N-016

変形する図形によるプログラムの実行状況の可視化
Visualizing Program Execution Using Diagram Animation

神山 拓哉 †　　六沢 一昭 ‡
KAMIYAMA Takuya　　ROKUSAWA Kazuaki

1. はじめに

本稿では, Perl 言語を対象とした, 変形する図形によるプログラムの実行状況の可視化について述べる.

この可視化は, 頂点の位置が変数の値に対応した図形の描画によって実現する. プログラムの実行によって変数の値が変化すると, 対応した頂点の位置も変化するため, 図形が変形する. このように, プログラムの実行状況は図形の変形によって表現される.

本可視化では図形の変形する様子を見ることで, 実行の全体像を知ることができる. そのため, プログラムの実行状況の全体像の理解が期待される.

2. 可視化の実現

2.1 処理の流れ

本可視化では, 可視化対象プログラムの実行ログを用いて可視化をおこなう. まず, 可視化対象プログラムに, 変数の名前や値などの情報を出力するためのコードを追加し, ログ出力プログラムに変換する. このログ出力プログラムを実行することで, 実行ログを得られる. そして, 可視化プログラムはこの実行ログを元に可視化をおこなう.

可視化の処理の流れを図 1 に示す.

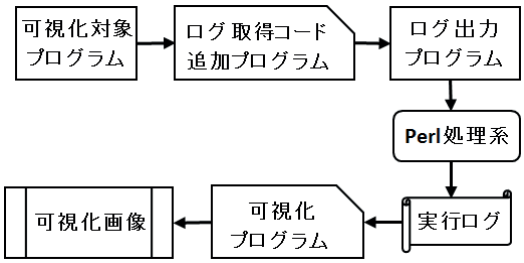


図 1: 処理の流れ

2.2 実行ログ

実行ログの例を図 2 に示す. 実行ログの各行は変数への値の代入を表している. 各行は変数名, 変数の値, 行番号の 3 つの情報からなる.

2.3 図形の形成

本可視化は, 頂点の位置が変数の値に対応した図形の描画によって実現する.

```
i,1,10
j,0,11
cha,40,14
num[0],30,15
num[1],40,16
j,1,11
cha,40,14
num[1],35,15
num[2],40,16
j,2,11
  . . .
```

図 2: 実行ログ

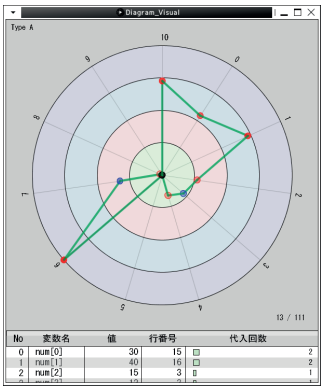


図 3: 実行画面

頂点

各頂点は, 原点との距離が数値に対応している. また, 各頂点は, 変数の値が正ならば赤, 負ならば青, 0 ならば黒で表現する.

3. 指示文

指示文は可視化プログラムへの指示を表すものであり, 以下の 2 種類がある.

3.1 可視化対象の変数指定

必要な変数のみを可視化したい場合は, 可視化対象プログラムのソースコードに以下の行を追加する.

`#Var:変数名, 変数名, 変数名, 変数名`

なお, この指示文がない場合は, 全ての変数が可視化対象となる.

使用例

配列@num, 変数$a, $b, $c, $d, $e を可視化対象に指定する指示文を以下に示す.

```
#Var:@num,$a,$b,$c,$d,$e
```

また, 配列の要素を指定したい場合には, 以下のように指定する.

```
#Var:$num[0],$num[1],$num[2],$num[3]
```

3.2 条件文

特定の条件をある位置で満たしているか確認したい場合には, 可視化対象プログラムのソースコードの確認したい位置に以下の行を追加する.

`#If:条件式`

この指示文が実行されると, 評価結果が実行ログに出力される. これにより, プログラムの実行中に指定した条件を満たしているか確認することができる.

†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

使用例

変数$a と$b が等しいことを確認するための指示文を以下に示す.

```
#If:$a==$b
```

変数$i が 10 未満であるか確認するための指示文を以下に示す.

```
#If:$a<10
```

4. 表示画面

可視化プログラムの実行画面例を図 3 に示す. 画面上部は可視化領域であり, 下部は変数情報領域である.

