ハイパフォーマンス
コンピューティング 63－7
（1996. 10. 4）

並列化支援のためのデータ依存の3次元視覚化

木和田 智子° 笹倉万里子† 中西 恒夫° 城 和貴° 福田 晃° 荒木 啓二郎‡
°: 奈良先端科学技術大学院大学 †: 岡山大学工学部 ‡:九州大学大学院

概要

近年、グラフィックワークステーションの普及により、3次元を使った情報視覚化が多く行なわれるようになってきている。しかし、単に量的に多くの情報を表示するためだけでなく、3次元を使わなければ表示が難しい情報を視覚化した例はまだ少ない。本稿で、我々は、3次元を有効に使った視覚化法の一つとして、並列化支援のためのデータ依存関係の視覚化法を提案する。この方法は、3次元空間内に複数の変数のデータ依存関係を一度に表示することができる。この方法で視覚化された図を見ることで、ユーザは、並列化を妨げるデータ依存や無駄に使われている変数／配列要素の発見を容易に行なえるようになる。

3D Visualization of Data Dependence for Parallelization Support

Satoko Kiwada° Mariko Sasakura† Tsuneo Nakanishi° Kazuki Joe° Akira Fukuda° Keijiro Araki‡
°: Nara Institute of Science and Technology †: Okayama Univ. ‡: Kyushu Univ.

Abstract

Recent improvement of graphic workstations allows a frequet use of 3D visualization of information. 3D visualization has a great advantage of treating complex information as well as a large amount of data. In this paper, we propose a 3D visualization method for data dependence which parallelizing compilers may detect. Using the method, users can analyze their parallel programs to find redundant dependence which keep their programs to be parallelized. This is a function of NaraView, which is an integrated environment of a parallelizing compiler and its user-interface with 3D visualization.

1 はじめに

情報視覚化とは、ソフトウェア開発、情報検索、ユーザインタフェース等の分野において、扱われる様々なデータを視覚化することであり、情報を文字ではなく、図にすることにより、人間にそれに対するより早く、より深い理解をあたえることを目的としている。

近年では、低価格で高性能なグラフィックワークステーションが普及し、より簡単にコンピュータグラフィクスによる図を作成できるようになってきた。また、計算機の能力の向上によって、より複雑で多くの情報を扱えるようになってきた。このため、従来の2次元に代わる3次元視覚化が、研究開発されてきている。

このような先駆的な研究には、大規模知識ベースをノードとリンクで視覚化したSemNet[4]が知られている。また、階層構造を木で表示したCone Tree[6]、線形データを壁上に表示するPerspective Wall[5]等、その多くは、3次元の奥行き方向を利用した表示領域の効果的利用を行なっている。

しかしながら、上記視覚化システムでは、3次元視覚化を行なう必然性は明示的に示されてはおらず、2次元に比べ、図の複雑さが増した分、情報に対する誤認や認知負荷をもたらす傾向がうかがわれる。

本稿では、3次元視覚化の必然性を示すために、並列化支援のためのデータ依存関係の視覚化を提案し、これをもって、3次元情報視覚化の有効性を示す。

データ依存関係とは、あるプログラムが並列に実行できるかどうかを判定するための主要な情報の一つであり、主に自動並列化コンパイラで用いられる。

自動並列化コンパイラは、逐次プログラムを、自動的に並列プログラムに書き換えることを目的としているが、現時点では、完全に自動的に並列プログラムに変換することは困難である。その主たる理由の一つとして、逐次プログラムの並列化すべき箇所と、そこに適用すべき並列化手法の選択、およびその適用順序の判断を、自動化できないことが挙げられる。このことから、ユーザが並列化の指示を行なえるような、並列化支援環境が必要とされている。

われわれは、並列化を支援するために、視覚化が有効であると考え、並列化支援視覚化システムNaraView[7]を構築している。本稿で述べるデータ依存の3次元表示は、そのNaraViewの一機能として実現されている。

以後、2節では、並列化支援における視覚化システムの意義と、われわれが提案する視覚化システムNaraViewの概要について述べ、3節ではデータ依存の3次元表示法について解説し、4節でその3次元表示の例をいくつか提示するとともに、それらが並列化に有効であることを示す。最後に5節で結論と問題点、今後の課題を述べる。

