B-020

変数の可視化と検査機能を持つプログラミングシステム
A Programming System with Visualization and Checking Values

實川 祐児 † JITSUKAWA Yuji
六沢 一昭 ‡ ROKUSAWA Kazuaki

1. はじめに

プログラム実行中の変数の値の変化の様子を見ることができると、プログラムの振舞いの理解が容易になる。また、プログラム実行時の任意の時点での変数群の値の関係を調べることができると、これは、プログラムの動きが正しいことの確認の助けになる。

本稿は、Perl 言語を対象とし、変数の値の可視化と、変数群の値の関係の検査を行う、初心者向きのプログラミングシステムについて述べる。

2. 設計方針

可視化の対象とする変数　すべて表示されることが初心者にはわかりやすいであろう。このことより、**値の代入の起きたすべての変数を可視化の対象とする**。初心者のプログラムは長くなく、登場する変数も膨大とはならないことが予想されるので、表示項目が多いことによる問題は起きにくいと思われる。

検査可能な関係　プログラムの任意の場所に、その場所を実行する時点で満たされているべき変数の値についての関係を書いておくと、その関係はシステムによって自動的に検査されるとする。この検査されうる関係は自由度の高いことが望ましい。このことから、任意の関係を指定することができるとする。

3. 可視化の実現

3.1 可視化の流れ

値の書き換えられた変数の名前や値などを出力する表示命令を可視化対象ソースプログラムに追加し、ログ取得用プログラムを作成する。そのプログラムを実行することによりログを取得し、そのログを元に可視化を行う。

3.2 非常に多い数の変数の表示

本システムでは、実行時に代入処理の行われた全ての変数を対象とする。そのため、多くの変数を表示する可能性がある。多くの変数が登場すると表示するためのスペースが膨大になってしまう。また、1 つのウインドウに納めようとして、一つ一つの変数の表示を小さくすると、表示が細かいものになり、分かり難くなってしまう。

そのため、一部の変数を表示する**部分表示ウインドウ**と、変数全体を縮小して表示する**全体表示ウインドウ**を用意する。図 4 の下半分が部分表示ウインドウであり、右上の長方形部分が全体表示ウインドウである。

3.3 部分表示ウインドウの構成

表示する変数には、大きく分けて一般変数と配列要素の 2 つがある。2 つを混在して表示すると初心者には分かりにくくなってしまう。そのため、ウインドウを左右の 2 つに分けた。一般変数は左のウインドウに、配列要素は右のウインドウに表示する。

4. 変数検査機能

これは、プログラムの実行中、任意の時点で、変数の値がユーザの意図する条件を満たしているか検査する機能である。

以下の変数検査コマンドを、検査を行いたい位置に書き加えることで、その位置での検査と、結果の表示が出来る。

```
#? 条件式
```

このコマンドは先頭が #であるので、可視化対象プログラム自身を動かした場合、コメント文となりプログラムの動作に影響を与えない。

ログ取得用プログラムの作成では、コマンドの条件式を、if{ 条件式 } という形に変換する。そしてログ取得用プログラム実行時に検査が行われ、結果はログとして出力される。

```
#!/usr/bin/perl
$sum=0;
for($i=1;$i<=10;$i++)
{
   $sum+=$i;
}
#? $i==11
```

(a) 正しいプログラム

```
#!/usr/bin/perl
$sum=0;
for($i=1;$i<10;$i++)
{
   $sum+=$i;
}
#? $i==11
```

(b) 間違えたプログラム

図 1: 変数検査コマンドの使用例

変数検査コマンドの使用例

図 1 は、1 から 10 までの加算処理を行うプログラムである。図 1(a) を実行すると、加算が 10 回繰り返され、ループを脱出した時点での変数 $i の値は 11 となる。そのため、最後の行の変数検査コマンドによる検査では、「条件を満たしている」という結果が表示される。

