読解問題作成のためのプログラム動作表現 プロジェクションアレイの実現

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 プログラムの動作内容を問う問題(以降，読解問題と呼ぶ)には，文が実行される前後の変数の値を問う問題や，変数間の値関係を問う問題，文の実行順を問う問題などがある．読解問題の作成では，実行時の変数の値の変化や文の実行順序などを知る必要がある．一方，デバッガの出力から，プログラムの実行順序に沿って各文に含まれる変数の値が得られる．そのため，デバッガを用いて各文の実行前後の変数の値を取得したい場合には，その変数に係る複数の文からの出力をまとめる必要がある．また，文の実行順を取得したい場合には，すべての文の出力から実行順を求めなければならない．そこで，本稿では以下の特徴をもつ読解問題作成のためのプログラム動作表現プロジェクションアレイを提案し，その実現法について述べる．本稿では，また，プロジェクションアレイを用いた読解問題作成システムについて述べる．プロジェクションアレイは次の 3 つの特徴を持つ．(1)プログラム中の各実行文に対して，その文の前後の文のデバッガからの出力を抽出し，各文の実行前後の変数の値の変化を表現．(2)デバッガから出力される文の順番を求め，次に実行される文へのポインタを作成して文の実行系列を表現．(3)デバッガの出力を制御構造ごとにまとめて繰り返し制御などの実行動作を表現

キーワード E-Learning, 読解問題, 実行履歴

1. はじめに

学習者がプログラミング演習を円滑に進められないことの一因として，プログラムに対する読解力不足が挙げられる．読解力が不足していると，プログラム実行時の変数の値の変化や，分岐などを理解することができない．そのため，学習者は読解力を身につけることが求められる．この読解力を向上させることを目的として，指導者は読解問題を用いる．

読解問題には，文が実行される前後の変数の値を問う問題や，変数間の値関係を問う問題，文の実行順を問う問題などがある．図 1 が含まれるプログラムに対する読解問題の例を図 2(a),(b)に示す．

```
... ... ...
18:  void sort(int data[], int n){
19:     int  k = n - 1;
20:     int  i, j;
21:     while (k >= 0){
22:        for (i = 1, j = -1; i <= k; i++){
23:           if (data[i - 1] > data[i]) {
24:              j = i - 1; swap(&data[i], &data[j]);
25:           }
26:        }
27:        k = j;
28:     }
29: }
... ... ...
```

図 1 プログラムの一部

i = 3,k = 4の場合について，
24行目のswap(&data[i], &data[j])実行前後の配列dataの値を答えよ

(a) 変数の値の変化を問う問題

i = 2,k = 3の場合について，
23行目のif(data[i-1]>data[i])実行後に実行される文は何行目か答えよ

(b) 文の実行順を問う問題

図 2 問題例

指導者が図 2(a)(b)のような読解問題を作成する場合，各文の実行時点での変数の値や文が実行される順番を取得しなければならない．取得するための方法として GDB[1]等のデバッガを用いる方法が挙げられる．デバッガとは，プログラムのバグの発見や修正を支援するソフトウェアであり，プログラムを一行ずつ動作確認しながら実行するステップ実行機能や，ステップ実行中のメモリや変数などの値を表示することでプログラムの実行過程を調べるトレース機能がある．ここで，トレース機能により表示されるメモリや変数の値は，ステップ実行により実行される行が実行される直前の値である．指導者はこれらの機能を用いることで，文の実行順や変数の値を取得することができる．図 1 が含まれるプログラムに対して，GDB を用いたトレース結果の一部を図 3 に示す．

