プログラミング演習支援システムにおける実行履歴の構造化方式

岩間　信介† 　立岩佑一郎† 　山本　大介† 　高橋　直久†

† 名古屋工業大学 大学院工学研究科　〒 466-8555 愛知県名古屋市昭和区御器所町
E-mail: †iwama@moss.elcom.nitech.ac.jp, ††{tateiwa,yamamoto.daisuke,naohisa}@nitech.ac.jp

あらまし　本稿では，プログラミング演習における答案プログラムの振る舞いを解析し動作を構造化して，構造化実行履歴 (SET:Structured Execution Trace) を生成する方式を提案する．ここで SET とは，実行時の変数の変化や制御構造，関数の呼び出し構造を XML で表現したデータである．提案方式の実現上の特徴として GDB などの汎用的なツールを使って SET を生成する点がある．本稿では，提案方式の実現法について述べる．また，答案プログラムのためのステップビューワーなどのプログラミング演習支援ツールを SET を用いて実現する方式について述べる．
キーワード　e-learning，プログラム解析，XML

A Structuring Method of Execution Trace for a Computer-Aided Programming Exercise System

Shinsuke IWAMA†, Yuichiro TATEIWA†, Daisuke YAMAMOTO†, and Naohisa TAKAHASHI†

† Graduate School of Engineering, Nagoya Institute of Technology　Gokiso, Showa, Nagoya, 466-8555 Japan
E-mail: †iwama@moss.elcom.nitech.ac.jp, ††{tateiwa,yamamoto.daisuke,naohisa}@nitech.ac.jp

Abstract　This paper presents a method to generate Structured Execution Trace (SET) by the analysis of program behavior in a computer-aided programming exercise system. SET, which consists of the changes of values in variables, control structures, and call structures of the functions, is described by using XML. Moreover, this paper presents a method to realize a step viewer, a programming exercise support tool, by using SET.
Key words　e-learning, program analysis, XML

1. はじめに

我々は繰り返し学習 [1] を支援するプログラミング演習システム CAPES [1] [2] の研究を進めている．ここで，本研究が対象としている学習は指導者が提示した問題の題意を満たすプログラムを受講者に作成させるものである．

CAPES では，答案と正解例プログラムの実行時の出力結果を比較し，答案の妥当性の評価を即座に行うことで繰り返し学習を支援する．この実行結果による評価方法では次に示す問題点がある．

問題点 1　受講者が実行結果を見てもプログラムの間違い箇所が分からない場合がある

問題点 2　関数や制御構造に間違いがあっても正解と判定されてしまう場合がある

そこで，上記の問題点を解決するアプローチとして実行結果だけではなくプログラムの振る舞いを調査する [3]．そして，その結果を用いて，答案プログラムの動作を受講者に提示する．これにより，間違い箇所を分からせることが可能になる．また，振る舞いを調査した結果より，答案プログラムの振る舞いについても誤りがあるかの判定が可能になる．

以上より本研究では，答案プログラムの振る舞いを解析し，その結果を構造化した構造化実行履歴 (SET) を生成する方式を提案する．SET 生成方式における実現上の特徴を以下に示す．

特徴 1　プログラム変換機能
プログラムを文単位でトレース可能な形式としてアセンブラ対応形式に変換する機能を実現する．これにより，汎用的なデバッガでプログラムを実行させて文単位で詳細にトレースできるようにする．

特徴 2　実行履歴生成コマンド生成機能
汎用的なデバッガの動作を制御して，実行時の値などを出力させるコマンドを生成する機能を実現する．これにより，汎用デバッガを用いてプログラム動作時の変数の値や構造に関するデータなどを取得することが可能となる．

特徴 3　実行履歴解析機能
汎用的なデバッガより得られたプログラムの動的解析結果を，プログラムの静的解析結果と結びつけて XML 形式に変換する機能を実現する．これにより，汎用的なデバッガから得られたデータより SET を生成することが可能となる．

さらに，本稿では答案プログラムのためのステップビューワーなどのプログラミング演習支援ツールを SET を用いて実現する方式を提案する．SET を用いたステップビューワーの実現上の特徴を以下に示す．

実現上の特徴 1　SET を用いて実行履歴を辿ることにより，前方と後方のトレース機能を有する高機能なビューワーを web 上に実現する．これにより，開発環境がなくてもプログラムの動作を確認することが容易となる．

実現上の特徴 2　答案の SET と正解例の SET とを比較して両者の差異を求めることにより，答案の誤り箇所を提示する機能を実現する．これにより，実行時のトレースを提示しながら答案プログラムの誤り箇所を受講者に提示することが可能となる．

実現上の特徴 3　変数の名前と値を検索キーとして SET を検索して，その結果に基づいて実行トレースの開始位置を決める機能を実現する．これにより例えば，変数 i が初めて 1 となる時点からのトレースが可能となる．

2. 関 連 研 究

プログラムの解析ツールとして，Sapid [4] など多くの静的解析ツールが開発されている．Sapid では，C 言語のソースコードを字句解析，構文解析して，12 種類のクラスと，構成関係や定義参照関係などを表す 29 種類の関連としてモデル化を行っている．そして，モデル化されたデータを制御依存関係とデータ依存関係などを表すビューや，波及解析などの基盤技術ライブラリを用いることで，さまざまな CASE ツールの基礎を提供する．

