流れ図作成支援システムのための誤り診断機能の性能評価

岡田 信一郎[†] 冨士池 均[†,☆] 藤原 祥隆[†]

筆者らは，これまでにプログラミング演習の質的向上を目的としたプログラミング演習支援システムを提案してきた．この支援システムは，流れ図記法によるアルゴリズム記述を支援するための流れ図作成支援システムを有する．この流れ図作成支援システムは，流れ図の質的向上と教官の負担を軽減することを目的とした，誤りを指摘する機能を持つ．学習者の誤りは，流れ図が正しい表記法に従って書かれていない「表記の誤り」，無限ループや分岐の誤りなどの論理的な誤りおよび構造的プログラミングの規則に対する違反を含む「論理的な誤り」，表記の誤りも論理的な誤りもないが，与えられた課題に対する正しい解答が得られていない「不正解」の3つに分類される．流れ図作成支援システムはこの分類に従って階層的に流れ図の誤りを診断する．特に，不正解の診断には動的診断法を利用しているため，より柔軟な診断処理が行える．本稿では，誤り診断機能を備えた流れ図作成支援システムのプロトタイプを実装し，過去の演習授業で作成された流れ図を診断させ診断能力の評価を行った．その結果，誤りのほとんどを検出することができ，本システムの誤り診断機能が実用的であることが分かった．

Performance Evaluation of Error Diagnostic Function for Flowchart Tutoring System

Shin-ichirou Okada,[†] Hitoshi Fujiike[†,☆] and Yoshitaka Fujiwara[†]

We have proposed a computer aided exercise support system in order to achieve a qualitative improvement of a programming exercise. This system includes the flowchart tutoring system for supporting algorithm description with a flowchart. It is useful to point out a mistake in the flowchart written by a student both for improving the quality of the flowchart and for reducing burden of teacher. The system supports to diagnose flowchart's validity. Errors by a student are classified into three categories, syntax error, logic error and uncorrect algorithm. The flowchart tutoring system hierarchically diagnoses errors according to the categories. Furthermore, it can diagnoses flowchart flexibly with the dynamic diagnosis method for validity diagnosis. In this paper, the performance of the error diagnosis algorithms are assessed using their prototype and sample flowcharts made by students in the past programming exercise class. This assessment denotes that the prototype can detect almost errors in the samples. This shows that the algorithms are practical.

1. はじめに

情報教育において，個々の学習者が実際のプログラミング言語などを使用し，プログラムを作成，実行することによりプログラミング技能を習得するプログラミング演習は重要な位置を占める．しかし，実際の大学などの講義において，数人の教官が数十人規模の学習者を同時に指導することは教官にとって大きな負担であり，さらに進捗状況の異なる個々の学習者に対して適切な指導を行うことは困難である．

筆者らは，これまで情報教育におけるプログラミング演習授業をネットワークでつながれた複数の計算機によって支援するシステム（プログラミング演習支援システム）の開発を行ってきた[1)]．このシステムは少人数（2，3人）の教官が，同時に数十人の学習者を指導するプログラミング演習の支援を行うもので，教官の負担を軽減するとともに学習者には個々の進捗に合わせたより高度な指導を行うことを目的としたものである．筆者らは，プログラミング教育において最も重要なポイントは良いアルゴリズムによりプログラムを作成する能力を身につけることにあると考える．アルゴリズム構築は特定のプログラミング言語に依存することなく行え，直観的に理解できる表現方法によって

† 北見工業大学工学部
Faculty of Engineering, Kitami Institute of Technology
☆ 現在，日鉄日立システムエンジニアリング株式会社
Presently with Nittetsu Hitachi Systems Engineering, Inc.

行うことが望ましい．そこで，上記プログラミング演習支援システムはアルゴリズムの記述にフォーサイスらの「構造的プログラミングによる流れ図表記法」[2]を採用し，この流れ図表記法によるアルゴリズム記述支援を中心としたシステム構成となっている．

良い流れ図（アルゴリズム）の作成を支援するためには，流れ図作成のための使い勝手の良いインタフェースを提供すること，学習者の流れ図を診断し学習者に誤りを指摘すること，が重要である．そこで筆者らは，これらの要件を満たす流れ図作成支援システムを提案してきた[3),4)]．流れ図作成支援システムは前述のプログラミング演習支援システムの一部として機能するものである．これまでに，流れ図入力のインタフェースとして GUI による流れ図エディタ，階層化知識に基づく流れ図診断機能，そして GUI と診断機能の双方に適するよう黒板モデルに基づいて設計されたデータ構造とシステム構成が提案されている．

プログラミング演習における初心者のプログラムを診断するシステムとしては PROUST[5)]や ALPUS[6)]をはじめとするシステムが多数提案されている[7),8)]．これらのシステムは，課題ごとに設計された知識ベースを用いて学習者のソースコードを解析したり，あらかじめ用意された変形規則により正解と学習者解のソースコードの双方を統一的な表記へと変形したうえで比較をするといった診断を行う．これに対し，本システムは流れ図を実際に実行した際に主要な変数がどのような変化をするかに着目し，正解と学習者解を比較する動的診断法[3),4)]を用いている．そのため，流れ図の細部に予期せぬ差異が存在しても柔軟に比較が行える特徴を持つ．また，課題ごとに特殊な診断知識を構築する必要がなく，正解知識を学習者解と同じ流れ図の形式で与えることができる．したがって，正解知識の追加が容易であり，正解知識を作成する教官の負担も少ない．

