プログラムの動作の空間的表現手法の提案と読解問題作成への応用

肥田 悠汰† 立岩 佑一郎‡ 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: †yuta@moss.elcom.nitech.ac.jp, ‡{tateiwa, naohisa}@nitech.ac.jp

あらまし プログラミング演習の正誤判定や読解問題の作成では，プログラムの実行履歴を用いることにより高度な機能を実現できる．しかし，一般にデバッガやトレーサーなどの多くのツールでは，プログラムの実行履歴は実行される文と変数の値が時系列順に出力される．そのため，配列の値を直接変更したり，ポインタ変数を用いて間接的に変更したりするような配列の値の変化を調べたい場合には，配列名とポインタ変数名を用いて値をトレースする必要がある．また，繰り返し実行される文に関する実行履歴を得たい場合には，広範囲から対象の文に関する出力を探しまとめる必要がある．これらの問題点を解決するために，本稿ではプログラムの動作の空間的表現手法を提案し，その実現法について述べる．提案手法では，プログラムの実行履歴を文，アドレス，出現を軸とする3次元空間に展開して表現し，文の軸に射影したデータ構造(プロジェクションアレイと呼ぶ)に変換する．プロジェクションアレイ及び，その操作関数の実現上の特徴は次の通りである．(1) ポインタの値と変数の確保したメモリのアドレスから，ポインタ変数の参照先変数を求める機能を実現する． (2) プログラムの静的解析結果に基づいてデバッガの出力を解析し，文の実行順やメモリの値を文，出現ごとにまとめる機能を実現する．(3) プロジェクションアレイから変数の値の変化の系列や文の実行系列などを取得するための操作関数を実現する．
本稿では，さらにプロジェクションアレイを用いた読解問題作成について述べる．

キーワード E-Learning, 読解問題, 実行履歴

1. はじめに

指導者はC言語の初学者に対して，C言語の文法や関数の使用方法，アルゴリズムを学習させることを目的として，プログラムを作成する課題や，プログラムを読む課題を出題する．これらの課題では，課題の対象となるプログラムが必要となる．各課題に対して課題の対象とするプログラムを作成することは手間がかかるため，指導者は既存のプログラムを利用することがある．既存のプログラムから，学習させたい文法や関数を含んだプログラムを検索し，課題の対象とするプログラムとして利用する．しかし，学習させたい文法や関数の有無を検索条件として設定した場合，プログラムの動作が考慮されない．そのため，初学者が動作を理解することが難しく出題に適さないプログラムや，プログラムへの入力値によって学習させたい文法や関数が実行されないプログラムが検索結果に含まれるといった問題がある．指導者が初学者に対してforを学習させたい場合の検索結果の例を図 1(a)(b)に示す．

```
... ... ...
for(i = 0; i < N; i++) {
    for(j=0; j < N; j++) {
        for(k = 0; k < N; k++) {
            c[i][j] += a[i][k] * b[k][j];
        }
    }
}
... ... ...
```

(a) 行列の積の計算

```
... ... ...
scanf("%d", &n)
if(n > 0){
    for(i = 1; i <= n; i++) {
        factorial = factorial *  i;
    }
}
... ... ...
```

(b) 入力された変数の階乗の計算

図 1 検索されるプログラム例

図 1(a)はforを用いてN×N行列の積を計算するプログラムである．このプログラムは，学習させたい文法forを含んでいる．しかし，forが入れ子構造になっている．forをはじめて学習する初学者にとって，入れ子構造のあるプログラムの動作を理解することは難しい．また，Nの値が大きくなるとプログラムのステップ数が多くなり，プログラムの動作を追うことが難しくなる．そのため，図 1(a)のプログラムは初学者への出題に適さない．また，図 1(b)は，入力値nの値が正の場合，nの階乗値を計算するプログラムである．このプログラムは，学習させたい文法forを含んでいる．しかし，入力値nの値によってforの実行の有無が変化する．このプログラムを課題の対象とする場合，指導者がforが実行されるような入力値を考えなければならない．

これらの問題を解決するためには，プログラムに入力を与え実行時の動作を取得し，実行時の動作を検索条件とした検索を行う必要がある．あらかじめ取得した実行時の動作を検索条件とすることで，プログラムのステップ数や入れ子の有無などを検索条件とすることができる．また，検索結果として得られるプログラムに対して実行時の動作を得るために与えた入力値を与えることで，検索条件の文法や関数が確実に実行される．