Sapid は，静的なプログラム解析を用いている．しかし，提案方式では静的解析だけではなく，動的解析も用いることで，実際にプログラムを動作させた時のプログラムの振る舞いを構造化している．これにより，静的解析だけではわかりにくいプログラム実行時のソースコードの実行順や，変数の変化などについて解析可能である．Sapid など従来の静的解析ツールを提案方式の字句・構文解析で用いることにより，より精度が高く，細かな構文解析データを取得可能になる．

また，実行履歴の解析手法として，実行履歴からのシーケンス図の生成手法に関する研究 [5] がある．この手法は，オブジェクト指向プログラムの実行履歴を基にシーケンス図を作成することで，プログラム実行中に生成される各オブジェクトの動作を視覚的に示す．このシーケンス図と，設計段階で作成されたシーケンス図を比較することで，実装されたプログラムの振る舞いと設計段階での振る舞いとの違いについて調べることが可能である．

文献 [5] では実行履歴を解析する際に，実行履歴中に出現する繰り返し構造や再帰構造を簡素な表現で書き換えることにより情報量を圧縮する手法を提案している．この手法を用いて提案方式の SET を図式に変換する機能を実現すると，プログラムの動作を図などで表現するアプリケーションの開発が可能になる．

3. 構造化実行履歴 SET

答案評価の方法として，答案と正解プログラムの実行時の出力結果を比較する方法がある．この方法では，最終的な出力結果だけを比較するため，出力結果を得るまでの構文や関数に誤りがあっても，誤りを発見することができない場合がある．

この問題を解決するために，本研究では，受講者へのプログラム実行時のトレース提示と，プログラムの動作を詳細に調べた結果に基づく判定の実現に必要なデータを生成する．具体的には提案方式では，プログラム実行時のトレース結果とプログラムの静的解析を組み合わせて，実行時の変数の変化や，制御構造や関数呼び出しを構造化したデータを XML で表現した SET を生成する．SET は，実行結果だけではなく表 1 と表 2 に示す制御や代入に関する項目を保持する．ここで，type とは，表示項目の種類を表している．

表 1　制御に関する SET における項目の一覧

表示項目	type	データ	データの意味
関数呼び出し	func	fname	関数名
		in-args	関数に入る時の引数の値
		out-args	関数から出る時の引数の値
		in-global	関数に入る時の大域変数の値
		out-global	関数から出る時の大域変数の値
		return	戻り値
繰り返し構造	loop	ltype	繰り返し構造の種類
		cvalue	制御変数の名前
		in-value	繰り返し構造に入る時の変数の値
		out-value	繰り返し構造から出る時の変数の値
		value	STEP 毎の変数の値
条件分岐の分岐状態	branch	btype	条件分岐の種類
		in-value	分岐構造に入る時の変数の値
		out-value	分岐構造から出る時の変数の値

表 2　代入文に関する SET における項目の一覧

表示項目	type	データ	データの意味
代入操作	assign	vname	変数名
		value	変数の値
配列操作	array	aname	配列名
		value	配列内の要素の値
構造体操作	struct	sname	構造体名
		mname	メンバ名
		value	メンバの値

SET で用いるタグと属性を表 3 に示す．ここで SET では，制御構造や関数などの制御に関するものをスコープとし，制御構造内部や関数内部での変数の変化などはスコープ内に含むとする．ここで，表 1 と表 2 におけるデータ value は，$<assign>$ 要素における内容の文字データとして扱う．

表 3 で定義したタグの関係は以下の関係式で定義することができる．

```
<scope> ::= {<scope> | <assign>}
```

表 3　SET で用いるタグと属性

タグ	意味	属性として用いるデータ
$< scope >$	スコープ	type,fname,in-args,out-args,in-global, out-global,return,ltype,cvalue,in-value, out-value,btype
$< assign >$	代入文	type,vname,aname,sname,mname

以上より，表 3 と上記の関係を用いて SET ではプログラム動作時の制御構造や関数呼び出しを構造化し表現する．例を用いてどのように表現するか示す．例えば，図 1(a) は関数 Func が関数 Proc から呼び出されているプログラムである．このプログラムから図 1(b) のような SET が生成できる．この SET では，まず関数 Proce があることを関数 Proc の $< scope >$ で表現している．また，関数 Proce の $< scope >$ 内において，関数 Func の $< scope >$ が存在することにより，関数 Proce が関数 Func が呼ばれていることを表現している．

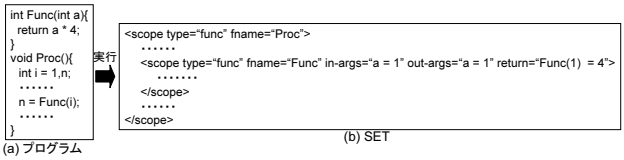

図 1　関数呼び出しに関する SET 生成例

次に，図 2(a) は関数 Proc 内に制御構造 while があるプログラムである．このプログラムから図 2(b) のような SET が生成できる．この SET では，関数 Proc の $< scope >$ 内において，制御構造 while の $< scope >$ が存在することにより，関数 Proc 内に制御構造 while があることを表現している．また，制御構造 while の $< scope >$ 内において変数 i に関する $< assign >$ が存在することにより，制御構造 while 内において変数 i の値が変更されていることを表現している．