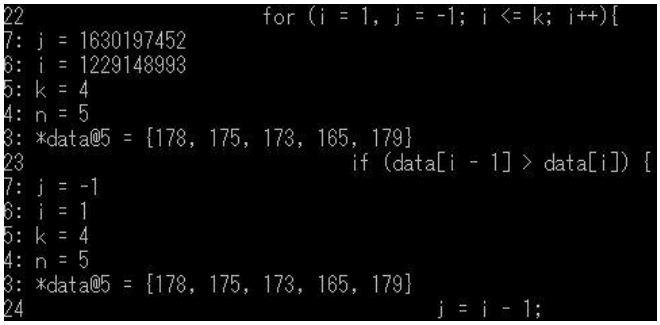

図 3 GDB を用いたトレース結果の一部

ステップ実行機能により，22 行目，23 行目，24 行目とプログラムが実行されていくことが分かる．また，トレース機能により 22 行目が実行される前の変数の値，23 行目が実行される前の変数の値が表示される．指導者がこのようなデバッガの出力から，読解問題を作成する場合，以下のような問題点がある．

問題点 1 文実行後の変数の値を取得するために，複数の文からのデバッガの出力をまとめなければならない

問題点 2 すべてのステップの出力から文の実行順を取得することに手間がかかる

問題点 3 デバッガの出力から制御構造の範囲を求めることに手間がかかる

本稿ではこれらの問題点を解決するためにデバッガの出力を各文についてまとめることで変数の値の変化を表し，文と文をポインタでつなぐことで文の実行系列を表したプロジェクションアレイの作成システムを提案する．また今回はデバッガに GDB を用いる．

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．これに対して提案するプロジェクションアレイでは，プログラムの流れを問う問題や変数間の関係を用いた問題を作成するという点から，文ごとに関数で使用されているすべての変数の値を格納する．

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，問題が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，変数間の関係や実行される文の流れを用いて問題作成可能な箇所を定める場合があるという点から，プロジェクションアレイは，変数の値や文の遷移先といった動的情報を持つ．

3. プロジェクションとプロジェクションアレイ

3.1 プロジェクション

プロジェクションとは各文の動作を表したデータである．また，プロジェクションには 1 つの文の動作をまとめた文のプロジェクションと，制御構造をまとめた制御構造のプロジェクションがある．プロジェクションのデータ構造を図 4 に示す．

```
[ "statement" : 文番号, "line" : 行番号, "location" : 行の何番目の文か,
  "code" : 文, "var" : 変数名リスト, "iteration" : 文実行回数,
  "function" : 関数名, "plane" : 各実行時の変数の値と次に実行される文]
```

図 4 プロジェクションデータ構造

プロジェクションは文の実行で変化が起きるデータと，文の実行で変化が起きないデータで構成される．表 1 に，プロジェクションの構成要素を示す．

表 1 プロジェクションの要素

キーワード	値	変化の有無
statement	文番号	無
line	行番号	無
location	行の何番目の文か	無
code	文	無
var	変数名リスト	無
iteration	文実行回数	無
type	タイプ	無
function	関数名	無
plane	変数の値と次に実行される文へのポインタ	有

ここで，変数名リストはその文を含むスコープで定義された全ての変数の変数名を格納する．また，ポインタ変数が定義されている場合，ポインタを何回辿れば，変数の値にたどり着くかを示すために，ポインタ変数に*をつけ，変数名リストに格納する．例えば，ポインタ変数 a がポインタであり，ポインタ変数 b がポインタのポインタであった場合，変数名リストには*a と**b が格納される．また，変数名リストには配列や構造体の変数名も格納される．変数名リストに格納されたすべての変数について文実行前後の値を plane に格納する．タイプは，1 つの文のプロジェクションと制御構造のプロジェクションを区別するために用いる．また，文の実行回数により変化する変数の値と次に実行される文について，文の何度目の実行かを区別するために plane を配列形式のデータとする．この配列の第 n 要素が n + 1 回目の実行時の変数の値と次に実行される文へのポインタとなる．plane の定義を図 5 に示す．

```
"plane":([ "value" : 文実行前の値と文実行後の値, "sequence" : (次の実行文)])
```

図 5 plane 定義

plane は文実行前の値と文実行後の値，次の実行文へのポインタを持つ．ここで，制御構造のプロジェク

ションの場合，次の実行文は制御構造内の文へのポインタと，制御構造終了後の文へのポインタの 2 種類のポインタを持つ．そのため，sequence 要素は配列形式のデータとする．この 2 種類のポインタにより，制御構造の範囲が取得可能になる．

