原著論文

教育システム情報学会誌 Vol. 14 No. 2 Jul. 1997

再帰プログラミングを対象とした問題の比較に基づく問題機能の実現

松田憲幸*, 柏原昭博*, 平嶋 宗*, 豊田順一*

Recursive Programming Exercises Based on Comparison between Problems

Noriyuki MATSUDA*, Akihiro KASHIHARA*,
Tsukasa HIRASHIMA*, Jun'ichi TOYODA*

It is very important task for a student to compare between problems in exercises. To help him/her to compare between problems in an educational system, it is necessary to define a problem representation and control the comparisons by using the representation. In this paper, we propose a representation for comparing between two problems in education of recursive programming. Second, we present a design of functions of comparing between problems which are (a) generating explanation of differences and similarities between two problems, (b) searching problems for comparisons, and (c) generating exercises of grouping problems. We have also implemented an educational system which helps a student to compare between two problems. We describe an experimental evaluation of the functions.

1. はじめに

問題解決の初心者の中には，経験した問題は解くことができても，同様の解法で解決できる類似した問題を解くことができない場合がよく見受けられる。この原因として，初心者が問題解決知識を個々の問題に依存して理解しており，同種の問題に共通する知識として理解できていないことが考えられる。このような初心者を対象とした問題解決の支援では，問題ごとに問題解決過程を説明するだけではなく，問題間における問題解決過程の類似点や相違点に注意を向けさせることが重要となる。

問題間の比較を促す教育は，実際の教育現場で頻繁に行われている方法であり，文献(1)においてある程度定式化されている。また，文献(2)及び文献(3)において，力学を対象とした，問題解決過程に基づく問題の特徴記述（以下インデックスと呼ぶ）の提案と，その利用手法について検討してる。本稿では，プログラミングを対象として，プログラムの振舞いを問題の特徴記述として利用することにより，問題間の比較を促す支援機能の設計・開発を行った。本研究では，プログラム言語については一とおりの学習を終えている学習者を対象として，プログラムの振舞いの学習とそれを用いた問題演習を行う教育支援システムの実現を目指している。本稿で設計・開発した演習支援機能は，このような教育支援システムにおいて必要となるものである。

プログラミング教育を対象とした，従来の教育支援

* 大阪大学産業科学研究所
The Institute of Scientific and Industrial Research, Osaka University

システムに関する研究では，主に個々の問題解決に対する支援が中心に取り扱われており，問題間の比較を促すことを目的とした問題演習支援機能の設計・開発手法を暗黙的に扱うことが多かった[4][5][6][7]。筆者らはすでに，初心者の再帰プログラミング教育を対象に，U-behaviorと呼ぶプログラムの振舞い表現と，この表現を利用したU-solutionと呼ぶプログラミング解法を提案した[8]。さらにプログラミング教科書との比較を行う実験によって，U-solutionの有効性を確認した[9]。

プログラミング教育では，少なくともプログラムの機能（プログラム仕様）と振舞いについての説明が不可欠となる。特に再帰プログラムは，字面上の命令の順序と振舞いが大きく異なり，複雑となる傾向がある。このため，再帰プログラミングは初心者にとって最も困難な課題の一つとなっている。本稿では，特に再帰プログラムの振舞いにおける理解・設計を扱う教育支援を取りあげる。

U-solutionは，与えられたプログラム仕様に対して，プログラムの振舞いを考えることによって，プログラムを作成する解法である。ここでは再帰呼び出しを1つしか含まない再帰プログラムを対象に，初心者が理解容易となる振舞いの表現を作成した。これをU-behaviorと呼ぶ。U-solutionでは，複数の再帰プログラムのU-behaviorに共通する構造とその構造からの差異を利用して，再帰プログラムを作成する。ここでは，この共通構造を「U-template」，再帰プログラムと共通構造との差異を表す処理を「振舞い部品」と呼ぶ。また，U-templateをプログラム言語で表現したものを「Program-template」と呼び，振舞い部品をプログラム言語で表現したものを「プログラム部品」と呼ぶ。本稿で対象とした範囲では，CおよびPrologにおいて，各U-templateおよび振舞い部品に対して，1対1にProgram-templateおよびプログラム部品を用意することができている。

U-solutionによる再帰プログラムの作成では，まず，プログラム仕様を満たすU-behaviorを作成するために，初心者がU-templateに適当な振舞い部品を代入する。次に全ての振舞い部品を代入して得られたU-behaviorにおいて，初心者が意図した通りの振舞いが得られるているかどうかを確認する。このとき意図したとおりの振舞いが得られるまで，振舞い部品の代入を繰り返す。最後に，Program-templateにプログラム部品を代入することによって，プログラムを作成する。このように，U-solutionによりプログラムを作成するには，U-behaviorにおける部品タイプ，振舞い部品の働きを十分に理解している必要がある。

本稿では，U-solutionの教育を目的とした問題演習を対象として，問題間の比較を促す支援機能について述べる。教育支援システムにおいて問題演習機能を実現するためには，各問題のインデックスが必要となる。本稿では，インデックスとしてU-behaviorを利用することにより，(1)問題間の差について自然言語およびアニメーションによる説明の生成，(2)比較を促す問題の検索，(3)プログラムの分類問題の提示機能を作成した。また，(1)については，開発したシステムによる教育と，従来の教科書を用いた教育との比較を行う実験を行った。その結果，本システムにより差の説明を行った教育が有望であることを示唆する結果が得られた。

