重ねあわせを用いたビジュアルプログラミングの表記法

Discription Method for Visual Programming Using “Overlapping.”

奥村 穂高† 田中 二郎††
Hotaka OKUMURA Jiro TANAKA

†筑波大学 大学院修士課程 理工学研究科
Master’s program in Science and Engineering,Univ. of Tsukuba
††筑波大学 電子・情報工学系
Institute of Information Sciences and Electronics,Univ. of Tsukuba

概 要

本論文は、ビジュアルプログラミング (VP) においてコマンド等をアイコン化したものを基に、データフローにノードの「重ねあわせ」を、制御フローに「有向グラフ」を使った表現を用いることによって、２つのフロー図を組み合わる記述法を提案する。また、それを利用した直接操作可能なグラフィカルなプログラミング環境を提案し、プログラムの編集を行なえるようなプロトタイプシステムを作成した。

1 はじめに

ビジュアルプログラミングは、「いくつかの意味のある図的な表現をプログラミングの過程で使用すること」と定義される [1]。プログラミングは主に理論的、言語的な能力によって成立する活動であるが、そこに、図的な表現を導入することによって、直観的能力をより有効に活用することができるようになる。このことは、プログラミングの効率化につながる。また、VP は「複雑な事柄をシンボル化し、コンピュータにより直接操作できるものに置き換える技術」とも定義できる [2]。直接操作を利用することにより、対象物に対して直に指示を反映させることができるため、扱う対象を認知するための負担が軽減され、プログラミングがより容易になる。グラフィカルユーザインターフェイスの発達により、ビジュアルプログラミングはより手軽で実用的な技術となった。今後、プログラミングについても従来のコマンドベースのインターフェースから、よりヒューマンフレンドリーなインターフェースへとプログラミング環境は変化していくと思われる。

2 ビジュアルプログラミングの表記

2.1 ノードの重ねあわせ

「ノードの重ねあわせ」とは、２つのノードを重ねることによって、その間に意味を持たせるものである (図 1) [3]。本論文では、重ね合わせる動作ではなく、重ねあわせている状態に着目した。データフローを重ねあわせで表現し、手前のノードから奥のノードへとデータが流れていると見なす。また、引数もノードで扱い、他のノードと同様な扱いをする。一つのノードに二つ以上の重ねあわせを行った場合、左上の位置から時計回りに引数等の順序を割り当てる。

重ねあわせを利用することにより、グラフ構造を

簡略化し表示することができるようになる。それによって、利用者のグラフ全体の把握や部分的なパターン認識の効率を向上できる。

図 1　重ねあわせ

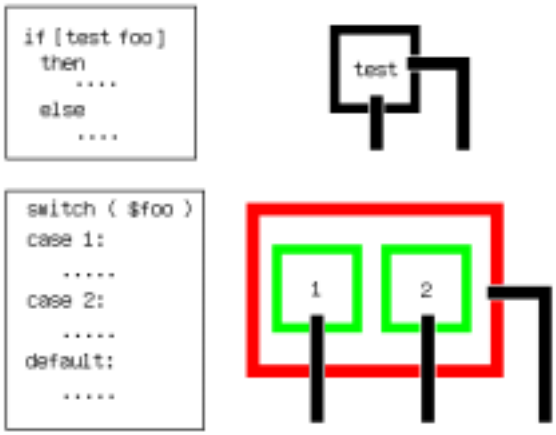


図 2　条件節・多段分岐

2.2　制御フロー

制御フローはエッジ（矢印）を使って表記する。プログラムの先頭は最上方左端にあるノードであり、基本的にグラフの左から右、上方から下方に向かって制御が流れていく形となる。また、次のノードに続かなくなった時点でそのプログラムは終了する。この点については、フローチャートとほぼ同様な記述となっている [4]。

フロー中の条件分岐は、真であるときは下からのエッジ、偽であるときは右からのエッジで記述する (図 2 上)。また、case 文などの条件節の場合、それぞれのノードに記されている条件に該当する場合はそのノードの下につながるエッジに制御が移る。どの条件も該当しない場合は、複数のノードをまとめているプレートの右から出ているエッジに制御が移る (図 2 下)。