図 1 の 24 行目の swap(&data[i],&data[j])に対するプロジェクションの plane の第 0 要素を図 6 に示す．

```
{ "value" : { "n":["5","5"] , "data" : ["&height", "&height"],
              "*data":["{178, 175, 173, 165, 179}","{175, 178, 173, 165,179}"],
              "j" : ["0","0"] , "k" : ["4","4"] , "i" : ["1","1"]} ,
  "sequence":[{"statement":"0","planeid":0}]}
```

図 6 plane の例

図 6 のように，文実行前後の値を value に，次の実行文へのポインタとして文番号と配列の第何要素かを sequence に格納する．また，value には，その関数で使用されているすべての変数の値と，ポインタが指している変数名を格納する．図 6 の plane では，変数 data がポインタ変数であるため，data にポインタ変数が指している変数名の変化，*data にポインタ変数が指している変数の値の変化が格納されている．

3.2 プロジェクションアレイ

プロジェクションアレイはプロジェクションと，プログラムで 1 番最初に実行される文，1 番最後に実行される文から構成される．1 番最初に実行される文のプロジェクションから次に実行される文のプロジェクションへとたどり続けることで，プログラムのトレースが可能となる．

4. 提案システム

4.1 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 文の実行前後の変数値を文と対応付ける

デバッガから出力された変数の値を文ごとにまとめ，文実行前後の変数の値を作成する．これにより，文の実行による変数の値の変化が容易に取得可能になる．

特徴 2 ポインタ変数の参照関係の作成

デバッガから出力された変数のアドレスをまとめ，ポインタ変数がどの変数を指しているかという参照関係を作成する．ポインタ変数の参照関係を用いることで，ポインタ変数がさしている変数の値の変化の取得や，領域外へのアクセスの検知が可能になる．

特徴 3 制御構造のプロジェクション

制御構造内部にある文のプロジェクションは，制御構造のプロジェクションから遷移する．また，制御構造から出る場合は，制御構造の内部の文のプロジェクションから,制御構造のプロジェクションに遷移する．これにより，制御構造のスコープや条件式の真偽が取得可能になる．図 1 の 23 行目の if 文のプロジェクションからの遷移を図 7(a),(b)に示す．

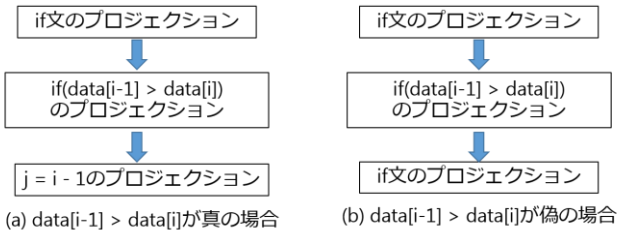

図 7 プロジェクションの遷移

図 7(a)は if 文の条件式が真の場合，(b)は if 文の条件式が偽の場合である．真の場合，制御構造全体の if 文のプロジェクションから，if 文の 1 行目のプロジェクションに遷移し，j = i − 1 のプロジェクションに遷移する．偽の場合は，制御構造全体の if 文のプロジェクションから，if 文の 1 行目のプロジェクションに遷移し，再び制御構造全体の if 文のプロジェクションに遷移する．条件式があるプロジェクションの遷移先が，制御構造全体のプロジェクションとなっているかどうかで，条件式の真偽が判断できる．

4.2 提案システムの構成

提案システムは図 8 のように，文分割機能，コマンド作成機能，デバッガ実行機能，参照関係作成機能，結合機能，実行系列作成機能，プロジェクションアレイ作成機能からなる．文分割機能は，入力されたソースコードを 1 行に 1 文となるように分割する機能である．コマンド作成機能は，変数のアドレスを出力させる GDB のコマンドファイルと，変数の値を出力させるコマンドファイルを作成する機能である．デバッガ実行機能は，作成されたコマンドファイルを用いて，GDB を動作させ GDB の出力を取得する機能である．参照関係作成機能は，GDB から出力された変数のアドレスを用いて，どの変数がどの変数を指しているかといった変数の参照関係を作成する機能である．結合機能は，GDB から出力された変数の値を用いて，文実行前後の変数の値を作成する機能である．実行系列作成機能は，GDB から出力される実行文を用いて，現在実行中の文から次に実行される文へのポインタを作成し，プログラムの実行系列を作成する機能である．プロジェクションアレイ作成機能では，変数の参照関係，文実行前後の値，実行系列をまとめて，文ごとにプロジェクションを作成し，プロジェクションアレイを作成する機能である．