プログラムを読む課題では初学者が間違えやすい部分やアルゴリズムの理解に必要な変数について読解問題を作成する．図 2(a)のソートアルゴリズムのプログラムでは，ソート対象の配列 score の値がアルゴリズムの理解に必要である．また，実行回数によって次に実行される文が変化する 23 行目の if 文の遷移先が間違えやすい部分となる．指導者は間違えやすい部分やアルゴリズムの理解に必要な変数の値を，プログラムの動作から取得する必要がある．

図 2(a)のプログラムに対して GDB[1]を用いて配列 score の値をトレースした実行履歴の一部を図 2(b)に示す．

```
... ... ...
12: void swap(int *x, int *y){
13:     int temp = *x;
14:     *x = *y;
15:     *y = temp;
16: }
... ...
19: void sort(int *data, int n){
20:     int i,k = n – 1;
21:     while (k >= 0){
22:       for (i = 1; i <= k; i++){
23:           if (data[i - 1] > data[i]) {
24:              swap(&data[i], &data[i-1]);
25:           }
26:       }
... ...
32: int main (void){
33:     int n = 5;
34:     int score[] = {65,74,32,56,94}
35:     sort(score, n);
36:     return(0);
37: }
```

(a) 配列をソートするプログラム

```
33: int n = 5;
    score = null
34: int score[] = {65,74,32,56,94}
35: sort(score,5)
    score = {65,74,32,56,94}
20: int i,k = n - 1;
    *data@5 = {65,74,32,56,94}
... ... ...
22: for (i = 1; i <= k; i++){
23: if(data[i-1]>data[i]){
24: swap(&data[i], &data[i-1]);
    *data@5 = {65,74,32,56,94}
... ... ...
22: for (i = 1; i <= k; i++){
    *data@5 = {65,32,74,56,94}
23: if(data[i-1]>data[i]){
24: swap(&data[i], &data[i-1]);
    *data@5 = {65,32,74,56,94}
... ... ...
22: for (i = 1; i <= k; i++){
    *data@5 = {65,32,56,74,94}
... ... ...
```

(b)scoreの値をトレースした実行履歴

図 2　プログラムと実行履歴

図 2(b)の実行履歴から，プログラムが開始されると main 関数 33 行目の int n = 5;が実行されることが分かる．また，35 行目の sort 関数呼び出し文実行前の配列 score の値が{65,74,32,56,94}であることが分かる．ここで，sort 関数呼び出し文の次に実行される文は sort 関数内の 20 行目の int i,k=n-1;である．そのため，配列 score の値をトレースしたい場合，ポインタ変数 data の参照先の値をトレースする必要がある．また，sort 関数内の 23 行目 swap 関数の呼び出し文は for,while によって繰り返し実行されるため，swap 関数の呼び出し文に関する出力が複数の箇所に出力されている．

このような実行履歴を用いて，プログラムの検索や読解問題の作成を行うシステムを考えた場合，以下の問題点がある．

問題点 1　配列名でトレースしても，ソート過程の配列の変化を得ることができない

問題点 2　繰り返し実行される文に関する実行履歴を得たい場合，広範囲から探しまとめなければならない

問題点 3　繰り返し実行される文の n 回目の実行履歴を得たい場合，n 回目まで出力を数え上げなければならない

本稿では，これらの問題点を解決するためにプログラムの実行履歴をアドレス，文，出現を軸とした 3 次元空間に展開したプログラムの動作の空間的表現手法を提案する．

2. 関連研究

文献[2]は，ソースコードのトレース結果にスコープ情報などを付与し，プログラムの構造化実行履歴を作成する手法を提案している．文献[2]は変数の値の変化に着目し，値に変化があった変数について，変数名，変数の値，変化があった行を 1 つの要素として格納している．これに対して空間的表現手法では，ポインタ変数により定義された変数名と異なる変数名でメモリの値が変更される可能性があるという点から，各変数のアドレス，ポインタ変数の値を求め，アドレスごとに値を配置する．

文献[3]は，プログラミング学習のためのソースコード読解問題類題生成システムを提案している．具体的には，ソースコードの静的解析結果から，問題作成可能な箇所を指導者に提示し,問題の作成を行っている．これに対して，問題作成には文の実行系列や，変数の値の系列が必要となるという点から，動的な解析結果を用いて空間的表現手法を作成する．

