コンピュータと教育 13－7
（1990. 10. 22）

プログラムのアルゴリズム診断を中心としたＩＴＳの研究（5）

安田恭一郎　　岡本敏雄
トッパン・ムーア システムズ（株）　　東京学芸大学

本研究ではＣ言語を学ぶ初級プログラマの学習支援を行なうためのＩＴＳの枠組みを提案する．本システムでは，アルゴリズムの理解レベルでのプログラム理解を目的として，木構造グラフ表現されたプログラム・プランおよびプロダクション・ルールで記述されたプログラミング知識と，課題アルゴリズム知識を用いて学習者のバグの同定を行い，その背景にある誤概念の同定を行う．また，開発したシステムを利用することによって学習者の理解がどのように改善されるかを分析する．

The study of ITS to diagnose the algorithm of the novice's program(5)

Kyoichiro Yasuda　　Toshio Okamoto
Toppan Moore Systems Ltd.　　Tokyo Gakugei University

This paper describes a framework of ITS(Intelligent Tutoring System) supporting novices to learn C programming. To understand "computer program" requires the complicated skills which include programming activities such as debugging, modification and documentation. In order to understand "computer program", we can classify the following categories. These are skills to abstract the essential algorithm from a given task, and to represent it's algorithm by a certain computer language.

This system attempts to find out the misconception that might cause the bug, using knowledge on "programming skill" and algorithm of a specific programming task. Then, it presents the student with remedial instruction.

1　はじめに

コンピュータ支援によるプログラミング教授やデバッグの自動化は，初級プログラマのプログラミングにおける認知過程のコンピュータ上でのモデル化や，高度なプログラム理解を含む困難な課題である．自然言語によって与えられた課題プログラムの仕様の理解から，プログラミング言語を用いた実際のコーディングまでのプログラム開発過程の認知的過程の研究がこれまで多くなされ，初級プログラマと経験を積んだプログラマのプログラミングにおける差異も明らかとされている[1]．

プログラムのアルゴリズムの抽象的表現の方法については、RuthらのPGM[2]、Adamらのデバッグ・システムLAURAにおけるプログラム・プラン[3]、上野らによる手続きグラフ[4]、SolowayらのPROUSTにおける戦略グラフ[5]などの多くの手法が提案されている。

一方、バグとその背景に存在する誤概念との関連の表現を目的としたシステムには、SolowayらのMENO-Ⅱ[6]がある。MENO-Ⅱでは既知の誤概念がネットワーク形式で組織化され、バグと直接関連付けられている。

2　研究目的

本研究の目的は，アルゴリズム・レベルのプログラム理解に焦点を当てたプログラムの分析を行い、学習者のバグを同定すると共に、その背景に存在する誤概念を指摘するＩＴＳの枠組みを提案することである．

このために次の個別の目標を掲げる．

（１）プログラムを構成する個々のテクニックの記述のため、エキスパート・プログラマと初級プログラマのプログラミング知識に対する認知的構造を抽出、適切に知識表現する．

（２）プログラミングの目標知識となる課題アルゴリズムをコーディング・テクニックの系列として抽象的に表現する方法を考案する．

（３）プログラム知識の観点からのボトム・アップのアルゴリズム診断と、課題アルゴリズムをテンプレートとしたトップ・ダウンの診断を巧みに組み合わせ、効率的なバグの同定メカニズムを実現する．

