 つの観点については，評価対象プログラムと事例のプログラムが同一と見なせない場合に，同一と見なすことがないように，一般化レベルを低く設定する．

3.2 インデックスの階層

前節で述べた方針に基づき，**図 2** に示すような 3 階層のインデックスを，個々の問題ごとに構築する．

(1) ルート：インデックスのルートは，検索の際のスタート地点となる．ルートノードに，第 1 階層の

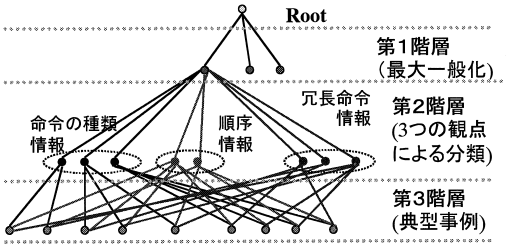

図 2 事例ベースインデックスの階層構造
Fig. 2 Three level index of cases.

ノードへのリンクとインデックス情報を持たせる．インデックス情報は，第 1 階層の個々のノードがカバーするプログラムリストにおける各命令数（CASL では 27 命令分）の上限値と下限値である．

(2) 第 1 階層：プログラムリストを可能な限り一般化したノード（プログラムリストに，本システムが持っているすべての一般化ルールを適用して得られる一般化表現を表すノード）である．

(3) 第 2 階層：第 2 階層のノードは前節で述べた 3 つの観点に対応する 3 種類のノードのグループを形成し，それら 3 つの観点に関して事例を分類する役割を果たす．「命令の種類」グループのノードは命令の種類に関して事例を分類する．つまり，「命令の種類」グループのノードの下位には，命令の種類に着目したときに同じと判断される複数の事例を配置する．同様に，「命令の順序関係」グループのノードの下位には，命令の順序関係が同じと判断される複数の事例を配置する．「冗長命令」グループのノードについても同様である．この結果，1 つの事例は個々のグループについて必ず 1 ノードとリンクされる．

(4) 第 3 階層：典型的な事例を配置する．ここでは，プログラム中のラベル名，レジスタ名が異なる場合も同一と見なす．

3.3 命令の種類に関する一般化レベルのカスタマイズ法

命令の種類に関する一般化レベルは，チェックリスト形式の説明文に教員がチェックを入れることで設定できるようにする．教員が適用すべき一般化ルールを直接選択するためには，一般化ルールに関する知識が必要となるからである．具体的には，一般化ルールとそれに対応するチェックリスト用の説明文をあらかじめ用意しておき，第 2 階層のノードを生成する際には，チェックが付けられた説明文に対応する一般化ルールを適用する．

図 3 に，CASL を対象とした評価支援システムにおけるチェックリスト設定のためのウェブベースのインタフェースを示す．現在のところ，本システムにおけるチェックリストの項目は，図 3 に示されているものですべてである．それぞれの説明文は 1 つ以上の一般化命令に関連付けされている．このようなインタフェースを提供することで，第 2 階層で適用する一般化ルールを教員が問題ごとに個別に設定可能とする．図 3 ですでにチェックが入っている項目はデフォルト値であり，教員が設定を行わない場合は，それらが使われる．

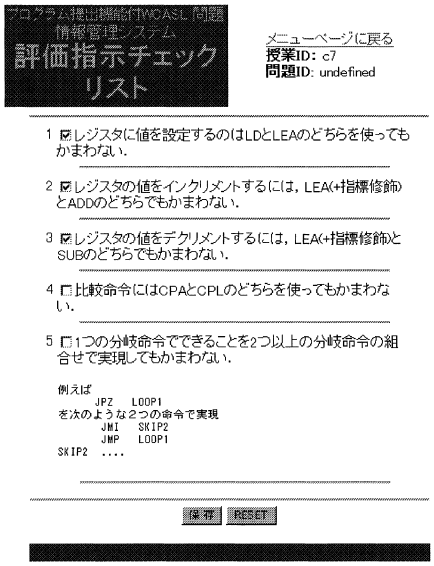

図 3 一般化レベルカスタマイズのためのインタフェースの例
Fig. 3 An example of an interface for customizing the generalization level.