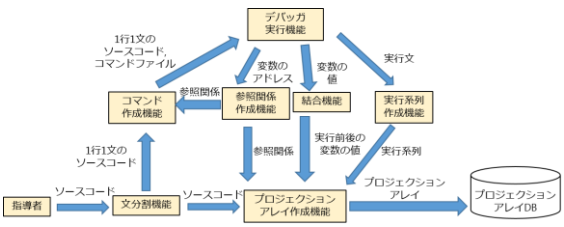

図 8 提案システムの構成図

提案システムを用いて，プロジェクションアレイを作成する場合，指導者は，プロジェクションアレイを作成したいソースコードをシステムに入力する．提案システムは文分割機能により，ソースコードを 1 行に 1 文になるように分割する．分割されたソースコードは，コマンド作成機能とプロジェクションアレイ作成機能に入力される．コマンド作成機能では，まずソースコードの解析結果のみを用いて，各ステップで各変数のアドレスが出力されるような GDB のコマンドファイルを作成する．作成したコマンドファイルを用いて，デバッガ実行機能により GDB を動作させる．デバッガの出力として，変数のアドレスを参照関係作成機能に入力する．参照関係作成機能により，各ステップでのポインタ変数がさす変数を求め，どの時点でどの変数を指していたかという参照関係を作成する．参照関係はコマンド作成機能とプロジェクションアレイ作成機能に入力される．再度コマンド作成機能により，ソースコードの解析結果と参照関係を用いて，各ステップで各変数の値が出力されるような GDB のコマンドファイルを作成する．同様にデバッガ実行機能により，GDB を動作させ変数の値を出力させ結合機能に入力する．このとき，変数の値と同時に各ステップで実行される文が GDB により出力されるので，実行文を実行系列作成機能に入力する．結合機能により，変数の値をまとめ各文実行前後の変数の値を作成する．実行系列作成機能により，実行される文の順番を求め，文から文へのポインタを作成することで実行系列を作成する．これらをプロジェクションアレイ作成機能に入力することで，各文のプロジェクションを作成し，プロジェクションアレイを作成する．

4.3 提案システムの機能の実現法

4.3.1 文分割機能

プログラムは，1 行に複数の文を書くことが可能である．また，読解問題を作成する場合，すべての文について読解問題を作成可能にした方が，1 つのソースコードからより多くのパターンの問題を作成できる．そのため，すべての文についてトレース結果を取得し，プロジェクションを作成しなければならない．しかし，デバッガによるステップ実行では行に対するトレース結果が得られる．そのため，1 行に 1 文となるように，ソースコードの分割を行う．具体的には，構文解析により，各文の終了位置を見つけ，そこに改行記号を挿入する．

4.3.2 コマンド作成機能

デバッガを実行するためにはコマンドファイルを作成する必要がある．GDB のコマンドとして表 2 のコマンドを用いる．

表 2 使用する GDB のコマンド

コマンド	意味
break 関数名	関数名の関数に breakpoint を設置
commands	breakpoint で実行するコマンドの設定
display 変数名	変数名の変数の内容を各ステップで表示
step	プログラムをステップ実行する