（４）プログラマの犯しやすいバグを体系的に分類し、その背景にある誤概念との関連を分析し、適切に知識表現する．

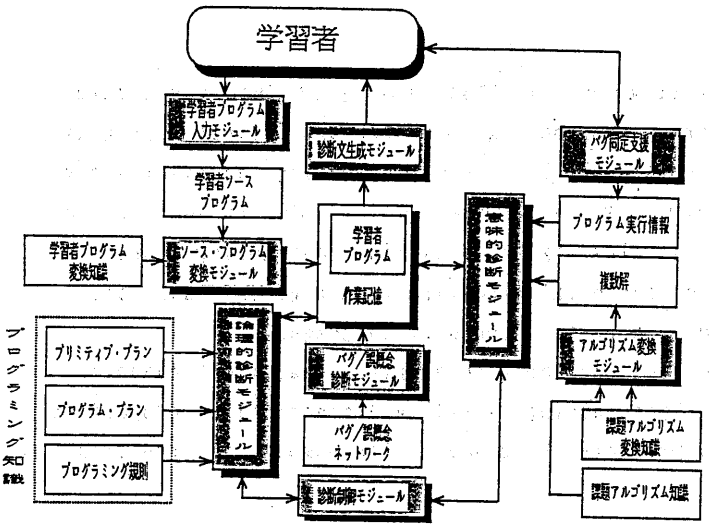

図１　システムの構成

3．システムの概要

本システムでは，学習者はシステムの支援を受けながらＣ言語のソース・プログラムを作成する．プログラミング課題は一次元配列データに対する逐次ソート程度のものである．システムは完成した学習者プログラムをネットワーク形式の内部表現に変換し，これに対して様々な診断を行い，診断結果を学習者に報告する．このような機能を実現するため，図１に示すような処理モジュール，知識ベースを用意した．

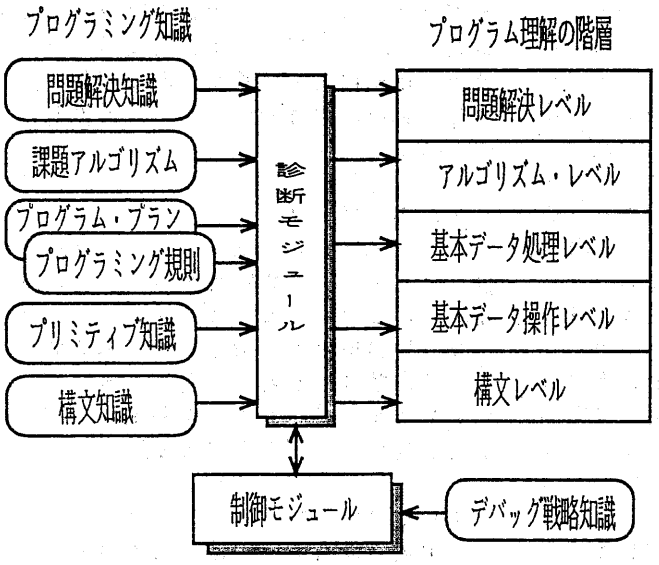

図2　プログラム理解の階層とプログラミング知識

4　プログラム理解

プログラム理解は、プログラムのデバッグや修正、文書化標準の設定といった作業に共通に包含される基本的なタスクである．

プログラム理解はその理解の対象のレベルによって、次のような階層的構造を持っていると考えられる[7]．

（1）問題解決レベル

プログラムが何を行なうかを理解することを目的とする．すなわち、プログラム理解システムにプログラムを与えたときに、「このプログラムは与えられた配列の要素を、逐次ソート・アルゴリズムによって昇順に並べ替えるプログラムである．」と答えることができたならば、このレベルでの理解が達成されたといえる．

問題解決レベルでのプログラム理解においては、プログラミングに関する知識のみではなく、問題領域に関する知識が必要となる．

（2）アルゴリズム・レベル

プログラムがどのように機能を実現しているかを理解するレベルである．すなわち、プログラムが実現しているアルゴリズムを抽出できたときにこのレベルでのプログラム理解が達成されたということができる．

（3）基本的データ処理レベル

アルゴリズムを構成する部品となる基本的な処理ルーチンの存在や正しさを理解するレベルである．各種のループや2変数値の交換のような典型的なデータ処理ルーチンが対象となる．

（4）基本的データ操作レベル

基本的データ処理レベルで理解の対象となった処理ルーチンを組み立てる基礎となる最も基本的な、命令レベルでの機能を理解するレベルである．データの代入、読み込み、書き出し、四則演算などが対象となる．

さらに、これ以下のレベルに構文レベルでの理解がある．これはプログラムが構文規則に従っているかどうかを調べるレベルである．

本研究では（1）の問題解決レベルを除いた（2）以下のレベルのプログラム理解の実現を目指す．図2にプログラム理解の各レベルとプログラミング知識との対応を示す。

5　プログラミング知識の表現

本システムでは，プログラミング知識は経験を積んだプログラマが獲得している典型的なコーディング・パターンと一般的なプログラミング規則によって構成される．

5．1　プログラム・プラン

プログラム・プランとはアルゴリズムを構成する典型的なコーディング・パターンである．初級プログラマに対して経験を積んだプログラマは，このようなプログラム・プランを豊富に持ち，プログラミングにおいて有効に用いていると考えられる．