4. 事例検索と事例ベース更新

4.1 データ構造

本節では，インデックスノードや事例を表すためのデータ構造について説明する．4.2 節以降では，本節で述べたデータ構造を前提として，処理内容を説明する．

命令の種類の数を N_{Inst} とする．CASL では，DS，DC，END の各命令を除いて，N_{Inst}=27，CASL II では，DS，DC，END の各命令を除いて，N_{Inst}=43 とした．また，第 2 階層ノードには 3 つのグループが存在するが，グループを以下のような a，b，c で表現する．

a. 命令の種類に関するノードグループ
b. 命令の順序関係に関するノードグループ
c. 冗長命令に関するノードグループ

4.1.1 インデックスルート

問題ごとに，インデックスルートが存在する．インデックスルートには第 1 階層へのリンクのエントリを

登録する．リンクのエントリは以下のような情報から構成する．

- リンク先の第1階層ノードの識別子（文字列）
- 命令数の上限値（N_{Inst} 個の整数値配列）
- 命令数の下限値（N_{Inst} 個の整数値配列）

命令の数の上限値と下限値は，リンク先の第1階層ノードのプログラムリストと照合する可能性のあるプログラムリストにおける命令の数の範囲を示している．

4.1.2 第1階層ノード

第1階層ノードは以下のような情報から構成する．

- 識別子（文字列）
- プログラムリスト（一般化表現）
- 第2階層aグループノードへのリンク
- 第2階層bグループノードへのリンク
- 第2階層cグループノードへのリンク

「第2階層aグループへのリンク」のエントリは，リンク先のノードの識別子，および命令数の上限値と下限値から構成する．これはインデックスルートにおける第1階層へのリンクのエントリと同様である．

「第2階層bグループへのリンク」のエントリは，リンク先のノードの識別子と命令の順序情報から構成する．命令の順序情報は，このbグループノードでカバーされるプログラムリストの各命令と対応する第1階層ノードのプログラムリストの行番号を順に並べたものである．

「第2階層cグループへのリンク」のエントリは，リンク先のノードの識別子と冗長命令情報から構成する．1つの冗長命令に関する情報は以下の情報で構成する．

- 冗長命令の位置する行（整数値）
- 冗長命令の直前の命令の行（整数値）
- 冗長命令の命令コード（文字列）

「冗長命令の位置する行」は，事例のプログラムリスト中で冗長命令が位置している行である．「冗長命令の直前の命令の行」は，事例のプログラムリスト中で冗長命令の直前に位置する冗長でない命令が対応する，第1階層ノードのプログラムリスト中の命令が位置している行である．

4.1.3 第2階層ノード

第2階層ノードは3種類存在するが，いずれも以下のような情報で構成する．

- 識別子（文字列）
- グループ固有情報
- 第3階層ノードへのリンク

「グループ固有情報」は，aグループでは一般化表現のプログラムリスト，bグループでは命令の順序情報，cグループでは冗長命令情報である．命令の順序情報，冗長命令情報の形式は 4.1.2 項で述べたものと同じである．「第3階層ノードへのリンク」のエントリはリンク先のノードの識別子のみである．

4.1.4 第3階層（事例）ノード

第3階層ノード，すなわち事例ノードは以下の情報で構成する．

- 識別子（文字列）
- プログラムリスト
- 合否判定結果（accept または reject）
- アドバイス文（文字列）
- 事例更新日時（年月日時分）
- 事例更新者（文字列）
- 命令数（N_{Inst} 個の整数値配列）
- 命令の順序情報
- 冗長命令情報

命令の順序情報，冗長命令情報の形式は 4.1.2 項で述べたものと同じである．

4.2 事例検索処理

事例検索処理は，評価対象となるプログラムリスト（P）と冗長命令情報を入力として事例検索を行い，最も類似する事例（*CASE*），事例と評価対象の照合度（*MDEGREE*），差異（*DIFF*），および対応関係情報（*MINFO*）を出力する．「照合度（*MDEGREE*）」は，完全照合のとき Perfect，類似照合のとき Similar，未照合のときは Miss とする．「差異（*DIFF*）」は，メンバ a，b，c を持ち，それぞれのメンバに a. 使用される命令の種類情報，b. 命令の順序情報，c. 冗長命令情報に関して異なる個所をカウントし，差異の値として保持する．*DIFF*=0 とは a，b，c のいずれも 0 であることを示す．また，差異の比較においては，a，b，c の順に優先して比較を行う．「対応関係情報（*MINFO*）」は命令，ラベル，レジスタの対応関係に関する情報で，事例を適用する際に用いる．**図 4** に事例検索アルゴリズムを示す．

