

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)

The study of ITS to diagnose the algorithm of the novice's program.

安田 恭一郎

Kyoichiro YASUDA

トッパン・ムーア システムズ(株)

TOPPAN MOORE SYSTEMS LTD.

岡本 敏雄

Toshio OKAMOTO

東京学芸大学

TOKYO GAKUGEI UNIVERSITY

あらまし: 本研究ではC言語を学ぶ初級プログラム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。

キーワード: 知的CAI プログラム理解 情報処理教育 拡張解析木 誤概念の同定

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、初級プログラム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ている⁽¹⁾。

プログラムの分析は、プログラムに関する理解なしには不可能であるが、プログラム理解はプログラムが遂行するタスクの意味的理解のレベルから、アルゴリズム・レベル、データ処理技法レベル、基本データ操作レベル、さらに個々のプログラミング言語の構文の理解のレベルまでの複数のレベルからなる階層的構造をなしている。また、プログラミング知識についても同様の階層性を持っていると考えられる⁽²⁾。

さらに、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⁽³⁾、Adamらのデバッグ・システムLAURAにおけるプログラム・プラン⁽⁴⁾、上野らによる手続きグラフ⁽⁵⁾、SolowayらのPROUSTにおける戦略グラフ⁽⁶⁾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⁽⁷⁾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

本研究では、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を枠組みを提案することを目的とする。プログラムの分析においては、プログラムを構成する個々のテクニックに着目すると共に、課題アルゴリズムの抽象的な記述を準備することによって学習者プログラムのデバッグを行う。このために次の目標掲げる。

(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) プログラミングのゴール知識となる課題

(4) プログラマの犯しやすいバグと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。

知的プログラミング環境の構築という観点からは、プログラム中のプリミティブな論理的バグや、ある程度固定的な課題アルゴリズムの記述によって指摘できる意味的なバグを同定することさえもプログラミング言語の習得において学習者に対する大きな支援となると考える。

学習領域としてC言語を扱い、課題としては与えられた一次元配列に対する並べ替えプログラム程度の題材を選択している。

本システムでは、学習者にC言語のソース・プログラムを対話形式で入力させ、この学習者プログラムを拡張解析木 (Argumented Parse Tree) による内部表現に変換し、これに対してさまざまな診断を行う。

したがって、本システムは、対話形式で学習者のソース・プログラムを取り込むフェイズと、バッチ方式でプログラムを診断するフェイズの二つのフェイズに分けることができる。

本システムはワークステーション、SONY NEWS上に、対話モジュールはC言語、診断モジュールはK-Prologによって実現されている。システムの全体的な構成を図1に示す。

本システムでは、定型的なプログラミング知識のプリミティブの木構造グラフ表現をプログラミングにおけるプランとして記述している。プランとはアルゴリズムを実現するためのコーディングのパターンのことである。プランには、変数の役割に関する変数プラン（カウンタ・プラン、累計プランなど）、ソース・コードの行単位の意味を解釈する行プラン（カウンタのカウンタ・アップ、値の累計など）、複数の行から構成されたり、行プランの組合せである一般的なプラン（各種のループ・プラン、2値の入

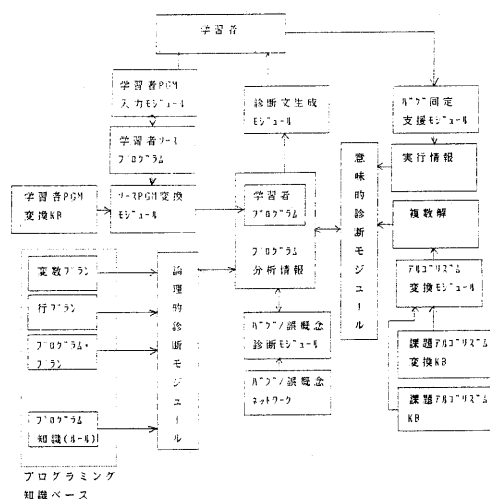


図 1. システムの構成

れ替プランなど)がある。プランは正しい知識のみではなく、想定される誤りのパターンをバグ・プランとして含んでいる。

本システムはつぎのような処理モジュールによって構成される。

学習者はシステムによる支援を受けながらソース・プログラムを入力する。この段階で学習者プログラムの構文エラーが指摘される。IF文や各種のループの仕様などはあらかじめ準備されており、メニュー選択によってこれらの枠組みが画面に提示される。これによって診断モジュールに渡される学習者プログラムに構文上のエラーや条件分岐の混乱が含まれることを防止でき、無用な診断の負荷が軽減される。