これらのコマンドを組み合わせて，GDB のコマンドファイルを作成する．具体的には，以下の手順によりコマンドファイルを作成する．

手順 1) ソースコードの各関数について，break コマンドで breakpoint を設定

手順 2) commands コマンドと display コマンドを用いて，各関数で使用されている変数名の値を表示するように設定

手順 3) step コマンドでプログラム終了までステップ実行

ここで，ポインタ変数が指している変数の値を display コマンドで表示する場合，ポインタ変数が配列として用いられているのか，ポインタ変数が指す変数がポインタ変数かなどを知る必要がある．そこで，まず変数のアドレスを出力させるコマンドファイルを作成し，デバッグを行い，ポインタ変数の参照関係を取得する．参照関係により，ポインタ変数で出力させるものを定め，変数の値を出力させるコマンドファイルを作成する．このように，2 段階のデバッグを行うことでポインタ変数が指す変数の値を取得可能にする．図 1 のプログラムに対する変数の値を出力させるコマンドファイルの一部を，図 9 に示す．

```
break qu.c:swap    #swap関数のbreakpoint
commands
display *x          # ポインタ変数xは1つの変数を指す
display *y          # ポインタ変数yは1つの変数を指す
display temp
end
break qu.c:sort    # sort関数のbreakpoint
commands
display *data@5    # ポインタ変数dataは長さ5の配列を指す
display n
display k
display i
display j
end
break qu.c:main   # main関数のbreakpoint
commands
display i
display height
end

run                # プログラムの開始
step               # プログラムのステップ実行
step
step
step
 :
```

図 9 変数の値を出力させるコマンドファイル

4.3.3 参照関係作成機能

デバッガから，出力された変数のアドレスを用いて，プログラムの各ステップでポインタ変数が指している変数を取得し，参照関係を作成する．具体的には，以下の手順により参照関係を作成する．

手順 1) デバッガから出力された変数のアドレスをまとめて，変数が取るアドレスのリストを作成．

手順 2) ソースコードの解析を行い，ポインタ変数の名前と，配列の変数名と型，長さを取得．

手順 3) 配列の先頭アドレスに，型の byte 数を長さ分足すことで，配列の各要素が格納されているアドレスを求め，手順 1 で取得した変数が取るアドレスリストを拡張

手順 4) 拡張したアドレスのリストで，ポインタ変数が取るアドレスと，ポインタ以外の変数が取るアドレスを比較することで，ポインタ変数が指す変数名を決定．

図 1 のプログラムを例に説明する．手順 1 により，表 3 のような変数が取るアドレスリストが作成される．ここで変数名は，関数ごとに区別するため変数名_関数名の形を取る．

表 3 変数が取るアドレス

変数名	アドレス
x_swap	0x28abe8, 0x28abec, 0x28abf0, 0x28abe8, 0x28abec, 0x28abe8
y_swap	0x28abe4, 0x28abe8, 0x28abec, 0x28abe4, 0x28abe8, 0x28abe4
data_sort	0x28abe4
height_main	0x28abe4

手順 2 により，x,y,data はポインタ変数であり，height は int 型で長さ 5 の配列であると分かる．手順 3 により，height の先頭アドレスに int の byte 数である 4 を 4 回足すことで．配列 height の各要素のアドレスを求め，アドレスリストを拡張する．拡張後のアドレスを表 4 に示す．

表 4 拡張後のアドレス

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

手順 4 により，表 5 のような参照関係が作成される．

表 5 参照関係

変数名	参照先
x_swap	&height[1] , &height[2] , &height[3] , &height[1] , &height[2] , &height[1]
y_swap	&height[0] , &height[1] , &height[2] , &height[0] , &height[1] , &height[0]
data_sort	&height[0]

4.3.4 結合機能

デバッガから，出力された変数の値をまとめて各文の実行前後の変数の値を作成する．デバッガからは図 3 のように変数の値が出力される．デバッガでのある文のステップ実行により，その文を含むスコープにおいて定義されたすべての変数の値を得られる．ここで得られる値は，その文の実行直前のものである．よって，第 n ステップに実行される文として出力された実行文の実行前の変数の値が第 n ステップに出力されている変数の値であり，実行後の変数の値は第 n+1 ステップに出力される．このため，第 n ステップと第 n+1 ステップに出力される変数の値をまとめることで，第 n ステップに実行される文の実行前後の値を作成する．

4.3.5 実行系列作成機能

デバッガから出力された文の順番を元に，現在実行中の文から次に実行される文へのポインタを作成することで，実行系列を作成する．文から文へのポインタには，次に実行される文の文番号と，次に実行される文が何回目の実行であるかを示す planeid の組を用いる．