(1) 最初に照合する可能性のある第1階層ノードの候補を選択して *NLIST* に設定する（図4の **1.**）．ここでは，「N_{Inst} 個の各命令の数が下限値以上」という条件を満たすノードを複数選択する．2.3.4 項で述べた照合条件を採用しており，事例のプログラムの全命令が評価対象プログラムリストに対応先を持てば，照合する可能性があるので，上限値に関しては検査しない．

(2) *NLIST* から取り出した第1階層ノードと評価対象のプログラム P とのプログラム照合（図4の **2.1**）を行い，照合条件を満たす場合は，そのノードの下位を調査する．ここでの照合条件は，図4の**照合条件 1** のとおり，「ノードのプログラムリストのすべての命令が評価対象プログラムの命令に対応先を持つこと」であり，評価対象プログラムにはノードのプログラムリストに対応先を持たない命令があってもよい．

事例検索アルゴリズムで用いられる変数

P	評価対象となるプログラムリスト.
CASE	最も類似する事例.
DIFF	*P* と *CASE* との差異.
MINFO	*P* と *CASE* の対応関係情報.
MDEGREE	*P* と *CASE* との照合度. 照合度として取り得る値は，Perfect(完全照合), Similar(類似照合)，Miss(未照合). ただし，Perfect > Similar > Miss
NLIST	検索対象となる第 1 階層のノードリスト.
N1	検索中の第 1 階層のノード.
N3	検索中の第 3 階層のノード.
DIFF3	*P* と *N3* との差異.
MINFO3	*P* と *N3* の対応関係情報.
MDEGREE3	*P* と *N3* との照合度. 取り得る値は *MDEGREE* と同じ.

事例検索アルゴリズム

0. 初期化 (*CASE* ← 空, *DIFF* ← 最大値, *MDEGREE* ← 空)
1. *NLIST* ← 各命令数の下限値が *P* に含まれる各命令の数以下である複数の第 1 階層ノード.
2. *NLIST* 中の個々のノード *N1* について以下を行う.
  2.1 *N1* と *P* のプログラムを照合.
  2.2 もし，**照合条件 1** を満たすなら，以下を行う.
    2.2.1 *N3* ← *N1* の下位で *P* との差異が最小の事例.
    2.2.2 *N3* と *P* の照合処理.
    2.2.3 *DIFF3* ← *N3* と *P* との差異.
    2.2.4 *MINFO3* ← *N3* と *P* の対応関係情報.
    2.2.5 *MDEGREE3* ← *N3* と *P* の照合度.
    2.2.6 もし，(*MDEGREE3*>*MDEGREE*) または，(*MDEGREE3*=*MDEGREE* かつ *DIFF3*<*DIFF*) ならば以下を行う.
      2.2.6.1 *CASE* ← *N3*.
      2.2.6.2 *DIFF* ← *DIFF3*.
      2.2.6.3 *MINFO* ← *MINFO3*.
      2.2.6.4 *MDEGREE* ← *MDEGREE3*.
    2.2.7 もし，*MDEGREE*=Perfect かつ，*DIFF*=0 なら，検索処理を終了.

照合条件 1

・ノード *N1* のプログラムリストのすべての命令が評価対象プログラム *P* の命令に対応先を持つ.

図 4 事例検索アルゴリズム
Fig. 4 An algorithm of case retrieval.

(3) *P* に照合する第 1 階層ノードの下位の第 2 階層ノードを調べ，最も差異の小さい事例（第 3 階層ノード）を選択する（図 4 の **2.2.1**）.

(4) 選択した事例 *N3* と評価対象プログラム *P* の照合処理を行い，対応関係情報を生成し，照合度を求める（図 4 の **2.2.2**〜**2.2.5**）. *N3* と *P* の照合処理で，*N3* と *P* を直接照合できない場合は，上位のノードのプログラムリストを用いる．すなわち，*N3* が選択された時点で，インデックスは**図 5** のようになっており，インデックスノードのうち，*N2a* と *N1* には一般化表現のプログラムリストが保持されている．*N2a* は第 2 階層の命令の種類グループのノードである．*N3* と *P* が直接照合できない場合は，図 5 の白抜きの矢印で示すように，*N3* と *N2a* の照合，*P* と *N2a* の照合を行い，その結果から，*N3* と *P* の対応関係を求める．*N2a* が利用可能でないとき，*N1* を用いる.