今回，筆者らは，誤り診断機能を備えた流れ図作成支援システムのプロトタイプシステムを実装し，実際の演習授業で学習者が作成した流れ図の診断を試みた．その結果，ほとんどの誤りを検出することができ，本システムの流れ図診断機能は十分実用的であることが明らかになった．また，流れ図作成支援システムの実用化において改善すべき点として，学習者に変数の型を正しく記述させなければ正しく診断が行えないこと，課題の内容によっては診断に数十秒の時間を要すること，などを示した．

以下，2 章では流れ図作成支援システムの概要を，3 章では流れ図作成支援システムの流れ図診断性能の評価と考察について述べる．

2. 流れ図作成支援システム概要

流れ図作成支援システムは，学習者からの流れ図の入力を受け付けるユーザインタフェース機能（流れ図エディタ），入力された流れ図の妥当性を検証する階層化知識による流れ図診断機能を備える．本章では，流れ図作成支援システムおよびそのプロトタイプの概要について述べる．

2.1 流れ図エディタ

流れ図エディタは流れ図作成支援システムの中でユーザインタフェースを担当する部分であり，GUI により流れ図を視覚的に作成することを支援する．

流れ図作成支援システムが対象とする構造的プログラミングによる流れ図表記法において，流れ図は「段階的分解」と呼ばれる手法によりアルゴリズムを単純な関数（または手続き）へ分解された形で記述される（**図 1**）．このように記述された流れ図は，式の記述，関数の記述など複数の記述レベルにより成り立っており，それぞれの記述レベルで固有の処理が必要であると同時に，それぞれのレベルの処理は密接な関わりを持つ．このような処理には協調分散処理の導入が適しており，流れ図エディタでは黒板モデル[9)]による協調分散処理によりデータ管理が行われている．黒板モデルの導入により，流れ図エディタの内部では学習者の入力した流れ図は画面上での表示位置などを記録する「表示情報」，個々の箱の型，箱の中に記述される命令文を記録する「動作情報」，個々の関数の型，引数などを記録する「機能情報」，関数どうしの依存関係を記述する「構造情報」，の 4 つの階層を持つデータ構造により保存され，それぞれの階層を担当する知識源

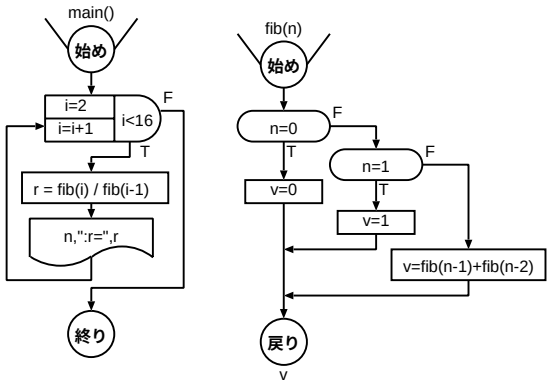

図 1 構造的プログラミングによる流れ図の例
Fig. 1 An example of flowchart by structured programming.

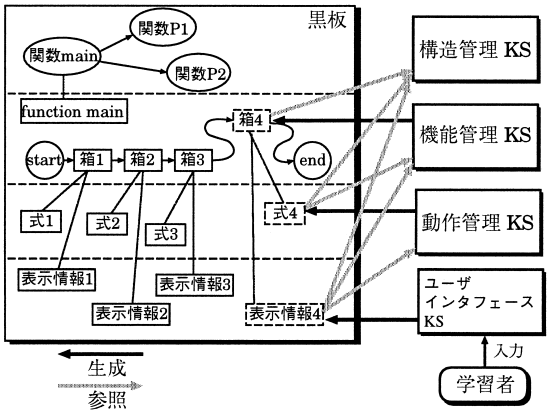

図 2 黒板モデルによる流れ図データの構造
Fig. 2 The data structure of flowchart with blackboard model.

(Knowledge Source，以下 KS),「ユーザインタフェース KS」,「動作管理 KS」,「機能管理 KS」,「構造管理 KS」の協調動作によって管理される．このような構造的プログラミングの持つ性質を反映したデータ構造による管理は，後に述べられる診断処理においても流れ図の解析が容易に行えるという利点もある．**図 2** に流れ図エディタにおけるデータ構造と黒板モデル各 KS との関係を示す．

2.2 階層化知識による流れ図診断法

本システムでは学習者が記述した流れ図から誤りを検出するにあたり，誤りを検出のためのルール（知識）の性質の違いから,「表記の誤り」,「論理的誤り」,「不正解」の 3 つに分類している．表記の誤りは流れ図の個々の命令が正しい文法で記述されていないことを示し，簡単な文法規則により検出することが可能である．これは一般のプログラミング言語におけるコンパイルエラーに相当する．論理的誤りは表記の誤りはないが，無限ループや構造的プログラミング規則に対する違反などを示し，流れ図の個々の命令を組み合わせた結果がどのような動作をするのかを理解できる診断知識が必要となる．そして，不正解は表記の誤りや，論理的な誤りはないが，課題の解釈の誤りや間違ったアルゴリズムの使用に起因し，アルゴリズムが課題に対する正解になっていないことを表す．この誤りを検出するためには課題ごとにその内容に適した知識が必要になる．