学習者が入力したソース・プログラムを学習者プログラム変換知識を用いて、拡張解析木表現に変換する。

プログラミングにおけるプリミティブな知識
およびその組み合わせを表現した正しいプランお

よびバグ・プランと、プランの解釈規則などを記述したルール・ベースを用いて論理的なバグの診断を行なう。

(4) 意味的診断モジュール

プログラミングのゴールとなる正答のアルゴリズム(課題アルゴリズム)は、プランをノードとする拡張解析木として表現される。意味的診断モジュールは、この課題アルゴリズムと学習者プログラムとの差異に着目し、プログラムの意味的な診断を行なう。

(5) アルゴリズム知識変換モジュール

上述のようにプランによる拡張解析木としてシステムに格納されている模範的な課題アルゴリズムを変換して複数解を準備する。

(6) バグ同定支援モジュール

バグの同定を容易にするために、学習者から課題プログラムの実行情報を取り込む。

(7) バグ/誤概念診断モジュール

予想されるバグとその背景に存在する誤概念との関連をネットワーク表現したバグ/誤概念ネットワーク知識を用いて、同定されたバグの組み合わせにより、その背景にある誤概念を同定する。

(8) 診断文生成モジュール

システムによって同定されたバグや誤概念情報などの診断結果を学習者に提示する。

3. 2 知識ベース

本システムではアルゴリズムの診断において必要となるプログラミング知識や課題アルゴリズム知識を記述するためにつぎのような知識ベースを準備している。

(1) プログラミング知識ベース

プログラム中の論理的バグを同定するために論理的診断モジュールで利用される知識である。木構造グラフ表現されたプリミティブ知識とそれらの解釈や一般的プログラミング規則を記述したルール・ベースから構成される(5節で詳述する)。

(2) 学習者プログラム変換知識ベース

学習者によって入力されたソース・プログラムを拡張解析木表現に変換するための知識である。

(3) 課題アルゴリズム知識ベース

課題アルゴリズム知識は、課題プログラムの正しいアルゴリズムを表現したもので、意味的診断モジュールで使用される。アルゴリズムを構成するプランをノードとし、それらの処理順序をリンクで示した拡張解析木によって記述される(6. 2節で詳述する)。

(4) アルゴリズム変換知識ベース

課題アルゴリズムに対して、プランの実現のためのコーディング技法の選択や、実行順序の入れ替えによって可能な複数解を生成するための変換知識である。

(5) バグ/誤概念ネットワーク

プランやプロダクション・ルールによって同定されるバグとその背景にある誤概念との関係をネットワーク表現したものである。学習者が理解すべきプリミティブなプログラミング知識を階層的にネットワーク表現し、それらの知識の欠如や予想できる誤った理解と、あらかじめ想定されるバグを直接的に結合させたものがバグ/誤概念ネットワークである。図2にバグ/誤概念ネットワークの例を示す。

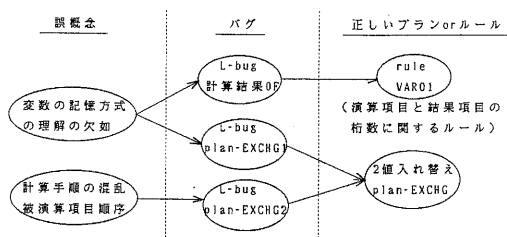


図2. バグ/誤概念ネットワーク

4. システムの基本動作

4. 1 プログラミングの課題

本システムで扱う課題は値が初期セットされた一次元配列データに対するバブル・ソートで

あり、並べ替え後のデータを画面に表示することが求められる。

学習者によるプログラムの入力にはある程度の制約を課しており、goto文の使用は許さず、また、使用する変数、ループの形式などは、あらかじめ指定されている。また、ひとつの変数を更新を伴う異なった作業に併用してはならないという制限を課している（たとえば、カウンタ用変数*i*を複数のループで別の目的で使用する）。また、課題プログラムの変数定義部分は出題者によってあらかじめ与えられている。この課題に含まれる初心者にとって困難な点はカウンタ制御型のループの正しい使用方法である。