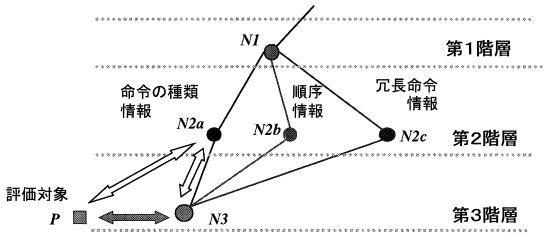

図 5 選択された事例へのインデックス
Fig. 5 Index of the selected case.

(5) 今回選択された事例の照合度 *MDEGREE3* が *MDEGREE* より高い場合，あるいは同じ照合度のとき差異 *DIFF3* が *DIFF* より小さい場合は，*N3* を *CASE* に設定し，*MDEGREE*，*MINFO*，*DIFF* も更新する（図 4 の **2.2.6**）.

(6) 完全照合で差異が 0 である事例を得た場合，もしくは第 1 階層ノードの候補 *NLIST* 中のノードをすべて調査したら，検索終了とする.

4.3 事例ベース更新処理

事例ベース更新処理では，評価対象のプログラムリスト *P*，その評価結果 *E*，冗長命令情報を入力として事例ベースを更新する．**図 6** に事例ベース更新アルゴリズムを示す.

(1) まず，*P* に対して 4.2 節で述べた事例検索を行う（図 6 の **1.**）. *P* に完全照合し，差異が 0 である事例が存在すれば，その事例の評価結果（合否判定とアドバイス文）を *E* で上書きすることで更新し，更新処理を終了する（図 6 の **2.**）.

(2) *P* から冗長命令を削除したプログラムリスト *P2* を求め，これらを使ってインデックスノードを検索する（図 6 の **3.** と **4.**）. ノードの検索は 4.2 節で述べた事例検索と同様であるが，以下の点が異なる.

- 第 1 階層ノードの候補を選択する際に命令数の上限値と下限値の両方を調査する.
- ノードとのプログラム照合では *P2* を用い，照合条件は「2 つのプログラム間で 1 対 1 に命令の対応がとれること」である.

ここでは照合するノードの検索を目的としているので，ノードとの照合処理を事例検索時よりも厳しく行うためである.

(3) 次に新事例を追加するために，事例ノードを生成する（図 6 の **5.**）. また，ノード検索の結果，照合するノードが存在しない場合は，新しくノードを生成し，リンクを付加する（図 6 の **6.** と **7.**）. 第 2 階層

事例ベース更新アルゴリズムで用いられる変数

P	評価対象となるプログラムリスト.
E	*P* に対する評価結果.
P2	*P* から冗長命令を削除したプログラムリスト.
CASE	最も類似する事例.
DIFF	*P* と *CASE* との差異.
MDEGREE	*P* と *CASE* との照合度. 照合度として取り得る値は，Perfect(完全照合)，Similar(類似照合)，Miss(未照合).
N1	検索中の第 1 階層のノード.
N2	第 2 階層のノード. *N2a*,*N2b*, または *N2c* が入る.
N2a	第 2 階層の命令の種類に関するノード.
N2b	第 2 階層の命令の順序に関するノード.
N2c	第 2 階層の冗長命令に関するノード.
N3	第 3 階層のノード.

事例ベース更新アルゴリズム

1. *P* に対して事例検索.
2. もし，*MDEGREE*=Perfect かつ，*DIFF*=0 なら，
  2.1 *CASE* の評価結果を *E* で書き換える.
  2.2 事例ベース更新処理終了.
3. *P2*← *P* から冗長命令を削除したプログラムリスト.
4. *N1, N2a, N2b, N2c* ← *P* と *P2* を用いて検索した結果 (**照合条件 2** を満たすノード).
5. *N3*← *P* と *E* を用いて生成した事例ノード.
6. {*N2a, N2b, N2c*} のうち，空なものがあれば，個々を *N2* として以下を行う.
  6.1 *N2*← 第 2 階層ノードを生成.
  6.2 *N2* の固有情報を設定.
  6.3 *N2* に *N3* へのリンクを追加.
  6.4 *N1* が空でなければ，*N1* に *N2* へのリンクを追加.
7. もし，*N1* が空なら，
  7.1 *N1* ← 第 1 階層ノードを生成.
  7.2 *N1* のプログラムリスト ← *N2a* のプログラムリストを一般化したプログラムリスト.
  7.3 インデックスルートに *N1* へのリンクを追加.
  7.4 *N1* に *N2a,N2b,N2c* へのリンクを追加.