ここで論理的誤りの検出を考えると，対象となる流れ図には表記の誤りが存在しないことが必須の条件となる．なぜなら，たとえば無限ループの検出を考えるとき，対象となるループの終了条件式に表記の誤りがあれば，そのループが無限ループに陥るか否かを判断することは不可能なためである．同様に，対象となる流れ図に論理的誤りがある場合，それが不正解に該当するか否かを判断することはできない．

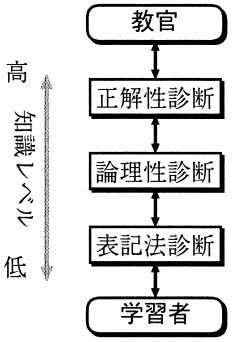

図 3 階層化知識による流れ図診断
Fig. 3 Flowchart diagnosis based on hierachical knowledge.

そこで，流れ図作成支援システムは診断処理を，表記の誤りに対応する「表記法診断」，論理的誤りに対応する「論理性診断」，そして不正解に対応する「正解性診断」の互いに独立した 3 つの診断処理に分解し，表記法診断，論理性診断，正解性診断の順番で診断を行う．これらの診断処理は表記法診断，論理性診断，正解性診断の順に診断に必要な知識が高度となり，診断知識のレベルに関して**図 3** のような階層構造となる．そのため，このような階層構造による流れ図の診断を階層化知識による流れ図診断と呼ぶ．

2.3 正解性診断

表記法診断，論理性診断に必要な知識は課題の内容に依存しないため，固定のルールとして実装することができる．しかし，正解性診断に必要な診断知識（正解知識）は課題ごとに異なるため，教官が各課題ごとに与える必要がある．

正解知識を特殊な形式で与えることは正解知識を作成する教官の負担を大きくする．そこで，流れ図作成支援システムでは，正解知識を流れ図の形式で与えることで，教官にとって正解知識を作成することが負担とならないようにした．以後，流れ図の形式で記述された正解知識を正解例と呼ぶ．

正解性診断はこの正解例と学習者の作成した流れ図（学習者解）の比較によって行われる．しかし，学習者解には同じ課題に対するものでも，様々な別解が存在しうる．それらの別解は，正解例と同じアルゴリズムでありながら，わずかな差異により別解となるものから，アルゴリズムのまったく異なるものまで種々多様である．

まず，わずかな差異による別解に対しては，より柔

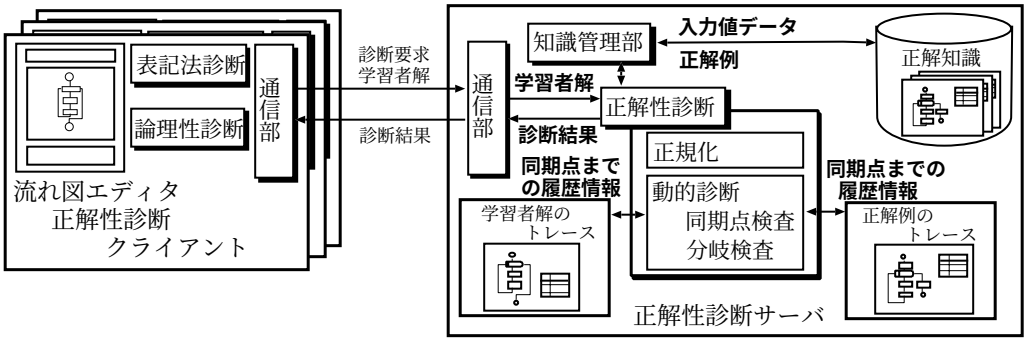

図 4 流れ図作成支援システムプロトタイプのシステム構成図
Fig. 4 Structure of the prototype flowchart tutoring system.

軟な流れ図の比較を行う必要がある．流れ図作成支援システムでは 2 つの流れ図を実際に実行し，主要な変数値の変化を比較する手法を用いた．本稿ではこの診断方法を動的診断法と呼ぶ．

さらにアルゴリズムの大きく異なる別解に対しては，正解例を複数用意することで対応する．学習者解は複数の正解例と比較され，1 つでも学習者解と同等と判断される正解例があれば，正解と診断される．教官があらかじめすべての別解を予想することは難しいが，本手法における正解例はそれぞれ独立した流れ図であるため，運用中に動的に正解知識を追加することも容易である．

ところで，正解性診断では正解例と学習者解が異なるとき，それが誤りに起因するものなのか，アルゴリズムの違いに起因するものなのかを判断することは難しい．そのため，表記法診断，論理性診断が学習者解の誤り検出を目的とするのとは異なり，正解性診断は学習者解が教官より与えられた正解例と同じ処理を行っていれば正解と診断することに重点を置く．

2.4 動的診断法

動的診断法は正解例と学習者解の両方の流れ図をインタプリタにより実際に実行し，あらかじめ 1 つ以上指定された主要な変数（着目変数）の値の変化を比較することで，学習者解の正当性を検証する診断法である．ここで着目変数は，計算結果を格納する変数や結果を配列へ格納するためのインデックスなど，その課題において特徴的な変化をするものが選択されるものとする．正解例と学習者解の実行は着目変数が変化する箱で一時停止され，双方の着目変数の値が比較される．この着目変数の値が変更される箱を同期点と呼ぶ．双方の流れ図の比較はこの同期点を基準に行われる．正解例と学習解双方の実行終了までの同期点の個数が同じで，それぞれ対応する同期点における着目変数の値がすべて同じであれば，このとき学習者解は正解と診断される．

