3次元ビジュアルプログラミング環境における視覚化手法

Program Visualization in 3D Visual Programming Environment

岡村 寿幸 † 田中 二郎 ††
Toshiyuki OKAMURA Jiro TANAKA

† 筑波大学大学院 博士課程 システム情報工学研究科
Doctoral Program in System and Information Engineering, University of Tsukuba
†† 筑波大学 電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba
{okamura, jiro}@iplab.is.tsukuba.ac.jp

我々は，プログラムを 3 次元オブジェクトを用いて視覚化する，3 次元ビジュアルプログラミングシステム “3D-PP” の開発を進めている．3D-PP によって視覚化される GHC プログラムが理解しやすいものであるためには，引数の区別の容易さ，引数の入出力モードの明示，オブジェクトの分散防止が重要である． これらを実現する手法として，引数を表し，引数の入出力モードを明示するオブジェクトを付加する手法と，オブジェクトの中にオブジェクトを入れ，分散を防止する，オブジェクトの階層表示の手法を提案する．また，これらの手法を 3D-PP 上に実装した．

1 はじめに

ビジュアルプログラミングとは，ユーザが，2 次元，またはそれ以上の次元を用いてプログラムを表示することを可能にするシステムである [1].

これまでの多くのビジュアルプログラミングシステムでは，2 次元図形を用いてプログラムを表現していた [2, 3, 4]．一方，我々は，3 次元空間上で，3 次元オブジェクトを用いてプログラムの視覚化を行う 3 次元ビジュアルプログラミングシステム 3D-PP の研究 [5, 6] を進めてきた.

2 背景

3D-PP は，2 次元ビジュアルプログラミングシステム PP[4] を，3 次元に拡張した 3 次元ビジュアルプログラミングシステムである．3D-PP は PP と同様に，並列論理型言語 GHC[7] を視覚化している．

GHC プログラムは，図 1 のような形をしており，実行時の述語の呼び出しを表すゴールと，ゴールの処理の条件と，その条件のときの処理を記述する節から成っている．GHC では，ゴールがどのような処理を行うかは，ゴールと同じ名前を持ち，同じ数の引数を持つの節 (以下同名，同引数の節と呼ぶ) によって記述される． 同名，同引数の節は通常，複数個存在する．節は

```
name(Arg1,Arg2,...) :- guard | body.
```

という形をしている．先頭から “:-” 記号まではヘッダと呼ばれ，どの述語の節であるかと，この節での引数の数と引数名を表している．“:-” から “|” まではガードと呼ばれ，節を選ぶ際の条件を表している．“|” から “.” まではボディと呼ばれ，節が選ばれたときの処理を表している．また，GHC の実行は，節に記述されたルールに従って，ゴールを書換えるリダクションによって行われる．図 1 はユークリッドの互除法によって最大公約数 (GCD) を求めるゴール gcd を定義している 4 個の節である．

PP では，ゴール等のプログラム要素を表す 2 次元図形をエッジで結線し，グラフを作成することで GHC プログラムを表現している．一方，3D-PP ではプログラム要素を 3 次元オブジェクトで表し，それらをエッジで結線して，3 次元グラフ作成することで GHC プログラムを表現している．

3D-PP では，マウスを用いて 3 次元オブジェクトを操作することによってプログラム編集を行うため，3 次元オブジェクトを操作，配置するための手法が用いられている．具体的には，強化された直接操作手法 [8] に基づく，オブジェクトの位置と上下関係を把握するための地面のオブジェクトと，オブジェクトが 3 次元空間上で重なり合わない様に自動的に再配置する，自動レイアウト [9] の機能がある．

これまでの 3D-PP では，3 次元空間上で 3 次元グラフを編集することができ，3 次元空間上での編集操作，プログラム表現手法について様々な検討が行われてきた．[6] では，並列論理型言語の特徴である，

リダクションの過程や論理変数の具体化に着目した表現手法が提案されている．[5] では，拡張された直接操作手法の 3D-PP への適用が提案されている．しかし，これまでの 3D-PP では，引数のための表現はなく，引数同士の区別が明確ではなかった．