4. 2 基本動作

システムの基本動作について述べる。システムは次の手順で各処理を行う。診断モジュールの動作の流れを図3に示す。

(1) 学習者プログラムの入力

学習者が対話形式でソース・プログラムの入力を行う。この段階では論理的な誤りは指摘せず、構文エラーのみをチェックする。

(2) 学習者プログラムの変換

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拡張解析木表現に変換される。変換後の拡張解析木に対して次の2種類の診断が行なわれる。

(3) 論理的バグの診断

変数プラン、行プラン、プログラム・プラン（バグ・プランを含む）によってアルゴリズムを実現するための変数役割や部分的なコーディング・プランが同定され、プログラミング知識を記述したプロダクション・ルールによって論理的なバグ（L-bugと称する）が同定される。

(4) 意味的バグの診断

論理的診断が終了すると、課題アルゴリズムを利用した意味的なバグ（S-bugと称する）の診断を実行する。ここでは、あらかじめ格納されているプラン表現された理想解のアルゴリズム

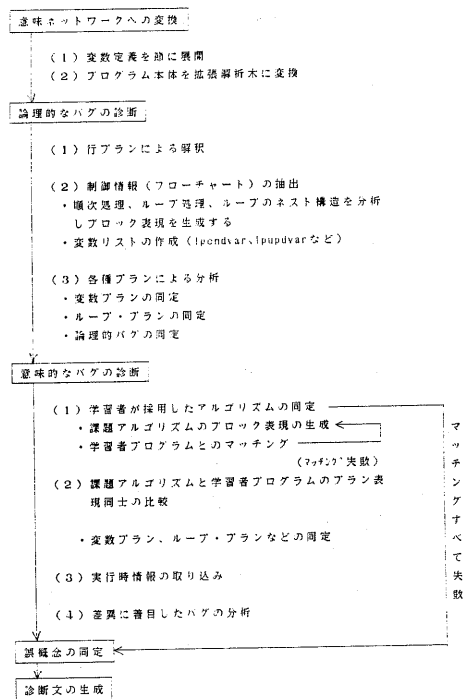


図3. システムの基本的動作

およびその変形バージョンと学習者のプログラムの比較を行う。マッチングがとれなかった場合には、これらの間の差異（処理ブロックの不一致、判別条件の演算子の逆転など）に着目しバグの同定を試みる。

さらにこの診断においては、プログラム実行時のエラー情報（0で除算を行ないプログラムが中断したなど）を利用し、診断の負荷を軽減する。

(5) 誤概念の同定

前段階までで同定された学習者プログラムのバグをバグ／誤概念ネットワークを用いて解釈し、バグの背景にある誤概念を同定する。

(6) 診断文の作成及び提示

バグ、誤概念に関する診断結果を学習者に提示する。

5. プログラミング知識の表現

プログラミング知識は、プログラミングに関するプリミティブな知識を木構造グラフ表現し

た各種のプランと、プランの解釈規則や一般的なプログラミング上の規則を記述したルール・ベースで構成される。

5. 1 各種のプラン

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ランについて説明する。

(1) 変数プラン

プログラム上で変数の果たす役割に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する。つぎにいくつかの例を示す。