学習者解が正解と診断されなかった場合，学習者解は誤りと診断され，誤り原因の推定が行われる．実行終了までの同期点の個数が異なる場合，学習者解の方が少ないときには処理が足りない，逆の場合には余分な処理があると仮定される．同期点における着目変数の値が異なる場合，同期点で使用された着目変数以外の変数の個数とその値が比較される．これを同期点検査という．同期点において誤り原因が推定できない場合には，実行時に作成された実行履歴を用いてプログラムスライシング[10),11)]におけるデータ依存関係にある箱を遡り，同期点で参照された変数の値の代入状況を比較する変数値検査，および制御依存関係にある箱を遡り，同期点に至るまでの条件分岐を比較する分岐検査により，誤り原因の推定を試みる．

2.5 流れ図作成支援システムプロトタイプの概要

前節までに述べられた流れ図作成支援システムの各機能を評価するために，流れ図作成支援システムのプロトタイプを設計・実装した．そのシステム構成図を**図 4** に示す．2.3 節で述べたように，流れ図作成支援システムは正解知識として教官の作成した正解例を用いており，それはつねに追加可能でなければならない．そのため，本プロトタイプは流れ図エディタと正解性診断サーバにより構成されるクライアント・サーバシステムとなっており，正解知識はサーバ計算機において一元管理される．

各学習者は，1 人 1 人に割り当てられたクライアントにおいて，流れ図エディタを用いて流れ図の作成を行う．表記法診断，論理性診断に必要な診断知識は固定的でよいためプロトタイプシステムでは流れ図エディタの一部として実装されている．

正解性診断の必要な流れ図は流れ図エディタの通信

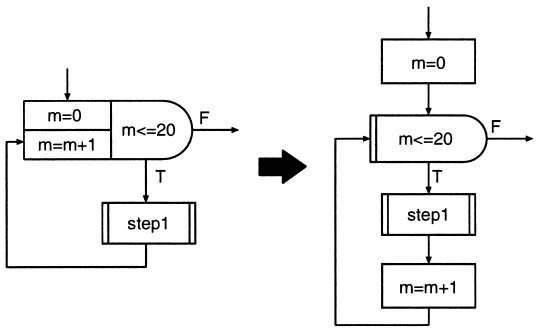

図 5 流れ図の正規化処理
Fig. 5 Normalization of flowchart.

部を通して，正解性診断サーバへ送られる．正解性診断サーバは流れ図エディタから学習者解を受け取り，まず，正規化と呼ばれる前処理を行う．これは簡単な変形規則により学習者解と正解例の表記を統一するためのもので，この処理をあらかじめ行っておくことで後の診断作業の負荷が軽減される．変形の例を**図 5** に示す．これは for ループを while ループに展開する例である．そして，正解性診断サーバは，知識管理部において管理されている正解知識から正解例を取り出し，2.4 節で述べられた手順により動的診断を行う．そのため，正解性診断サーバは流れ図を直接解釈・実行できるインタプリタ機能を備えている．着目変数は課題ごとに教官が決定し，正解例とともに登録することとした．学習者解の着目変数は教官が課題提示時に変数名を指定することとし，変数名の一致によって正解例との対応をとる仕様とした．なお，課題によっては，標準入力からの入力が求められる場合がある．その場合には，流れ図の実行時に入力動作をエミュレートする必要がある．そのため，プロトタイプシステムでは，入力動作で流れ図に入力するためのテストデータを正解例とともに正解知識へ登録できるようにしてある．診断が終了したら，その結果は流れ図エディタへ返され，流れ図エディタによって学習者に提示される．

本システムが支援対象として想定している本学情報システム工学科のプログラミング演習では，整数・実数計算にはじまり，入出力，条件分岐，ループ，配列・文字列処理，関数，再帰までの初級プログラミング技術の習得を目的としている．現在のプロトタイプシステムは関数の引数に配列を使用できないなど一部に制限があることを除き，上記項目のほとんどをサポートしている．

表 1 実験システム
Table 1 Exeperiment system.

	クライアント	サーバ
OS	IRIX 6.3	Linux 2.0.36
CPU	R5000 180 MHz	Pentium II 400 MHz
Memory	64 Mbyte	256 Mbyte

3. 診断性能評価

3.1 実験概要

流れ図診断機能の有効性と実用化の問題点を探るため，過去に行われたプログラミング演習授業で実際に学習者が作成した流れ図をプロトタイプシステムに診断させる実験を行った．

実験システムとしては，前章で述べたプロトタイプシステムを 1 クライアント・1 サーバの構成で用い，両者を結ぶネットワークには 10BASE-T を用いた．クライアント計算機およびサーバ計算機のそれぞれの仕様は**表 1** のとおりである．

