三次元ビジュアル・プログラミング・システム 3D-PP

Three-dimensional visual programming system 3D-PP

宮城 幸司[†] 大芝 崇[††] 田中 二郎[†††]
Koji MIYAGI Takashi OSHIBA 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, University of Tsukuba
††筑波大学 大学院 博士課程 工学研究科
Doctoral Program in Engineering, University of Tsukuba
†††筑波大学 電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

概 要

我々は従来の視覚化手法である二次元の視覚化ではなく、三次元による視覚化の新しい手法を取り入れた実用的なビジュアル・プログラミング・システム (VPS) を提案する。本システムは並列論理型言語を VPS の対象にしており、並列論理型言語の特徴であるゴールの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を提案する。また三次元図形を扱うことで、スケーラビリティ、レイアウトの自由度、表現力を向上し、実用的なプログラムを記述できるようにした。ユーザは三次元アイコンを使用して図形を組合せていくだけで、プログラミングすることができる。実行の様子も記述したビジュアル・プログラムがアニメーションにより変化していく形で視覚化した。

1 はじめに

VPS は二次元図形を使用して数々の研究がなされてきた [1] [3] [4] [6]。しかし二次元 VPS は、コンピュータの限られた画面サイズでは、一度に大規模なプログラムを扱えないという問題 [9]を指摘されてきた。そこで、三次元図形を用いた VPS が研究されるようになった [2] [5] [10]。三次元図形を使用すれば、次元が一つ増え一画面で大規模なプログラムを扱うことができるようになる。これらの三次元 VPS の内 [2] [5]は我々と同様、並列論理型言語を視覚化の対象としている。また、 [10]も、パターン書き換えによって実行を行うという点では、類似しているといえる。我々は、並列論理型言語の特徴である実行過程に着目し、その様子を素直に視覚化した。そのため従来の表現手法では理解しづらかった、並列論理型言語の振る舞いをユーザに提供できる。

2 VPS の三次元化

我々は三次元図形による視覚化によって、ユーザが享受できるメリットに以下のようなものがあると考えている。

2.1 プログラムの量

ビジュアル・プログラミングは、図形でプログラミングをするため量がかさばる傾向にある。その解決法として、 fisheye-view [8]などいくつかの手法が提案されている。それでも二次元図形では、一画面に表示できるプログラムの量に限界がある。プログラムを三次元空間に配置することにより、一画面で扱えるプログラムの量は飛躍的に向上する。

2.2 リアルな表現力

三次元図形を用いることにより、よりリアルにプログラムを表現することができる。本システムでは、親ゴールとサブゴール群との関係を包含関係で表示するが、その様子を、まさに親ゴールの中にサブゴール群が存在する形で表現できる。また、それぞれのプログラム要素に三次元図形を割り当てるわけだが、同類の要素には上下方向から見た形を統一し、横から見ると各々の要素に合わせた形にするといった表現ができる。このような表現は二次元図形では難しい。また、実行の過程をアニメーションで表示する場合、二次元図形と三次元図形では表現力に多大な差が出てくる。

2.3 レイアウト

二次元図形では、いくら見やすくレイアウトしようとしても、図形が交差したり重なったりすることが多々ある。しかし三次元図形では、自由度が一つ増えるため、重なりや交差を避けることができる。

3 並列論理型言語と VPS

VPS の設計に際して、並列論理型言語 KL1 [7]を対象にした。並列論理型言語はプログラムが簡潔であるなど VPS の対象としてふさわしい [11]。我々は、並列論理型言語の特徴を前面に押し出し、並列論理型言語を素直に視覚化した。よって、ユーザは対象言語のプログラム構造や、実行過程を理解しやすく、可読性のよい VPS ということができる。並列論理型言語 KL1 においてプログラムの実行は、ゴールをサブゴールに書き換えることによって行われる。各々のゴールは、条件が満たされたものから順次書き換えられていく。ゴールは書き換えることのできないゴール “true” になった時点で書き換えを終了する。ゴールを “true” になるまで書き換えていく過程をリダクションと呼ぶ。