・新規変数プラン

変数が標準入力やファイルからの読み込み（読み込みプラン）によって初期化され、読み込みプランによって更新される。

・カウンタ変数プラン

数値変数が、代入（定数による初期化プラン）によって初期化されカウント・アップによって更新される。ループ内での使用が一般的である。

・累計変数プラン

数値変数が、代入（定数による初期化プラン）によって初期化され、累計によって更新される。ループ内での使用が一般的である。

・作業変数プラン

2 値の入れ替えなどのプランの中で更新、参照され、その後プログラム中で参照されない。

(2) 行プラン

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。

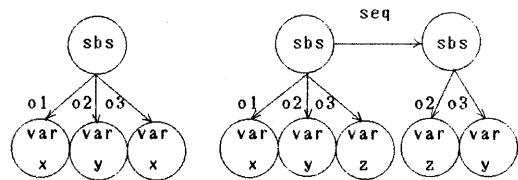
・定数代入プラン

・四則演算プラン

・読み込みプラン

・書き出しプラン

・累計プラン
などがある。



(a) 累計プラン

(b) 冗長な累計プラン

図4. 累計プラン

(3) その他のプラン

すでに述べた変数プラン、行プラン以外の典型的なコーディング・テクニックをプリミティブな知識の組み合わせによって記述したプログラミング知識である。これには2つの値の入れ替えプランやWHILEループ、DO/WHILEループ（UNTILループ）プランなどがある。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。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わせによってループ・プランを定義する。これらのプランのいくつかを次に示す。

・2つの値の入れ替えプラン

・新値ループ：新値変数によって制御される。

・新値累計ループ：新値ループにおいて新値変数が累計されるもの。

・カウンタ・ループ：カウンタ変数によって制御される。

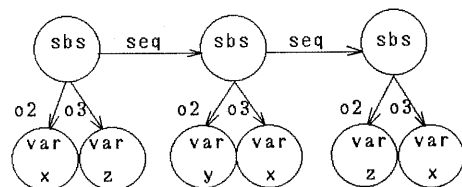


図5. 2値の入れ替えプラン

5. 2 ルール・ベース

木構造グラフ表現されたプランによって記述できない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(ループの判別条件に含まれる変数がループ本体で更新されない誤りなど)。また、変数プランの同定においても初期化と更新に適用されるプランを結び付けるためにはこのルールが用いられる。これらのルールは、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作業メモリとして参照してルールの条件部分が評価され発火される。

5. 3 プランの適用(論理的診断の方法)

前述のプランとルール・ベースを用いた学習者プログラムの解釈を論理的診断と称する。

ここでは、まず学習者プログラムに対して行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、すべてのコードが行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、ループについては条件判定式に現われる変数、

lpcndvar (ループ番号, [変数リスト])
ループ本体で更新される変数、

lpupdvar (ループ番号, [変数リスト])

などの変数リストが抽出され、その制御の階層構造が抽出される。

つぎに変数プランや一般的な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、あるループについて条件判定式中的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、未決定のままトップダウンの診断に引き継がれる。

さらにプロダクション・ルールの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。たとえば、ループの条件判定式中に現れる変数が、ループ本体において更新されていない場合、あるいはカウンタ・ループ・プランの使用が確認された場合にループの前までにカウンタが初期化されていないといった論理的なバグが同定される。

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、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。

6. 学習者プログラムと課題アルゴリズムの表現

6. 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知識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の関係や実行順序を表現したものである(図6参照)。

ノードの種類は演算(op)、変数(var)、定数(val)の3種類である。リンクの種類を表1に示す。

表1. 学習者プログラム表現に用いられるリンクの意味

リンク名	リンクの意味
seq	順次処理
cond	ループの判別条件
c_in	条件成立時の処理
c_out	条件不成立時の処理
o1	演算項目 1
o2	演算項目 2
o3	演算結果項目