可視化領域

変数に対応した頂点を持つ図形が描画される. 頂点に付随する番号は, 変数情報領域の番号と対応している.

変数情報領域

変数の情報を表示する. 各項目は, 変数の番号, 変数名, 変数の値, 代入行の番号, 代入回数を示す. 代入回数は棒グラフでも示している.

5. 実行例

5.1 実行 1

図 4 は, for ループの制御変数$i と$j の和と差をそれぞれ$num1, $num2 に代入するプログラムである. このプログラムの実行状況の可視化を図 5 に示す.

```
1  #!/usr/bin/perl
2
3  for($i = 1; $i < 10; $i++){
4      for($j = 1; $j < 10; $j++){
5          $num1 = $i + $j;
6          $num2 = $i - $j;
7      }
8  }
```

図 4: 可視化対象プログラム

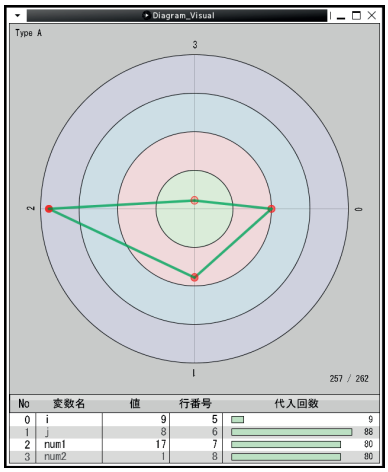


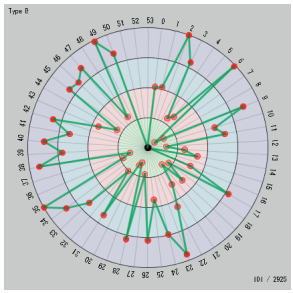
図 5: 可視化の実行例 1

図 4 のプログラムには 4 つの変数が登場する. そのため, 可視化によって図 5 のような 4 つの頂点を持つ図形が描画される. 右側の頂点から時計回りに$i, $j, $num1, $num2 である.

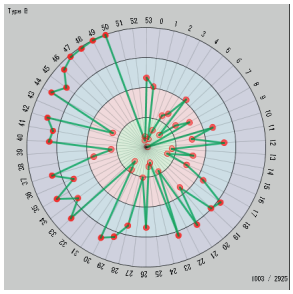
5.2 実行 2

図 6 は, 50 個の値に対してバブルソートをおこなうプログラムの実行状況を可視化である.

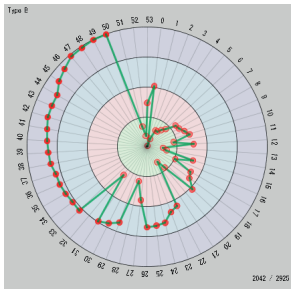
図 6(a) は, ソート前の状態であるため, 棘が生えているような図形が描画されているが, ソートによって徐々に図形の棘がなくなり, 滑らかになっていくことがわかる. 図 6(d) では, ソートが終了して渦を巻くような図形になっている.



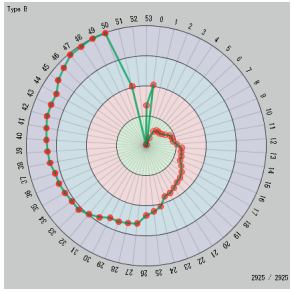
(a) 開始前



(b) 序盤



(c) 終盤



(d) 終了

図 6: 可視化の実行例 2

6. おわりに

プログラムの実行状況を, 変形する図形によって表現する可視化プログラムを作成した.

図形が変形していく様子は印象に残りやすく, プログラムの実行状況の全体像をイメージしやすくなると考えられる. 本可視化はプログラムの実行状況を理解する一助になることが期待される.

参考文献

[1] Stephan Diel, Software Visualization, Springer, 2007.

[2] 實川 祐児, 六沢 一昭, 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, 第 6 回 情報科学技術フォーラム (FIT 2007), B-020, pp.125–126, 電子情報通信学会, 情報処理学会, 2007.