照合条件 2

・*N1*：*P2* を構成する各命令の数が，命令数の下限値以上，上限値以下である．かつ，*N1* のプログラムリストと *P2* の間で命令が 1 対 1 に対応がとれる.
・*N2a*：*N1* の下位のノードである．かつ，*N2a* のプログラムリストと *P2* の間で命令が 1 対 1 に対応がとれる.
・*N2b*：*N1* の下位のノードである．かつ，命令の順序情報が *P2* と等しい.
・*N2c*：*N1* の下位のノードである．かつ，冗長命令情報が *P* と等しい.

図 6 事例ベース更新アルゴリズム
Fig. 6 An algorithm for updating the case-base.

ノードのノード固有の情報（図 6 の **6.2**）は，*N2a* では事例のプログラムリストの一般化表現，*N2b* では命令の順序情報，*N2c* では冗長命令情報である．*N2a* に設定する一般化表現は 3.3 節で述べたチェックリストでチェックされている説明文に対応する一般化ルールのみを *P2* に適用して生成する．一方，第 1 階層ノード *N1* に設定するプログラムリストは，チェックリストでチェックの付いていない説明文に対応する一般化ルールを *N2a* に設定されたプログラムリストに適用して生成する（図 6 の **7.2**）.

5. 実験および考察

5.1 実験条件

提案した手法の有効性を評価することを目的として，事例の検索実験を行った．実験システムは，個々の問題に対して提出されたプログラムを学籍番号順（同じ学生が複数提出した場合は提出順）に読み込み，事例の検索を行う．事例の検索結果を記録した後，新しく事例を追加する必要があれば，新事例の追加を行う．すなわち，事例ベースは，最初は空であり，実験が進むにつれて漸増的に大きくなる．このような事例検索と更新操作を提出されたプログラムすべてに対して行った結果,「全プログラム数」に対する「照合する事例を得たケース」の割合を事例ベース活用率 (%) とする.

実験に利用したデータは，過去の授業において出題した問題とそれに対して提出された動作の正しいプログラム群の組である．具体的には 1999 年度と 2000 年度に CASL を教材として行った授業において出題した延べ 17 問と 2001 年度に CASL II を教材として行った授業において出題した延べ 14 問を用いた．これらの中には，年度や授業クラスが異なる場合に同じ問題を出題したケースも含まれる．1 問に対するプログラムの数は，少ないもので 61 本，多いものでは 180 本であった．CASL を教材とした場合の問題は，“1-通し番号”，CASL II を教材とした場合の問題は，“2-通し番号” で表すことにする.

提案した手法による事例ベース構築と事例検索を行う実験システムを *index*，インデックスを用いずにフラットな事例ベースを構築する実験システムを *flat* と表す.

5.2 実験結果と考察

5.2.1 事例ベース活用率

CASL と CASL II に対する事例ベース活用率の平均値を**表 1** に示す．いずれの場合も，*index* では *flat* に比較して平均で 10%程度，事例ベース活用率を向上できている．また，提案手法を採用すれば，初等レベルの問題では平均で評価対象の 80%以上で，評価事例を参照しながら評価作業が行えることが分かる．各問題での事例ベース活用率（**図 7**，**図 8**）からは，*flat* で事例ベース活用率が低い問題ほど，*index* では大幅に向上する傾向がみられる.

これらについては，CASL か，CASL II かには関係

表 1 事例ベース活用率の平均の比較
Table 1 Comparisons of the *index*-system and the *flat*-system in average percentages of successful retrievals.