診断対象となる流れ図は平成 10 年度に本学情報システム工学科の 2 年生を対象に行われたプログラミング演習授業で出題された 4 つの課題に対して学習者が実際に作成し，レポートとして提出したものを用いた．4 つの課題は，課題 1 はきわめて簡単な例，課題 2 は着目変数が複数ある例，課題 3 は正解例と異なるアルゴリズムによる正解が存在する例，課題 4 はループ回数が非常に多い例，であり，それぞれのケースにおける正解性診断の診断特性を示すために選択された．これらの課題は本システムが対象とするプログラミング演習の配列・文字列処理を除く内容の大半を含んでいる．

実験は以下の手順で行われた．まず，前述の演習においては，学習者に対して着目変数の指定をしていなかったため，課題ごとに主要な変数を着目変数と決め，学習者の作成した流れ図の対応する変数を着目変数となるよう修正を加えた．次に，この学習者解に対し表記法診断，論理性診断を行った．それぞれの診断で誤りが検出されたもののうち，明らかに単純なミスであり，修正がアルゴリズム全体に影響を及ぼさないと判断されるものについては，手作業で修正を施し，より上位の診断機能の評価に利用した．最後に，それぞれの課題に対して最も一般的な解を 1 つ正解例として与え正解性診断を行った．以上のことをすべての学習者解に対して行った．

実験結果は表記法診断，論理性診断，正解性診断のすべての結果を示した．表記の誤り，論理的誤りは一

般のプログラミング言語においてはコンパイルエラー，実行エラーなどで容易に知ることができるが，流れ図にはそれらに相当する手段がないため，一般には教官が手作業で評価するものである．実際，実験に用いられた流れ図も，表記の誤りや論理的誤りを多く含んだものである．これらの誤りをシステムが自動的に診断することにより，システム全体として教官の負担軽減にどれだけ貢献するかを示すため，これらの評価結果もすべて提示することとした．結果の表はそれぞれの診断で実験を行ったデータ数と診断結果を示している．表記法診断および論理性診断の診断結果はどれだけの誤りを正しく検出できたかで示した．分母は人間が判定した実際の誤りの個数，分子は検出できた誤りの個数である．正解性診断は学習者解が正解であるか否かを検出することが主な目的であるため，診断結果はどれだけの正解を検出できたかで示した．分母が人間が判定した実際の正解の数，分子が検出された正解の数である．また，表記法診断，論理性診断では誤りのない流れ図の数，正解性診断では不正解の数をそれぞれ括弧内に付記した．括弧内も人間の診断結果が分母，システムの診断結果が分子となっている．

3.2 課題 1：フィボナッチ数列

問題：

うさぎの繁殖モデルを表すフィボナッチ数列について，初期値として 0 月目の値を 0，1 月目の値を 1 とするとき，2 月目から 20 月目まで，各月のうさぎの総数の前月比 r（今月の総数/前月の総数）を計算し，出力するプログラムを作成せよ．またこの値が黄金比 1.618 に近づくことを確認せよ．

特徴：

・前月比 r を着目変数とする．
・アルゴリズムは比較的単純である．

結果：

診断結果は**表 2** のとおり．表記法診断では 53 件の誤りを検出できた．そのほとんどが，1 つの箱に対して複数の式を記述する際に必要となる式と式の区切り文字「,」の記述洩れのような単純な誤りであった．論理性診断では変数 SUM のつづりを誤って SUN と記述することによる未定義変数の使用や，その他の誤りを 4 件検出できた．課題 1 では表記法，論理性診断ではアルゴリズムに重大な影響を及ぼす誤りは存在しなかった．そのため，ここまでのすべての誤りに対して修正を行い，正解性診断を行った．

正解性診断では，正しい流れ図 36 件すべてに対して正答であることが診断でき，誤りのある流れ図に対しても 21 件すべてを誤りと診断することができた．

表 2 課題 1 診断結果
Table 2 Diagnosis results of the exercise 1.

総学習者データ数	57
表記法診断実験データ数	57
表記法診断で誤りがある（ない）と診断	53/53 (4/4)
論理性診断実験データ数	57
論理性診断で誤りがある（ない）と診断	4/4 (53/53)
正解性診断実験データ数	57
正解性診断で正解（不正解）と診断	36/36 (21/21)
正解性診断で診断失敗	0

誤りの内訳は，ループ回数にかかわる誤りが 8 件，前月比 r の計算にかかわる誤りが 6 件，アルゴリズムが大きく異なるものが 7 件であった．ループ回数の誤りについては 8 件ともループに誤りがあることを検出できたが，それがループカウンタの初期値にあるのか終了条件にあるのかまで特定することができなかった．前月比 r の計算にかかわる誤りでは，計算式の誤りと，計算に使用された変数の初期値の誤りを正しく区別することができず，正しく誤りの位置を判定できたものは 1 件であった．アルゴリズムが大きく異なるものは，誤りの範囲が大きく具体的な誤りの位置を示すことはできなかった．

3.3 課題 2：制御コードの出現回数

問題：

エスケープシーケンスを含む任意の文字列を入力するとき，この中に存在する空白，タブ，復改，の個数をそれぞれ数えるプログラムを作成せよ．ただし，文字列の終りを示す EOF の文字は \ とする．

特徴：

・空白，タブ，改行，それぞれの出現回数を着目変数とするため着目変数が複数ある．
・アルゴリズムは比較的単純である．
・入力値データを必要とする．

結果：

診断結果は**表 3** のとおり．表記法診断では課題 1 と同様な簡単な誤りを 16 件検出できた．論理性診断では，入力された文字が \ であることをループの終了条件としながらも，ループの内部で入力を行わないために無限ループとなる誤りなど 11 件の誤りを検出することができた．ここで，表記の誤りがある流れ図のうち 14 件，論理的誤りがある流れ図のうち 2 件が容易に修正可能であったため，それぞれ修正して論理性診断，正解性診断の診断対象へ加えた．

正解性診断では，正しい流れ図に対してすべて正答であることが診断でき，着目変数が複数ある場合でも診断可能なことが示せた．また正解性診断では 1 件の

表 3 課題 2 診断結果
Table 3 Diagnosis results of the exercise 2.