以下，2.ではU-behaviorおよびU-solutionについて概略を示す。3.ではU-behaviorをもとに比較を促す問題演習を実現する手法について論じる。4.ではプロトタイプシステムを用いた予備実験について述べ，5.で本論文のまとめを行う。

2. 再帰プログラムの振舞い表現

本章ではまず，筆者らがこれまでに提案した，初心者にとって理解容易となるような再帰プログラムの手続き表現U-behavior，およびその表現を利用した再帰プログラミング解法U-solutionの概要について述べる。その詳細と評価は，文献(8)(9)において報告している。

2.1. U-behavior

U-behaviorでは，初心者向け教科書によく見られる再帰呼び出しを1つしか含まないような単純な構造のプログラムを対象に，再帰処理を2つの反復に対応させて表現する。図1にU-behaviorの概念図を，図2にPrologのappendプログラムのU-behavior表現を示す。1つ目の反復は，再帰関数における関数の

先頭から再帰呼び出しまでの間の処理の反復である（図1(a)）。appendでは，第1引数において入力リストの先頭要素を取り出す処理，第2引数では入力リストに対して操作を加えずに再帰する処理がこれに相当する（図2(a-1)(a-2)）。2つ目の反復は，再帰呼び出しの次の行から関数の最後の行までの間の処理の反復である（図1(d)）。appendでは，再帰により得られたリストの先頭に先程取り出した先頭要素を追加する処理がこれに相当する（図2(d)）。また再帰処理において必要となる変数のスタックへの保存／取り出しといった操作を，U-behaviorでは前処理から後処理へのデータの移動として表現する（図1(e)）。appendでは，第1引数で取り出した先頭要素の移動がこれに相当する（図2(e)）。

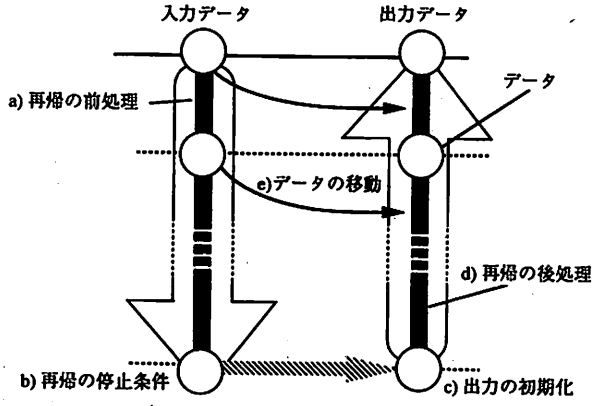


図1　U-behaviorの概念図

さらにU-behaviorでは，再帰処理の過程を次のような4種類の構成要素に分割している。1つ目の要素は1回目の反復において繰り返される処理で，これを「再帰の前処理」と呼ぶ（図1(a)）。2つ目の要素は，再帰関数を終了させるための入力データの条件で，これを「再帰の停止条件」と呼ぶ（図1(b)）。3つ目の要素は，この停止条件が成立した際に実行される出力データ（再帰関数の戻り値）の設定で，これを「出力の初期化」と呼ぶ（図1(c)）。4つ目の要素は，2回目の反復において繰り返される処理で，これを「再帰の後処理」と呼ぶ（図1(d)）。このような4種類の要素を「部品タイプ」と呼び，その具体的な処理単位を