2.3　全体の表記

データフローを重ねあわせで、制御フローをエッジで示すことによって 2 種類の流れを同時に表記することを可能にしている。また、重ねあわせを併用することによって Pict [5] のように、すべてエッジで表現する方法と比べ、エッジの数を減らすことができ、グラフの複雑さを緩和できる。プログラムは、コマンドやデバイスを示すノードと、処理の流れを示すエッジによってあらわされる。ノードは使用目的によりいくつかにわかれ、それぞれ色のついた枠によって区別する。プレートは、ある一連の処理をまとめて実施するときなどに利用し、その該当する範囲を枠で囲み 1 つにまとめることによって、1 つのノードと同等に扱うようにする。プレートを使うことにより、ループや再帰の記述が容易になる。各ノードの中央部分には、そのノードが保持するコマンド名やファイル名を記述しておく。

3　試作システム

3.1　試作システムの対象

以上に述べた、ビジュアルプログラミングの表記方法を UNIX の Shellscript に対して適用し、マウスを利用して Shellscript を生成する試作システムを作成した。

Shell とは UNIX 上で使われるコマンドインタプリタであり、そこで使われるコマンドとその手順を記述した実行可能なプログラムを Shellscript という [6] [7]。Shell コマンド は基本的に 1 入力 1 出力のため、重ねあわせを利用するのに適している。例えば、Shell で使われるパイプは図 3のようにあらわすことができる。

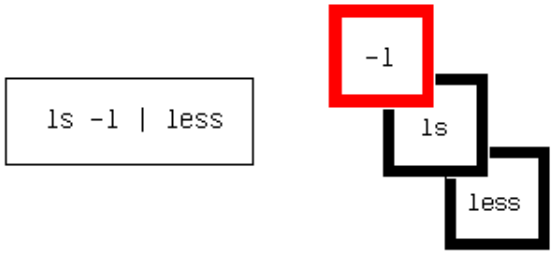


図 3　パイプ と 引数

3.2　システム構成・操作法

今回、我々は実装言語として Java を用いて実装を行った。本システムは主にノードの種類やエッジなどを選択するウインドウと、選択したものを使って実際にプログラムを作成するウインドウから構成される。この 2 つのウインドウをつかい、プログラムに相当するグラフ構造を作成する。

ノードは選択肢側のウインドウで種類を選択する。その際、持たせるファイル名やコマンドを指定しておき、作成側のウインドウ内のノードの無い場所でクリックすることによって目的のノードを作成する。

プレートの場合もノードなどと同様に作成する。ただし、サイズが可変なため、サイズ決定には、カット＆ペーストを行なう場合と同様の操作によってサイズを指定する。

次に、プログラムの作成の手順について述べる。

1. ノードの作成
   選択ウインドからノードの種類を選択し、制作側のウインドウに配置する。
2. ノードの重ねあわせ
   ノードを動かし、重ねあわせをおこなう。
3. 制御フローを記述する
   選択側のウインドウにより、エッジを選択し、分岐の作成、制御フローの指定などを行う。
4. グループ化を行う
   必要によって、プレートをもちいてグループ化を行う
5. １～４の手順を必要に応じて繰り返す。

3.3　糊しろ

2 つのノードをそのまま重ねてしまうと、下のノードが隠れてしまうことがあり、ノードの内容が確認できなくなる場合がある。そのようなことを避けるために、ノードにはあらかじめ、「糊しろ」と呼ばれる部分を想定しておく。

もし、糊しろをはみ出して重ねあわせを行った場合、置いた直後に、重ねようとするノード（drag していたノード）は自動的に糊しろの部分まで移動し、下のノードの内容が隠れないようにする。

