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

小西　達裕†　　鈴木　浩之††　　伊東　幸宏†

A Method of Automated Evaluation of Learners' Programs for Assisting Teachers

Tatsuhiro KONISHI†, Hiroyuki SUZUKI††, and Yukihiro ITOH†

あらまし　本論文では，プログラミング演習における教師支援を目的として，プログラム理解技術の応用により学習者のプログラムを自動的に評価する手法を提案する．学習者プログラムの評価はプログラムの正誤だけではなく，教師が演習問題を出題する際の教育目標を満たすか否かに基づいて行う必要がある．そのためにまず，教育目標は演習問題の解答プログラムの標準アルゴリズムとして表現可能であることを示し，その具体的表現手法として拡張PADを提案する．拡張PADは処理順序の任意性やある機能の実装方法の任意性を陽に表現できるため，教師が記述の詳細度を教育目標に応じて調整できるという特徴がある．次に学習者プログラムと拡張PAD形式で記述されたアルゴリズムの照合手法を提案する．最後に提案した手法に基づいて試作システムを構築し，初等プログラミング演習のモデルコースを想定してシステムの実用性に関する評価実験を行った結果を報告する．

キーワード　教師支援システム，プログラム評価，初等プログラミング教育，知的教育システム

1. まえがき

我々は，初等プログラミング演習において，教師が学習者プログラムを評価する作業を支援するシステムの構築手法について検討している．本論文では，このシステムの中心的機能であるプログラムの自動評価の手法を提案する．また，試作した実験システムを用いた評価実験の結果について述べる．

プログラム診断技術のプログラミング教育への応用に関しては，これまでにも多くの研究が行われている．しかし，そのほとんどは学習者支援を目標とするものである[1]～[5]．これに対し本論文では，プログラム診断技術の教師支援への利用を考える．

一般にプログラミング演習を行う教師は，まず学習者に学習させるべき項目（教育目標）を整理し，その項目に関連する問題を出題する．教師は多数の学習者から提出されたプログラムを読み，はじめに設定した教育目標を満たしているか否かに基づいて評価し，学習者に評価結果をフィードバックする．ここで，多数のプログラムを評価する作業は教師にとって非常に大きな労力を要する．したがって，学習者プログラムを，教育目標を満たすものとそうでないものに自動的に弁別できれば，教師に対する大きな支援となる．

本研究と同様に教師支援を目的としたプログラム評価システムの例として，服部らのシステム[6]がある．このシステムでは教師が評価したプログラム例をデータベースに蓄積しておき，学習者のプログラムを，それらのプログラム例とデータフローレベルで突き合わせ，等価性を判断する．等価な場合，教師はデータベースに蓄積された評価を再利用できる．データベース中に学習者プログラムと等価なプログラムがない場合には，教師が自力でプログラムを評価し，評価結果をデータベースに追加する．空のデータベースから始めれば，教師による事前の準備は不要である．これに対し我々は，教師が演習問題の教育目標を陽に記述してシステムに与え，これをもとに学習者プログラムを評価するというアプローチをとる．

そこで，本論文では以下に焦点を当てて論ずる．

（i）教師が教育目標をシステムに伝達する手法

（ii）教育目標に基づく学習者プログラムの評価手法

以降，**2.**では教育目標の表現方法について述べ，**3.**では従来の学習者支援システム・教師支援システムと我々のシステムの類似点と相違点について考察する．

† 静岡大学情報学部，浜松市
Faculty of Information, Shizuoka University, Hamamatsu-shi, 432-8011 Japan
†† 静岡大学大学院理工学研究科，浜松市
Graduate School of Science and Engineering, Shizuoka University, Hamamatsu-shi, 432-8011 Japan

4.では教育目標に基づいて学習者プログラムを評価する手法について述べる．5.では構築した実験システムの構成を示し，6.では評価実験の結果を報告する．なお，本研究ではPascalを対象言語とする．

2. 教育目標の表現法

2.1 教育目標の分類

我々はプログラミング演習を担当する教師へのインタビューから，教師が演習問題に設定する教育目標を表1のように分類した．このうち本論文ではAタイプの教育目標の取扱いについて述べる（Bタイプの目標についても具体的方法を検討している[7]）．

ここでa1.～a3.の教育目標は，演習問題を解く標準アルゴリズムを以下のように記述すれば，その中に反映させることができる．

a1., a2.：標準アルゴリズム中に，目標である制御構造，データ構造を用いることを陽に指定する．

a3.：目標である問題解決方法を適切な抽象度のアルゴリズムとして記述する．

したがって，学習者プログラムが教育目標を満たすか否かは，各プログラムが標準アルゴリズムに対応するか否かを調べれば判定できる．以上より，本システムでは，教師が教育目標をシステムに伝える媒体として，標準アルゴリズムを利用することにする．

2.2 標準アルゴリズム表現の基本要素

標準アルゴリズム表現は，以下の条件を満たす必要がある．

（1） 記述単位についての条件

教師がアルゴリズムを記述するためには，アルゴリズムの記述単位が明確に定義されており，かつその種類が把握できる程度の数に収まること，更にそれらの記述単位が，初等プログラミング演習に現れる様々なアルゴリズムやデータ構造に対する十分な表現能力をもつことが必要である．

表1 教育目標の分類
Table 1 Pedagogical goals of exercises.

A.特定のプログラミング言語に依存した目標	a1.ある言語における制御構造を学習する． Pascalのwhile文，repeat文，if文など． a2.ある言語におけるデータ構造を学習する． Pascalの配列，レコード，ポインタなど． a3.それらのデータ構造，制御構造を用いた問題解決方法を学習する． Pascalを用いて配列から最大値を求める方法など．
B.プログラミング言語によらない目標	b1.プログラミング言語によらず一般的に利用されるデータ構造（抽象データ構造）を学習する． 木，リスト，表など． b2.抽象データ構造を用いた問題解決方法を学習する． 木の接点の探索，リストのソーティングなど．

（2） 記述の抽象度についての条件

教師が，自分の意図する抽象レベルで標準アルゴリズムを記述できることが必須である．教師がa1., a2.のタイプの教育目標をもつとき，目標である制御構造やデータ構造については学習者がどのような命令を用いて使用しているかまでをチェックしなければならないが，それ以外はある程度自由にプログラミングしてあっても許容できる．またa3.のタイプの教育目標をもつ場合も，問題解決の本質にかかわる部分は詳細な手順までチェックする必要があるが，例えば結果に添えるメッセージの出力など，本質的でない部分については実装方法にこだわる必要はない．したがって教師が教育目標を反映した標準アルゴリズムを記述する際には，実装方法まで具体的に指定したい部分と，大まかな機能のみ実現されていれば詳細は指定する必要はない部分が生じる．前者の部分を具体的に，後者の部分を抽象的に記述できなければならない．