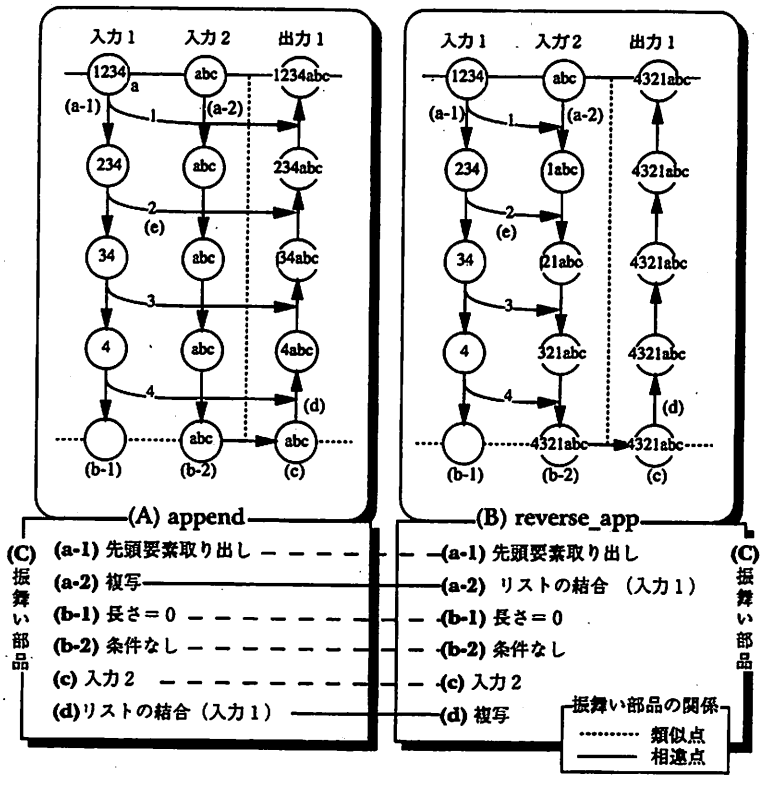


図2　appendとreverse_appのU-behaviorの類似点・相違点

表1 再帰の前処理と再帰の後処理における振舞い部品とプログラム部品

部品名称	記号	Prolog	C
部品なし	C_0, E_0	——	——
複写	C_1, E_1	*V1* = *V2*	*V1* = *V2* ;
先頭要素取り出し	C_2, E_2	*V1* = [*Head*|*V2*]	*head_list* (*Head* , *V2*); *tail_list* (*V1* , *V2*);
先頭に追加	C_3, E_3	*V2* = [$Head_m$|*V1*]	*compose* ($Head_m$, *V1* , *V2*);
Kを足す	C_4, E_4	*V1* =*V2* + *K*	*V1* =*V2* + *K* ;
入力$_m$を足す	C_5, E_5	*V1* =*V2* + $Input_m$	*V1* =*V2* + $Input_m$;
Kを引く	C_6, E_6	*V1* =*V2* - *K*	*V1* =*V2* - *K* ;
入力$_m$をかける	C_7, E_7	*V1* =*V2* * $Input_m$	*V1* =*V2* * $Input_m$;

V1, V2, Head, Inputは変数．添字のmは，第m引数を表す．Input は入力引数，Headは入力データの先頭要素を表す．

表2 再帰の停止条件における振舞い部品とプログラム部品

部品名称	記号	Prolog	C
部品なし	S_0	——	——
条件なし	S_1	——	——
長さ＝0	S_2	*Input* = []	null_list(*Input*)
長さ＝1	S_3	*Input* = [_]	one_element_list(*Input*)
数値*K*	S_4	*Input* = *K*	*Input* == *K*

Inputは変数．null_list(X)は，変数Xがnullリストかどうかを調べる関数．
one_element_list(X)は，変数Xが1要素からなるリストかどうかを調べる関数．

表3 出力の初期化における振舞い部品とプログラム部品

部品名称	記号	Prolog	C
部品なし	I_0	——	——
入力$_m$	I_1	Out = $Input_m$	Out = $Input_m$
入力$_m$の先頭要素	I_2	Out= $Head_m$	Out = $Head_m$
Constant : *A*	I_3	Out = *A*	Out = *A*

Out, Input, Head は変数．添字のmは第m引数を表す．
Inputは入力引数を，Headは先頭要素を表す．

「振舞い部品」と呼ぶ。append では，図2(C)に示すように，第1引数および第2引数においてそれぞれ再帰の前処理と再帰の停止条件の振舞い部品が，第3引数において出力の初期化と再帰の後処理の振舞い部品が存在するため，計6個の振舞い部品に整理できる。

このようにU-behavior は，再帰処理をU字型の図を用いて，2つの反復と4種類の振舞い部品で表現する。筆者らは，初心者のためのPrologとCのプログラミング教科書に用いられている単純な再帰プログラムを対象に，振舞い部品を整理した。整理した振舞い部品の一部を表1，2，3に示す。また対象となるPrologの再帰プログラムの一部を付録に示す。

2.2. U-solution

U-behavior を利用して，プログラム仕様からプログラムを作成するための解法をU-solution と呼んでいる（図3）。U-solution では，複数のU-behavior における共通部分を残して，相違部分を空欄としたU-templateと，空欄に代入する振舞い部品を利用して，U-behavior を作成する（図3手順1）。得られたU-behavior がプログラム仕様を満足することを確認するまで，振舞い部品の代入を繰り返す（図3手順2）。次に，U-behavior が得られたら，U-templateに代入した振舞い部品に対応するプログラム部品をProgram-templateに代入することによってプログラムを作成する（図3手順3）。このように，U-solutionでは，振舞い部品の選択が重要であり，そのた