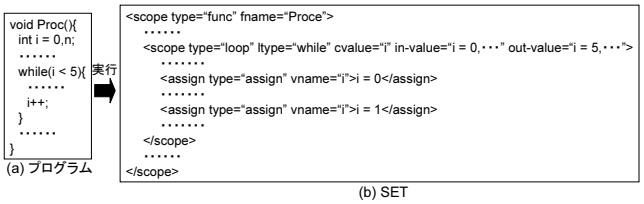

図 2　繰り返し構造と代入に関する SET 生成例

SET の具体例を以下に示す．以下の例では SET のタグを解釈し，実行時の構造をわかりやすい形としてインデントスタイル（以下 IS）で表現した形式で SET を表現する．

図 3(a) のプログラム p は，繰り返し構造 for において制御変数 i で繰り返しを制御しながら，関数 Keisan を繰り返し呼び出す．プログラム p より生成した SET は図 3(b) になる．この SET から，プログラムが繰り返し構造 for を使用し，制御変数 i を用いて繰り返し構造を制御していることと，for 文内のおいて関数 Keisan が繰り返し呼ばれていることが確認できる．また，指導者が設定した正解例プログラムの SET と答案プログラムの SET を比較することにより，制御変数 i の変化の形や関数 Keisan の呼び出し方の差異をシステムが自動的に見つけることが可能になる．この差異を受講者に提示することで，受講者が関数や制御構造の誤りを見つけることができるようになる．また，図 3(b) の SET より，変数 i の変化の様子を受講者

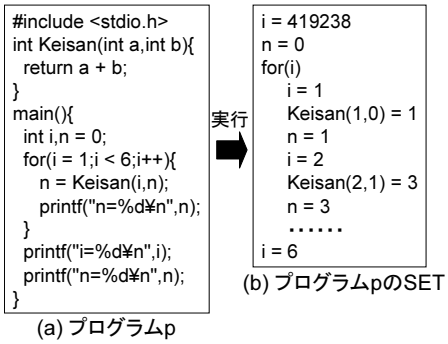

図 3　IS で表現した関数呼び出しと繰り返し構造に関する SET 生成例

に提示することが可能である．

図 4(a) のプログラム q は，繰り返し構造 while を 2 重に使用することで 2 次元配列の値を計算する．プログラム q より生成した SET は図 4(b) になる．この SET から，制御変数 i を使用した繰り返し構造 while の内側において制御変数 j を使用した制御構造 while があることが確認できる．また，内側の while 内において変数 n が変化していることが確認できる．このことにより，プログラム b において繰り返し構造を 2 重に用いることによって変数 n を変化させていることが確認できる．

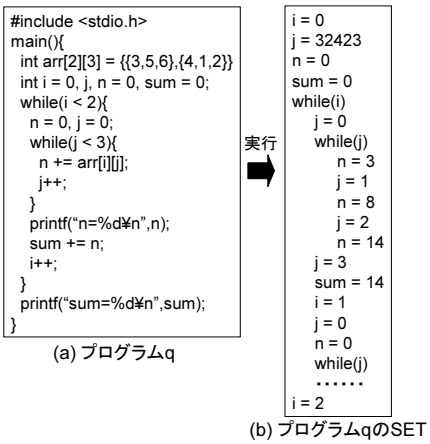

図 4　IS で表現した 2 重の繰り返し構造に関する SET 生成例

図 5(a) のプログラム r は，再帰関数を用いてフィボナッチ数を求める．プログラム r より生成した SET は図 5(b) になる．この SET から，関数 Fibo の内部でさらに関数 Fibo が呼ばれていることが確認できる．例えば，関数 Fibo の引数 n が 3 の時は引数 n が 2 の時の関数 Fibo と，引数 n が 1 の時の関数 Fibo が呼ばれていることが確認できる．このことより，このプログラムにおいて関数 Fibo が再帰構造を持つことが確認できる．

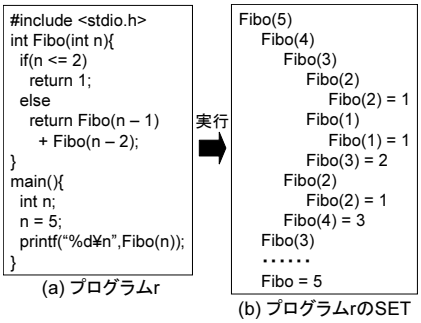

図 5　IS で表現した関数の再帰構造に関する SET 生成例

上記のまでの例のように，SET は実行時の変数の変化や制御

構造，関数呼び出しを構造化し表現したデータとなっている．

4. SET 生成機能の実現法

SET を生成する SET 生成機能の実現法について述べる．

SET 生成の手順を図 6 に示す．図 6 に示す順に STEP を実行することでプログラムより SET を生成する．ここで，検査シンボル定義表 (CSD) とはプログラム中の調査したい動作構造を指定する条件のことで，変数や関数，制御構造などを検査シンボルとして設定する．

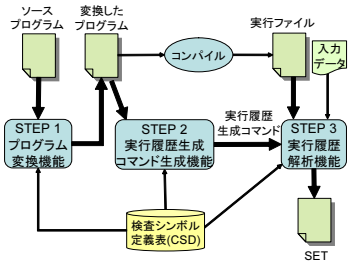

図 6　SET 生成機能の流れ

以下に，各機能の実現法の詳細について述べる．

4.1 プログラム変換機能

プログラムを文単位でトレース可能な形式としてアセンブラ対応形式に CSD を用いてプログラムを変換するプログラム変換機能の実現法について述べる．プログラム変換機能の実行手順を以下に示す．