変数への値の代入や四則演算のような1行の命令コードに対応するもっとも基本的

なパターンをプリミティブ・プランと呼ぶ．プリミティブ・プランではこのような副作用を持つ命令コードのみを考え、プログラム理解の階層の基本的データ操作レベルに対応するものである．while文や、if文のような構造を持った命令コードは、プリミティブなコードを結合して基本的データ処理レベルのパターンを生成するものと考え、プリミティブ・プランには含めない．

プリミティブ・プラン以外のプログラム・プランは基本的データ処理レベルのパターンに対応するもので、プログラム内での変数の役割を記述する変数プラン、典型的なループ・タイプを記述したループ・プランが診断上重要である．これらの他に２つの変数値の交換のような決まりきった部分的コーディング・パターンを記述したプログラム・プランが多数含まれる．これらはプリミティブ・プランの組み合せとして表現される．

5．2　各種のプログラム・プラン

本システムでプログラミング知識を表現するために使用される各種のプログラム・プランについて説明する．

（１）プリミティブ・プラン

プログラム中の各行を解釈するための詳細なプランである．これによってプログラム各行の役割が明らかになるとともに、演算項目の数やリンクの種類によらない一定の形式で表現することができる．構文的に正しい命令文はすべて解釈することができるので、学習者プログラムの入力段階で構文チェックが終っている本システムでは、すべての命令行がこのプランによって解釈される．プリミティブ・プランは、定数代入、四則演算、読み込み、書き出しプランなどのもっとも基本的なプランである。

（２）変数プラン

プログラム上で変数の果たす役割を分類し、記述したプランである．変数プランは、変数が働く一区切りの意味のあるタスクを対象とし、変数の初期化、更新に用いられるプログラム・プランの組み合せによって記述する．これによって新値変数（データの読み込みによって更新される変数），カウンタ変数，累計変数などの変数役割が記述される．

（３）ループ・プラン

ループ・プランとは、ループの構造に着目し、あらかじめ想定できるループ構造を抽出したものである．ループの実行条件判定部分に出現する変数とループ本体で更新される変数の組み合せによってループ・プランを定義する．カウンタ制御ループ，新値変数制御ループなどがある．

（４）その他のプログラム・プラン

上記以外にも２つの値の入れ替えプランやカウントアップ・プランなど多くの一般的なプログラム・プランが準備されている．この中には２つの値の入れ替えプランのような抽象度の低いプランがかなり含まれるが、これらは低レベルのバグを指摘する必要性による．これらは命令コードと対象変数や定数をノードとし、それらの間をリンクで結合した木構造グラフで表現されている．

5．3　プログラミング規則

プログラミング上の一般的規則などはプロダクション・ルール形式で記述される．これらのルールは、作業記憶上の学習者プログラムの拡張解析木、変数定義、およびプログラムの分析情報を参照してルールの条件部分が評価され発火される．

6　バグ知識

本システムにおけるバグの同定は，あらかじめ抽出された誤りパターンとのマッチング，プログラミング規則からの逸脱のチェック，および課題アルゴリズムとの差異のチェックによって行われる．

誤りパターンには初級プログラマに特有の誤りと，頻繁に観察される一般的な誤り

パターンが含まれている．初級プログラマが犯し易い誤りのパターンは，プログラミング教授に携わる専門家のインタビューや，初級プログラマのコーディングの分析の結果得られたものである．

誤りパターンには，一般性を持ったものとプログラミング課題に依存したものがある．一般性を持ったパターンはプログラミング課題によらず観察される誤りであり，変数値を累計する場合に代入演算子の右辺と左辺に同じ変数がくるのを嫌い，作業用変数を用いて値を一旦待避してから改めて加えるというパターンや，配列の指標が１から始まると誤解している（正しくは０から始まる）パターンなどがある．

課題に依存する誤りパターンとしては，カウンタ制御ループにおけるループ実行条件式中のカウンタの上限値の誤りなどがある（カウンタｉが１０未満であれば処理を繰り返すべきところを，等号を含め１０以下とするなど）．課題依存のバグには問題解決過程に起因するバグが多く含まれる．