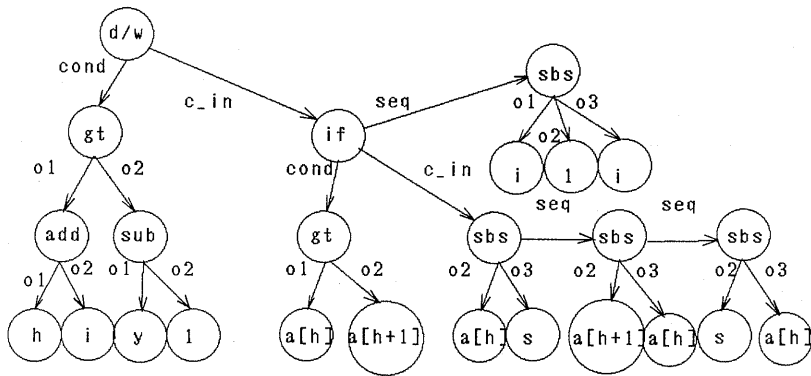


図 6. 学習者プログラムの拡張解析木表現 (部分)

学習者プログラムは前節で述べたプランによって解釈され、プランをノードとするネットワークに変換される。さらにノードにループの入れ子を含まないようなプランの系列、ブロック (順次処理、ループ処理) をノードとするネットワークが生成される。

これらの各学習者プログラム表現の階層を図 7 に示す。

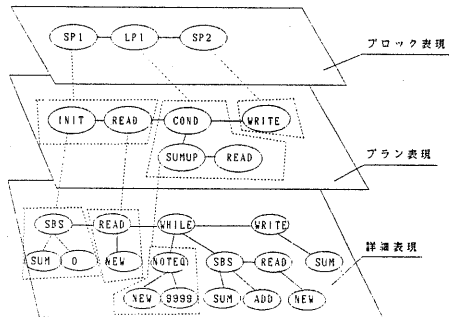


図 7. 学習者の拡張解析木の階層

6. 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 (図 9 参照)。

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。

課題アルゴリズムから、プログラミング知識を利用して順次処理、ループ処理からなる処理ブロックをノードとするネットワークが生成される。このネットワークが学習者プログラムのブロック表現と比較され、学習者のプログラミング戦略が大枠で出題者の意図した通りであるかを調べることができる。

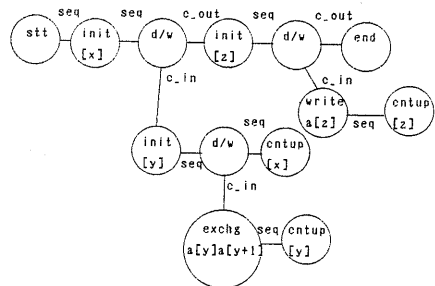


図 8. 課題アルゴリズムの拡張解析木表現

6. 3 課題アルゴリズムと学習者プログラムの比較 (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。

まず、格納されている課題アルゴリズムのブロック表現を生成する。そして、学習者プログラムと課題アルゴリズムのブロック表現同士を比較し、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ブロック表現間のマッチングがとれない場合には、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、ボトムアップの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。

ブロック表現において一致がみられた場合には、一致した課題アルゴリズム（またはそのバリエーション）を採用し、プラン表現同士の比較を行う。比較は各処理ブロック単位で行い、ボトムアップの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。

各処理ブロック単位での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。

また、学習者から実行時情報が得られる場合には、バグの診断にこれを利用する。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。

7. 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／誤概念ネットワークが固定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。また、現状ではこの知識は経験を積んだプログラマの知識を基にして想定され構成されているが、C言語を学び始めた大学生を対象にしてバブル・ソートのプログラミング例を収集し、バグ・誤概念ネットワークを充実させたいと考えている。

8. 引用・参考文献

- (1)Wenger,E.,Intelligent Tutoring System, Morgan & Kaufmann,(1987)
- (2)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-KE5-8902-4(6/7),(1989)
- (3)Ruth,G.R.,Intelligent Program Analysis, Artificial Intelligence,pp.65-85,(1976)
- (4)Adam,A.,Laurent,J.P.,Laura,A system to Debug Student Program,Artificial Intelligence,pp.75-122,(1980)
- (5)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)
- (6)Soloway,E.,Johnson,W.E.,PROUST:Knowledge-Based Program Understanding,IEEE Trans. on Soft.Eng.,vol.SE11,no.3,pp.11-19,(1986)
- (7)Soloway,E.,Meno2 an Ai-based Program Tutor,Journal of Computer-based Education,vol.10.no.1&2,pp.20-34,(1983)