以上の条件を満たすアルゴリズム記述手法について検討する．

アルゴリズムとはあるデータ構造に対する操作と一連の操作群の制御構造とを記述したものである．よって，データ構造に対する操作と制御構造指定のための記述がアルゴリズムの記述単位となる．制御構造指定のための記述は，逐次構造（Pascalではbegin文に対応），選択構造（if文，case文），反復構造（for文，while文など）の指定方法を定めておけばよい．

問題解決に利用されるデータ構造のタイプは，初等プログラミングに限定すれば，必要なものをすべて列挙できる．本研究ではこれらを「典型的データ構造」と呼ぶ．我々は市販教科書[8],[9]並びに筆者らの所属した静岡大学工学部情報知識工学科における初等プログラミング演習の問題例について事例研究を行った．その結果，抽出された典型的データ構造は，単変数（整数型，実数型，文字型など），集合型，配列型，レコード型などの10種類であった．

また，この典型的データ構造に対して行われる典型的な操作も，各データ構造ごとに列挙できる．これらを「典型的操作」と呼ぶ．上述の10種の典型的データ構造に対し，上と同様の範囲での事例研究の結果，「整

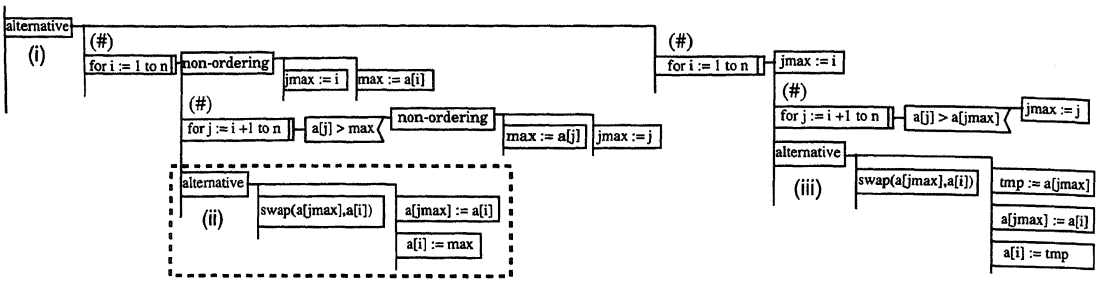

図1 拡張 PAD の記述例
Fig. 1 An example of extended PAD.

数型単変数に対するデータの代入」,「文字集合に対する和集合計算」,「配列に対するデータの読込み」など,38 種類の典型的操作を挙げることができた.

典型的操作の中には,他のより小さな粒度の操作の組合せで構成されるものがある.例えば「配列へのデータの読込み」という操作は,配列の要素に対するデータの読込みの反復によって構成できる.この関係に従って,典型的操作間に粒度に基づく階層を定義できる.この階層の末端には,プログラミング言語で用意されている一つの命令で実現できる操作がくる.これらの操作をプリミティブ操作と呼ぶ.

この階層中のどのレベルの操作でもってアルゴリズムを記述するかによって,前述の(2)の条件をクリアできる.例えば,教師は「配列へのデータの読込み」という操作全体を1単位として記述するか,「配列の1要素へのデータ読込み操作」と反復構造とを陽に記述するかを選択することによって,望ましい抽象度でこの操作を記述できる.

事例研究の範囲では,初等プログラミング演習で取り扱われる問題のうち,**2.1** で述べた A タイプの教育目標をもつものに対し,列挙した典型的操作の適当な階層のものを利用することで,その教育目標を十分に反映できることが確認できている.

2.3 標準アルゴリズムの表現方法

一般にアルゴリズムは逐次構造・選択構造・反復構造という3種類の基本構造からなる.したがってアルゴリズム表現は少なくともこの三つを表現できなければならない.また上述の操作の階層性を取り扱うために,ある操作とそれを構成するより粒度の小さい操作群(以後,部分操作群と呼ぶことにする)の関係の表現が必要である.これを部分操作構造と呼ぶことにする.更に,アルゴリズムを抽象的に表現するためには,以下の2種類の構造を記述できる必要がある.

(1) 任意順序構造(non-ordering)
任意の順序で実行してよい操作列を表現する.

(2) 任意手段構造(alternative)
ある操作を構成する部分操作群が複数通り考えられるとき,それぞれの群を並列に記述して実現手段が任意であることを表現する.

以上より,標準アルゴリズム表現は6種類の基本構造をもつ必要がある.

本研究では,図的表現である PAD 表現を,上述の2種類の構造が記述できるように拡張して利用する.拡張 PAD によるアルゴリズム記述例を図1に示す.図1は,最大値選択法ソーティングのアルゴリズムである.図1中の "alternative" は任意手段構造であり,(i) では最大値のインデックスのみを変数に記憶する方法と,最大値そのものを記憶する変数も用意する方法のどちらを用いてもよいこと,(ii),(iii) では発見した最大値を配列の適切な位置におく処理は swap 文と代入文のどちらを用いてもよいことが表現されている.また "non-ordering" は任意順序構造を示し,代入の順序はどちらが先でもよいことが表現されている.

この記述は,以下の教育意図を想定している.

(1) Pascal の for 文を用いて多重反復を記述すること(a1. ある言語における制御構造の学習)

(2) Pascal の配列を用いてデータ列を表現すること(a2. ある言語における制御構造の学習)

(3) Pascal で最大値選択法ソーティングを正しく行うこと(a3. 問題解決方法の学習)

(1)は図中(#)のステップにより,(2)はアルゴリズム中で配列が使用されていることにより表現される.(3)は拡張 PAD 全体によって示されている.