これらの誤りパターンは，デバッグにおける専門家のバグ発見に関する着眼点であり，詳細レベルのバグを含んだプランは正しいプログラム・プランと同様，学習者プログラムの各部分とマッチングが試みられる．また，課題に依存したバグは学習者プログラムと課題アルゴリズムとの比較において，課題アルゴリズムを摂動するために利用される．

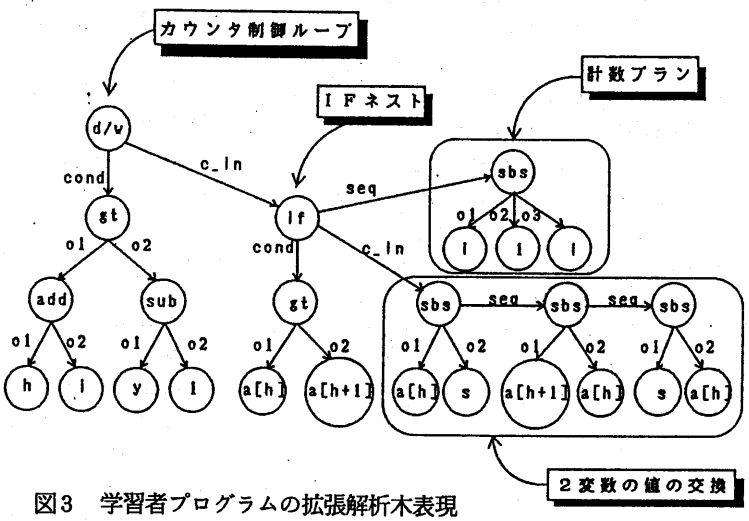

図3　学習者プログラムの拡張解析木表現

7　学習者プログラムと課題アルゴリズム

本章では、学習者プログラムの表現と課題アルゴリズムの表現について述べる．

7．1　学習者プログラムの表現

学習者が入力したソース・プログラムは学習者プログラム変換モジュールによって内部表現に変換される．学習者プログラムの内部表現は，変数定義とプログラム本体の拡張解析木表現である．

学習者プログラムの拡張解析木表現は，命令と対象となる変数，定数をノードとし，リンクによってそれらノード間の関係や実行順序を表現したものである．拡張解析木の各部分とプログラム・プランのマッチングにより，各部分の役割が同定され，プログラム・プランをノードとするネットワーク（プランの系列）に変換される（図３参照）．

7．2　課題アルゴリズム知識

課題アルゴリズム知識は，専門家によってシステム内に記述される正しいアルゴリズムの表現である．これは学習者プログラムの正当性を調べる際のテンプレートとなる知識である．課題アルゴリズム知識は、学習者プログラムを解釈するのに用いられるプログラム・プランをノードとし，リンクによって実行順序を表現する拡張解析木で記述される．

プランの系列として記述されているため、個々のプランを実現するためのコーディング・テクニックについては学習者プログラムにおける多様な選択（誤ったものを含めて）に対応することができる．

課題アルゴリズム変換知識を利用して，プランをコードとして実現するテクニックの選択，順不同である処理順序の入れ替えの可能性を考慮した複数解が生成される．これにより制限されたものではあるが、より多様な学習者プログラムを解釈することができる．図4に課題アルゴリズムの拡張解析木表現の例を示す．

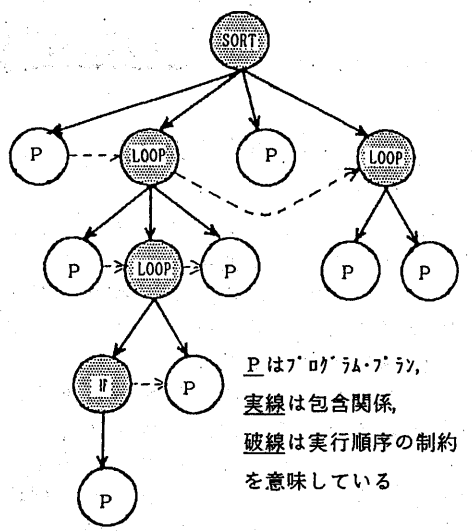

図4．課題アルゴリズムの拡張解析木表現

8 診断モジュール

本システムでは，プログラミング知識を参照し，学習者プログラムを構成する個々の基本的データ処理レベルのコーディングの論理的整合性を分析する論理的診断と，プログラムが実現すべき機能の観点から課題アルゴリズムと学習者プログラムを比較し，その差異に着目する意味的診断を行う．診断モジュールは全体として一種の黒板モデルを構成しており，診断制御はデバッグ戦略知識を参照する制御モジュールが行う．図5に診断モジュールの構成を示す．