総学習者データ数	56
表記法診断実験データ数	56
表記法診断で誤りがある（ない）と診断	16/16 (40/40)
論理性診断実験データ数	54
論理性診断で誤りがある（ない）と診断	11/11 (43/43)
正解性診断実験データ数	45
正解性診断で正解（不正解）と診断	44/44 (1/1)
正解性診断で診断失敗	0

誤りを検出した．これは改行以外の文字を改行として数えた誤りであり，改行の数を表す変数が正常な値より大きくなったため，余分な処理を行っているものとして検出された．

3.4 課題 3：再帰的なフィボナッチ数列

問題：

フィボナッチの数列の第 n 項を再帰的に求める関数 $fib(n)$ を作成せよ．ただし，フィボナッチの数列は第 0 項から始まるものとする．またこの関数を用いて，n の値が 2 から 15 まで，前の項との比

$$r = fib(n)/fib(n-1)$$

を求め出力するプログラムを作り，r の値が黄金比 1.618 に収束することを確認せよ．

特徴：

・アルゴリズムに関数の再帰を用いる．
・着目変数は 1 つで前月比 r である．

結果：

診断結果は**表 4** のとおり．表記法診断では条件式で記述の誤りを犯す誤りなど 9 件検出できた．論理性診断では関数の戻り値に値が代入されていない誤りなど 16 件が検出できた．しかし，再帰の終了条件が正しく記述されておらず無限に再帰を繰り返す誤り 1 件を検出することができなかった．表記の誤りのすべてと，論理的誤りのうち 8 件が容易に修正可能であったため，修正の後それぞれ論理性診断，正解性診断の診断対象へ加えた．

正解性診断では，単純に再帰を行う解法と，アルゴリズムとしては同じだが再帰の方法に工夫をし再帰回数の少ない解法の 2 つの解答があったが，学習者のアルゴリズム表現は違っていても結果である比の変化は同一であるため，2 つの解答に対して 1 つの正解例で診断することができた．

一方，学習者が変数の型を不正確に記述したため，数値計算で正しい結果が得られず不正解と診断される学習者解が多数存在した．このとき，誤りの場所は特定できたが，変数の型に誤りがあることは検出できなかった．そこで，学習者解の変数の型や関数の戻り値に厳密な型の定義を行い，もう 1 度診断を行った．その場合，正しい流れ図をすべて正答であると診断できた．残り 5 件の誤りは，関数の戻り値に誤った値を与えているものが 3 件，関数 fib() のアルゴリズムが大きく異なるものが 2 件であった．このうち，戻り値に誤った値を与えているもの 1 件を関数 fib() に誤りがあると診断した以外は，正確な誤り位置の特定をすることができなかった．

表 4 課題 3 診断結果
Table 4 Diagnosis results of the exercise 3.

総学習者データ数	52
表記法診断実験データ数	52
表記法診断で誤りがある（ない）と診断	9/9 (43/43)
論理性診断実験データ数	52
論理性診断で誤りがある（ない）と診断	16/17 (36/35)
正解性診断実験データ数	43
正解性診断で正解（不正解）と診断	6/38 (37/5)
型を指定した場合の	
正解性診断で正解（不正解）と診断	38/38 (5/5)
正解性診断で診断失敗	0

3.5 課題 4：完全数

問題：

500 までの整数を調べ，完全数だけを出力するためのアルゴリズムを作成せよ．完全数とは，その数がそのすべての約数の和に等しい数のことをいう．最小の完全数は 6 $(= 1 + 2 + 3)$ である（1 は完全数からは除かれる）．

特徴：

・アルゴリズムとして 2 重ループを用いており，実行回数が多い．
・着目変数は 1 つで完全数の値である．

結果：

診断結果は**表 5** のとおり．この課題のときのみ実際の演習現場において，流れ図の作成時に表記法診断が運用されていた．そのため，表記の誤りは 2 件と少ない．論理性診断では無限ループや，値が代入されていない変数の参照やその他の誤りが 29 件検出された．しかし，ここで 1 件の無限ループの検出に失敗している．その無限ループは，ループカウンタとなる変数へループの内部で加算，減算を交互に繰り返した結果，ループの内部で値が変化せず無限ループに陥るものであった．表記の誤りの 2 件，論理的誤りのうち 11 件は容易に修正可能であったため，修正の後それぞれ論理性診断，正解性診断の実験データへ加えた．

正解性診断では最初，型の定義があいまいなため正答を 3 件しか検出できなかった．そこで，変数の型や関数の戻り値に厳密な型の定義を行ったのち再度診断

表 5 課題 4 診断結果
Table 5 Diagnosis results of the exercise 4.