図中,破線で囲んだ部分では swap 命令若しくは代入文の組合せによりデータの交換を行うことが指定されている.仮に Pascal の swap 命令を学習した直後の

演習で，swap 命令を用いることが教育意図に含まれる場合には，この部分の記述に任意手段構造を使用せず，単に swap 命令のみを指定すればよい．

システムへの入力は，将来的にはこの拡張 PAD 形式で図形的に行うことを考えているが，現在は同等の情報をもつ Lisp 言語の S 式を用いている．

3. プログラム評価手法に関する基礎的考察

まず，教師支援のためのプログラム評価手法と，従来の学習者支援を目的としたプログラム評価手法の共通点，相違点について考察する．

（1） 共通点

従来のシステムでは，ターゲットプログラムの構造を，抽象的に記述されたアルゴリズムと静的に突き合わせて評価する手法が多く用いられている．例えば LAURA [1] ではプログラムモデルと呼ばれるグラフ構造で記述された抽象的アルゴリズムの等価性を判定する．PROUST [2] ではプランと呼ばれる階層的記述手法で表現されたアルゴリズムと学習者プログラムを突き合わせる．我々の提案する方法もこの範疇（ちゅう）に属する．

（2） 相違点

(2-1) 従来の学習者支援システムでは，プログラムの同等性の評価において，計算結果の等価性を基準としている．このため，静的解析システムにおいてはしばしば抽象的アルゴリズムを変形ルールを用いて変形するという手法がとられている（例えば LAURA では，プログラムモデルのグラフ構造をヒューリスティックスにより変形することで，ブロック構造の異なるプログラムの等価性を判定している）．

また，プログラムに一定の初期値を与えて実行シミュレーションを行い，その結果を用いてプログラムを評価する動的解析システム [10], [11] も，基本的には計算結果の等価性を基準に評価を行っている．

しかしながら教師支援システムにおけるプログラムの評価は，教師が想定した教育目標を満たしているかを基準に行われるべきである．正しい結果を返すプログラムでも，教育目標を満たしていなければ，教師は教育的指導を行う必要があるからである．例えば教師支援システムでは，教師がプログラムの表層構造まで言及した教育目標（読みやすい条件分岐の書き方，処理の意味に対応したブロックの分割位置など）をもつ場合，計算結果が同じであっても，必ずしも同等のプログラムとみなすべきではない．

(2-2) 学習者支援システムでは，評価誤りが直ちに学習者の混乱や誤解を引き起こすため，多様なバグを含む学習者プログラムを高い精度で診断する能力が必須となる．そのために多くのバグ知識やバグ発見のためのヒューリスティックスを用いる場合が多く，評価プロセスの計算量も多大になりがちである．しかし教師支援システムでは，教育目標を満たさないと判定されたプログラムの最終的評価は教師が下すので，誤ったプログラムを正しいと判定することさえなければ，ある程度の判定誤りが含まれても実用上大きな問題とはならない．そこで，精度よりも速度を優先した照合アルゴリズムを採用することが可能である．我々のシステムでは，プログラム表層におけるブロック構造の一致性に重点をおいた照合アルゴリズムを採用する．この手法は照合対象をブロック単位で限定できるので，ステートメントの比較回数を減らすことができる．一方で，教育目標がプログラムの表層構造にない場合には，正しいプログラムに対して教育目標を満たさないものと判定してしまうこともあり得るが，上述の理由により，このことは致命的な問題にならない．

次に，教師支援システムである服部らのシステムと我々のアプローチを比較する．我々のシステムは事前に教育目標すなわち標準アルゴリズムを記述する労力が必要となるが，服部らのシステムではその必要がない．しかしながら，

（1） 教師がプログラムの評価基準をシステムに与える際，自分の意図する抽象レベルを選択することができなければならない（**2.2**（2））．服部らのシステムでは，データベースに蓄積されるプログラム事例がこの評価基準にあたる．このシステムではプログラム事例を蓄積する際，プログラムを自動的にデータフローレベルへ抽象化するため，教師が抽象レベルを選択することが容易ではない．

（2） プログラミング演習において，小グループごとに少しずつ問題内容を変えたり，同一テーマで例年出題している問題を若干変更して出題することがよく行われる．我々のシステムではこのような場合，標準アルゴリズムを部分的に修正することにより再利用が可能である．一方，服部らのシステムにおいては，問題が変わればデータベースを空の状態から再構築するか，若しくはデータフローレベルに変換されたプログラムを書き換える必要がある．データフローレベルへの変換はシステムによって自動的に行われているので，教師がこれを適切に書き換えるためには，データ

フローを再解釈する必要があり，一般に容易ではない．

以上より，我々のアプローチは服部らのシステムと比較して，より労力を要する部分もあるが，応分の利点もあると考える．

4. 教育目標に基づくプログラム評価

ここでは学習者プログラムを標準アルゴリズム表現と比較照合して評価する手法について述べる．

4.1 前 処 理

比較照合のために，学習者プログラムと標準アルゴリズムに以下の前処理を行う．

（1） 数式の比較のために，プログラムと標準アルゴリズムに含まれる数式を正規化する．正規化処理では，各計算式の括弧を展開して多項式に変形し，定数項の計算と同類項の係数の計算を行って，可能な限り項をまとめる．

（2） 標準アルゴリズムの典型的操作のうち，プリミティブでないものは，そのままではプログラムと照合できない．よってそのような操作を部分操作群に展開する．展開に用いる知識は，非プリミティブの典型的操作各々に対して，それをより粒度の小さい操作群に変換する手続きとして実装する．例えば，典型的操作「データ交換」(2変数の値の交換）に対して，これを代入文の組合せによる手順に変換する手続きを知識ベースに格納する．この例では swap 命令を用いる手順も考えられるが，このように非プリミティブ操作の実現方法が複数ある場合には，それぞれの方法を任意手段構造の選択肢として統合した形式に変換する．

（3） 学習者プログラムに含まれる各制御命令を拡張 PAD の選択構造及び反復構造に置換する．またその他の命令をプリミティブな操作に置換する．

以上により，標準アルゴリズム，学習者プログラムともに拡張 PAD 形式のデータ構造に変換される．

4.2 照 合 処 理

本システムでは，学習者プログラムと標準アルゴリズムの照合を静的に行う．照合処理は基本的にパターンマッチによる．しかし，単純なパターンマッチでは比較の際の仮説空間が爆発し，仮説間の競合解消処理を行いきれなくなることが予想されるため，適当な戦略により探索空間を縮小する必要がある．我々は比較的単純で有効な方法として，Pascal プログラムのブロック構造に着目したヒューリスティックスを用いることにした．

Pascal プログラムは，最もマクロに逐次構造ブロックをもち，その内部に選択構造ブロック，反復構造ブロック，及び逐次構造ブロックが階層的に包含される構造をもつ．よってまず，標準アルゴリズム側のプログラム全体と，学習者プログラム全体を最もマクロなブロックから順にブロック単位で比較する．