8．1 論理的診断

プログラム・プランとプログラミング規則を用いた学習者プログラムの診断を論理的診断と称する．論理的診断は次の手順で行われる．

（1）プリミティブ・プランによる解釈／制御の流れの同定

まず学習者プログラムに対してプリミティブ・プランによるノード，リンクの解釈が行われる．前述したように本システムでは構文的に正しいプログラムが診断モジュールに渡されるので、すべてのコードがプリミティブ・プランで解釈可能である．この段階で順次処理、ループ処理の個々の処理ブロックが識別され、学習者プログラムの制御構造（フローチャートに相当する）が明らかになる．すなわち、個々の順次処理やループ処理について連番が振られ，ループについては各種の変数リストが生成され、制御の階層構造が抽出される．

（2）プログラム・プランによる解釈

つぎに変数プランや一般的なプログラム・プランが調べられる．たとえば，変数プランについては，その初期化，更新に適用されるプログラム・プランがそれぞれ唯一に決まればこの段階で変数の役割が決定される．また、ループについては、条件判定式中の変数とループ本体で更新される変数のうち一致するものが一つにしぼられ，変数プランが決定されていれば、そのループに対応するループ・プランが決定される．

この段階で使用されているプランが決定できない変数やループについては、複数の可能性が中間仮説として作業記憶に書き出され、未決定のまま意味的診断に引き継がれる．

（3）プログラミング規則による診断

さらにプロダクション・ルール形式で記述されたプログラミング知識によって論理的なバグが調べられる．

各ルールは分析の対象となるループ、変数に関する情報を参照して発火されるが、

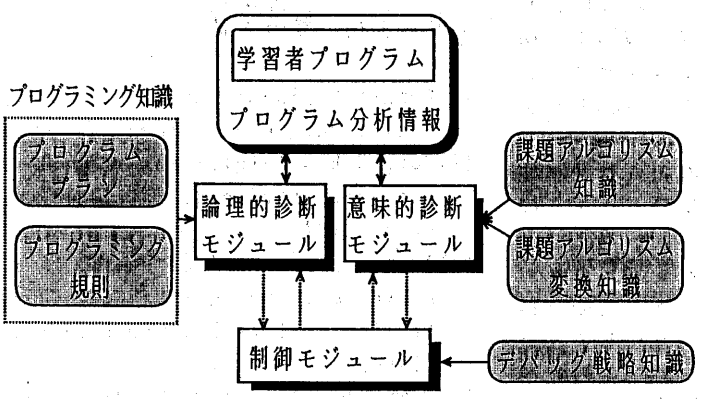

図5　診断モジュールの構成

個々のループ、変数に対しては一回ずつだけ適用され、推論の終了条件が成立した場合、また適用可能なルールがなくなった時点で推論を終了する.

8.2 意味的診断

課題アルゴリズムと学習者プログラムの比較によるアルゴリズムの診断を意味的診断と称する. 意味的診断は次の手順で実行される.

（1）制御構造の比較

学習者プログラムと課題アルゴリズムの制御構造を比較し, 不一致があれば, 課題アルゴリズムの変形を試み, さらに比較を繰り返す. システムが準備できるすべてのアルゴリズムのバリエーションについて制御構造の一致が得られない場合には、学習者とシステムの間で根本的なプログラミングの戦略が異なっていると考えられるため, 論理的診断の結果のみから誤概念を分析し、学習者に提示して診断を終了する.

（2）プランの系列同士の比較

制御構造が一致した場合には, プラン表現同士の比較を行い, 論理的診断段階で未決定となっている変数役割, ループのタイプなどを同定しようと試みる. ここで、学習者プログラム上のノードとマッチングのとれた課題アルゴリズム上のノードには学習者によって考慮されていることを示すマークが付与される.

プラン表現同士で完全なマッチングが行われない場合には、これらの間の差異に着目して学習者のバグを同定しようと試みる. ここで着目される差異とは、条件判別式中の演算子の逆転、変数の数の不一致、課題アルゴリズム中に現れないプランの使用などである. これらのバグ. パターンもプログラミング知識に含まれ, 課題アルゴリズムを摂動することによって学習者プログラムと同様の構造が得られるかどうかを検討する.