一方、図 1(b) では、for 文の条件判定を間違えているため、加算が 9 回しか繰り返されず、ループを脱出した

†千葉工業大学 大学院 情報科学研究科 情報科学専攻
‡千葉工業大学 情報科学部 情報工学科

時点での変数 $i の値は 10 となる。そのため、「条件を満たしていない」という結果が表示される。

5. システム

5.1 処理の流れ

図 2 に本システムにおける可視化までの流れを示す。

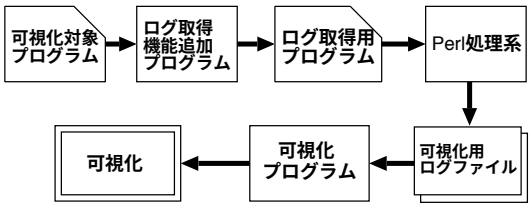

図 2: 可視化の流れ

5.2 表示形式

図 4 に表示画面例を示す。下半分は**部分表示ウインドウ**、右上は**全体表示ウインドウ**である。残りの左上の部分には**実行中の命令**などが表示されている。

部分表示ウインドウには変数とその値が表示され、変数の値が書き換えられると内容が更新される。全体表示ウインドウには全ての変数が縮小して表示され、新しい変数が登場するとウインドウの内容が更新される。

6. 実行例

図 3 のソースプログラムを対象として可視化をした一場面を図 4 に示す。

```
#!/usr/bin/perl
$sum=0;
for($i=1;$i<=10;$i++)
{
   $sum+=$i;
   $ar[$i]=$sum;
}
#? $i==11
```

図 3: 可視化対象ソースプログラム

図 3 は、1 から 10 までを加算し、途中結果を配列に格納するプログラムである。図 4 は for ループ中の代入処理の一場面であり、配列要素 $ar[5] への値の代入を示している。

書き換え中の変数 ($ar[5]) は、部分表示ウインドウでは矢印で、全体表示ウインドウでは網掛けで示されている。

図 5 はプログラム終了時の場面である。左上の**実行中の命令**欄の右端の T は、 $i==11 に対する検査結果 (条件を満たしている) である。

7. 初心者からの初期評価

Perl プログラミング初心者の学部学生 8 人に本システムを利用してもらった。そしてシステムを利用した感想をアンケートとして提出してもらった。以下は寄せられたコメントの一部である。

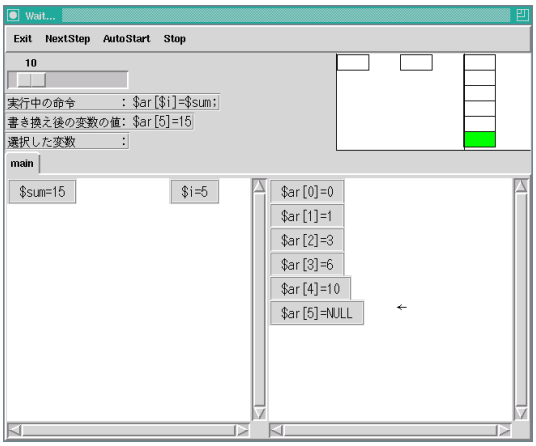

図 4: 可視化時の一場面

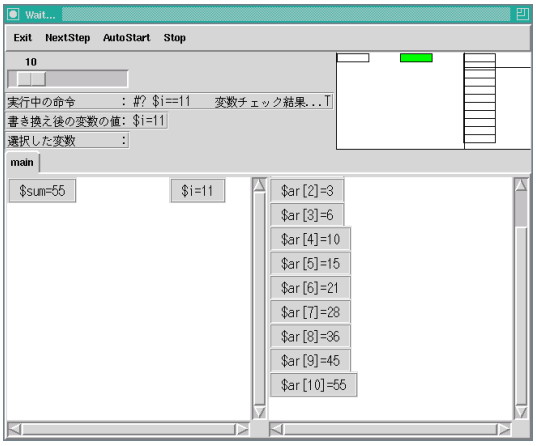

図 5: 変数検査結果の表示

- 変数の値の変化を理解するのに非常に役に立った。
- 変数の値の書き換えを見ることが出来たので、容易に予想外の書き換えを発見することが出来た。
- 変数の値の変化が分かりやすかった。
- 大きい配列がある場合、どの配列要素が書き換えられているか分かりやすかった。

上記のように、プログラムの振る舞いを理解する事や予想外の変数の値の書き換えを発見する事に有効であるという意見を得られた。また、画面表示が見やすいという感想を得られた。

8. まとめ

変数の値の変化の可視化機能と変数の値の検査機能を持ったプログラミングシステムを作成した。

本システムはプログラム実行状況の理解や、予想外の書き換えなどのプログラムミスの発見に役立つことが期待される。