3.1 ゴールのリダクションの表現法

並列論理型言語は、手続き型言語と異なりどのゴールからリダクションされるかは処理系依存である。処理系は処理可能なゴールに対して、順次リダクションを試みる。リダクションされるか否かはガード部分に記述される条件を満たすデータが揃うか否かで決まる。データが揃わない場合は、そのゴールのリダクションを中断し、別の処理可能なゴールのリダクションを試みる。データが揃えば、ゴールがリダクションされ、サブゴールが処理可能なゴールとして生み出される。処理可能なゴールが無くなった時点で実行は終了する。リダクションの条件が満たされたものが複数ある場合は、それらが並列にリダクションされる。この様子を明示的に表現するために、各々のゴールを三次元空間に浮遊させて示した。

まず処理系は、各ゴールに対してリダクションの試みるが、これをゴールの点滅表示で表現する。リダクションの条件が満たされるかどうかを、入力データと条件部分とを照合するアニメーションで表現する。条件が満たされない場合は、ゴールの点滅表示をやめ、他のゴールへのリダクションを試みる。リダクションを行えるデータがそろっている場合は、サブゴール群を生み出して消滅する。プログラムの実行は、このようにリダクションの繰り返しによって進んでいく。画面中にリダクション可能なゴールがなくなったとき、実行は終了する。

3.2 論理変数

各ゴールの関係は論理変数によって表現される。論理変数は手続き型言語でいう変数と異なり、一度値が具体化すると二度と変化しない。具体化された論理変数と具体化されていない論理変数の表現を異なるものにした。このことによりテキストでは分かりにくかった論理変数の振る舞いを、明示的に表現することを可能とした。

4 三次元 VPS“3D-PP”

我々は、三次元 VPS“**3D-PP**” を実装した。このシステムの画面イメージを図 1に示す。これは、1000 番目の素数を計算し出力するプログラムを記述したところを表わしている。このビジュアル・プログラムに対応する KL1 のコードを図 2に示す。primes は 1000 番目の素数を計算するゴールを表わす。 gen_primes は素数列を生成し、 pkup がその素数列から 1000 番目の素数を取出す。 primes が出力したデータを io:outstream が格納する。

ユーザは任意の階層までを画面に表示でき、図 1では gen_primes、 pkup 以下の階層は表示させていない。さらに gen_primes、 pkup の中身として

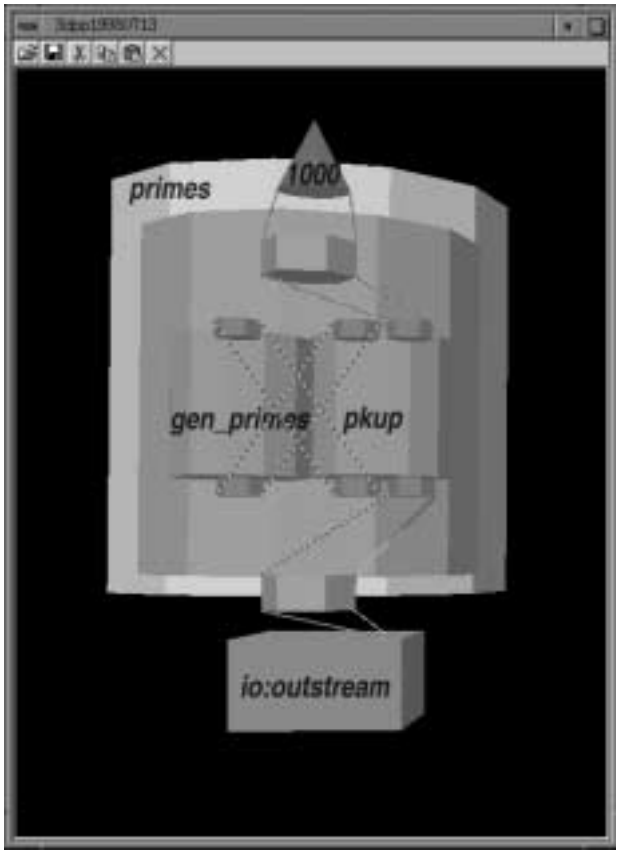

図 1　3D-PP の画面イメージ

gen、 shift、 pkup(再帰ゴール) などがある。

4.1　3D-PP の操作・実行例

本システムでは、ゴールやアトムなどのプログラム要素は三次元図形で表現される。ユーザはこれらのプログラム要素を表わす三次元アイコンを使用して三次元図形を組合わせることでプログラミングをすることができる。プログラミングの過程を図 1のプログラムを例にとって説明する。