ブロック同士の比較は以下のように行う．

（1） 標準アルゴリズム側ブロックのトップレベルの構成要素がブロックのときは，学習者プログラム側ブロックの中から対応しそうなブロック（制御構造のタイプが一致するもの）を選び，階層を下りて比較する．

（2） 標準アルゴリズムの中で任意手段構造が用いられている場合には，すべての手段について学習者プログラムとの対応を調べ，最も強く対応している手段との対応関係を採用する．対応の強さは，後述する仮説の量で判定する（仮説生成方法の詳細は付録参照）．

（3） 標準アルゴリズム側ブロックのトップレベルの構成要素が操作のときは，学習者プログラムのブロック中から対応する操作を探し，両者が対応づくという仮説を生成する（その際，変数間の対応も同時に保存する）．操作の対応関係の判定では，まず操作の種類を比較し，一致していれば操作に含まれる式を比較する．式の比較に失敗した場合，式に含まれる各変数に同一ブロック内で代入が行われているならば，その変数を代入文の右辺の式に置き換えた式も考慮する．更に，その時点までに蓄積された仮説との矛盾をチェックし，矛盾していればこの仮説を棄却する．仮説間の矛盾のチェックは以下の方法で行う．

（i） 変数間の対応関係の一貫性を検査する．例えばある仮説中で標準アルゴリズム中の変数 x が学習者プログラム中の変数 x' と対応しているとする．このとき，以後の照合ではこれに矛盾する対応関係（例えば「変数 x と変数 y が対応する」）は棄却される．

（ii） 操作順序の無矛盾性を検査する．標準アルゴリズム中のブロック B と学習者プログラム中のブロック B' が対応しているという仮説があるとき，逐次構造中でブロック B より前に置かれているブロック A と，ブロック B' より後に置かれているブロック C の対応関係は棄却される．

ブロック同士の照応が終了すると，ブロック内の各要素を対応づけて得られる仮説の和が，ブロック同士が対応づく場合に想定できる仮説群として蓄積される．

仮説生成時には，互いに矛盾する複数の仮説を生成可能な場合がある．このときシステムは，これらの仮

説各々を採用した可能世界を生成して処理を続行する．ただしある可能世界において，照合に失敗したステートメント数が一定値を超えている場合，その可能世界に関する処理を打ち切る．

システムは最終的に，最も多くの仮説を含む仮説集合の生成された世界（すなわち，最も一貫した解釈が可能であった世界）を選択し，照合結果とする．

一般には，同一のアルゴリズムで書かれたプログラムでもブロック構造が異なる場合があり，上記の方法ではこの種のプログラムと標準アルゴリズムの照合に失敗する．しかしながら，初等プログラムに現れる程度の単純な選択構造，反復構造はほぼ安定したブロック構造で記述される場合が多い．また逐次構造についてはブロックの区切り位置がずれることで多数の異なるブロック構造が現れ得るが，これらは連続した逐次実行ブロックを一つのブロックとして扱うことにより同一の構造とみなすことが可能である．

照合処理アルゴリズムの詳細[12]は，付録として本論文末尾に示す．

5. 試作システムの構成

図2に試作システムの構成を示す．教師は拡張PAD形式で標準アルゴリズムを記述し，システムに入力する．システムは前処理として，プリミティブでない操作をプリミティブ操作に展開する（図中，Derivation）．また学習者プログラム群が前処理ユニットに入力され，拡張PADと比較可能な形式に変換（Transform）される．照合部（Comparison）は前処理の結果を受け取り，両者を比較して対応関係を教師に示す．

本システムはCommon Lispで記述されており，

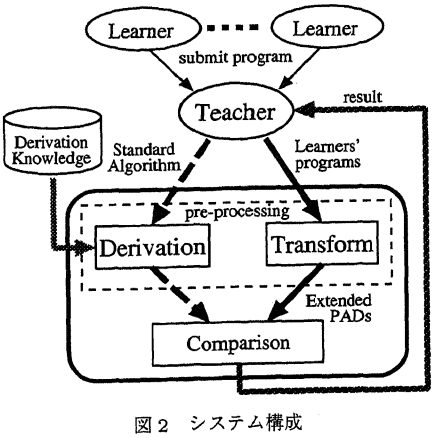

図2　システム構成
Fig. 2　Architecture of our system.

UNIX上若しくはMS-DOS上のGCLで動作する．ソースコードの大きさは約364 KBである．

6. 評価実験

試作システムを用いた評価実験の結果を示す[13]．本実験では日本電算機JU2/2300（CPU: Ultra SPARC-II 300MHz*2, SPECint95:12.3, SPECfp:20.2）を用いた．

6.1 プログラム評価能力の評価

6.1.1 評価実験のねらい

教育システムの実用性を評価するには，そのシステムを現実的な教育課程に適用した場合にどの程度の能力をもつかを検証する必要がある．我々は一つの方法として，初等プログラミングの標準的な教育項目をまとめてモデルコースを設計し，ここで出題される演習に対して本システムを適用した．このモデルコースは筆者らの所属した静岡大学工学部において，プログラミング入門科目として開講された半年間の授業科目を母体にしており，実験に用いた学習者プログラムはこの科目でレポートとして提出されたものである．モデルコースは12回（1回3時間）の授業からなり，以下の6テーマについて講義と演習を各1回行う．

（1）　逐次構造プログラム
（2）　様々なデータ型
（3）　選択構造プログラム
（4）　反復構造プログラム
（5）　配列と反復構造プログラム
（6）　学習のまとめ

我々は以下の点に着目して実験的評価を行った．

（a）　標準アルゴリズムを記述する教師の労力は過大にならないか．

（b）　提出された実際のプログラムを処理した結果に，「完全照合」若しくは「部分照合」が十分高い割合で含まれるか．

（a），（b）はトレードオフの関係にある．なぜなら教師が標準アルゴリズムを記述する際，任意手段構造を用いて可能なプログラムパターンを多く記述すればするほど，照合率は向上するからである．よって本評価実験では，標準アルゴリズムの記述量を，教師に負担にならない程度（ここでは模範解答のステップ数の2倍とした）に制限し，その条件下で教師にとって有用な水準の照合率が達成できるかを調べる．評価実験に用いた演習問題を表2に示す．これらは上述のモデルコースにおけるテーマ（1）～（5）に対応している．

表2 評価実験に用いた演習問題
Table 2 Target exercises for the experiment.