4.3.6 プロジェクションアレイ作成機能

参照関係，文実行前後の変数の値，実行系列を文ごとにまとめ，プロジェクションを作成する．具体的には，参照関係と文実行前後の変数の値から，plane の value 要素を作成し，実行系列から sequence 要素を作成する．また，ソースコードを解析することで変数名リストや文番号などを作成する．

5. プロジェクションアレイを用いた読解問題作成

プロジェクションアレイを用いた読解問題作成システムの構成図を図 10 に示す．読解問題作成システムは，プロジェクションアレイ DB からプロジェクションアレイを取得し，読解問題の種類や候補を提示する提示機能と，指導者が選択した読解問題を作成する作成機能からなる．また，今回は読解問題の種類として図 2(a)(b)に示した，変数の値の変化を問う問題(以下，問題 1 とする)と，文の実行順を問う問題(以下，問題 2 とする)を対象とする．

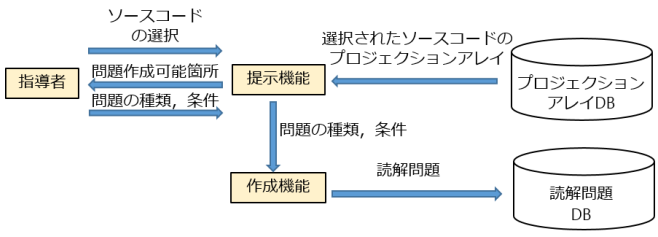

図 10 読解問題作成システム

指導者が読解問題を作成したいソースコードを選

択すると，プロジェクションアレイ DB から選択されたソースコードのプロジェクションアレイが取得される．提示機能は取得したプロジェクションアレイから，要素を抽出することで読解問題の種類や，読解問題に必要な条件の候補を指導者に提示する．指導者は提示された候補の中から，1 つの条件を選択する．作成機能は，指導者により与えられた読解問題の種類や条件に従って，読解問題を作成する．

ここで，今回対象とする問題 1,問題 2 は，どの時点での問題かを示す部分，問題に使用する文，問題の種類の 3 つの部分に分けられる．例えば，図 2(a)の問題は「i=3,k=4 の場合について,」と「24 行目の swap(&data[i],&data[j])」と，「実行前後の配列 data の値を答えよ」の 3 つに分けられる．よって，問題に必要な条件とその条件が必要な問題の種類を表 6 に示す．

表 6 問題に必要な条件

	問題に必要な条件	必要な問題の種類
条件 1	どの時点での問題か	問題 1,2
条件 2	どの文を問題に使用するか	問題 1,2
条件 3	どの変数を問題とするか	問題 1

指導者が問題 1,2 のような読解問題を作成する場合，表 6 の条件 1,2,3 の候補を挙げ，候補の中から条件を決定する必要がある．問題 1,2 の読解問題作成手順を図 11,12 にそれぞれ示す．

手順1) 問題とする変数の候補(条件3)を挙げる
手順2) 候補の中から問題で問う変数を決定
手順3) 問題とする文の候補(条件2)を挙げる
手順4) 候補の中から問題の対象とする文を決定
手順5) 問題とする文の実行回数の候補(条件1)を挙げる
手順6) 候補の中から問題の対象とする実行回数を決定
手順7) 手順2,4,6で決定した条件を元に問題文を作成
手順8) 問題の正解を作成

図 11 問題 1 の作成手順

手順1) 問題とする文の候補(条件2)を挙げる
手順2) 候補の中から問題で問う文を決定
手順3) 問題とする文の実行回数の候補(条件1)を挙げる
手順4) 候補の中から問題の対象とする実行回数を決定
手順5) 手順2,4で決定した条件を元に問題文を作成
手順6) 問題の正解を作成