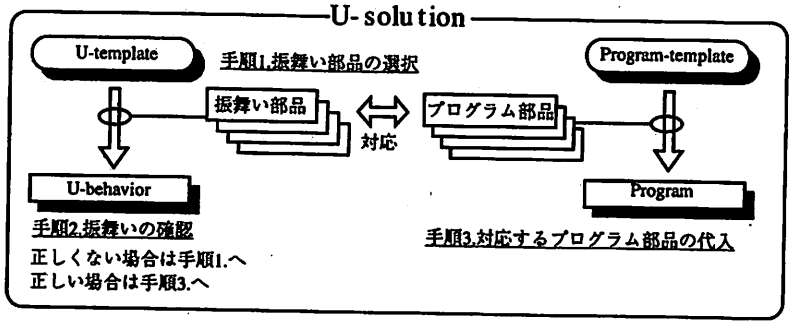


図3　U-solution

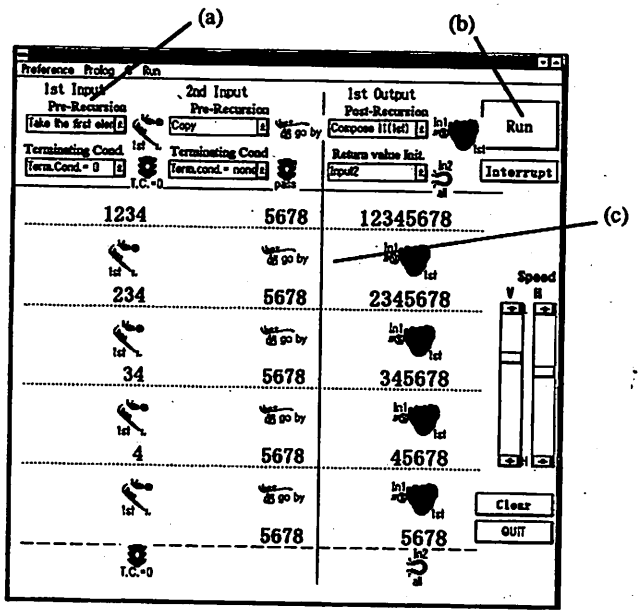


図4　append プログラムの U-behavior

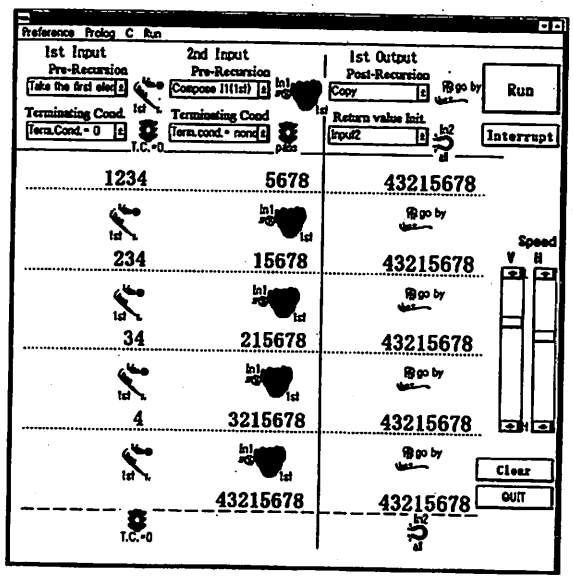


図5　reverse_app プログラム U-behavior

めには，部品タイプや振舞い部品の働きを理解する必要がある。

初心者プログラマの U-solution による再帰プログラミングを支援するツールを実装した（図4，5）。本ツールは，学習者がメニュー（図4(a)）から振舞い部品を選択して，実行ボタン（図4(b)）を押すと，選択された振舞い部品による U-behavior 上での振舞いのアニメーションを表示（図4(c)）する。

3．　比較を促す問題演習機能

U-solution の教育を目的とした問題演習において必要となる，比較を促す支援機能について述べる。本稿では，問題間の異同を U-behavior の振舞い部品を用いて表現することによって，支援機能を実現する。以下ではまず，振舞い部品による問題間の異同の表現方法について，次に実現した比較を促す支援機能について述べる。

3．1．　問題間の類似点・相違点の表現

問題の解となるプログラムの U-behavior を構成している振舞い部品を用いて，2つの問題間の類似点・相違点の表現とする。類似点は U-behavior の各部品タイプにおいて一致する振舞い部品とし，相違点は不一致となる振舞い部品とする。

図2 (A)(B) に，付録に示す Prolog の append と

reverse_appの類似点・相違点を振舞い部品によって表現する例を示す。appendは，第1引数（入力1）のリストと第2引数（入力2）のリストを結合したリストを第3引数（出力1）とするプログラムである。reverse_appは，第1引数（入力1）のリスト要素の順序を反転して，第2引数（入力2）のリストと結合し，結合したリストを第3引数（出力1）とするプログラムである。この2つのプログラムの振舞いの類似点は，再帰の前処理においてリストの長さが0になるまで先頭要素を取り出し，再帰が停止したら入力2で出力を初期化する点である。また相違点は，再帰の前処理において入力1から取り出した先頭要素を追加するタイミングである。すなわちappendでは再帰の後処理で追加するのに対し，reverse_appでは再帰の前処理において追加する点である。このようなU-behavior上の異同を振舞い部品を用いて表現する(図2(C))。append，reverse_appのU-behaviorにおける各部品タイプにおいて一致している振舞い部品（図2(C) 実線部分）を類似点の表現とし，不一致となる振舞い部品（図2(C)破線部分）を相違点の表現とする。

3．2．比較を促す問題演習支援機能

U-behaviorの振舞い部品によって表現された問題の類似点・相違点の表現をもとに，問題間の比較を促す問題演習支援機能を作成した。作成した支援機能は，(1) 問題間の差の説明，(2) 問題の検索，(3) 分類問題の作成機能である。以下にそれぞれの実現手法を述べる。

3．2．1　問題間の差の説明機能

問題間の比較を支援する最も基本となる機能が，問題間の差の説明機能であり，U-behaviorにおける異同を自然言語およびアニメーションによって説明する。説明文はプログラム間の類似点や相違点となる振舞い部品について自然言語で説明したものであり，プログラム仕様と対応づけて提示される。アニメーションは，それぞれのプログラムの振舞いをU-behavior表現の上でデータが処理される過程をアニメーション表現したものである。学習者にそれぞれのプログラムのアニメーションを提示して比較を促す。

まず，説明文の作成方法について述べる。ここでは，あらかじめ教師によって作成された，各問題のプログラム仕様の説明文と，各振舞い部品の働きに関する説明文を用いて，プログラムの説明文を作成する。作成には，各プログラムに共通の説明文のテンプレートを利用する。類似点の説明文は，類似点の説明文のためのテンプレートに対して，類似点を表す振舞い部品に設定されている説明文を代入することにより作成する。同様に，相違点の説明文は，相違点のテンプレートに，相違点を表す振舞い部品の説明文を代入することにより作成する。