演習問題（1）	鶴亀算を解くプログラムを作れ.
演習問題（2）	文字データを読み込み，大文字を小文字に変換して出力するプログラムを作れ.
演習問題（3）	"0～9" 及び "A～F" の文字データで表現された2けたの16進数を10進数に変換して出力するプログラムを作れ.
演習問題（4）	方程式 "ax^3+bx^2+cx+d=0" をニュートン法を使って解くプログラムを作れ.
演習問題（5a）	単純選択法により配列中の数値データをソーティングするプログラムを作れ.
演習問題（5b）	ガウスの消去法を用いて多元連立1次方程式を解くプログラムを作れ.

6.1.2 実験方法と結果

（1） 実験方法

(1),(2) の各問題について，実際に学習者から提出されたプログラム（(1) 42例,(2) 56例）を試作システムで評価した．これらの結果に基づき，以下の観点から試作システムの実用性を評価した．なお，ここでは，照合に失敗したステートメント数に基づく照合打切り処理ははずし，プログラムの先頭近くが全く照合しないプログラムでも，最後まで照合を続けさせるようにした．

（A） 各ステートメントの評価の妥当性

（B） 完全照合・部分照合の比率

（2） 実験結果

（A） 各ステートメントの評価の妥当性

試作システムによる評価の妥当性を検証するために，以下の分析を行った．

- 学習者が提出したプログラムについて人間の教師が評価し，ステートメントごとに標準アルゴリズムに対応するものとしないものに分類しておく．
- これをシステムの評価結果と比較する．人間の教師の評価結果と一致すれば「妥当な評価」，さもなくば「不適切な評価」として集計する．

結果を表3に示す（表中のステートメント数には，関数定義・変数宣言・ブロックの開始終了などを含まない）．この結果によれば，全ステートメントの4分の3以上を妥当に評価している．また，不適切な評価を行ったものを調査した結果，1例を除くすべてが，本来対応づくはずのものを対応づかないと判定したものであった．対応づけてはいけないものを対応づけた唯一の例は出力文であり，出力メッセージの内容が不適切というものであった（具体的には,「鶴の数は」と出力させる位置で「亀の数は」と出力したもの）．

（B） 完全照合・部分照合の比率

表3 試作システムによる評価の妥当性
Table 3 Validity of evaluation by our system.

	演習問題（1）	演習問題（2）	合計
全ステートメント数	228	244	472
「妥当な評価」	194	168	362
「不適切な評価」	34	76	110
「妥当な評価」の割合	85%	69%	77%

表4 試作システムによる各プログラムの評価結果
Table 4 The result of evaluation by our system.

	演習問題（1）	演習問題（2）	合計
プログラム数	42	56	98
完全照合	17(40%)	28(50%)	45(46%)
部分照合	4(10%)	2(4%)	6(6%)
照合失敗	21(50%)	26(46%)	47(48%)

演習問題 (1),(2) に対するプログラム全体の照合結果を表4に示す．表4において，完全照合とは以下のものを指す．

- 学習者プログラムのすべての命令が，標準アルゴリズムにマッチし，
- かつ，標準アルゴリズム中のすべての操作が（ただし任意手段構造については，列挙された手段のうちの一つが）学習者プログラムにマッチしたもの．

また部分照合とは以下のものを指す．

- 学習者プログラム中の命令のうち，70%以上が標準アルゴリズムにマッチし，
- かつ，標準アルゴリズム中で学習者プログラムとマッチしていないプリミティブな命令が全体の25%以下であるもの（ただし，任意手段構造を含む場合，選択されなかった方法に含まれる命令の数は無視する．またいずれの選択肢も学習者プログラムにマッチしなかった任意手段構造があれば，各選択肢に含まれる命令の最小値を数える）．

6.1.3 考　　察

本実験では，実際に学習者が作成したプログラム中の約7割程度のステートメントに対し，標準アルゴリズムとの対応関係を適切に評価でき，プログラムレベルで4割強を完全照合と判定することができた．プログラムレベルでの照合率は，部分照合を含めるとほぼ5割に達する．

ここで重要なのは，システムが学習者プログラムと標準アルゴリズムとの対応付けを行ったものは，例外的な事例を除き，すべて適切な評価であることである．これにより，システムが完全照合あるいは部分照合と判定した結果の信頼性は十分高いと考えられる．よって，教師は完全照合についてはほとんどプログラムを読む必要がなく，部分照合については照合に失敗した

部分だけを読めばよいことになる．

実験で用いた例の場合，詳細に読む必要があるプログラムを全体の約半分に減らすことができる．この結果は，模範回答のステップ数の2倍以下という条件で作成した標準アルゴリズムであっても，現実の学習者が正規の授業のレポートとして提出するプログラムがある程度安定して合致することを示しており，本実験の対象とした授業に本システムを適用した場合，レポートを評価する教師を有効に支援できると考えられる．

結果的に問題がないプログラムを照合失敗としたケースもあったが，その主要な原因は，学習者プログラムが問題の要求にない機能を実現していることであった．例えば，演習問題（2）のプログラムでは，小文字を大文字に変換する機能を加えた学習者が多数見られた．照合失敗若しくは部分照合となった28例のうち21例はこのケースである．我々は本研究の基本方針として，このように教師の要求を超えたプログラムに対しては，教師が自分の目で評価して標準より良い評点をつけるべきであると考えている．したがって，この種の照合失敗はむしろ望ましいことといえる．

6.2 典型的誤りへの対処能力の評価

（3）の問題について，学習者から提出されたプログラム49例中，誤りを含むプログラムが23例見られた．システムに二つの典型的誤りパターンを拡張PAD形式で与えて，これらの誤りプログラムと比較させたところ，13例については部分照合以上の結果を得ることができた．このことから，典型的誤りパターンの知識をシステムに蓄積することにより，誤りプログラムの自動評価も可能であることが確認できた．

6.3 多様なプログラムへの対処能力の評価

（4），（5a），（5b）の各問題について，正しいプログラム，軽微な誤りを含むプログラム，大きく誤ったプログラムの記述パターンを作成した．各問題ごとに一つの標準アルゴリズムを与え，これらのプログラムと照合させたところ，正しいプログラムとはすべて完全照合し，軽微な誤りを含むプログラムに対しては部分照合となることを確認した．大きく誤ったプログラムに対しては部分照合以上にはならなかった．ここで使用したプログラム数，平均ステートメント数，ブロック数，プログラム中に含まれる変数の数を表5に示す．

6.4 プログラム評価に要する計算量に関する考察

本論文で提案したプログラム評価手法の適用可能性をより明確にするために，（1）～（5b）の演習問題について，任意手段構造と任意順序構造を用いない標準アルゴリズムを定めて改めて評価させ，プログラム規模（学習者プログラム，標準アルゴリズム双方のステートメント数，ブロック数，変数の数）及び標準アルゴリズムの記述の抽象度（含まれる任意手段構造の数，任意順序構造の数）と，評価に要する時間並びに生成される可能世界フレームの数を整理した．結果を表6に示す．二つのプログラムを，変数間の対応関係を考慮して比較しようとすると，ステートメント間の比較

表5 多様なプログラムへの対処能力の評価実験
Table 5 Experiment on various exercises.