2 並列化支援と視覚化

図はあくまでもその情報に対する直観的な理解をあたえるものであり、正確な内容を示すものではない。並列化支援視覚化システムを、実用的なものとするためには、視覚化システムと、正確な内容を取り扱う解析システムを組み合わせ、対話させる必要がある。我々の提案する視覚化システム NaraView は、並列化コンパイラ Parafrase-2[3] と連動し、Parafrase-2 の解析結果をもとに、与えられたソース・プログラムを視覚化する。ユーザは、NaraView の出力する図から得られる情報をもとに、コンパイラに指示をあたえ、プログラムを並列化する。

NaraView の目的は、プログラムのどこが並列に実行されるか、また、どこに並列化できる可能性があるかを、視覚化されたプログラムから読みとることである。並列化のために、NaraView は次に挙げるビューを用意している。ビューとは、情報のある特定の特徴を切りだし、見易くしてくれるメカニズムである。

- プログラム構造ビュー
- CFG(control flow graph) ビュー
- データ依存関係ビュー
- ソースコードビュー

NaraView は、逐次またはすでに並列化されたプログラムを入力とし、まず、ユーザにプログラム構造ビューで、直観的にプログラム全体の情報を知らせる。プログラム構造ビューは、ノード（各ノードはソースコードの一文に相当する。）の集まりである**物体**を表示する。ノードは、プログラムのながれ、並列度、ループ構造に基づいた階層、の３つの値の組で定まる位置に表されている。ユーザはこのプログラム構造ビューから、次にどのループを並列化すればよいかを推定できる。プログラム構造ビューに関しては文献 [7] を参照されたい。次に、特定したノードをクリックしフォーカスすると、NaraView は、本稿で提案するデータ依存関係ビューを使い、そのループ内に存在するデータ依存を３次元表示する。ユーザは、データ依存関係ビューとソースコードビューを併用して、どのように並列化するかの指示をコンパイラに与える。

3 データ依存の３次元表示

データ依存には、様々な形態のものが知られているが、現在の自動並列化コンパイラに関する研究は、ループに関するものが主流を占めるため、われわれがこれから提案するデータ依存の３次元表示も、ループ内のデータ依存を対象とする。

3.1 背景と基本方針

従来、データ依存は、データ依存グラフ (DDG)[2] やイタレーション空間を表示するというような方法 [1] で、視覚化されている。

DDG はデータ依存の状態を表示しているが、どの並列化手法を使うべきかに関して、十分な情報をあたえてはくれない。なぜならば、どの変数やどの配列要素が、プログラムの並列化を阻害する依存を生じているかが、わからないためである。

イタレーション空間の表示により、ある配列内のデータ依存の詳細を、x-y 平面上に表すことができる。この時、各イタレーションの実行中に、個々の配列のどの要素がアクセスされるのかを示すことはできるが、異なった配列間にまたがる、依存関係は表現できない。

よって、イタレーション内の複数の配列や変数の動きを追跡する、より一般的なアプローチが必要である。

そこで、われわれはデータ依存の３次元表示を提案する。基本概念は、あるループ内の複数の配列要素や変数に関与する依存情報を、３次元的に視覚化することである。３次元空間において、配列や変数を x-y 平面上にマッピングし、イタレーションを z 軸方向に対応させる。ユーザは各イタレーションにおける、複数の配列や変数に関与する依存関係を知ることができ、データの分割や割り当てを考慮するのに有効である。

3.2 定義と説明

データ依存関係ビューは、ソースプログラムを入力として、データ依存を、色付けされた**キューブ**と**柱**（以後、柱のことを**ポール**と呼ぶ。）からなる**物体**として視覚化する。生成された３次元**物体**は、ループ内の変数や配列要素へのアクセスを順番に表示している。

キューブは四つの情報を有する。そのうち三つは以下に示すような情報であり、(x, y, z) の座標値として、３次元空間での配置に対応される。

x,y: 変数や配列要素の配置
z: イタレーション数

イタレーション数は、多重ループの場合、辞書式順序で並ぶ。

キューブの有するもう一つの情報は、変数や配列要素の各イタレーションでのアクセスのされ方であり、以下の３種類にわけられ、色の違いで表現化される。

- **読み込み**
- **書き込み**
- **同一イタレーションでの読み込みと書き込み**

３番目のキューブは、底面を**読み込み**、上面を**書き込み**と考える。キューブとキューブをつないでいるポールには以下の２種類がある。

- **ライフタイム・ポール** 同一変数や配列要素において、書き込みのキューブから、次の書き込みの直前の読み込みのキューブをつないだ物
- **アンチ-ディペンデンス・ポール** 同一変数や配列要素において、読み込みのキューブから、それ以後の最初の書き込みのキューブをつないだ物