本研究では，PP とこれまでの 3D-PP で検討されてきたプログラム表現を元に，3D-PP 上で編集可能な GHC プログラムを表現する手法を提案し，実装を行った．提案するプログラム表現は，ゴールの引数に関する表現とオブジェクトの中にオブジェクトを入れる手法を利用した，ゴールと節のプログラム表現である．

```
gcd(X,Y,Out) :- X=:=0 | Out = Y.
gcd(X,Y,Out) :- Y=:=0 | Out = X.
gcd(X,Y,Out) :- X=\=0, X=<Y |
     gcd(X1,X,Out), X1 := Y mod X.
gcd(X,Y,Out) :- Y=\=0, X=>Y |
     gcd(Y,Y1,Out), Y1 := X mod Y.
```

図 1: GCD の定義

3 提案手法

3.1 引数オブジェクト

我々は，3D-PP のプログラム表現として，GHC のゴールの引数に注目した．ゴールは与えられる引数によってその挙動を変えるため，ゴールの引数が何と対応しているかが明確に区別され，プログラマが引数を容易に判別できる様にすることが重要である．引数を判別することが容易になれば，プログラムの読み間違いを軽減することができる．

また GHC では，ゴールの引数が入力，出力のどちらとして使われるかは陽には記述しないが，プログラマは通常，引数の使い方についてあらかじめイメージを持っている [10]．そのため，引数が入力，出力のどちらとして使われるか (入出力モード) を明示出来る方法を導入すれば，プログラムの理解しやすさが向上すると考えられる．

3.1.1 引数の入出力モードの明示

引数の入出力モードは，引数オブジェクトの配置位置の違いで表現する．引数オブジェクトを付加するオブジェクトには，引数オブジェクトを配置するための面がオブジェクトの上部，下部，あるいは両方に存在する． 上部の面には入力の引数オブジェクト，下部の面には出力の引数オブジェクトを配置する．また，引数オブジェクトの色は入力，出力で異なっており，入力が赤色，出力が青色となっている．

入力を上部の面，出力を下部の面と固定したことによって，3D-PP の自動レイアウト機能によるオブジェクトの再配置の際に，上から下へと処理順に配置することが出来る．これにより，プログラマがオブジェクトを手動で，配置しなおさなくても読みやすい配置を得ることが容易となる．

3.1.2 引数同士の区別

同じ入出力モードの引数同士は，引数オブジェクトの色の濃度の違いと各面上での配置で区別する．より濃い色をした引数オブジェクトが，同じプログラムをテキストで表記した時に，ヘッダに先に現れる引数となる． また，現れる引数の順に，時計回りに配置される．

図 2 は，引数オブジェクトを付加したゴールのオブジェクトの例である．左のゴール `separator` には，入力が 1 個，出力が 2 個の引数オブジェクトが付加されている．出力の引数オブジェクトは，テキストで表記した時に現れる順に、色が薄い青となっているのが分かる．右のゴール `goal` には，入力が 3 個，出力が 1 個の引数オブジェクトが付加されている．入力の引数オブジェクトは，テキストで表記した時に現れる引数の順に色が薄い赤となり，時計回りに配置されていることが分かる．

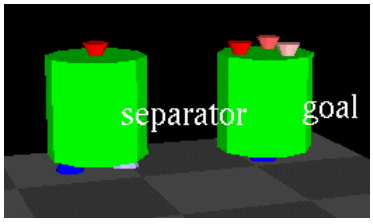

図 2: 引数オブジェクトを付加したゴールの例

3.2 オブジェクトの階層表示

3.2.1 ゴールと節

同名，同引数の節は，各々がゴールの処理の条件とその条件のときの処理を記述しており，同名，同引数の節の集まりで，ゴールの動作を規定している．よって，ゴールの動作を理解するには，そのゴールと

同名，同引数の節全てを見る必要がある．そのため，GHC プログラムをテキストで記述するときには，同名，同引数の節は近い場所に記述され，全ての同名，同引数の節をまとめて見れるようにしている．