問題	学習者プログラム数			平均ステップ数	平均ブロック数	平均変数数
	正しいもの	軽微な誤りを含むもの	大きく誤ったもの			
（4）ニュートン法	7	1	6	9.9	2.0	6.6
（5a）単純選択法	1	10	3	9.8	4.0	5.1
（5b）ガウス消去法	1	11	12	27.3	11.8	11.0

表6 プログラム規模，アルゴリズム記述の抽象度と評価処理時間，可能世界フレーム生成数の関係
Table 6 Relation between program size, abstract level of standard algorithm, processing time and number of possible world frames.

問題名	標準アルゴリズム					学習者プログラム				処理結果					
	ステートメント数	ブロック数	変数総数	任意手段数	任意順序数	プログラム数	平均ステートメント数	平均ブロック数	平均変数総数	完全照合数	部分照合数	不照合数	平均処理時間 [sec]	平均PWF数	
（1）鶴亀算	5	1	4	0	0	42	9.2	9.1	1.4	0	4	38	0.96	9.4	
（2）大文字小文字変換	4	1	2	0	0	56	9.3	2.1	3.1	3	17	36	0.30	8.1	
（3）16進10進変換	11	5	4	0	0	49	19.7	7.7	5.9	0	0	49	1.50	13.2	
（4）ニュートン法	9	4	5	0	0	14	9.9	2.0	6.6	7	1	6	6.23	72.4	
（5a）単純選択法	10	4	5	0	0	14	9.8	4.0	5.1	1	10	3	0.30	30.7	(*)(#)
ソーティング	10	4	5	1	0	14	9.8	4.0	5.1	1	10	3	0.31	35.4	(*)
	10	4	5	2	0	14	9.8	4.0	5.1	1	10	3	0.43	43.9	(*)
	10	4	5	3	0	14	9.8	4.0	5.1	1	10	3	0.44	47.8	(*)
	10	4	5	0	1	14	9.8	4.0	5.1	1	10	3	0.35	42.6	(#)
	10	4	5	0	2	14	9.8	4.0	5.1	2	9	3	0.52	79.1	(#)
（5b）ガウスの消去法	28	12	11	0	0	24	27.3	11.8	11.0	1	11	12	9.87	599.2	
	28	12	11	11	0	24	27.3	11.8	11.0	1	11	12	31.00	2930.0	

表7 市販教科書における例題・演習問題の規模
Table 7 Size of exercises in a standard textbook of programming.

問題クラス	主な問題	問題数	平均ステートメント数	任意手段構造数			任意順序構造数	
				反復に関するもの（平均）	反復以外（平均）	反復以外（最大）	平均	最大
初級（1章～5章）	四則計算，数の総和など	19	8.5	1.1	0.2	3	1.2	5
中級（6章～11章）	数値積分，ソートなど	34	17.9	4.0	0.5	3	1.1	5
上級（12章～17章）	線形リストの生成，木の探索など	14	20.9	3.5	0.3	2	1.1	3

回数はステートメント数に対して階乗オーダで増大し，実用時間でシステムが稼動しなくなることが懸念される．しかし表6によれば，対象プログラムの規模を大きくしたとき，ステップ数・ブロック数・プログラムに含まれる変数の数に対する処理時間・可能世界数の増大は階乗オーダよりもかなり低い割合にとどまっている．最も規模の大きい演習問題（5b）についても，処理時間は平均9.9秒（最大で32.4秒）であって，十分に実用的時間で動作している．

これは，本システムではブロック構造の対応関係に基づいて比較しているために，比較対象となるステートメントが強く限定されること，変数間の対応関係，ステートメントの順序関係に関する制約が徐々に蓄積され，仮説生成の必要がない組合せが棄却されること，更に部分照合以上の結果となる可能性がない可能世界フレームの処理を打ち切るという戦略をもつことによると考えられる．

更に同一問題（(5a) 単純選択法ソーティング）に対して，標準アルゴリズム中に含まれる任意手段構造・任意順序構造の数を変化させ，許容される手順のバリエーションを増やした際の処理時間を調査した．

任意手段構造については，2通りの手段を許容する箇所を0箇所から3箇所まで変化させた．結果は表6の右端に（*）が付されている部分である．3箇所に許容性をもたせた場合，許容する手順には2の3乗=8通りのバリエーションが発生するが，実際の処理時間は約1.5倍の増大にとどまった．

同様に任意順序構造についても，二つのステップの順序の任意性をもつ箇所を0箇所から2箇所まで変化させた．結果は表6の右端に（#）が付されている部分である．2箇所に許容性をもたせた場合，許容する手順には2!× 2!= 4通りのバリエーションが発生するが，実際の処理時間は約1.8倍の増大にとどまった．

発生するバリエーションよりも処理時間の伸びがかなり小さくなるのは，上述の処理打切り戦略が極めて有効に働いていることが主な理由であると考えられる．学習者プログラムと一致しない手順との照合処理では，ほとんどのステートメントが対応失敗となるので，早い時点で部分照合以上になる可能性がなくなり，速やかに処理が打ち切られる．

更に，現実のプログラミング演習におけるプログラム規模と，標準アルゴリズムに含まれ得る任意手段構造，任意順序構造の数に基づいて我々のシステムの適用性を評価するために，初等Pascalプログラミングの教科書[8]における例題・演習問題のプログラム67題について，事例研究を行った．結果を表7に示す．表7の任意手段構造・任意順序構造の数は，各問について教師が標準的に許容すると思われる実装方法をカウントしたものである．

まずステートメント数について考察する．上述のように，演習問題（5b）程度の規模のプログラムは，任意手段構造・任意順序構造を含まない場合，平均10秒程度で処理可能である．この問題のプログラムステートメント数は36であるが，これは上級プログラムの平均値を上回っており，実際ステートメント数で36を上回るものは67題中2題のみである．