図 12 問題 2 の作成手順

これらの手順のうち，今回の読解問題作成システムでは，図 11 の手順 1,3,5,7,8 と図 12 の手順 1,3,5,6 をシステムで行う．これ以外の図 11 の手順 2,4,6 と図 12 の手順 2,4 は指導者が決定し，システムに入力を行う．システム側が行う手順のうち，提示機能が図 11 の手順 1,3,5 と図 12 の手順 1,3 で必要な条件の候補を挙げ，作成機能が図 11 の手順 7,8 と図 12 の手順 5,6 で問題文と答えの作成を行う．

提示機能と作成機能の実現法について簡単に以下に述べる．

5.1 提示機能

図 11 の手順 1,3,5 と図 12 の手順 1,3 で条件の候補をプロジェクションアレイから抽出し，指導者に提示する．抽出するための関数として，次のような抽出関数を定義する．

関数 1 getvarlist(projectionarray)

プログラムで使用されているすべての変数について関数ごとに変数名を抽出する関数．プロジェクションアレイを入力とし，プログラム中の関数名とその関数に含まれる変数名リストを出力．

関数 2 getchangecode(projectionarray, var, function)

変数の値が変化する文を抽出する関数．プロジェクションアレイと変数名，関数名を入力とし，入力された変数名の変数の値が変化する文の文番号，行番号，文のリストを出力．

関数 3 getbranchcode(projectionarray)

次に実行される文が実行回数によって変化する文を抽出する関数．プロジェクションアレイを入力とし，次に実行される文が実行回数によって変化する文の文番号，行番号，文のリストを出力．

関数 4 getvalueset(projectionarray, statement)

文の実行回数が一意に決定可能な変数名と変数の値の組を抽出する関数．プロジェクションアレイを入力とし，文の実行回数ごとに一意に決定可能な変数名と変数の値の組のリストを出力．

これらの抽出関数により，図 11 の手順 1,3,5 と図 12 の手順 1,3 の候補の抽出が可能となる．問題作成の各手順と手順で用いる抽出関数を表 7 に示す．

表 7 手順と抽出関数

手順	用いる抽出関数
図 11 手順 1	関数 1
図 11 手順 3	関数 2
図 11 手順 5	関数 4
図 12 手順 1	関数 3
図 12 手順 3	関数 4

5.2 作成機能

図 11 の手順 7,8 と図 12 の手順 5,6 で，問題文と正解を作成する．

問題文の作成には，問題とする時点，問題とする文，問題の種類についてあらかじめ提案システムが用意したテンプレートを用いる．指導者に入力された条件を提案システムがテンプレートに当てはめることで，問題文を作成する．用意したテンプレートを表 8 に示す．

表 8 テンプレート

対象	テンプレート
問題とする時点	$W の場合について
問題に使用する文	$L 行目の$C
問題の種類 (問題 1)	実行前後の$V の値 を答えよ
問題の種類 (問題 2)	実行後に実行される文 は何行目か答えよ

ここで，$W,$L,$C,$V は選択された条件によって変化する変数部分である．この変数部分に提示機能から入力された問題の条件を入れることで，各テンプレートを完成させる．完成したテンプレートを組み合わせることで，問題文が完成する．

正解の作成は，プロジェクションアレイから変数の値や次に実行する文を抽出することで行う．

6. 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[4]を用いて実装した．また，プロジェクションアレイデータベースは，MySQL[5]を用いて実装した．

7. 評価実験

プロジェクションアレイが作成可能かを検証する評価実験と，プロジェクションアレイから読解問題が作成可能かを検証する評価実験の 2 つの評価実験を行った．

7.1 プロジェクションアレイの作成に関する評価実験

プロジェクションアレイ作成システムのプロトタイプシステムを用いて，プロジェクションアレイの作成を行い，プロジェクションアレイが作成可能かを検証することを目的とした評価実験を行った．プロジェクションアレイ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の授業で用いる教科書[6]に掲載されているプログラムを対象とした．教科書に掲載されている全プログラム数と，プロジェクションアレイが作成可能であったプログラム数を表 9 に示す．