	事例ベース活用率の平均 (%)		
	flat	index	増減
CASL（17 問）	71.3	82.0	+10.7
CASL II（14 問）	73.9	84.3	+10.4

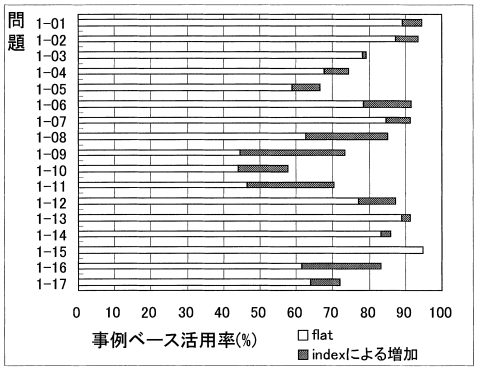

図 7 各問題における事例ベース活用率（CASL）
Fig. 7 Percentages of successful retrievals in each problem (CASL).

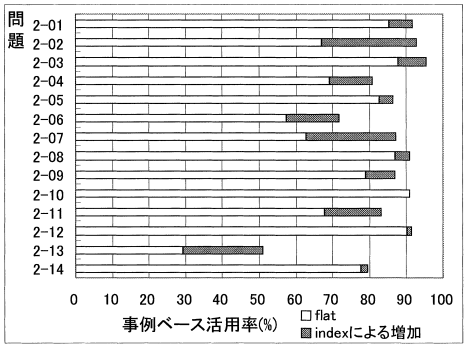

図 8 各問題における事例ベース活用率（CASL II）
Fig. 8 Percentages of successful retrievals in each problem (CASL II).

なく同程度の向上効果があったことから，提案手法は，初等アセンブラプログラミングであれば，具体的言語に依存せずに効果が期待できることが示唆される．以上より，提案手法が事例ベース活用率向上について有効であることが明らかになった．

5.2.2 検索と更新の処理時間

表 2 に事例検索と事例ベース更新の処理時間の比較を示す．これは，CASL と CASL II の両方の実験結果，すべてをまとめたものである．なお，実験に使用した計算機の CPU は Pentium III（周波数 1 GHz），メモリは 256 MB，OS は WindowsNT である．結論

表 2 事例検索・更新時間の比較
Table 2 Comparisons of the *index*-system and the *flat*-system in the processing time of retrieving cases and updating case-bases.

	flat	index
事例検索時間の平均 (sec)	0.082	0.052
事例検索時間の最大値 (sec)	0.69	0.58
事例ベース更新時間の平均 (sec)	0.085	0.076
事例ベース更新時間の最大値 (sec)	0.69	0.86

からいうと，*flat* と *index* のどちらも，検索時間，更新時間とも，1 秒以下であるので，実用上問題はない．2 つの実験システムを比較すると，次のようなことがいえる．

事例検索に関しては，*index* は *flat* よりも処理時間が短い．プログラムの照合処理に着目すると，1 つのプログラムに対する事例検索で *flat* は平均 13.0 回，*index* は平均 5.2 回であった．インデックスを付与することにより，プログラムの照合回数が少なくなるので，インデックスをたどるなどの余分な処理が増えても処理時間は短くなると考えられる．

一方，事例ベース更新に関しては，平均的には *index* は *flat* よりも処理時間が短いが，新事例を追加する場合には *index* の方が処理時間は長い．事例ベース更新は，事例検索後にその結果に基づいて行うので，事例検索時間と事例追加時間の合計となる．*flat* での新事例追加は，単に事例データを追加するだけであるのに対して，*index* ではノードの検索と生成処理が加わるためである．

6. むすび

初等アセンブラプログラミング評価支援システムのための事例ベース構築法を提案した．CASL と CASL II のプログラムを対象として，事例ベース検索・更新実験を行ったところ，本手法はフラットな事例ベースに比較して，事例ベース活用率を約 10%向上できることが示された．

本研究ではアセンブラプログラムを想定して検討を行ったので，提案した事例ベース構築手法は初等アセンブラプログラミングの範囲内であれば，ほとんどそのまま適用可能であると考えられる．これは実験において，CASL と CASL II とで事例ベース活用率向上に同様な効果があったことからも示される．

高級言語に拡張した場合でも，3.1 節で述べた基本的な方針は適用可能であると思われる．これらのうち，プログラム一般化における 3 つの観点は高級言語においては，(a) ある処理を行う文の種類，(b) 文の順序，(c) 冗長な処理を行っている文の有無ととらえること