ライフタイム・ポールは、変数の寿命を、アンチ-ディペンデンス・ポールは、逆依存の存在を意味している。

これらのキューブとポールが物体の主要素であるが、ユーザが物体を理解し易くするために、データ依存関係ビューは、次の2種類のz軸に垂直な半透明の平面を用意している。

- **ループ・グリッド** ネスト内の各ループの最初のイタレーションを示す。
- **AVD(array-variable disposition)** マップ 変数や配列のx-y平面上でのレイアウトを表示。

図2は、データ依存関係ビューが、キューブ、ポール、平面というそれぞれの要素をどのように組み合わせて、物体を形成しているかを示している。

```
DO J=1,2
  S=0                          [1]
  DO K=1,2
    S=S+TEMP[K]*B[K,J]         [2]
  EndDO
  A[I,J]=S                     [3]
EndDO
```

図 1: サンプルプログラム 1

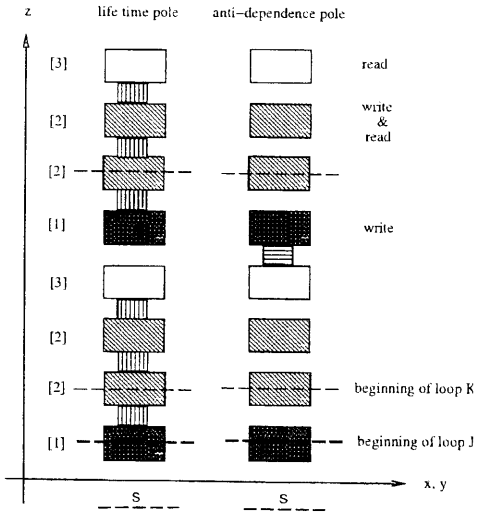
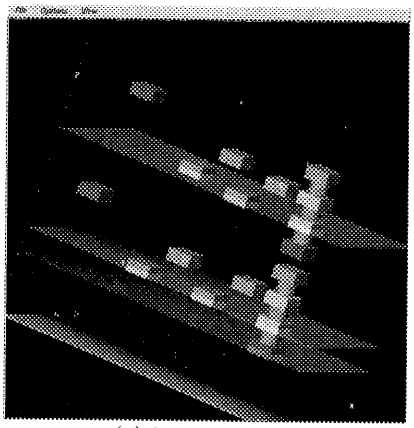


図 2: 図1中の変数Sへのアクセスの視覚化方法

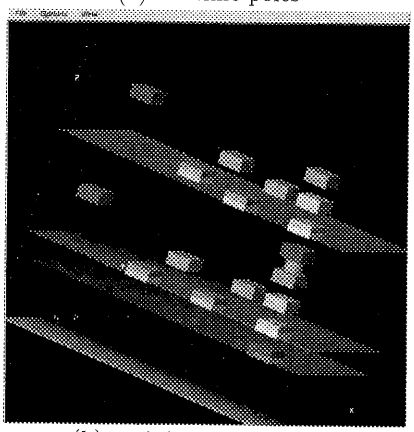
この図2は、図1のプログラム実行時の変数Sへのアクセスの状態を表している。

白、黒、斜線の長方形は、順に、**読み込み、書き込み、同一イタレーションでの読み込みと書き込み**のキューブである。それらのキューブをつないでいる縦縞の長方形はライフタイム・ポールを、横縞の長方形はアンチ-ディペンデンス・ポールを示している。キューブ上に位置している破線はループ・グリッドであり、一番下に位置する破線とSの文字がAVDマップである。左端に縦に並んでいる[数字]は図1のソースコードに書かれている[数字]と対応している。この図2においては、変数sへのアクセスの順番、つまり、プログラムの流れは、z軸方向の下から上へ進む。

NaraViewにおける実際の表示では、キューブやポールは色分けされているため見分けやすく、直観的に依存状況が理解しやすいので、並列化やデータ分割に有効な情報を得ることができる。図3が図1のプログラムの3次元データ依存表示である。