文献[4]は，初学者向け C プログラミング演習に用いられる記入式 Web 試験 DrilLs-F においての類題生成を支援する Web オーサリングツールについて記述している. DrilLs-F は初学者の C 言語の理解支援のため，ソースコードの一部を受講者に提示し，出力値などを回答させるソースコード読解問題の出題を行うことができるシステムである．文献[4]では入力データを与え，出力結果を問う問題に対する正答生成を行うプラグインの開発を行っている．このプラグインは，実際にプログラムの実行を行い複数行の出力結果から特定の行，フィールドを切り出すことで正答の生成を行う．これに対して，本稿では出力文以外の文の実行による変数の値を問う問題の作成が考えられるという点から，ソースコードのすべての文について変数の値を保持する．

3. プログラムの動作の空間的表現

図 2(b)の実行履歴の swap 関数, sort 関数の呼び出し文に，表 1 のような文番号を割り当て，変数 score に表 2 のようなアドレス識別子を割り当てた場合の 3 次元空間表現を図 3 に示す．ここで，文番号とはプログラムに記述されている文に対して，記述されている順番に割り当てられる番号である．この文番号が文軸の目盛りとなる．また，アドレス識別子とは実際の変数が確保したメモリのアドレスを一意に識別可能な識別子である．このアドレス識別子がアドレス軸の目盛り

となる.出現軸は文の何度目の出現かが目盛りとなる.

表 1 文番号

文番号	文
9	swap(&data[i],&data[i-1])
13	sort(score,n)

表 2 アドレス識別子

アドレス識別子	変数名
1	score[0]
2	score[1]
3	score[2]
4	score[3]
5	score[4]

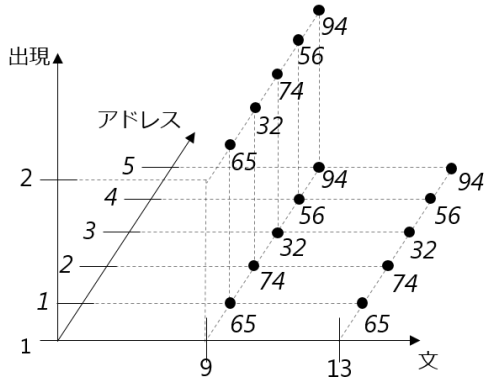

図 3 3 次元空間表現

図 2(b)の実行履歴において，sort 関数呼び出し文での配列 score の値は{65,74,32,56,94}である．そのため，文 13,出現 1 の直線とアドレス軸 1,2,3,4,5 の交点に，65,74,32,56,94 が配置される．1 度目の swap 関数の呼び出し文では，data の参照しているメモリの値が sort 関数呼び出し文と同じ値である．そのため，文 9,出現 1 の直線とアドレス 1,2,3,4,5 の交点に，65,74,32,56,94 が配置される．しかし，2 度目の swap 関数の呼び出しでは，score[1]と score[2]の値が変更されているため，文 9,出現 2 の直線とアドレス 1,2,3,45 の交点に，65,32,74,56,94 が配置される．

3.1 文への射影

3 次元空間表現を用いて，プログラム検索や読解問題作成を行う場合，文の動作などプログラムの部分的な動作を 3 次元空間から得ることに手間がかかる．そこで，3 次元に配置されている値を，文に射影したデータ構造プロジェクションアレイ(PA)を作成する．PA を作成するために，文 s,出現 o の直線に配置されているアドレスの値と，文の実行系列をもとに文,出現ごとの実行履歴を作成する．文，出現ごとの実行履歴の要素を表 3 に示す．

表 3 文，出現ごとの実行履歴の要素

キーワード	値
backsequence	1 つ前に実行される文の実行履歴へのポインタ
sequence	次に実行される文の実行履歴へのポインタ
value	ステップ実行前後の変数の値 ポインタ変数が参照するアドレスのメモリを確保した変数名

表 3 の要素をもつ文，出現ごとの実行履歴を文に射影することで文単位でプログラムの動作を表現するプロジェクションアレイを作成する．

4. 配列とポインタ変数の関係

ポインタ変数の値が配列の先頭アドレスの場合，ポインタ変数の値を変更せず，配列の各要素の値を扱うことができる．図 4 に例を示す．

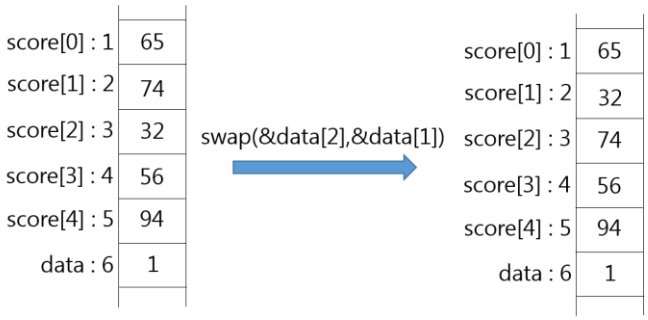

図 4 ポインタと配列の関係

図 4 のように，メモリのアドレス 1 から 5 が配列 score に，アドレス 6 がポインタ変数 data に割り当てられている場合，swap(&data[2],&data[1])とすることで，ポインタ変数の値を変更することなく，アドレス 2 とアドレス 3 の値を変更することができる．