例えば，Prologのappend，reverse_appにおいて，説明文を生成する場合について考える。図2(C)はこの2つのプログラムのU-behaviorとそこで使用される振舞い部品を示している。プログラムの振舞いについての説明文を生成するときは，振舞い部品にあらかじめ設定されている説明文を「再帰の前処理」，「再帰の停止条件」，「出力の初期化」，「再帰の後処理」の順に組み合わせることによって，プログラムの説明文を作成する。appendのU-behaviorの説明文は，次のようになる。「入力1は先頭要素の長さが0になるまで1つずつ取り出す。入力2は複写を繰り返す。再帰が停止したら，出力を入力2で初期化する。出力は先頭に入力1で取り出した要素を追加する。」下線部分が，各振舞い部品に設定されている説明文である。同様にreverse_appの振舞いの説明文は，「入力1は先頭要素の長さが0になるまで1つずつ取り出す。入力2は先頭に入力1で取り出した要素を追加する。再帰が停止したら，出力を入力2で初期化する。出力は複写を繰り返す。」となる。次に，この2つのプログラムの類似点と相違点についての説明文を作成する。2つのプログラムの間で，一致している振舞い部品を利用して類似点の説明文を，一致していない振舞い部品を用いて相違点の説明文を作成する。appendとreverse_appの間の類似点の説明文は次のようになる。「入力1の再帰の前処理において，先頭要素の長さが0になるまで1つずつ取り出す点が類似点である。出力の初期化は，出力を入力2で初期化する点が類似点である。」appendとreverse_appの間の相違点の説明文は次のようになる。「appendにおける相違点は，入力2の再帰の前処理において，複写を繰り返す点で

ある。また，再帰の後処理は，出力は先頭に入力1で取り出した要素を追加する点である。reverse_appにおける相違点は，入力2の再帰の前処理において，先頭に入力1で取り出した要素を追加する点である。また，再帰の後処理は，出力で複写を繰り返す点である。」

さらに，それぞれのプログラムのプログラム仕様について説明した文章を提示する。各問題のプログラム仕様についての文章は，あらかじめ教師により与えられている。appendのプログラム仕様は「入力1と入力2のリストを結合して出力3とする」であり，reverse_appのプログラム仕様は「入力1の要素の順序を反転して，入力2と結合して出力3とする」となる。このようなプログラム仕様についての説明文を，前述のU-behavior上の説明文と対応づけて学習者に提示することにより，プログラム仕様とU-behaviorの対応関係についての理解を促す。

次に，U-behaviorのアニメーションを利用した差の説明方法について述べる。U-behaviorはプログラムの振舞いを表す表現であるため，アニメーションを用いてデータの動きを表現することで，学習者にとって理解容易な表現となると考えられる。U-behaviorのアニメーション表示を利用したプログラム間の差の説明の具体例として，append，reverse_appのアニメーション画面を図4，5に示す。図4に示すように，appendの再帰の前処理である入力リストから1要素取り出す様子は，入力データが下方向に動く過程で表現される。すなわち，振舞い部品「先頭要素取り出し」を表すバットのアイコンにより，入力データが2つに分岐して，入力データの先頭要素が右方向に，残りのデータが下方向に移動する。appendでは，右方向に移動した先頭要素が，再帰の後処理における振舞い部品「先頭に追加」を表すグローブのアイコンによって，出力の先頭に追加される。一方，reverse_appでは，右方向に移動している先頭要素が，もう一つの入力引数の再帰の前処理に位置する振舞い部品「先頭に追加」のグローブアイコンにより，入力データの先頭に追加される。このように振舞い部品「先頭に追加」を表すグローブアイコンの位置の違いによって，取り出されたデータが追加されるタイミングの違いをアニメーションで表し，さらに，このような動作上の違いが，出力結果における要素順序の違いに表れることを表現する。

このようなU-behaivior に基づく問題間の差の説明が有効であるかどうかは，プログラムを比較するための表現としてのU-behaviorの適切さを示すものである。この説明機能における有効性の評価については，4.で述べる。他の支援機能については，問題演習全体を対象とした教育支援システムの完成後に行う予定である。

3.2.2 問題の検索機能

ここでは，比較を促す問題を検索する機能を作成した。比較において説明する内容によって2つの検索条件を設定した。

(A-1) 振舞い部品の説明

問題p_oとの比較を通して振舞い部品bの働きを説明することを目的に，問題p_o以外の問題p_sを学習者に提示する場合。この場合の問題p_sの検索条件を，「問題p_sは，説明する振舞い部品bを含み，かつ，問題p_sの振舞い部品が問題p_oの振舞い部品とできるだけ多く一致すること」とした。

(A-2) 部品タイプの説明

問題p_oとの比較を通して部品タイプtの説明を行うことを目的に，問題p_o以外の問題p_sを学習者に提示する場合。この場合の問題p_sの検索条件を，「問題p_sは，部品タイプtにおいて問題p_oと振舞い部品が異なり，かつ，部品タイプt以外の部品タイプにおいて問題p_sの振舞い部品が問題p_oの振舞い部品とできるだけ多く一致すること」とした。

さらに，検索対象とする問題によって，本機能が使われる状況を以下の2通りに分類できる。

(B-1) 例題の検索

学習者が過去に解いた問題を対象に問題検索する場合。この場合，検索された問題は，例題として学習者に提示する。

(B-2) 次の問題の検索

学習者がまだ解いていない問題を対象に問題検索する場合。この場合，検索された問題は，学習者が次に解くべき問題として提示する。

以上の検索条件をもとに，検索対象となる問題の中から，最もよく条件を満足する問題を全探索する。また，検索の結果，複数の問題が選ばれた場合には，競合を解消するための条件が必要となる。そこで，ここ