STEP1-1　字句・構文解析

プログラムに対し C コンパイラと同様に字句解析と構文解析を行う．これにより，入力されたプログラムから変数や関数が定義されている行番地と変数名，関数名，制御構造の種類などを構文解析データとして抽出する．構文解析データ内のデータの主要なものを表 4 と表 5 に示す．この表 4 と表 5 のデータ項目を用いた表を解析結果として出力する．

例えば，図 7 では，プログラムから関数 main の中で int 型の変数 i が 6 行目に定義されていることを表す表を出力する．

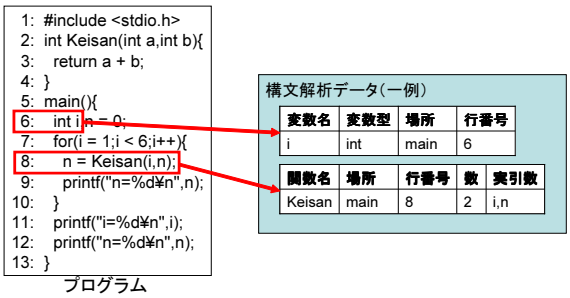

図 7　字句・構文解析の例

STEP1-2　文分割

プログラムの各行を分割して，全ての行が単一文になるようにプログラムを変換する．SET 生成機能では，プログラム動作時の変数の値などを取得するために行単位のデバッガを用いている．そのため，1 行に複数の文がプログラムに記載されている場合，正確に変数の変化などが取得できない可能性がある．取得不可を回避するために，プログラムの 1 行に複数の文がある箇所を 1 行ごとに分割することなどを行う．

表 4　制御に関する構文解析データ

データの種類	含まれる項目	項目の説明
関数定義情報	関数名	定義されている関数の関数名
	開始行	関数本体部の開始行番号
	終了行	関数本体部の終了行番号
	戻り値の型	戻り値の変数型
	引数の数	仮引数のして使用される変数の数
	引数の変数型	仮引数として使用される変数の変数型
	引数の変数名	仮引数として使用される変数の変数名
関数呼び出し情報	関数名	呼び出される関数の関数名
	場所	関数呼び出しがされている関数の関数名
	行番号	関数呼び出しが記載されている行番号
	引数の数	実引数として使用される変数の数
	引数の変数名	実引数として使用される変数の変数名
制御構造情報	種類	制御構造の種類
	制御変数	制御構造を制御する変数の変数名
	場所	制御構造がある関数の関数名
	開始行	制御構造を開始している行番号
	終了行	制御構造を終了している行番号
	条件式	制御構造の条件式
	初期化式	for における初期化式
	再初期化式	for における再初期化式

表 5　代入に関する構文解析データ（抜粋）

データの種類	含まれる項目	項目の説明
変数定義情報	変数名	定義されている変数の変数名
	変数型	定義されている変数の変数型
	場所	変数定義がされている関数の関数名
	行番号	変数定義が記載されている行番号
	仮引数	関数の仮引数となっているか

STEP1-3　制御構造変換

構文解析データと式分割を行ったプログラムを用いて，プログラム中に記載されている制御構造を分岐構造 if の形式などに変換する．変換ルールは文献 [6] に従い，以下の (1) から (3) に示す形式でプログラムを変換する．ただし，(2) の変換は，先に (1) における変換をしてから (2) の変換をする．

(1) 繰り返し構造の場合　繰り返し構造をラベルと if 文，ラベルにジャンプする goto 文を用いて表した形式に変換する．for の場合を例にとって述べる．図 8 のように変換する．なお，**Init** は初期化式で，**Update** は再初期化式を表している．**Body** 任意の C の文で，**Test** は条件式を表す．繰り返し構造 for の場合，一旦繰り返し構造 while の形式に変換した後，繰り返し構造 do-while の形式を経て分岐構造 if の形式にプログラムを変換する．

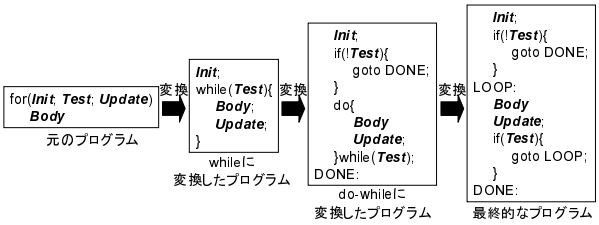

図 8　繰り返し構造 for の変換

(2)if 文の場合　図 9 のように変換する．ここで，**Type** は条件式 **Test** の評価値に応じた変数型を表している． if 文の場合，

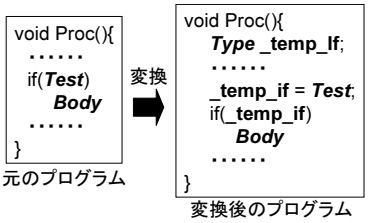

図 9　分岐構造 if の変換

条件式 **Test** を取り出し，**Test** の結果を別の変数に格納するプログラムに変換する．図 9 の場合，**Test** の値を変数`_temp_If`に格納している．なお，結果を格納する変数の定義も追加した形でプログラムを変換する．この変換により，条件式を評価したときの評価結果について調査可能となり，if 文で実際に分岐するか調査可能となる．

(3)return 文の場合　図 10 のように変換する．ここで，**exp** は戻り値の式を表している． return 文の場合分岐構造 if の変換

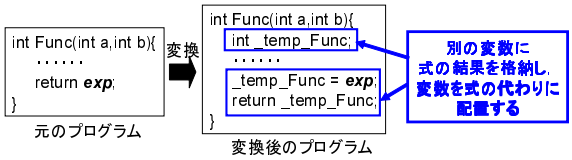

図 10　return 文の変換

と同様に，戻り値の式 **exp** をの結果を別の変数に格納するプログラムに変換する．図 10 の場合，**exp** の値を変数`_temp_Func`に格納している．この変換により，プログラムを動作させた時に呼び出された関数が戻る際の戻り値について調査可能となる．