表 9 実験結果

全プログラム数	172
作成可能なプログラム数	120
作成不可能なプログラム数	52

教科書に掲載されている 172 のプログラム中，プロトタイプシステムでプロジェクションアレイが作成可能なプログラムは 120 であった．プロジェクションアレイが作成不可能である要因としては，プログラムに do-while 文が含まれている，プログラムの構文解析が行えない，プログラムに再帰があるなどであった．この結果から，多くのプログラムでプロジェクションアレイが作成可能であるといえる．しかし，再帰，do-while など未対応の文法があるため，これらの文法への対応が必要である．

7.2 読解問題の作成に関する評価実験

読解問題作成システムのプロトタイプシステムを用いて，プロジェクションアレイから読解問題の作成を行った．読解問題の作成を行う対象のプログラムとして，図 1 のプログラムを用い，図 2(a)(b)の問題を作成した．以下に，プロトタイプシステムを用いて図 2(a)の問題を作成する流れを示す．

図 2(a)の問題の作成

プロトタイプシステムに図 1 のプログラムを指定し，変数の値を問う問題を選択すると，図 13 のように関数ごとに関数で用いられている変数名が提示される．問題とする変数名として，sort 関数の data を選択すると，図 14 のように変数 data の値が変化する文と行が提示される．問題とする文に 24 行目の swap 文を選択すると，図 15 のように実行回数とその実行回数を示す変数の値の組が提示される．実行回数として，k=4,i=3 を選択することで，図 16 のように問題が作成される．

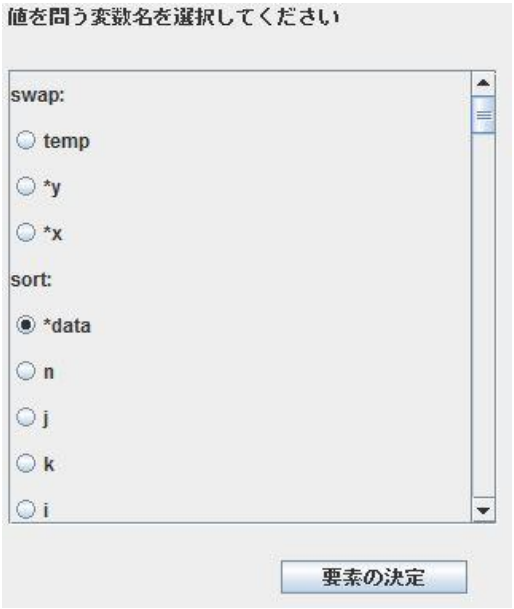

図 13 変数名の候補

図 14 文の候補

図 15 実行回数の候補

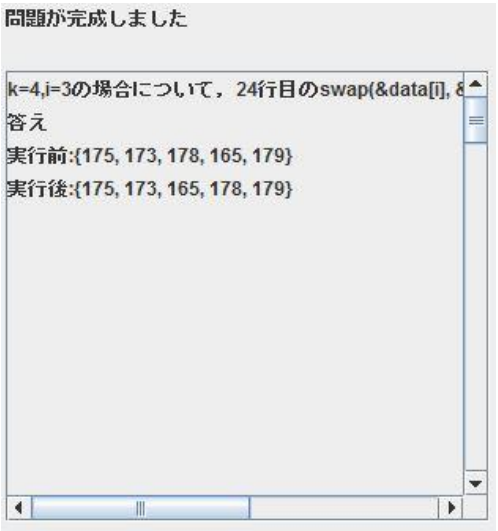

図 16 作成された問題

以上により，プロジェクションアレイを用いることで読解問題が作成可能なことを確認した．

8. おわりに

本稿では，デバッガのトレース結果を文ごとにまとめることで読解問題作成のためのプログラム動作表現プロジェクションアレイを作成するシステムを提案し，その実現法を述べた．また，プロジェクションアレイの活用方法として，プロジェクションアレイを用いた読解問題作成システムを提案し，その実現法を述べた．今後は，プロジェクションアレイの do-while 文や再帰への対応，指導者からの入力なしに読解問題を作成する方法の考案を行う予定である．

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

[5] MySQL , “http://www-jp.mysql.com/”

[6] 柴田望洋, “新版明解C言語入門編”, SB クリエイティブ