では不一致となっている振舞い部品について，学習者がその振舞い部品を過去に経験した回数を考慮して，よく経験した振舞い部品が多く含まれる問題を選択する条件を設定した。その手法として経験頻度を表す評価値を利用した。評価値は，表4に示すように，問題に属する各振舞い部品について，一致していれば最高値である1を，不一致の場合は経験回数により1未満0以上の値とした。問題の評価値は，その問題に属する全ての振舞い部品の評価値の合計値とした。競合している問題について評価値を求め，その中で最高値となる問題を検索結果として提示する。

問題検索の具体例を示す。例として，問題reverse_appとの比較を通して，出力の初期化に属する振舞い部品I1の説明を行うための問題を，過去に解いた問題から検索する場合について述べる。すなわち，本例は前述の(A-1)(B-1)の状況に相当する。なおここでは，学習者が表5に示す問題を解いてきたものとする。まず，学習者が解いた問題の中から，振舞い部品I_1を含まない問題を除くと，last，append，append_delが候補として残る。次に，これらの候補の中からreverse_appの振舞い部品と最も一致する振舞い部品が多いプログラムを検索する。この中では，appendが最も一致する振舞い部品が多いので，このプログラムを例題とし提示する。

表5 振舞い部品の経験回数による評価値

	一致	不一致				
		経験回数4回以上	3回	2回	1回	0回
評価値	1	0.8	0.6	0.4	0.2	0

3.3.3 分類問題の作成機能

学習者にU-behaviorについて，より深く理解させるためには，用意された問題のいろいろな組み合わせにおいて，U-behaviorの類似点や相違点を考えさせることが重要と考えられる。ここでは，一とおり問題演習を終えた問題について，プログラムの振舞いの観点で分類させる，分類問題を作成する。分類問題を用いた問題演習では，U-behaviorの上で複数の問題をあらかじめ分類した「分類図」を学習者に提示し，ここに新たなプログラムを追加させる。したがって分類問題を解くためには，学習者は追加するプログラムの振舞いと，既に分類されているプログラムとの振舞いを比較し，適切な位置を検討する必要がある。このような比較作業を学習者に行わせることで，振舞い部品の働き，部品タイプの働きなど，問題解決知識の理解を深めることが期待できる。

分類問題の作成では，学習者が過去に経験した問題の中からランダムに問題を1つ取り除き，残った問題において，問題の分類図を作成する。分類図の作成では，まず，4種類の部品タイプについて分類する順序を決める。このとき，学習者がどのような順序においても正しく分類できるように，ランダムに順序を決定する。次に，この順序にしたがって，最初の部品タイプについて共通する振舞い部品を持つ問題を同じグルー

表4 例題検索の例

問題系列	再帰の前処理 入力1	再帰の前処理 入力2	停止条件 入力1	停止条件 入力2	出力の初期化	再帰の後処理
select	C_2	C_1	S_6	S_1	I_2	E_3
last	C_2	C_0	S_3	S_0	I_1	E_1
append	C_2	C_1	S_2	S_1	I_1	E_3
append_del	C_2	C_1	S_3	S_1	I_1	E_3
suffix	C_2	C_1	S_5	S_1	I_0	E_0
現在の問題 reverse_app	C_2	C_3	S_2	S_1	I_1	E_1

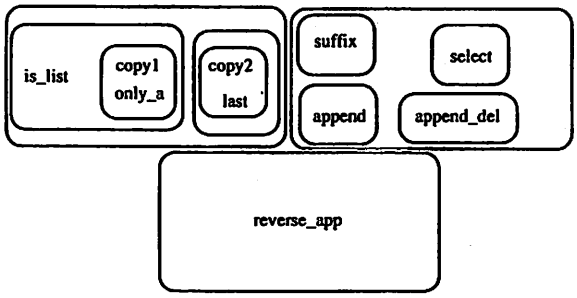


図6　プログラムの分類図の例

プとする。さらに各グループにおいて，次の部品タイプで分類したグループを作成する。このようにして作成した分類図の具体例を図6に示す。この分類図は，11個のプログラムを，「再帰の前処理」，「再帰の停止条件」，「出力の初期化」，「再帰の後処理」の順に分類したものである。分類問題では，１１個の中から任意の1つを取り除いておき，そのプログラムを学習者に代入させる。

4．　予備実験による評価

U-behaviorにもとづく問題間の差の説明の有用性を評価するために予備的な実験を行った。実験では，本研究で開発したシステムが生成する問題間の差に関する説明文を与えた場合と，市販されているプログラミング教科書における説明文を与える場合の比較を行った。利用した教科書は，個々の問題についての説明が行われており，問題間の差異については説明されていない。被験者は，実験前に行ったプログラミングテストにおいて，再帰プログラミングが行えないと判断できた，プログラミングの初心者19名（大学生，大学院生）である。

実験では，まず，被験者を2つのグループに分け，約45分間再帰プログラミングの教育を行った（これを教育フェーズと呼ぶ)。一方のグループ（12名）には，本システムが提示する問題間の差の説明を与えた(グループＡと呼ぶ)。他方のグループ（7名）には，市販されているプログラミング教科書を読んでもらった（グループＢと呼ぶ)。このとき両グループで説明する問題を図7に示す。グループAでは，問１，問３において個々の問題の説明を行い，問２，問４において，前問の問題（問１，問３）との差異についての説明文を与えた。両グループに与えた文章の量はほぼ同じになるように設定した。グループAの被験者が，システムで利用できる機能は，(1) 問題間の差の説明(問２，４のみ)，(2) 振舞い部品の説明文の提示，(3) 正解・不正解の提示，(4) U-behaviorのアニメーション表示である。U-behaviorを作成する際には，図4に示すU-behaviorの作成ツールを利用してもらい，作成ツールが出力するアニメーションにより意図したU-behaviorが得られているかどうかを確認してもらった。