STEP1-4　ダミー関数埋め込み

構文解析データと STEP1-3 で変換したプログラム，CSD を用いて，プログラムにダミー関数を埋め込む．ダミー関数とは，何も動作を行わない関数のことで，ダミー関数を制御構造や関数呼び出しの前後に埋め込むことにより，制御構造や関数呼び出しの実行前後における変数などのプログラムの状態が調査可能になる．構文解析データより関数呼び出しなどの位置を割り出しその前後にダミー関数をプログラムに埋め込む．そして，埋め込んだプログラムを用いて SET 生成する．例えば，図 11 では，ダミー関数`__break_Keisan_`と`__break_Keisan_end_`の定義文と，関数 Keisan の呼び出し前後に 2 つダミー関数の埋め込みを行っている．埋め込みにより，関数 Keisan の呼び出し前後における状態を調査することが可能となる．

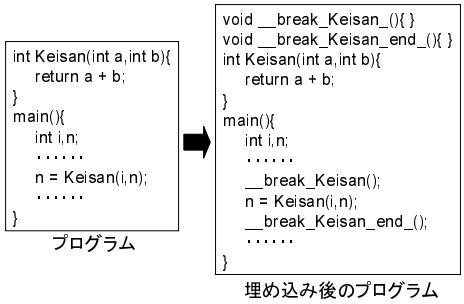

図 11　ダミー関数埋め込みの例

4.2　実行履歴生成コマンド生成機能

CSD とアセンブラ対応形式のプログラムを元に，実行履歴生成コマンドを生成する実行履歴生成コマンド生成機能の実現法について述べる．実行履歴生成コマンドとは，アセンブラ対応形式のプログラムを実行させて，CSD で指定された検査シンボルの種類に応じた値の出力を行うことを指定したデバッガのコマンド系列である．実行履歴生成コマンド生成の実行手順を以下に示す．

STEP2-1　字句・構文解析

アセンブラ対応形式のプログラムに対し，STEP1-1 で実行した字句・構文解析と同一の解析を行い，コマンド生成に必要な構文解析データをプログラムより取得する．

STEP2-2　コマンド生成

構文解析データと CSD を元に，実行履歴生成コマンドを生成する．構文解析データより CSD で指定されている検査シンボルに対応するデータを抽出する．抽出するデータは，検査シンボルが定義されている行番号などである．そして，抽出したデータを元に，実行履歴解析機能においてプログラムをデバッガ上で実行した際に，プログラムの実行を制御するコマンドや，ある時点での変数の値を出力するコマンドを組み合わせて実行履歴生成コマンドを生成する．例えば，図 12 の左にある変数 i に関する構文解析データと CSD の指定があったとする．この場合，プログラムの 6 行目以降の変数 i の値を調査することになるので，実行履歴生成コマンドは図 12 の右にあるものになる．この例では，break コマンドで 6 行目にブレークポイントを設定し，6 行目以降の変数 i の値を display コマンドを用いて出力するように指定している．

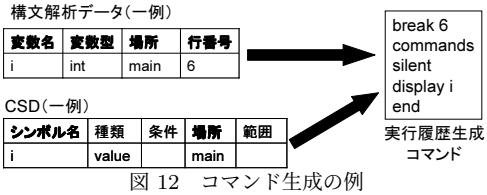

図 12　コマンド生成の例

4.3　実行履歴解析機能

実行履歴生成コマンドと入力データを用いてアセンブラ対応形式のプログラムを実行することで得られるデータを解析しプログラムの静的解析結果を組み合わせて XML 形式に構造化することで SET を生成する実行履歴解析機能を実現法について述べる．実行履歴解析機能の実行手順を以下に示す．

STEP3-1　デバッガ実行

実行ファイルと実行履歴生成コマンド，入力データの 3 つを用

いてデバッガ上でプログラムをトレース実行して検査シンボルの値を含んだ動的解析データを得る．ここで，動的解析データとは，実行履歴生成コマンドによってデバッガより出力されるCSD で指定した検査シンボルの値の変化を含んだプログラムの実行履歴である．例えば，図 13 のプログラムや変数 i に関して指定がある実行履歴生成コマンドを用いてデバッガ上でプログラムを動作させる．動作させるとデバッガより図 13 の右のような変数 i の実行した時の値が含まれた動的解析データを得る．

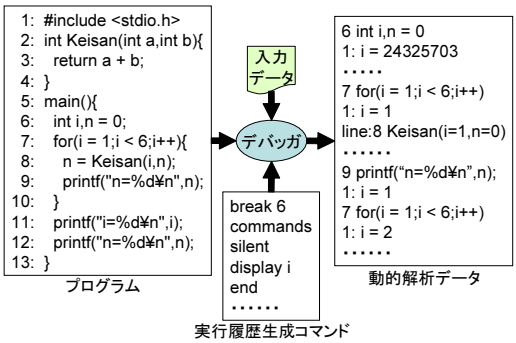

図 13　デバッガ実行の例

STEP3-2　SET 生成

STEP3-2 で得られた動的解析データより CSD を用いて CSD で指定された検査シンボルの値を抽出し SET を生成する．動的解析データより抽出した検査シンボルの値を集約し，プログラムの静的解析結果である構文解析データと組み合わせることで検査シンボルの値の変化と参照関係を構造化し SET を生成する．なお，構文解析データは，STEP1-1 で実行した字句・構文解析と同一の解析をアセンブラ対応形式のプログラムに対し行い取得する．例えば，図 14 の動的解析データと CSD があった場合，図 14 の右図のような SET を生成する．図 14 の CSD では変数 i と関数 Keisan に関して指定されているので，動的解析データより変数 i の変化した値と関数 Keisan が呼ばれた時の引数の値や戻り値などを抽出する．また，変数 i の値と変数 i に関する構文解析データとの結び付けを行う．そして，それぞれの値に対応するタグを付加することなどを行うことにより実行履歴を構造化し，SET を生成する．