ができる．これらの方針に基づいた，高級言語に対する具体的な事例ベース構築法の開発は，今後の重要な研究課題といえる．

謝辞 本研究の一部は文部省科学研究費補助金（No.12780293），人工知能研究振興財団研究助成（No.12AI320-1）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) Watanabe, H., Arai, M. and Takei, S.: Case-Based Evaluation of Novice Programs, *Proc. AI-ED2001*, pp.55 – 64 (2001).
3) 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).
4) Kolodner, J.: *Case-Based Reasoning*, Morgan Kaufmann Publishers, Inc. (1993).
5) Leake, D. (Ed): *Case-Based Reasoning*: *Experiences, Lessons and Future Directions*, AAAI Press/MIT Press (1996).
6) 仲谷善雄：事例ベース推論の動向，人工知能学会誌，Vol.17, No.1, pp.28–33 (2002).
7) 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, pp.2-93–94 (2000).
8) 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評価支援システムのための事例ベース構築法，情報処理学会研究報告，コンピュータと教育，Vol.2001, No.61, pp.23–30 (2001).
9) Watanabe, H., Arai, M. and Takei, S.: A Method of Constructing Case-base for Evaluation Assistant of Novice Programs, *Proc. ICCE2001*, Vol.1, pp.313–320 (2001).
10) Adam, A. and Laurent, J.P.: LAURA, A System to Debug Student Programs, *Artificial Intelligence*, Vol.15, pp.75–122 (1980).
11) Murray, W.R.: Automatic Program Debugging for Intelligent Tutoring Systems, *Computional Intelligence*, Vol.3, pp.1–16 (1987).
12) Johnson, W.L.: Understanding and Debugging Novice Programs, *Artificial Intelligence*, Vol.41, pp.51–97 (1990).
13) 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).
14) 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，電子情報通信学会論文誌，Vol.J78-D-II, No.V2, pp.321–332 (1995).
15) Kim, S. and Kim, J.H.: Algorithm Recognition for Programming Tutoring Based on Flow Graph Parsing, *Applied Intelligence*, Vol.6, Iss.2, pp.153–164 (1996).
16) 小西達裕，鈴木浩之，伊東幸宏：プログラミング教育における教師支援のためのプログラム評価機構，電子情報通信学会論文誌，Vol.J83-D-I, No.6, pp.682–692 (2000).
17) Micahlski, R.S.: A Theory and Methodology of Inductive Learning, *Machine Learning: An Artificial Intelligence Approach*, Michalski, R.S. (Ed), Tioga, Palo Alto, pp.83–129 (1983). 電総研 AI 研究グループ（訳）：機能学習の理論と方法論，知識獲得入門，pp.91–148, 共立出版 (1987).

付 録

A.1 事例ベースデータ構造の例

4.1 節で述べたデータ構造で表現されたデータの具体例を示す．ここで示す例は，次の問題に対するプログラムの評価事例とそのインデックスである．

> **問題：** メモリの連続した領域に N 個の整数が格納されている．指標レジスタを用いて，これらの合計を求め，結果をメモリに格納するプログラムを作成せよ．なお，データの個数 N はメモリに格納されているものとする．

A.1.1 インデックスルートのエントリの例

・識別子：f-001
・命令数の上限値：(0 0 2 1 1 1 2 0 0 0 2 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0)
・命令数の下限値：(0 0 0 0 0 0 0 0 0 0 0 0 2 0 1 0 3 0 3 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1)

A.1.2 第 1 階層ノードの例

・識別子：f-001
・プログラムリスト：

```
1   PGM     START
2           LD      GR1,N
3           SET     GR0 0
4   LOOP    INC     GR1 -1
5           ADDA    GR0,DATA,GR1
6           COMP    GR1 AA
7           J+-OR+  LOOP
8           ST      GR0,ZZ
9           RET
```

・第 2 階層 a グループノードへのリンク：s-001
- 命令数の上限値：(0 0 2 1 1 1 2 0 0 0 2 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0)
- 命令数の下限値：(0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1)

・第 2 階層 a グループノードへのリンク：s-004
- 命令数の上限値：(0 0 2 1 1 1 2 0 0 0 2 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0)
- 命令数の下限値：(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1)