まず、 1000、 primes、 io:outstream を画面上に作成する。 gen_primes、 pkup を作成し primes の中に入れる。各要素の引数を論理変数を意味する線で結ぶ。同様に gen_primes、 pkup の中身も作成していくことで、プログラムを完成することができる。

図 1のプログラムの実行イメージを図 3に示す。図 3-a は、 gen_primes がリダクションして、 gen と shift が生まれた時点のスナップショットである。 gen は自然数列を生成し、 shift はその数列から素数の倍数を除去してサブゴールに渡す。図 3-b は、 shift と pkup がリダクションした状態であり、 2 の

```
:-module main.
main:-primes(1000,LP),io:outstream([print(LP),nl]).
primes(Nth,LP):-gen_primes(AB,Ps), pkup(Ps,Nth,AB,LP).
gen_primes(AB,Ps):-gen(AB,2,Ns)@lower_priority,
        sift(Ns,Ps).
gen(abort,_,Ns):-Ns=[].
alternatively.
gen(AB,N,Ns):-Ns=[N|Ns2],N2:=N+1,gen(AB,N2,Ns2).
sift([],Zs):-Zs=[].
sift([P|Xs],Zs):-Zs=[P|Zs2],filter(P,Xs,Ys),sift(Ys,Zs2).
filter(_,[],Ys):-Ys=[].
filter(P,[X|Xs],Ys):-X mod P=\=0 |
        Ys=[X|Ys2],filter(P,Xs,Ys2).
filter(P,[X|Xs],Ys):-X mod P=:=0 |
                    filter(P,Xs,Ys).
pkup([P|_], Nth,AB,LP):-Nth=:=1 |
        LP=P, AB=abort.
pkup([_|Ps],Nth,AB,LP):-Nth=\=1 |
        Nth2:=Nth-1,pkup(Ps,Nth2,AB,LP).
```

図 2　KL1 コード

倍数を除去する filter が生まれている。素数が一つできたので、 pkup の数字が 999 に減っている。図 3-c は、素数が 2、 3、 5、 7 と生成され、 pkup の数字が 996 になった状態を表わしている。 pkup の数字が 1 になった時点の素数が 1000 番目のの素数として、出力される。図 3-d は、実行の最終段階を示しており、 1000 番目の素数が出力データとして io:outstream に格納された状態である。

4.2　3D-PP のシステム構成

本システムは 3 部構成になっており、「GUI 部」がメインシステムである。「GUI 部」で記述されたプログラムに対して、他のシステムが情報を与えるという形で VPS に求められる機能を満たす。新たな機能の追加は「GUI 部」の内部表現を加工して返すシステムを外付けすればよい。 3 つのサブシステムは本システム用に設計した内部表現によってデータを交換する。三次元図形は全て内部表現によって記述することができる。「GUI 部」はユーザとのインタフェース全般を扱う。プログラムの編集、実行などの命令を受けたり、その命令の結果を三次元図形としてフィードバックする。「トランスレータ」は内部表現と KL1 のコードとを相互に変換する。「KL1 エンジン」は記述されたプログラムの実行をトレースし、「GUI 部」に結果を返す。