もし，同名，同引数の節を全く関連のない場所に記述した場合，プログラムを読む際に，同名，同引数の節をプログラムのあちこちを見て探さなくてはならなくなる．3D-PP 上で視覚化されたプログラムにおいても，同名，同引数の節が表示空間上の，全く関連のない場所に配置されていた場合には同様のことが起きる．

我々は，同名，同引数の節や節の定義を表すオブジェクトをまとめて扱い、分散しない様にするための手法として，ゴールのオブジェクトの中に同名，同引数の節，節のオブジェクトの中に節の定義を表すオブジェクトをまとめていれる，オブジェクトの階層表示の手法を提案する．オブジェクトの階層表示には,「ゴールと同名，同引数の節」の階層表示と,「節とその定義」の階層表示の 2 つがある．

階層表示されているオブジェクトは，そのオブジェクトよりも上の階層 (外側) にある オブジェクトが表示されているときには，そのオブジェクトの中に隠れ，見えない状態になっている．中のオブジェクトを見る場合には，上の階層のオブジェクトを左ボタンダブルクリック操作し，拡大表示状態にする． この状態では，上の階層のオブジェクトはオブジェクトの輪郭のみの表示となり，中のオブジェクトが表示され，中のオブジェクトを操作することができる．

ゴールのオブジェクトをを拡大表示すると，その中にある節のオブジェクトが表示される．また，節のオブジェクトを拡大表示すると，その中の節の定義を見ることができる．

3.2.2 ゴールと同名，同引数の節の階層表示

図 3 は，ゴールと同名，同引数の節を階層表示した表現である．図 3 のように，ゴールと同名，同引数の節は全てそのゴールのオブジェクトの中に入る．節のオブジェクトは全て同じ大きさで，3 次元空間上で互いに重なり合わない様に配置される．

3.2.3 節とその定義の階層表示

図 4 は，図 3 の節のオブジェクトの 1 つを拡大表示した図である．節を拡大表示にした場合は，節の外側にあるゴールのオブジェクトは，一時的に表示されなくなる．また、画面右下に，現在定義されているゴールと同じ引数オブジェクトが付加されたアイコンが表示される．これは，他のゴールの呼び出し，または再帰的呼び出しを定義するために用いられる．

節のオブジェクトの内部では，半透明の円盤状オブジェクトを節オブジェクトの内部に設けることで，ガード部とボディ部とに分けている．円盤状のオブジェクトは GHC プログラムのガード，ボディの区切り “|” に相当する．図 6 のように，円盤状のオブジェクトの上にガード部，下にボディ部のオブジェクトを配置している．

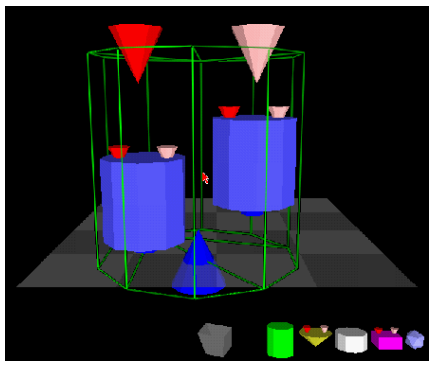

図 3: ゴールと同名，同引数の節の階層表示の例

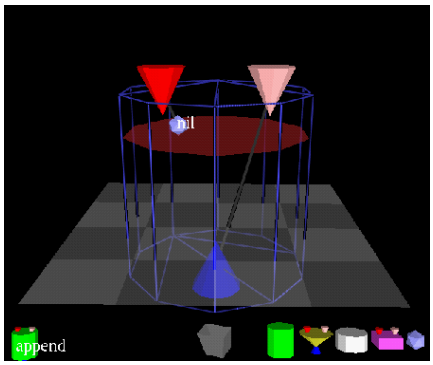

図 4: 節と節の定義の階層表示の例

4 プログラムの例

プログラム例として，GCD を求めるプログラムを用いる (図 1)．gcd は 4 つの節を持っているが，ここではそのうちの 1 つ (図 1 の 3 番目の節) を例として解説を行う．