次に任意手段構造数について考察する．表7より，中級以上の問題では平均して4箇所程度の任意手段構造が含まれる．しかしこのほとんどが反復構造の実装をfor文・while文・repeat文のいずれで行うかの任意性である．この種の任意手段構造では，各々の手段に応じた可能世界が生成されても，不適切な手段を学習者プログラムと比較したとき，反復ブロック全体が不照合となり，直ちに処理が打ち切られる．このため，処理時間への影響はそれほど大きくならない．このことを確認するために，演習問題（5a）について，11箇所の反復構造をすべて任意手段構造で記述した場合の処理時間を測定したが，平均処理時間で約3倍の増加にとどまっている（表6最下段参照）．なお，11箇所より多くの反復構造をもつ問題は，67題中1題のみ（14箇所）である．反復構造による任意性を除くと，たかだか3箇所程度の任意手段構造を含むのみであって，表6の演習問題（5a）に関する実験結果に従えば，任意手段構造を含まない場合のたかだか数倍程度の処

理時間で処理可能と推定される．

最後に任意順序構造数について考察する．表7より，各問のアルゴリズムには平均して1箇所程度の任意順序構造が含まれる．1問に含まれる数は最大で5箇所であった．表6の演習問題（5a）に関する実験結果に従えば，これも任意順序構造を含まない場合のたかだか数倍程度の処理時間で処理可能と推定される．

以上より，現実的なプログラミング教科書に含まれる標準的規模のプログラムに対して，我々のシステムは実用的時間で処理が可能であると推定できる．

7. むすび

本論文では教師により与えられた教育目標に基づいて学習者プログラムを評価し，教師を支援するシステムについて述べた．現在，教師が標準アルゴリズムを記述しやすいような視覚的インタフェースを構築中であり，このインタフェースを含めて教師によるシステムの利用評価を行うことが今後の課題となる．

謝辞 本研究の一部は，日本学術振興会科学研究費補助金基盤研究C (11680216) による．

文　献

[1] A. Adam and J. Laurent, "LAURA, A system to debug student programs," Artif. Intell., vol.15, pp.75–122, 1980.

[2] W.L. Johnson, "Understanding and debugging novice programs," Artif. Intell., vol.42, pp.51–97, 1990.

[3] W. Murray, Automatic program debugging for intelligent tutoring systems, Morgan Kaufmann Publishers, Inc., 1988.

[4] H. Ueno, "Integrated intelligent programming environment for learning programming," IEICE Trans. Inf. & Syst., vol.E77-D, no.1, pp.68–79, Jan. 1994.

[5] 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（D-II），vol.J78-D-II, no.2, pp.321–332, Feb. 1995.

[6] 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, pp.21–28, April 1997.

[7] T. Konishi, A. Sugiyama, H. Suzuki, and Y. Itoh, "Evaluation of novice programs based on teacher's intentions independent of programming languages," Proc. ICCE97, pp.916–918, Kuching, Malaysia, Dec. 1997.

[8] 米田信夫，疋田輝雄，桜井貴文，Pascalプログラミング増訂版，サイエンス社，1990.

[9] 阿部圭一，ソフトウェア入門第2版，共立出版，1989.

[10] A. Barr, M. Beard, and R. Atkinson, "The computers as a tutorial laboratory," Int. J. Man-machine Studies, vol.8, pp.567–596, 1976.

[11] E. Shapiro, Algorithmic program debugging, MIT Press, MIT. 1983.

[12] T. Konishi, A. Suyama, and Y. Itoh, "Evaluation of novice programs based on teacher's intentions," Proc. ICCE95, pp.557–566, Singapore, Dec. 1995.

[13] Y. Itoh, T. Konishi, and H. Suzuki, "On experimental evaluation of an educational system that supports teachers of novice programming," Proc. ICCE98, vol.2, pp.234–238, Beijing, China, Oct. 1998.

付　録　照合処理アルゴリズム

処理の枠組みと用語

システムは，標準アルゴリズムと学習者プログラム中のブロック同士，若しくは操作同士を比較し，それらの対応関係に関する仮説を蓄積する．この仮説を「操作間対応仮説」(ACO: Assumptions on Correspondence between Operations) と呼ぶ．生成されるACOを管理するために，仮説木と呼ばれる木構造を用いる．仮説木のノードは，解析中のある時点におけるシステムの内部状態を保持するフレームである．これを可能世界フレーム（PWF: Possible World Frame）と呼ぶ．可能世界フレームは，その時点で生成されているACO群を保持する「操作間対応仮説スロット」，その時点で仮定されている変数間の対応関係を保持する「変数間対応仮説スロット」，今後対応を調べるべき〈標準アルゴリズム中の操作群，学習者プログラム中の操作群〉のペアのリストを保持する「仮説候補スロット」からなる（仮説候補の生成方法については下記 (2)）．システムは，比較処理を行う過程で，ACO若しくは仮説候補を生成するごとにPWFを生成し，その直前の時点のPWFにその下位ノードとしてリンクする．このリンクが仮説木の枝となる．なお一般には，互いに矛盾する複数のACOや仮説候補を生成可能な場合がある．このとき，各ACO若しくは各仮説候補ごとにPWFを生成し，仮説木上では兄弟ノードの位置に配置する．以下では仮説候補をC*:〈標準アルゴリズム側操作，学習者プログラム側操作〉，操作をop_*，逐次的操作列を(op_* … op_**)と表記する．op_*はプリミティブ操作，反復構造ブロック，選択構造ブロックのいずれかである（以下の説明では簡単のため，逐次構造ブロックは，はじめから逐次的操作列に変換されているものとする）．

照合アルゴリズム（紙数の関係上任意手段構造・任意順序構造の扱いを省略する．）

(1) 初期設定として，〈標準アルゴリズムを構成する全操作群，学習者プログラムを構成する全操作群〉というペアを生成し，仮説候補とする．仮説木のルートノードとしてPWFを一つ生成する．上で生成した仮説候補をこのPWF中の，「仮説候補スロット」に格納する．このPWFに着目する．(2) へ．

(2) 着目しているPWFについて打切り判定を行う．PWFに含まれる照合結果から，照合に失敗したステートメント数を標準アルゴリズム，学習者プログラムの双方について数え，以後の照合がすべて成功しても部分照合以上になることが不可能であるか，若しくは既に生成されたPWF中の最良の照合率を上回ることが不可能であれば，このPWFを棄却して (3) へ．さもなくば着目しているPWFの「仮説候補スロット」を見る．仮説候補がなければ (3) へ．仮説候補があれば先頭の一つに着目し，その仮説候補のタイプを調べて以下のいずれかの処理を行う．