総学習者データ数	42
表記法診断実験データ数	42
表記法診断で誤りがある（ない）と診断	2/2 (40/40)
論理性診断実験データ数	42
論理性診断で誤りがある（ない）と診断	29/30 (13/12)
正解性診断実験データ数	23
正解性診断で正解（不正解）と診断	3/11 (14/12)
型を指定した場合の	
正解性診断で正解（不正解）と診断	5/11 (12/12)
正解性診断で診断失敗	6

を行った．その場合，正しい流れ図 5 件が正答であることが診断できた．この課題はアルゴリズムに 2 重ループが用いられており実行回数が多い．現在のプロトタイプシステムでは流れ図の履歴の保存をトレースしたすべての実行文に対して行っているため，大量の履歴情報の保存が行われサーバ計算機のメモリ不足が発生した．その結果，サーバ計算機のメモリ不足のため，6 例の正解の診断に失敗した．

その他の不正解の内訳は，ループ回数の誤りが 2 件，約数の計算誤りが 7 件，不要な入力処理があるものが 3 件であった．このうち，誤り位置を正確に特定できたものは約数計算の誤り 1 件であった．誤り位置を正確に指摘できなかった誤りに，約数の判定処理が欠落したものがある．この場合，誤っている位置は存在していない部分であるため，誤り位置特定に失敗している．

3.6 診断コスト

正解性診断の診断コストを評価するため，それぞれの課題に対して，1 回の診断にかかるステップ数（1 つの箱の実行を 1 ステップとする）と実時間による実行時間を計測した．誤りのある流れ図の診断は，誤りが検出された時点で実行が中断されるため，実行ステップ数にばらつきが大きく現れてしまう．そのため，計測は正解と診断された学習者解についてのみ行った．

実行時間の結果を**表 6** に示す．これより，課題の内容によっては実行ステップ数が大きくなり，それにともない実行時間も大幅に増加することが分かる．特に課題 4 は，回答に 2 重ループを含むため，ステップ数が他の課題よりも多く，平均実行時間も 60 秒を超えている．これは現在のプロトタイプシステムの流れ図実行の効率の悪さに起因するものであり，その原因は 2 つ考えられる．まず，流れ図の実行において箱の中の命令文をそのつど解析していることがあげられる．そのため，ループにより同じ箱を何度も実行する場合，同じ解析処理が何度も行われており，実行速度低下の

表 6 実行ステップ数と実行時間
Table 6 The number of execution steps and run time.

	正答数	平均ステップ数	平均実行時間 [sec]
課題 1	36	142	0.073
課題 2	44	1147	0.120
課題 3	38	33677	4.281
課題 4	5	432598	65.480

原因となると考えられる．次に，誤りが検出されたときに誤り原因を推定するためにすべての変数値の変更履歴を保存していることが考えられる．このことは診断処理が大量のメモリを必要とする原因にもなっており，課題 4 の実行においては，200 Mbyte 以上のメモリ使用が確認されている．

3.7 考 察

本実験において，表記法診断，論理性診断ではほとんどの誤りを検出し，正解性診断では正しい学習者解の大半を正答と診断することができた．多数の学習者解すべてに対して誤りを指摘し，正答であるか否かを検査することは教官にとって大きな負担となる．本システムの各診断機能はこの負担を大幅に軽減することができると考えられる．

なお，課題 3，4 において，それぞれ 1 件ずつ論理性診断で無限ループ（無限再帰）の検出に失敗した．これは，複雑な処理を経たうえで無限ループに陥ったもので，ループ内でカウンタ変数が書き換えられているか否かを静的に検査する簡単な検出規則で実装された現在のプロトタイプでは検出しきれなかったものである．実際に流れ図を実行し値の変化をトレースし，無限ループ（無限再帰）の可能性を検証する必要があると考えられる．

さらに課題 3 の結果から，動的診断法は使用しているアルゴリズムに少々の違いがあっても，着目変数の値の変化が同じであれば，正解性を正しく診断できることが示された．これにより，動的診断法を適用することにより，より少ない診断知識で流れ図の診断を行えることが分かる．特に，今回の実験では，結果的に動的診断法によってほとんどの学習者解がただ 1 つの正解例によって診断可能であった．その理由として，実験の対象とした本学情報システム工学科のプログラミング演習が講義の後に続けて関連する内容について演習を行うスタイルをとるため，学習者の作成する流れ図は極端にアルゴリズムの異なるものにならないことがあげられる．このように，学習者解に類似性が期待できる状況では，動的診断法による正解性診断は有効であると考えられる．

一方，今回の実験で次のような問題点が発見された．動的診断法で流れ図をトレースするとき，変数の型が厳密に定義されていない場合，診断が正しく行われないことが分かった．しかし，アルゴリズム記述の段階で学習者に変数の型の定義まで考慮して作成させるのは困難である．そのため，データ型のあいまいさに対して有効な手法を検討する必要がある．