そのため，ポインタ変数の値が配列の先頭アドレスである場合，ポインタ変数の値のメモリに格納されている値だけでなく，配列の各要素の値を求めなければならない．

5. 提案システム

5.1 提案システムの特徴

提案システムは以下の特徴を持つ．

特徴 1 配列とポインタの関係性解析

変数が確保しているメモリのアドレス，ポインタ変数の値からポインタ変数が参照する変数名を求め，ポインタと配列の関係性を解析する機能を実現する．この機能により，ポインタ変数を用いて配列要素の値を変更した場合でも，その値を配列要素のアドレスの位置に配置できる．

特徴 2 静的解析結果に基づくデバッガ出力の解析

静的解析結果に基づいてデバッガの出力を解析し，文の実行順やメモリの値を文，出現ごとにまとめる機

能を実現する．この機能により，文，出現ごとの実行履歴を作成できる．

特徴 3 PA の操作関数の実現

PA から変数の値の変化の系列や文の実行系列などを取得するための操作関数を実現する．これにより，PA から容易にプログラムの動作を得ることができる．

5.2 提案システムの構成

提案システムは図 5 のように，ポインタ変数参照先解析機能，実行履歴作成機能，PA 作成機能からなる．ポインタ変数参照先解析機能は，ポインタ変数の参照先変数名を求め，ポインタ変数の配列の関係性を求める機能である．実行履歴作成機能では，変数の値，文の実行系列をまとめ，文，出現ごとの実行履歴を作成する機能である．PA 作成機能では，文，出現ごとの実行履歴とプログラムの静的解析結果を組み合わせ，PA を作成する機能である．

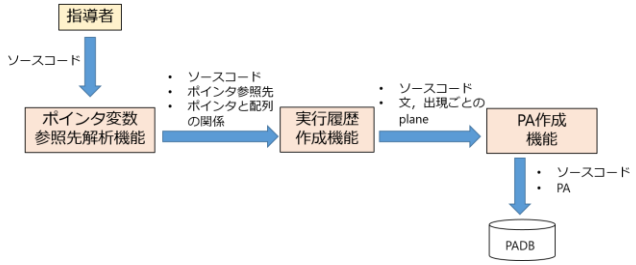

図 5 提案システムの構成図

提案システムを用いて，PA を作成する場合，指導者は PA を作成したいソースコードをシステムに入力する．提案システムは，ポインタ変数参照先解析機能により，変数が確保したメモリのアドレス，ポインタ変数の値を求め，比較することでポインタ変数の参照先を決定する．また，ポインタ変数の参照先を用いて，ポインタ変数と配列の関係を作成する．ソースコード，ポインタ変数の参照先，ポインタ変数と配列の関係を実行履歴作成機能に入力する．実行履歴作成機能では，変数の値，ポインタ変数の参照先のメモリの値を求め，文，出現ごとにまとめることで文，出現ごとの実行履歴を作成する．作成した，出現ごとの実行履歴とソースコードを PA 作成機能に入力する．PA 作成機能では，文，出現ごとの実行履歴とソースコードの静的解析結果を組み合わせ PA を作成する．

5.3 提案システムの機能の実現法

5.3.1 ポインタ変数参照先解析機能

ポインタ変数の参照先変数名と，ポインタ変数と配列の関係をデバッガの出力より求める．求める手順を以下に示す．

手順 1) ソースコードの静的解析結果から，ポインタ変数の値，変数が確保しているメモリのアドレスの値を出力するデバッガコマンドを作成する

手順 2) デバッガを実行することで，ポインタ変数の値，変数が確保しているメモリのアドレスに関するデバッガの出力を得る

手順 3) デバッガの出力から，ポインタ変数の値，変数が確保しているメモリのアドレスの値を求める

手順 4) ポインタ変数の値と変数が確保しているメモリのアドレスの値を比較し，ポインタ変数が参照する変数を求める．

手順 5) ポインタ変数の参照先から，ポインタ変数と配列の関係性を求める．

図 2(a)のプログラムを例に説明する．手順 1 により，図 6 のようなデバッガコマンドが作成される．

```
break qu.c:sort    # sort関数のbreakpoint
commands
display data    # ポインタ変数dataの値を出力
end
break qu.c:main   # main関数のbreakpoint
commands
display &score    # 配列scoreのアドレスを出力
end
…
```