(2-1) 仮説候補の標準アルゴリズム側が複数の操作からなっている場合：
このタイプの仮説候補は C0:⟨(op_A_1 … op_A_m), (op_B_1 … op_B_n)⟩ と書ける．このとき，二つの仮説候補 C1: ⟨op_A_1, (op_B_1 … op_B_n)⟩ 及び C2:⟨(op_A_2 … op_A_m), (op_B_1 … op_B_n)⟩ を生成する．C1 は op_A_1 と対応する操作を (op_B_1 … op_B_n) 中で探すための仮説候補である．可能世界フレーム PWF_1 を生成し，現在着目している PWF のスロット値をすべてコピーした後，それぞれの「仮説候補スロット」から C0 を取り除き，C1，C2 を加える．PWF_1 を着目中の PWF に下位ノードとしてリンクする．着目点を PWF_1 に移す．(2) へ．

(2-2) 仮説候補の標準アルゴリズム側が単一の操作である場合：

(2-2-1) 仮説候補の標準アルゴリズム側が反復ブロック（若しくは選択ブロック）である場合：このタイプの仮説候補は C0: ⟨op_A, (op_B_1 … op_B_n)⟩ と書ける．op_B_1 ～ op_B_n の中で，反復構造（選択構造）が一つもなければ，この仮説候補を棄却して (2) へ．さもなくば，以下の処理を行う．今，op_A と op_B_i (1≦i≦n) がともに反復構造（選択構造）であり，ブロック内部にそれぞれ条件命令 op_cond_A，op_cond_B と，ループ本体（選択構造の場合 then 部及び else 部）(op_body_A1 … op_body_Ak)，(op_body_B1 … op_body_Bj) をもつものとする．仮説候補 Clc: ⟨op_cond_A, op_cond_B⟩，Clb:⟨(op_ body_A1 … op_ body_Ak), (op_B1 … op_ body_Bj)⟩ を生成する．可能世界フレーム PWF_1 を生成し，現在着目している PWF のスロット値をすべてコピーした後，PWF_1 の「仮説候補スロット」から C0 を取り除き，Clc，Clb を加える．PWF_1 を着目中の PWF に下位ノードとしてリンクする（ただし，op_B_1 ～ op_B_n 中に反復構造（選択構造）が複数あった場合には，同様の処理を繰り返す）．
続いて，op_A が op_B_1 … op_B_n と対応しないと考えた場合の可能世界フレーム PWF_2 を生成する．現在着目している PWF のスロット値をすべてコピーした後，PWF_2 の「仮説候補スロット」から C0 を取り除く．PWF_2 を着目中の PWF に下位ノードとしてリンクする．着目点を PWF_1 に移す．(2) へ．

(2-2-2) 仮説候補の標準アルゴリズム側が選択・反復以外のプリミティブ操作である場合：このタイプの仮説候補は C0: ⟨op_A, (op_B_1 … op_B_n)⟩ と書ける．op_B_1 ～ op_B_n の中で，op_A とマッチング可能な操作が一つもなければ，この仮説候補を棄却して (2) へ．ただしここでマッチング可能とは以下の条件を満たすことである．
－操作の種類が同一である．
－双方ともに，着目中の PWF に蓄積された ACO で既に対応関係が確定している操作ではない．
－変数間の対応関係が着目中の PWF における「変数間対応仮説スロット」の内容と矛盾しない．
－標準アルゴリズムで指定された操作間の順序関係と，着目中の PWF に蓄積された ACO の内容が矛盾しない．
マッチング可能な操作があれば，以下の処理を行う．今，op_A と op_B_i (1≦i≦n) がマッチング可能であったとする．このとき，まずこの二つの操作が対応すると考えた場合の可能世界フレーム PWF_1 を生成し，現在着目している PWF のスロット値をすべてコピーした後，PWF_1 の「仮説候補スロット」から C0 を取り除く．「op_A と op_B_i が対応する」という ACO を生成し，PWF_1 の「操作間対応仮説スロット」に追加する．op_A と op_B_i が対応するときに生じる変数間の対応関係を，PWF_1 の「変数間対応仮説スロット」に追加する．PWF_1 を着目中の PWF に下位ノードとしてリンクする（ただし，op_B_1 ～ op_B_n 中に op_A とマッチング可能な操作が複数あった場合には，同様の処理を繰り返す）．
続いて，op_A が op_B_1 … op_B_n と対応しないと考えた場合の可能世界フレーム PWF_2 を生成する．現在着目している PWF のスロット値をすべてコピーした後，PWF_2 の「仮説候補スロット」から C0 を取り除く．PWF_2 を着目中の PWF に下位ノードとしてリンクする．着目点を PWF_1 に移す．(2) へ．

(3) 「操作間対応仮説スロット」を調べて，完全照合に成功しているか判定する．成功していれば比較終了処理 (4) へ．さもなくば仮説木の葉にあたる各 PWF を検査し，いずれかの PWF に (2) のステップで検査されていない仮説候補が残っていれば，その PWF に着目して (2) へ．もし仮説木中に未選択の仮説候補が一つもなければ比較終了処理 (4) へ．

(4) 完全照合に成功していればこれを出力結果とする．さもなくば，仮説木の葉ノード中で最も多くの ACO をもつ（すなわち，最も多くの対応関係を発見することに成功した）PWF を出力結果とする．

（平成 11 年 10 月 15 日受付，平成 12 年 1 月 8 日再受付）

小西 達裕 （正員）

1987 早大・理工・電子通信卒．1992 同大大学院博士後期課程了．1991 早大・理工・情報助手．1992 静岡大・工・情報知識工学科助手．現在，同大情報・情報科学科助教授．博士（工学）．知的教育システム，知的対話システムなどに興味をもつ．人工知能学会，情報処理学会，教育システム情報学会，日本認知科学会各会員．

伊東 幸宏 （正員）

1980 早大・理工・電子通信卒．1987 同大大学院博士課程了．同年，早大・理工・電子通信助手．1990 静岡大・工・情報知識工学科助教授．現在，同大情報・情報科学科教授．工博．自然言語処理，知的教育システムなどに興味をもつ，人工知能学会，情報処理学会，言語処理学会，教育システム情報学会，日本認知科学会各会員．

鈴木 浩之 （学生員）

1998 静岡大・工・情報知識卒，現在，同大大学院博士後期課程在学中．プログラミング教育支援システムの研究に従事．人工知能学会会員．