また，プロトタイプシステムによる誤り原因の推定は正確な位置を示さない場合が多かった．これは同期点検査，変数値検査，分岐検査が着目変数以外の変数の比較を行うため，正解例と学習解の着目変数以外の変数の対応が正しく推定されたときのみ正しい誤り原因が特定できることに起因すると考えられる．別解を与えることで正しい誤り原因を推定できる場合もあると考えられるが，複数ある正解例の中から誤り原因推定の基準となる正解例を決定する方法は，現在のところ着目変数の同期回数しか検討されておらず，同期回数の同じ正解例が複数存在する場合，基準となる正解例の選択を誤ることがある．今後，より厳密な基準正解例選択の方法を検討する必要がある．

診断にかかるコストに関しては，課題1〜3では実用上問題のない実行時間で診断結果が得られている．しかし，課題4のように実行回数が多くなる課題では，実行時間が実用上問題になるほど長くなる．また，そのような場合，消費するメモリ量も大きく，メモリの少ない計算機では診断不能に陥ることもある．これらの原因として，命令を実行するたびに構文解析処理を行っていることや，実行の履歴をすべて保存していることがあげられる．1度構文解析を行った命令文を中間コードに置き換えたり，不要になった実行履歴を削除するなど，実行効率やメモリ効率の向上を図る必要がある．

今回の実験において，着目変数は人間が課題に合わせて設定したが，実際の演習授業において，課題ごとに動的診断が効果的に働く着目変数を設定することは教官にとって負担となる．また，着目変数の存在を意識しながら問題を解くことは学習者にとっても負担となる．そのため，正解例，学習者解の双方から自動的に着目変数を抽出する手法を確立する必要がある．

4. むすび

本稿では，プロトタイプシステムと実際に学習者が作成した流れ図を用いた実験により，流れ図作成支援システムの診断機能が学習者に対するきめ細かい指導と教官に対して診断の負担の軽減という目的をほぼ達成していることを示した．また，実際の演習で運用するために，診断の正確性の向上，実行速度の向上，メモリ消費の低減，といった改良を行う必要があることが明らかになった．

今後は，上記の改良を行うとともに，演習で用いられるすべての機能の実装，および教官と診断サーバとの連携を容易にするための各種ツールの作成を行い実用システムとして完成させる予定である．

謝辞 本研究に関して流れ図作成支援システムの作成にご協力いただいた本学情報工学科第3期卒業生の田原迫将人氏，同第4期卒業生の石本恵氏，藤原一樹氏に感謝いたします．

参考文献

1) 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（D-II），Vol.J78-D-II, No.11, pp.1701–1709 (1995).
2) フォーサイス, A.I., キーナン, T.A., オーガニック, E.I., ステンバーグ, W.（著），浦 昭二（訳）：改訂コンピュータサイエンス入門 1，培風館 (1978).
3) 岡田信一郎，藤原祥隆，松西年春，大鎌 広：Program Slicing 技術を利用した誤り診断機能をもつ流れ図作成支援システム，電子情報通信学会論文誌（D-II），Vol.J78-D-II, No.11, pp.1669–1679 (1995).
4) 冨士池均，岡田信一郎，藤原祥隆：流れ図作成支援システムにおける誤り診断機能の評価，電子情報通信学会技術研究報告，Vol.98, No.563, pp.31–38 (1999).
5) Johnson, W.L. and Soloway, E.: PROUST: Knowledge-Based Program Understanding, *ICSE 84*, pp.369–380 (1984).
6) Ueno, H.: A Program Normalization to Improve Flexibility of Knowledge-Based Program Understander, *IEICE Trans. Info.*, Vol.E18-D, No.12 (1998).
7) 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 (1997).
8) 伊藤良二，小西達裕，伊東幸宏：プログラムの問題領域上での動作説明を行うプログラミング学習支援システムの構築，人工知能学会誌，Vol.15, No.2 (2000).
9) Fennell, R.D. and Lesser, V.R.: Parallelism in Artificial Intelligence Problem Solving: A Case Study of Hearsay II, *IEEE Trans. Comput.*, Vol.C-26, No.2, pp.98–111 (1977).
10) Weiser, M.: Programmers Use Slices When Debugging, *Comm.ACM*, Vol.25, No.7, pp.446–452 (1982).

11) 下村隆夫：プログラムスライシング技術と応用，共立出版 (1995).

(平成 12 年 4 月 5 日受付)
(平成 13 年 1 月 11 日採録)

岡田信一郎（正会員）

昭和 44 年生．平成 4 年北見工業大学工学部電子工学科卒業．平成 6 年同大学大学院工学研究科電気電子工学専攻修士課程修了．現在，同大学工学部情報システム工学科助手．知的教育支援システム．衛星通信を利用した知識ベース遠隔利用の研究に従事．電子情報通信学会会員．

冨士池　均

昭和 50 年生．平成 9 年北見工業大学工学部情報工学科卒業．平成 11 年同大学大学院工学研究科情報システム工学専攻博士前期課程修了．同年日鉄日立システムエンジニアリング（株）入社．在学中は知的教育支援システムの研究に従事．電子情報通信学会会員．

藤原　祥隆（正会員）

昭和 18 年生．昭和 43 年北海道大学大学院工学研究科電子工学専攻修士課程修了．同年電々公社（現 NTT）入社．平成元年より北見工業大学工学部教授．工学博士．エキスパートシステム，知的 CAI，衛星通信を利用した知識ベース利用環境等，知識を利用した情報処理全般に興味を持つ．IEEE，電子情報通信学会，人工知能学会各会員．