図 6 デバッガコマンドの一部

手順 2 により，図 7 のようなデバッガの出力が得られる．

```
… … …
35 sort(score,5)
    &score = 0x28abe4
20 int i,k = n – 1;
    data = 0x28abe4
… … …
```

図 7 デバッガの出力の一部

手順 3 により，表 4,5 のようなポインタ変数の値と，変数が確保しているメモリのアドレスが得られる．

表 4 ポインタ変数の値

ポインタ変数	値
data	0x28abe4

表 5 変数が確保しているメモリのアドレス

変数名	アドレス
score[0]	0x28abe4
score[1]	0x28abe8
score[2]	0x28abec
score[3]	0x28abf0
score[4]	0x28abf4

手順 4 により，ポインタ変数 data が参照する変数が score[0]であることが分かる．手順 5 により，ポインタ変数 data は配列を参照していることが分かる．

5.3.2 実行履歴作成機能

ポインタ変数と配列の関係を用いて，各変数のアドレスの値を出力させるようなデバッガコマンドの作成を行う．ポインタ変数と配列の関係によって，ポインタ変数が指すアドレスの値を出力させるデバッガコマンドが異なる．ポインタ変数と配列の関係ごとのデバッガコマンドを，表 6 に示す．

表 6 ポインタ変数 p に対するデバッガコマンド

ポインタ変数の参照先	デバッガコマンド
配列以外	display *p
配列	display *p@配列の長さ

ポインタ変数が配列以外を参照している場合は*をつけるだけで，アドレスの値が得られる．しかし，ポインタ変数が配列を参照している場合，ポインタ変数名の後に配列のサイズを指定する必要がある．これにより，ポインタ変数が配列を参照する場合も，配列の値すべてを得ることができる．

作成されたデバッガコマンドを用いて，デバッガを実行することで図 2(b)のようなデバッガの出力が得られる．デバッガの出力を各文の出現ごとにまとめることで，文，出現ごとの実行履歴を作成する．

5.3.3 PA 作成機能

ソースコードを解析することで，各文の行番号や関数名などの解析結果を取得する．取得した解析結果と，文，出現ごとの実行履歴をまとめることで PA を作成する．

5.4 PA 操作関数

プロジェクションアレイからプログラムの動作の一部を取得するための PA 操作関数を実現する．実現した操作関数を以下に示す．

変数の値の変化の系列を抽出する関数

PA から変数の値を取得し変数の値の変化の系列として，変数の値，変数の値が変化しているステップの文，出現を抽出する．入力として，PA，変数の値の変化の系列を抽出したい変数名と変数が宣言されている関数名を入力する．図 2(a)のプログラムの PA と変数名 score，関数名 main を入力した場合の出力を表 7 に示す．

表 7 score の値の変化の系列

文	出現	score の値
int score[] = {65,74,32,56,94}	1	{65,74,32,56,94}
swap(&data[i], &data[i-1]);	1	{65,32,74,56,94}
swap(&data[i], &data[i-1]);	2	{65,32,56,74,94}
swap(&data[i], &data[i-1]);	3	{32,65,56,74,94}
swap(&data[i], &data[i-1]);	4	{32,56,65,74,94}

ポインタ変数に参照される変数名を抽出する関数

PA からポインタ変数が参照している変数名を取得し，ポインタ変数に参照されている変数名，ポインタ変数に参照されている変数が宣言されている関数名，参照に用いられているポインタ変数名，参照に用いられているポインタ変数が宣言されている関数名を抽出する．入力として PA を入力する．図 2(a)のプログラムの PA を入力した場合の出力を表 8 に示す．

表 8 ポインタ変数に参照される変数名

変数名	変数の関数	ポインタ	ポインタの関数
score	main	data	sort
score[1]	main	y	swap
score[2]	main	x	swap
score[2]	main	y	swap
score[3]	main	x	swap
score[0]	main	y	swap
score[1]	main	x	swap

文の動作を抽出する関数

PA から文の出現ごとの次のステップで実行される文，出現，文実行前後の変数の値を取得する．入力として PA と文番号を入力する．図 2(a)のプログラムの PA と文番号 9 を入力した場合の出力を表 9 に示す．