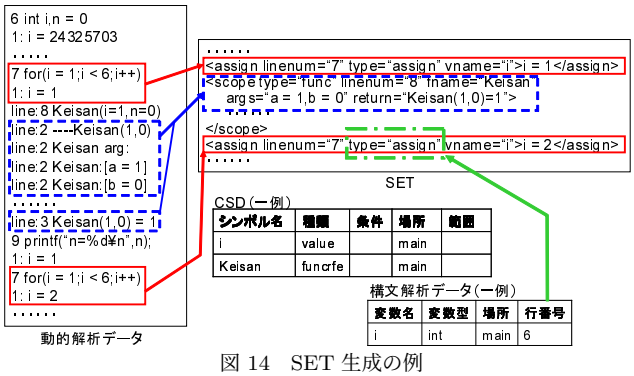

図 14　SET 生成の例

5.　構造化実行履歴のプログラミング演習支援システムへの適用

SET 生成機能を有するプログラミング演習支援システムの構成図を図 15 に示す．

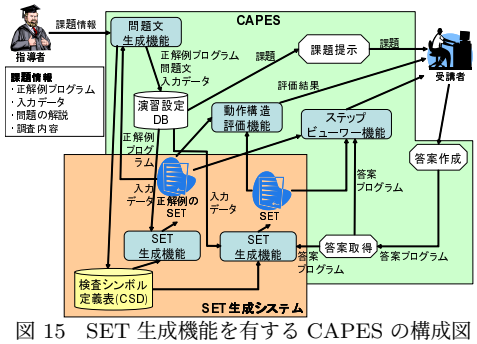

図 15　SET 生成機能を有する CAPES の構成図

SET 生成機能を含む CAPES の演習の流れを以下に示す．指導者は予め正解例プログラムや入力データなど課題として設定しておく．受講者は，CAPES から提示される課題を見て答案プログラムを作成し CAPES に提出する．CAPES は，答案プログラムと正解例プログラムの SET を生成する．

以下に問題文生成機能とステップビューワー機能，動作構造評価機能の実現法について以下に述べる．

5.1　問題文生成機能

正解例プログラムと調査項目，問題の説明を入力することで問題文を生成する問題文生成機能の実現法について述べる．指導者が正解例プログラムと，答案プログラムにおいて調査したい動作を構成する変数や制御構造，関数呼び出しなどの検査シンボルを機能に入力することで，指定された検査シンボルを答案プログラムの必要条件とすることを含んだ問題文のテンプレートを生成する．指導者は生成された問題文のテンプレートに問題の解説を加えるだけで，容易に調査したい動作のための必要条件を含んだ問題文を登録することが可能となる．また，入力した正解例プログラムより SET 生成機能で生成した SET を元に読解問題についても生成することが可能である．

5.2　ステップビューワー機能

答案プログラムより生成された SET と答案プログラムを用いて，受講者にプログラム実行時のトレースを提示するステップビューワー機能の実現法について述べる．SET を解析し，プログラムの各動作ステップ毎の変数の値や関数呼び出しなどを，該当するステップのソースコードと共に受講者に提示する．このとき，正解例プログラムより生成される SET と答案プログラムの SET を比較した結果を用いて，正解例プログラムと異なる箇所の提示も行う．SET の解析では，SET のデータとソースコードとの結び付けを行う．結びつける方法としては，SET のタグの属性である linenum を読み込むことにより，ソースコードとの結び付けを行う．linenum とは，変数の値などのデータが得られた時の答案プログラム上での行番号である．そして，結びつけた内容を時系列順に提示することで受講者に実行時のトレースを提示する．例えば図 16 の場合，変数 i に関する $< assign >$ における属性 linenum の値が 7 であるため，答案プログラムの 7 行目と変数 i に関する $< assign >$ との結び付けを行う．

5.3　動作構造評価機能

答案プログラムと正解例プログラムより生成されたそれぞれの SET と，評価したい構造と評価方法について記載した評価

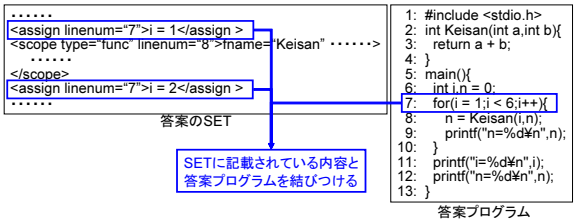

図 16　SET と答案プログラムの結びつけ

条件より，答案プログラムの動作構造について評価する動作構造評価機能の実現法について述べる．

この機能では，評価したい構造と評価方法について記述した評価条件を元に，答案プログラムと正解例プログラムの SET を比較することで，関数呼び出しの関係と順序や，制御構造と変数の変化の関係などの動作について評価を行う．評価条件は，答案プログラムにおいて評価したい（もしくは調査したい）構造と，SET を比較した際の動作構造に誤りがないと判定する評価方法の組で構成する．そして，評価結果を受講者に提示する．