4.1 ゴールの引数，節

図 5 はゴール gcd に引数，節を設定したものを拡大表示状態にした図である．

ゴール gcd は X，Y の 2 つの入力の引数と，出力の引数 Out を持っているので，上部の面に，濃い赤 (X)，薄い赤 (Y) の 2 つの引数オブジェクトがあり，下部に濃い青 (Out) の引数オブジェクトがある．

引数オブジェクトの追加は，引数が入力ならオブジェクトの上の面を，出力なら下の面を左ダブルクリックすることで行う．引数オブジェクトを，予定より多く追加してしまった場合は，引数オブジェクトを画面下の中央にあるゴミ箱アイコン上にドロップすることで削除することができる．

図 5 のゴール gcd の中には，4 個の節のオブジェクトがある．これは gcd が 4 個の節を持っていることを表している．

節の数を設定は，ゴールに節のオブジェクトを追加することで行う．ゴールのオブジェクトを右ダブルクリックすることで，ゴールのオブジェクトの内部にゴールと同じ引数オブジェクトを持った，節のオブジェクトが追加される．予定より多く節のオブジェクトを追加してしまった場合は，ゴールを拡大表示した状態で，削除したい節のオブジェクトをゴミ箱アイコン上にドロップすることで削除することができる．

ゴールを拡大表示状態にするには，ゴールのオブジェクトを左ダブルクリックする．拡大表示状態を解除するには，中にあるオブジェクトに触れていない状態で，オブジェクトを左クリックする．

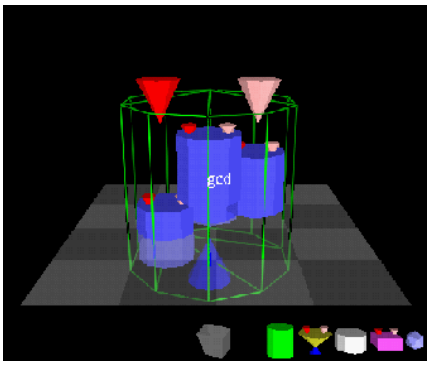

図 5: ゴール gcd の節

4.2 節の定義

節の定義の編集は，編集する節を拡大表示状態にして行う．節を拡大表示状態にするには，節のオブジェクトを左ダブルクリックする．また，拡大表示状態を解除する場合は，中にあるオブジェクトに触れていない状態で，オブジェクトを左クリックする．

拡大表示した状態で、節のオブジェクトのガード部，ボディ部にオブジェクトを追加し，それらを結線することで定義のグラフを作成する．

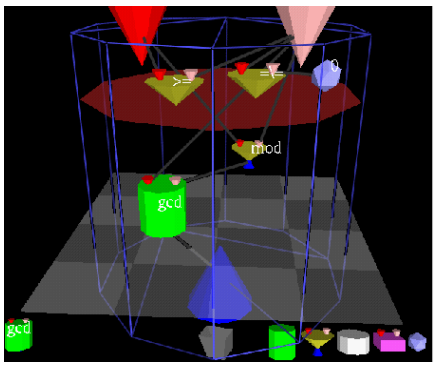

図 6: 節 gcd の定義

5 議論

5.1 引数の表現

引数の区別については，ゴールと引数として与えられる要素のオブジェクトを結ぶエッジに色をつけて区別する，という手法が考えられる．この手法は，引数オブジェクトが色の系統と濃淡で区別されているように，エッジの色の濃淡で引数の区別を行う．

引数オブジェクトを用いた場合には，引数オブジェクトを配置するための面を，上下ではなく左右に用意するという手法も考えられる．この手法では，オブジェクトが左右に広がるように配置されやすくなる．上下に配置される本論文での手法と比較して，プログラムの規模が大きくなったときに表示しやすい，という利点があると考えられる．