教育フェーズの直後に，3問の問題（図8）によるテストを行った（これをテストフェーズと呼ぶ)。時間は1問10分とし，各問題についてアルゴリズムの図とＣプログラムを作成してもらった。このとき，グループAには作成ツールを利用してもらった。

テストフェーズにおける問題の採点では，U-behaviorの4つの部品タイプと再帰部分の計5カ所における正解個数を点数とした。したがって，U-behavior，Ｃプログラムそれぞれ満点は5点となる。採点の結果を表6に示す。

グループＡにおける平均得点は，アルゴリズムの作成，Ｃプログラムの作成においても，グループＢの平均得点を上回り，いづれも高い得点となっている(いづれも分散分析による1％有意)。この結果は，本システムにおける問題間の差の説明を用いたU-solutionの教育が，再帰プログラミングに対して有望であることを示唆している。

本実験では，さらに，実験終了後にグループAの被験者に対して，問題間の差についての説明がプログラミングに効果があったかどうかを問うアンケートを実施した。その結果，12中7人の被験者が問題間の差の説明がプログラミングに有効であったと述べ，4人はどちらとも言えないとし，1人は効果がないと答えた。どちらともいえないと判断した被験者のうち2名から，実験に使用したプロトタイプシステムにおけるウィンドウの配置等のインタフェースに問題があるとの指摘があった。この指摘を行った被験者はいずれも，この問題は，システム上の問題であり，問題間の差に関する情報はプログラミングに有益であると述べた。

問1．与えられたリスト（入力1）の末尾要素を削除して，残ったリストを取り出す再帰関数を作成せよ．
　入出力例：　入力1（1，2）→出力（1）
　　入力1（あ，い，う，え）→出力（あ，い，う）
問2．与えられたリスト（入力1）の末尾要素を要素zと置換する再帰関数を作成せよ．
　入出力例：　入力1（1，2）→出力1（1，z）
　　入力1（あ，い，う，え）→出力1（あ，い，う，z）
問3．与えられたリスト（入力1と入力2）において両方とも要素の順序をそのままにしてリスト（入力1）の後ろにリスト（入力2）をつないであらたなリスト（出力1）とする再帰関数を作成せよ．
　入出力例：　入力1（1，2，3，4），入力2（あ，い，う，え）
　　→出力1（1，2，3，4，あ，い，う，え）
　　入力1（1，2），入力2（あ，い，う，え）→出力1（1，2，あ，い，う，え）
問4．与えられた2つのリスト（入力1と入力2）において，リスト（入力1）の要素の並ぶ順序を反転させその後ろに要素の並ぶ順序をそのままにしたリスト（入力2）をつなげる再帰関数を作成せよ．
　入出力例：　入力1（1，2，3，4），入力2（あ，い，う，え）
　　→出力1（4，3，2，1，あ，い，う，え）
　　入力1（1，2），入力2（あ，い，う，え）→出力1（2，1，あ，い，う，え）

図7　教育フェーズにおける問題

問1．与えられたリスト（入力1）の最後に要素zを挿入する再帰関数を作成せよ．
　入出力例：　入力1（1，2）→出力（1，2，z）
　　入力1（あ，い，う，え）→出力（あ，い，う，え，z）
問2．与えられた2つのリスト（入力1と入力2）において，リスト（入力1）の末尾要素を取り除いて，残ったリストの順序を反転させたリストの後ろにリスト（入力2）をつなげる再帰関数を作成せよ．
　入出力例：　入力1（1，2，3，4），入力2（あ，い，う，え）
　　→出力1（3，2，1，あ，い，う，え）
　　入力1（1，2），入力2（あ，い，う，え）→出力1（1，あ，い，う，え）
問3．与えられた2つのリスト（入力1と入力2）において，リスト（入力1）の並ぶ順序をそのままにしたリストのうしろに，リスト（入力1）の順序を反転させたリストをつなげて，さらにその後ろにリスト（入力2）の順序をそのままにしたリストをつなげる再帰関数を作成せよ．
　入出力例：　入力1（1，2，3，4），入力2（あ，い，う，え）
　　→出力1（1，2，3，4，4，3，2，1，あ，い，う，え）
　　入力1（1，2），入力2（あ，い，う，え）
　　→出力1（1，2，2，1，あ，い，う，え）

図8　テストフェーズにおける問題

これらの事実は，再帰プログラミング教育において，本システムの説明が有用であることを示していると考えられる。

5．　むすび

本稿では，問題の特徴を表現するインデックスとして，U-behaviorを利用することにより，U-solutionの教育を目的とした問題演習における比較を促す支援機能を実現する手法について述べた。本稿で作成した支援機能は，(1)問題間の差の説明，(2)問題の検索，(3)分類問題の作成機能である。さらに，これらの支援機能を実装したプロトタイプシステムを利用した，問題間の差の説明支援機能の有望性を確認する予備実験について述べた。残りの支援機能についての評価，および，問題演習全体を対象とした教育支援システムの設計・開発が今後に残された課題である。

表6　実験結果

	人数	平均得点（標準偏差）	
		アルゴリズム	C言語
グループA	12	92 (0.6)	87 (3.3)
グループB	7	30 (18.4)	36 (9.3)