6.　プロトタイプシステムの実装

提案方式により生成される SET の有効性を検証するためにプロトタイプシステムを実装し，演習支援システム CAPES に組み込みを行った．プロトタイプシステムでは，SET 生成機能とステップビューワー機能を実装した．プロトタイプシステムの実装には Java 言語 [7] を用いた．また，構文解析データや CSD などを格納するデータベースとして MySQL [8] を使用し，デバッガとしては GDB [9] を使用した．各種設定における Web ページの生成には Tomcat [10] を使用した．プロトタイプシステムを有する CAPES の構成を図 17 に示す．

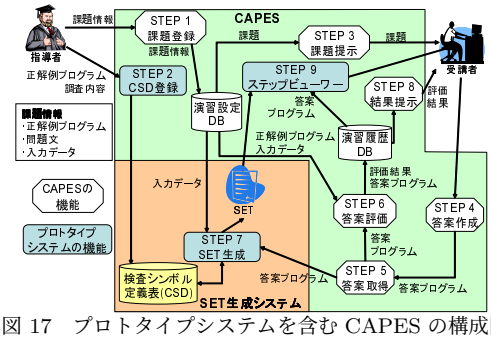

図 17　プロトタイプシステムを含む CAPES の構成図

プロトタイプシステムを有する CAPES では，図 17 の STEP で演習が行われることを想定して実装した．なお，演習設定 DB には，課題として受講者に提示される課題のタイトルや問題文，正解例プログラム，入力データなどを登録され，演習履歴 DB には，受講者が課題を提出した際の答案プログラムや答案提出日時，評価結果，答案を提出した受講者の情報などを登録される．

プロトタイプシステムでは正解例プログラムと調査項目を入力することで CSD を登録する CSD 登録機能を実装した (STEP2)．この機能では，正解例プログラムを Web ページに入力することで図 18 の Web ページより CSD を登録可能とした．図 18 では，入力された正解例プログラムに対して字句・構文解析を行うことで取得できる構文解析データを指導者に提示する．そして，指導者は提示されたデータを選択することで，簡単に CSD を登録可能としている．

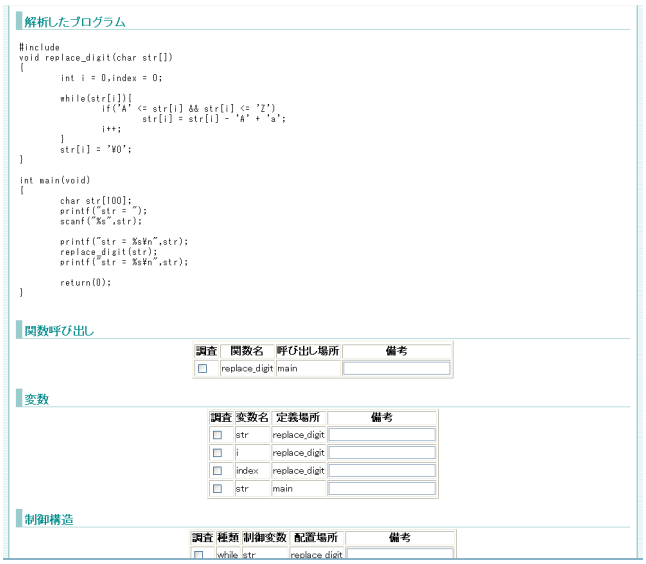

図 18　問題設定登録における CSD 登録ページ

実装したステップビューワーでは図 19 のようなインターフェースを用いて実行時のトレースを提示する． 図 19 のインターフェースでは答案プログラムと，現在のステップを実行する前の変数の値などを表示する．また，このインターフェースでは，以下のボタンを実装した．

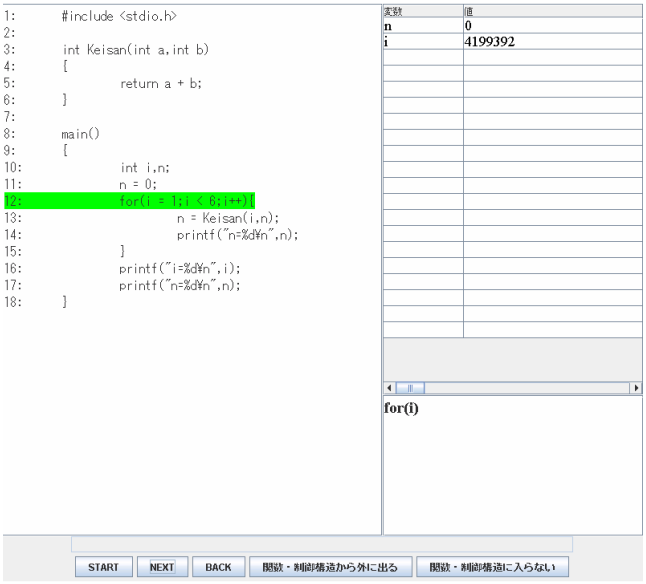

図 19　ステップビューワー

START ボタン　トレースを開始する．

NEXT ボタン　次のステップに移動する．

BACK ボタン　前のステップに移動する．

「関数・制御構造から外に出る」ボタン　現在いる関数や制御構造から一つ外に出て，現在いる関数や制御構造の最終ステップ直後まで移動する．

「関数・制御構造に入らない」ボタン　次のステップで入る関数や制御構造に入らず，次の関数や制御構造を飛ばして次のステップに移動する．

以上のボタンを実装することで受講者が自由にステップを動かしながら答案プログラムの動作を確認できるようにした．

7. 評　　価

7.1 実験項目

提案方式を実装したプロトタイプシステムを作成し以下の 2 つの項目に関して実験を行った．

(1) 課題の解答時間による評価

(2) アンケート調査による評価

7.2 実験方法

(1) 実験手順

手順 1 被験者は，読解問題を 2 題解く．このとき，1 題はステップビューワーを利用して，もう 1 題はステップビューワー以外の任意のツールを利用して解く．それぞれ正解に到達するまでの時間を測定する．