また，同じ入出力モードの引数の区別には， 時計回りに配置する等の配置方法による区別の他に，引数オブジェクトに番号を付けて区別するという手法も考えられる．

5.2 節の定義の表示，同名，同引数の節の表現

節の定義の表示については，節の定義を見たり，編集したりする際には，その節の定義だけを表示する，別のウィンドウを表示する，という手法が考えられる．オブジェクトを表示する領域を分け，各ウィンドウに必要なオブジェクトしか表示しないため，各ウィンドウの表示するオブジェクトが少なくなる．

本論文で提案した同名，同引数の節の階層表示では，ゴールのオブジェクトの中には，節のオブジェクトが全て同じ大きさになるように入っている．これは，どの節も同じように操作できるようにするためである．しかし，節が多くなってくると，ゴールのオブジェクトの中に一杯となり，操作がしにくくなる．

これを解決する方法としては，節のオブジェクトのうち，1 つだけを大きく表示することで，操作がしやすいようにし，残りは小さく表示して節の数を知ることが出来るようにする，という手法が考えられる．また，同名，同引数の節のうち，重要な節を 1 つだけ表示する，という手法も考えられる．もし，他の節を見たい場合には，何らかの操作で，表示する節を入れ替える．

節のオブジェクトの表示に関する，これらの手法については，今後の実装を検討していきたいと思っている．

6 まとめ

本論文では，引数を表す引数オブジェクトの付加による，引数の区別，引数の入出力モードの明示を提案し，実装した．この手法により，引数を明確に表現することが可能となった．また,「ゴールと同名，同引数の節」,「節のその処理の定義」のオブジェクトを階層表示で表示するオブジェクトの階層表示の手法を提案し，実装した．この手法により，同名，同引数の節や節の定義のグラフが分散することが防止でき，まとめて扱えるようになった．

今後は，節のオブジェクトの表示方法や引数オブジェクトの操作のしやすさを考慮して，引数オブジェクトとオブジェクトの階層表示や他の表現手法を検討していく予定である．また，3D-PP の機能として，テキストから 3 次元グラフに変換する、または 3 次元グラフからテキストに変換する機能の実装を行う予定である．

参考文献

[1] B. A. Myers. Taxonomies of Visual Programming and Program Visualization. *J. Visual Languages and Computing*, Vol. 1, pp. 97–123, 1990.

[2] Kenneth M. Kahn and Vijay A. Saraswat. Complete Visualizations of Concurrent Programs and their Excutions. *Workshop on Logic Programming Environments*, pp. 30–34, 1990.

[3] Ken Kahn. Concurrent Constraint Programs to Parse and Animate Pictures of Concurrent Constraint Programs. Technical Report SSL91-16/P91-00143, XEROX PARC, 1991.

[4] 田中二郎. ビジュアルプログラミング. ビジュアルインタフェース –ポスト GUI を目指して–, bit 別冊, pp. 65–78. 共立出版, 1996.

[5] Takashi Oshiba and Jiro Tanaka. “3D-PP”: Visual Programming System with Three-Dimensional Representation. *Proceedings of International Symposium on Future Software Technology*, pp. 61–66, 1999.

[6] 宮城幸司, 大芝崇, 田中二郎. 三次元ビジュアル・プログラミング・システム 3D-PP. 日本ソフトウェア科学会第 15 回大会論文集, pp. 125–128, 9 1998.

[7] 古川康一, 溝口文雄. 並列論理型言語 GHC とその応用. 共立出版, 1987.

[8] 大芝崇. 3 次元モデリングツール “Claymore”:付加情報によって強化された直接操作. 日本ソフトウェア科学会第 15 回大会論文集, pp. 161–164, 1998.

[9] 宮下貴史, 田中二郎. 3 次元スプリングモデルと拡張直接操作手法の統合. 情報処理学会論文誌, Vol. 42, No. 3, pp. 565–576, 2001.

[10] 田中二郎, 太田裕紀子. GHC プログラムの視覚的入力システム:FE’92. 電気情報通信学会 研究報告 COMP 90-67, pp. 53–62, 1990.