得点は満点が100

謝　辞

実験にご協力をいただいた徳島大学工学部知能情報工学科矢野研究室の諸氏，ならびに大阪大学産業科学研究所知能アーキテクチャ豊田研究室の諸氏に感謝致します。また，プログラミング言語についてご教授を頂いた九州工業大学情報工学部砥崎賢一助教授に感謝致します。本研究の一部は，文部省科研費一般研究(B)（課題番号 No.07458070）の援助による。

（1996年8月19日受付）

付　録

```
append([],A,A).
append([A | B],C,[A | D]):-append(B,C,D).
reverse_app([],A,A).
reverse_app([A | B],C,D):-reverse_app(B,[A|
  C],D).
copy1([],[]).
copy1([A | B],C):-copy1(B,C).
copy2([A],[A]).
copy2([A | B],C):-copy2(B,C).
only_a([],a).
only_a([A | B],C):-only_a(B,C).
is_list([]).
is_list([A | B]):-is_list(B).
last([A],A).
last([A | B],C):-last(B,C).
symmetry_app([],A,A).
symmetry_app([A | B],C,[A | D]):-symmetry_
  app(B,[A | C],D).
```

参　考　文　献

(1) Polya, G.：“いかにして問題を解くか”，丸善，1954

(2) Hirashima, T., Niitsu, T., Hirose, K., et al.：“An Indexing Framework for Adaptive Arrangement of Mechanics Problems for ITS”, IEICE TRANS. INF & SYST., VOL.E77-D, No.1. JANUARY 1994, pp19-26

(3) 平嶋宗，東　正造，柏原昭博，豊田順一：“補助問題の定式化”，人工知能学会誌，Vol.10, No.3, pp.83-90 (1995)

(4) Timothy, S., GEGG-HARRISON：“Adapting Instruction to the Student's Capabilities”, JI. of Artificial Intelligence in Education, pp.169-181 (1992)

(5) Ueno, H.：“Integrated Intelligent Programming Environment for Learning Programming”, IEICE TRANS. INF. & SYST., VOL.E77-D, NO.1 JANUARY 1994, pp.68-79

(6) Itoh, K., Itami, M., Fukukawa, K., et al.：“A Workbench System for Novice Prolog Programmers : Visually-Structured Interactive Tracer and Prototype-Based Programming Support”, IEICE TRANS. INF. & SYST., VOL.E77-D, No.1 JANUARY 1994, pp.57-67

(7) 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，D-II Vol.J78-D-II No.2, pp.321-332 (1995)

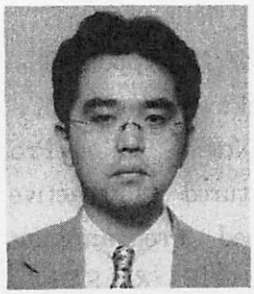
(8) Matsuda, N., Kashihara, A., Hirashima, T., at al.：“An Instructional System for Constructing Algorithms in Recursive Programming”, Proc. of the Sixth International Conference on Human-Computer Interaction, (HCI International '95), pp.889-894, Tokyo, Japan, (1995)

(9) 松田憲幸，柏原昭博，平嶋宗，豊田順一：” プログラムの振舞いに基づく再帰プログラミングの教育支援”，電子情報通信学会論文誌，D-Ⅱ Vol. J80-DⅡ No.1, pp.326-335 (1997)

著　者　略　歴

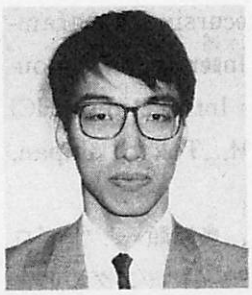
松田　憲幸

1992年関西大学工学部管理工学科卒業。1994年大阪大学大学院博士前期課程修了。1997年大阪大学大学院博士後期課程修了。同年，大阪大学産業科学研究所助手。工学博士。学習支援システム，情報フィルタリングの研究に従事。人工知能学会，電子情報通信学会，情報処理学会，教育システム情報学会各会員。



柏原 昭博

1987年徳島大学工学部情報工学科卒業。1989年同大学院修了。1992年大阪大学大学院博士課程修了。同年，大阪大学産業科学研究所助手。1996年～1997年，ドイツの GMD-FIT 客員研究員，工学博士。説明，認知孵化，知識外化の研究に従事。1993年度人工知能学会全国大会優秀論文賞受賞。Outstanding Paper Award at ED-MEDIA '95 受賞。人工知能学会，電子情報通信学会，情報処理学会各会員。



平嶋 宗

1986年大阪大学工学部応用物理学科卒業。1991年同大学院博士課程修了。同年，大阪大学産業科学研究所助手，1996年講師，工学博士，人間を系に含んだ計算機システムの高度化に興味を持っており，特に知的学習支援システムおよび情報フィルタリングの研究に従事している。1993年度人工知能学会全国大会優秀論文賞，ED-MEDIA'95 優秀論文賞，1996年度人工知能学会研究奨励賞，人工知能学会，電子情報通信学会，情報処理学会，教育工学会，教育システム情報学会，AACE，The International AI-ED Society 各会員。

豊田 順一

1961年大阪大学工学部通信工学科卒業，1966年同大学大学院博士課程単位取得退学，同年，大阪大学基礎工学部助手。1969年助教授。1982年大阪大学産業科学研究所教授。工学博士。現在，概念の形成，Visual Fidelity に関する研究に従事。情報処理学会，電子情報通信学会，日本認知科学会各会員。