・第 2 階層 b グループノードへのリンク：s-002
- 命令の順序情報：1 (2 3) 4 5 6 7 8 9

・第 2 階層 c グループノードへのリンク：s-003
　-冗長命令情報：　空
・第 2 階層 c グループノードへのリンク：s-011
　-冗長命令情報：(7 6 JPL)
・第 2 階層 c グループノードへのリンク：s-018
　-冗長命令情報：(2 1 LAD)

命令の順序情報の丸括弧は，順序の違いが些細であり，順序の入れ換えが可能であることを示す．冗長命令情報は，冗長命令が位置する行，直前の行，命令コードの組で構成される．

A.1.3 第 2 階層ノードの例

A.1.3.1 第 2 階層「命令の種類」ノードの例

・識別子：s-004
・命令数の上限値：(0 0 2 1 1 1 2 0 0 0 2 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0)
・命令数の下限値：(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1)
・プログラムリスト：

```
1    SUM     START
2            LD      GR1,N
3            SET     GR0 0
4    LOOP    INC     GR1 -1
5            ADDA    GR0,DATA,GR1
6            CPA     GR1,ZERO
7            JPL     LOOP
8            ST      GR0,ZZ
9            RET
```

・第 3 階層ノードへのリンク：c-002 c-005 c-006 c-010

A.1.3.2 第 2 階層「命令の順序」ノードの例

・識別子：s-002
・命令の順序情報：1 (2 3) 4 5 6 7 8 9
・第 3 階層ノードへのリンク：c-001 c-002 c-005 c-006 c-007 c-010

A.1.3.3 第 2 階層「冗長命令」ノードの例

・識別子：s-018
・冗長命令情報：(2 1 LD)
・第 3 階層ノードへのリンク：c-010

A.1.4 第 3 階層ノードの例

・識別子：c-010
・プログラムリスト：

```
1    SUM     START
2            LAD     GR1,0
3            LAD     GR0,0
4            LD      GR1,N
5    LOOP    LAD     GR1,-1,GR1
6            ADDA    GR0,DATA,GR1
7            CPA     GR1,C
8            JPL     LOOP
9            ST      GR0,ZZ
10           RET
```

・合否判定結果：accept
・アドバイス文：2 行目の LAD 命令はなくてもかまいません．
・事例更新日時：2002 9/15 16:39
・事例更新者：渡辺 博芳
・命令数：(0 1 1 2 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1)
・命令の順序情報：1 3 2 4 5 6 7 8 9
・冗長命令情報：(2 1 LAD)

命令の順序情報は，冗長命令を削除し，以降の行番号を詰めた形で表現している．

(平成 14 年 2 月 22 日受付)
(平成 14 年 12 月 3 日採録)

渡辺　博芳（正会員）

1963 年生．1986 年宇都宮大学工学部情報工学科卒業．1988 年同大学院工学研究科修士課程修了．工学博士．栃木県工業技術センターを経て，1991 年より帝京大学理工学部情報科学科助手．知識工学，事例ベース推論，機械学習，知的教育/学習システム等に興味を持つ．2000 年情報処理学会第 60 回全国大会大会優秀賞．2002 年私立大学情報教育協会第 10 回情報教育方法研究発表会奨励賞．電子情報通信学会，人工知能学会，教育システム情報学会，AAAI 各会員．

荒井　正之（正会員）

1958 年生．1981 年東京理科大学理工学部経営工学科卒業．1995 年宇都宮大学大学院工学研究科生産・情報工学専攻博士後期課程修了．工学博士．1988 年帝京大学理工学部情報科学科助手，現在同大学専任講師．ニューロコンピューティング，文字認識，自然言語理解，知識工学等に興味を持つ．著書「文字認識技術」(トリケップス社）等．電子情報通信学会，IEEE Computer Society 各会員．

武井　惠雄（正会員）

1938 年生．1961 年東北大学理学部天文及地球物理学科第二卒業．1963 年同大学院理学研究科地球物理学専攻修士課程修了．理学博士．東北大学理学部助手，同大学情報処理教育センター助教授を経て，1992 年帝京大学理工学部情報科学科教授．力学系，知的信号処理，分散環境における協調学習環境の実現，情報教育の哲学的側面等に興味を持つ．地球電磁気・地球惑星圏学会，日本教育工学会各会員．