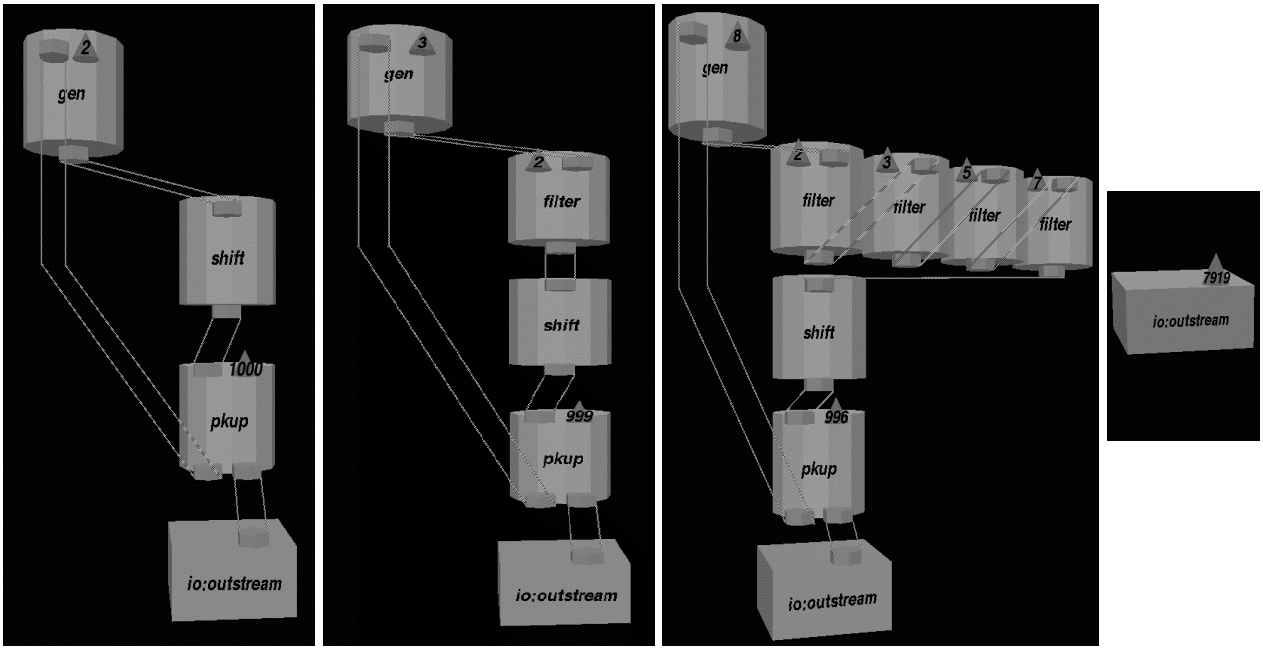

図 3　実行イメージ

5 結　　論

我々は並列論理型言語を対象に VPS を設計、実装した。並列論理型言語の特徴である、ゴールの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を提案した。これによって、テキスト言語や従来の表現手法では理解しづらかった並列論理型言語の振る舞いをユーザに提供できる。また、視覚化を三次元図形によって行ったため、ユーザは大規模プログラムを扱えること、二次元に比べ表現力が向上しプログラムの可読性が向上、レイアウトの自由度の増加による図形の一覧性の向上などのメリットを得ることができる。

その反面、三次元空間に大規模プログラムを表示することは、かえってユーザに過大な情報を提供することになり、混乱を招く危険性もある。今後の課題として、いかにユーザに有用な情報だけを提供し、その他の情報を目立たなく表示するかを検討する必要がある。

参考文献

[1] Masashi Toyoda, Buntarou Shizuki, Shin Takahashi, Satoshi Matsuoka and Etsuya Shibayama：Supporting Design Patterns in a Visual Parallel Data-flow Programming Environment, *Proc. 1997 IEEE Symposium on Visual Languages*, 1997.

[2] 高田 哲司, 小池 英樹：VisuaLinda: 並列言語 Linda のプログラムの実行状態の３次元視覚化, インタラクティブシステムとソフトウェア II: 日本ソフトウェア科学会 WISS'94, pp. 215-223, 1994.

[3] P. T. Cox, F. R. Giles and T. Pietrzykowski：Prograph: A Step towards Liberating Programming from Textual Conditioning, *1989 IEEE Workshop on Visual Languages*, Rome, pp. 150-156, 1989.

[4] http://www.toontalk.com/toontalk.htm/

[5] Marc-Alexander Najork：Programming in Three Dimentions, *Thesis for Ph.D. in Computer Science of the University of Illinois*, 1994.

[6] 田中二郎：ビジュアル・プログラミング・システム PP, ビジュアルインタフェースの研究開発報告書, 財団法人 日本情報処理開発協会, March 1994.

[7] KLIC 講習会テキスト -KL1 言語編-, 財団法人 新世代コンピュータ技術開発機構 作成, 財団法人 日本情報処理開発協会開発研究室 改訂, 1995.

[8] George W. Furnas：Generalized fisheye views, *In Proceedings of ACM CHI'86 conference on Human Factors in Computing Systems*, pp. 16-23, Association for Computing Macinery, 1986.

[9] Margaret Burnett, et al：Scaling Up Visual Programming Languages, *IEEE Computer*, Vol.28 No.3, pp.45–54, March 1995.

[10] 山本格也：ビットマップ型言語におけるモジュール機能, 情報処理学会論文誌, Vol.38, No.12, pp. 2544-2551, 1997.

[11] 財団法人 機械システム振興協会：並列システムのための視覚的インタフェースの構成に関する調査研究報告書, 財団法人 新世代コンピュータ技術開発機構, 1995.