手順 2 被験者は，ステップビューワーに関するアンケートに回答する．

(2) 被験者のグループ分け

読解問題に対する慣れによる評価結果への影響を取り除くため，先に解く課題とステップビューワーを利用する課題を被験者毎に変える．両者の組み合わせより被験者 14 名を A から D の 4 グループに分ける．

7.3 実験結果と考察

(1) 課題の解答時間による評価の結果

A から D の各グループの解答時間の合計と全体の合計，平均，標準偏差を表 6 に示す．なお，規定時間（1 時間）に終らなかった 2 名の解答時間は結果より除いた．表中の丸数字は何問目にその読解問題を解いたかを表し，SV はステップビューワーを表す．例えば，表 6 より，グループ A は 1 問目に乗算プログラムに関する課題をステップビューワー以外の方法で解き，2 問目に英文処理プログラムに関する課題をステップビューワーを利用して解いたということになる．表 6 の平均時間より，各問題共にステップビューワーを利用した方が正解までにかかった時間が短くなったことが分かった．

表 6　解答時間の測定結果

	乗算プログラム		英文処理プログラム	
	SV利用なし	SV利用あり	SV利用なし	SV利用あり
A	① 2:12:46	-	-	② 0:40:25
B	② 1:10:09	-	-	① 1:11:20
C	-	① 1:06:07	② 1:14:44	-
D	-	② 1:16:29	① 1:24:11	-
合計	3:22:57	2:22:36	2:38:55	1:51:45
平均	0:33:50	0:23:46	0:26:29	0:18:38
標準偏差	0:12:35	0:08:49	0:14:21	0:09:33

(2) アンケート調査による評価の結果

アンケートの集計結果の一部を図 20 に示す．図 20 の Q1 より，86 %の被験者がプログラム理解にステップビューワーが役立った回答している．また図 20 の Q7 より，65 %の被験者がステップビューワーの方が他の方法よりも役立ったと回答していることが分かる．

上記の 2 つの評価結果より，ステップビューワーがプログラムの動作を把握し，プログラムを理解することに有効であると言える．

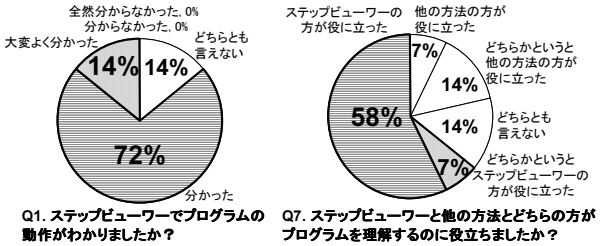

図 20　アンケートの集計結果（一部抜粋）

8. おわりに

本研究では，プログラミング演習支援システムにおける実行履歴の構造化方式と，SET を用いたアプリケーションの実現法について提案した．そして，プロトタイプシステムを実装し，ステップビューワーの有効性について検証した．今後は問題文生成機能と動作構造評価機能の実装と有効性について検証を行う予定である．

文　　献

[1] 中島秀樹，高橋直久，細川宜秀：プログラミング演習のための QA サイクル-受講者の習得度に応じた問題提示メカニズム-，電子情報通信学会論文誌，VOL.J88-D-I，NO.2，pp439-450(2005).

[2] 中島秀樹，宮地恵佑，高橋直久：プログラミング演習支援システム CAPES のための答案評価機構の実現，情報処理学会 研究報告，2006-CE-83(18)，pp127-134(2006).

[3] 岩間信介，高橋直久：プログラミング演習における答案診断のためのプロファイル生成システムの実現，FIT2007 第 6 回情報科学技術フォーラム，2007.

[4] 福安直樹，山本晋一郎，阿草青滋：細粒度リポジトリに基づいた CASE ツール・プラットフォーム Sapid，情報処理学会論文誌，Vol.39 No.6，pp1990-1998(1998).

[5] 谷口考治，石尾隆，神谷年洋，楠本真二，井上克郎：プログラム実行履歴からの簡潔なシーケンス図の生成手法，日本ソフトウェア科学会学会誌，Vol.24 No.3，pp153-169(2007).

[6] R.E.Bryant and D.R.O'Hallaron, "Computer Systems: A Programmer's Perspective", Prentice Hall 2002.

[7] JavaTM 2 Platform Standard Edition 5.0 API 仕様，http://java.sun.com/j2se/1.5.0/ja/docs/ja/api/index.html

[8] MySQL，http://www.mysql.com/

[9] Richard M.Stallman, Roland H.Pesch（コスモ・プラネット訳）：GDB デバッキング入門，アスキー，東京，1992.

[10] The Apache Software Foundation, "Apache Tomcat", http://tomcat.apache.org/