（3）学習者プログラムの実行情報の獲得

プログラム実行ウィンドウにおいて, 学習者にプログラムのコンパイル, 実行を行なわせ, 実行時情報を取り込む. 実行時情報とは、学習者のプログラムが無限ループに陥って終了しない、0による割り算のためや配列の指標が不正なため実行が中断されたといった情報である.

8.3 診断モジュールの制御

前節までで説明してきた2種類の診断モジュール, すなわち論理的診断モジュールと意味的診断モジュールは, デバッグ戦略知識を参照する診断制御モジュールの制御のもとで診断を進める.

一般に, 論理的診断において一意に同定できない変数プランやループ・プランなどは, 複数の説明可能な仮説を与えた後, 意味的診断にその同定が委ねられる. ここで, 学習者プログラムの誤りのため, 論理的診断モジュールが性急な結論や仮説を与えていた場合には, 意味的診断の結果によって論理的診断の再起動が行われる.

制御構造の抽出やプリミティブ・プランの同定のような, いわば手続き的な診断を別とすれば, 変数役割やループ・タイプの同定が診断における主目的である. 論理的

診断で未決定のこれらの情報が意味的診断において決定できれば，これらの結果を持って再び論理的診断においてコーディングの論理的整合性がチェックされる必要がある．

例えば，学習者プログラムのカウンタ制御ループにおいてループ内でのカウンタのカウント・アップが欠落している場合には，論理的診断においては，この変数がカウンタであることを同定することは不可能であり，当該ループが無限ループとなることが指摘されるのみである．そして，この変数プランは未解決のまま意味的診断に委ねられる．意味的診断において，学習者プログラムと課題アルゴリズムの対応によって，このループはカウンタ制御ループであるべきであるとの結論を出せれば，作業記憶中のループ制御変数，変数プラン情報を更新し，論理的診断の再実行を促す．論理的診断は，プリミティブ・プランのマッチング以外の処理を改めて実行する．すなわち，このループについては，変数プランをカウンタ変数プランとし，当該のループをカウンタ制御ループとして論理的整合性を再検討し，変数のカウント・アップが欠如していることが指摘される．

9 今後の課題

課題アルゴリズムの表現がプログラム・プランをノードとする意味ネットワーク表現であり，より深い表現によらないと，汎用的なアルゴリズムが表現できないという問題がある．ただし，そのような表現をシステムにもたせるためには，教材開発者に教材知識の格納のための高度なスキルが必要とされるという問題点が残る．

また，バグと誤概念の関係が直接的に表現されているため，バグとその根元にある誤概念との関連が固定的にしか指摘できない点にも問題がある．バグとその背景にある誤概念の関連付けをより知的に行い，より深い診断を行うために、バグの分類、誤概念の分類、それらの対応付けを行ない、これらの知識を用いたプロダクション・システムを組み込むことを検討している．

10 引用・参考文献

(1)Wenger,E., Intelligent Tutoring System,Morgan & Kaufmann,(1987)

(2)Ruth,G.R., Intelligent Program Analysis,Artificial Intellience,pp. 65-85,(1976)

(3)Adam,A., Laurent,J.P., Laura, A system to Debug Student Program, Artificial Intelligence,pp.75-122, (1980)

(4)上野晴樹,知的プログラミング環境,情報処理vol.28 no.30,pp.1280-1296, (1987)

(5)Soloway,E., Johnson,W.E., PROUST: Knowledge-Based Program Understanding,IEEE Trans.on Soft.Eng., vol.SE 11,no.3,pp.11-19,(1986)

(6)Soloway,E., MenoII an Ai-based Program Tutor,Journal of Computer-based Education,vol.10.no.1&2,pp. 20-34,(1983)

(7)上野晴樹，アルゴリズムに基づくプログラム理解の枠組み，人工知能学会研究会資料，SIG-KB5-8902-4(6/7), (1989)

(8)安田,岡本,プログラムのアルゴリズム診断を中心としたＩＴＳの研究（１），情報処理学会研究報告,90-ce-10,pp. 87-94,(1990).

(9)安田,岡本,プログラムのアルゴリズム診断を中心としたＩＴＳの研究（２），人工知能学会第４回全国大会論文集, (1990).