表 9 文の動作(*data の値)

出現	次の文	次の出現	実行前	実行後
1	1	1	{65,74,32,56,94}	{65,32,74,56,94}
2	1	2	{65,32,74,56,94}	{65,32,56,74,94}
3	1	3	{65,32,56,74,94}	{32,65,56,74,94}
4	1	4	{32,65,56,74,94}	{32,56,65,74,94}

6. PA を用いた読解問題作成システム

指導者は，ポインタ変数の参照による値の変更などの学習者が間違えやすいプログラムの動作や，ソートアルゴリズムにおけるソート対象の配列の値などのアルゴリズムの理解に必要なプログラムの動作について，読解問題を作成する．本稿では，図 8 のように PA の操作関数を用いて読解問題作成に必要なプログラムの動作を PA から取得し，指導者に提示する読解問題作成システムを実現した．

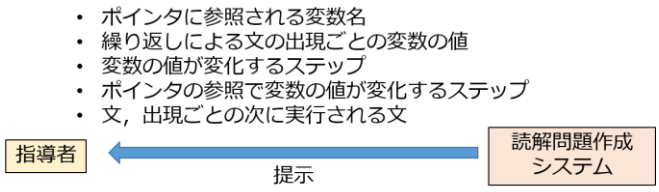

図 8 読解問題作成システム

7. プロトタイプシステム

提案したプロジェクションアレイ作成システムと読解問題作成システムのプロトタイプシステムの開発を行った．プロトタイプシステムは Java[5]を用いて実装した．

8. 評価実験

PA は各文についてその文が含まれるスコープ中のすべての変数の値をすべてのステップで保持している．そのため，PA のデータ量は大きくなると考えられる．そこで，PA 作成システムのプロトタイプシステムを用いて，PA を作成し作成された PA のデータ量を調査することで，PA のデータ構造を用いてプログラムの動作を表現可能かどうかを検証した．PA を作成する対象として，名古屋工業大学のプログラミン

グ演習で用いられている演習支援システムに登録されている 187 組のソースコードと入力値の組を用いた．PA のデータ量を図 9 に示す．

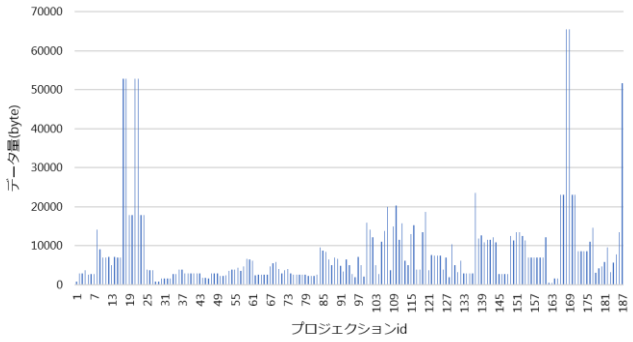

図 9　PA のデータ量

PA のデータ量は最大でも 65418byte となった．この PA が作成されたソースコードは，1 つの int 型変数を用いて 100 以下の奇数を表示するプログラムで，254 ステップで実行が終了するプログラムであった．このようなプログラムでも，データ量は Mbyte にも達していない．そのため，演習で用いられるようなソースコードと入力値に対して，提案手法を用いてソースコードの動作を表現できることが分かった．

9. おわりに

本稿では，プログラムの実行履歴を文，アドレス出現を軸とする 3 次元空間に展開して表現し，文の軸に射影したデータ構造 PA を作成するシステムの実現法を述べ，プロトタイプを作成した．今後は，プログラム検索や正誤判定などの応用システムを開発する．

参　考　文　献

[1] GDB: The GNU Project Debugger, ” https://www.gnu.org/software/gdb/”

[2] 岩間信介，立岩佑一郎，山本大介，高橋直久，“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", DEIM2009，2009.

[3] 吉田拓己，立岩佑一郎，山本大介，高橋直久，”プログラミング学習のためのソースコード読解問題類題生成システムの実現”，情報処理学会第 74 回全国大会講演論文集 2012(1), pp.743-744, 2012.

[4] 国信真吾，富永浩之，林敏浩：記入式 Web 試験 DrilLs における C プログラムのソースコード読解問題 - 類題生成を支援する Web オーサリングツールの開発 - , 電子情報通信学会技術研究報告. ET, 教育工学 108(470), 217-222, 2009-02-28

[5] Java , https://java.com/ja/