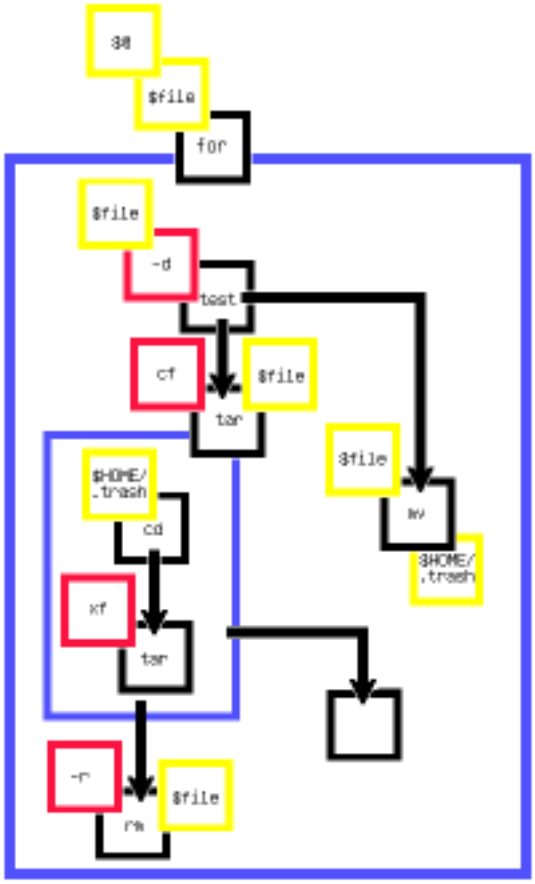
4　図形からの Shellscript 生成

試作システムではグラフ構造で記述された図形から、 Shellscript の生成を行う。基本的にグラフの左上から右下へとノードが配置されているので、それに基づいてコードを生成する。パイプ、リダイレクト、代入など Shellscript の 1 行分を重ねあわせを使って表現している。また、各行の関係や、分岐などをエッジを使って表現している。

図 4のプログラムは、 作成したプログラムの例である。引数で指定したファイルやディレクトリを、ホームの下の .trash というディレクトリに移動させ、元のファイルを消す、というプログラムである。指定されたファイルがディレクトリならば、その中のファイルもまとめて .trash へ移動する仕組みになっている。

試作システムのコード生成は、次の手順で行われる。

1. プレートの有無を調べる
   プレートによってグループ化されている個所を探す。もしある場合は、そのプレート内の記述について、手順１より再び行う。
2. 重ねあわせから行単位の Shellscript を生成する
   重ねあわせは、記述の性格上、あるノードＡに対して、Ａの前後 1 段階に重ねられたノードが持つ情報が直接影響してくる。よって、前の段階のノードから渡された情報を、そのノードが与えられた目的に従い、加工、選択し、後のノードに引き渡す。本試作システムでは、もっとも手前のノードから同じ深さのノードを左上のものから順に解析、それによって Shellscript




```
#!/bin/sh -f

for file in $@
do
   if test -d $file
   then
      tar cf - $file |(cd $HOME/.trash ;\
      tar xf -) && rm -r $file
   else
      mv $file $HOME/.trash
   fi
done
```

図 4　Shellscript プログラムの図形による記述

を行単位で生成する。

3. エッジから制御フローの関係を構築する

行単位で生成された Shellscrip に対して、エッジが示す制御フローに従い、左上から、制御関係の構築を行う。

5　結論と今後の課題

本論文では、データフローに重ねあわせを、制御フローに有向グラフを用いることによって、この 2 種類のフロー構造を同時に表記する手法を提案した。この手法を利用することにより、既存のビジュアルプログラミングシステムをより直観的なものにすることができる。

また、提案した手法を用いた直接操作によるビジュアルプログラミングシステムを実装した。このシステムには UNIX 上の Shellscript に対して利用し、実装言語に Java を用いた。「重ねあわせを用いた際に、ノードが隠れてしまう」という問題点を指摘して、その対処法として「糊しろ」を使うことを提案した。

今後、 Shellscript 以外の言語への応用することにより、汎用性を高め、他の言語も統一的な操作で利用できるようにしていきたい。

参考文献

[1] Shu,Nan C. : Visual Programming , Van Nostrand Reinhold (1988) (邦訳： 西川博昭, ビジュアルプログラミング, 日経 BP 社 (1991))

[2] 田中二郎: ビジュアルプログラミング, ビジュアルインターフェース Bit 2月号別冊, pp.65–78, 共立出版 (1996)

[3] Hirakawa, M., Tanaka, M. and Ichikawa, T. : An Iconic Programming System, HI–VISUAL, *IEEE Transaction on Software Engineering*,Vol.16, No.10, pp.1178–1184(1990)

[4] JIS X 0121 , 情報処理用流れ図・プログラム網図・システム資源図記号

[5] Glinert, E., Tanimoto, S. : PICT: An Interactive Graphical Programming Environment, *IEEE Computer* , Vol.17, No.11, pp 7–25 (1984)

[6] UNIX System V プログラマ・リファレンス・マニュアル 第 2 版リリース 3.0, 共立出版 (1986)

[7] 山口和紀 他, The UNIX Super Text [上], 技術評論社 (1992)