(a) life time poles



(b) anti-dependence poles

図 3: 図1のデータ依存の3次元表示：
右端のポールが、変数Sのデータ依存表示である。左から順に、配列A, B, TEMPと並んでいるが、キューブだけでポールが見られないことから、これらの配列には依存が存在しないことがわかる。

3.3 操作

3次元表示を行なう上で重要な問題の1つは、その操作性、対話性である。2次元の場合、視点の移動は、おもに垂直または平行な方向に限られるが、3次元の場合、回転が意味を持つようになる。重なっ

て見えなくなってしまったものを見るためだけではなく、さまざまな角度からの観察により、新たな情報を得ることが可能である。また、1次元増えたことによる、情報量の増加からくる認知負荷を軽減するためには、表示内容に適した情報の省略または抽象化が必要である。データ依存の3次元表示では、メニューから、次のような操作が可能である。

- 物体の回転・拡大・縮小
- 2種類のボールの表示切り替え
- 半透明平面の表示・非表示
- AVD マップの表示・非表示

ユーザの希望・目的を反映した視覚化システムとするために、物体に対して次のような変更も行なえる。

- AVD マップ上での、変数や配列要素のマッピングの変更
- ループ・インターチェンジ等のための、イタレーションの順番の変更
- データ依存に関する情報の選択
  1. データ依存に関する全ての情報を視覚化
  2. 深いループに関する情報の非表示
  3. 個々の配列要素ではなく、配列要素の塊に対するアクセスの表示

```
      DO 2000 i = 1,n - 1
        DO 1600 k = i + 1,n
           m = aary(idx(k),i) / aary(idx(i),i)
           CDOALL 1550 j = i + 1,n + 1
             aary(idx(k),j) = aary(idx(k),j)
                            - m * aary(idx(i),j)
1550         CONTINUE
1600       CONTINUE
2000     CONTINUE
```

図 4: サンプルプログラム 2

```
      do 2000 i = 1,n - 1
        do 1600 k = i + 1,n
           m(k) = aary(idx(k),i) / aary(idx(i),i)
           cdoall 1550 j = i + 1,n + 1
             aary(idx(k),j) = aary(idx(k),j)
                            - m(k) * aary(idx(i),j)
1550         continue
1600       continue
2000     continue
```

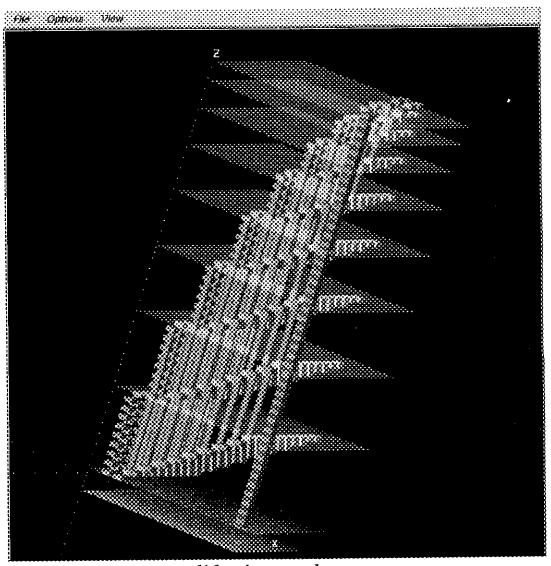
図 5: サンプルプログラム 3

4 プログラムへの適用

ここでは、データ依存の3次元表示の例を二つ用いて、その有効性を示す。一つ目は並列化手法の適用を、二つ目はメモリーの効果的利用を、促進する例である。

4.1 表示例 1

図 4はガウスの消去法のプログラムを、並列化コンパイラ Parafrase-2 で並列化したプログラムの一部分であるが、この並列化は十分とは言えない。[1]まず、データ依存関係ビューを使って、ループ内に存在している変数や配列要素の依存を観測する。図 6がそのライフタイム・ボールと AVD マップの表示である。これを見ると、変数 m に対するライフタイム・ボール間にアンチ-ディペンデンス・ボールが観察される。[2]このことから、変数 m に依存があり、並列化の妨げとなっていたことがわかる。そこで、変数 m に対して並列化手法の一つであるスカラー・エクスパンション [2] を適用してみる。その結果が図 7である。変数 m へのアクセスを表すキューブは、図 6では z 軸方向に平行に一直線状に並んでいたのが、図 7では階段状に分解されている。このことから、変数 m が展開されている様子がうかがえる。このようにして、より並列化されたプログラム、図 5をえることができた。

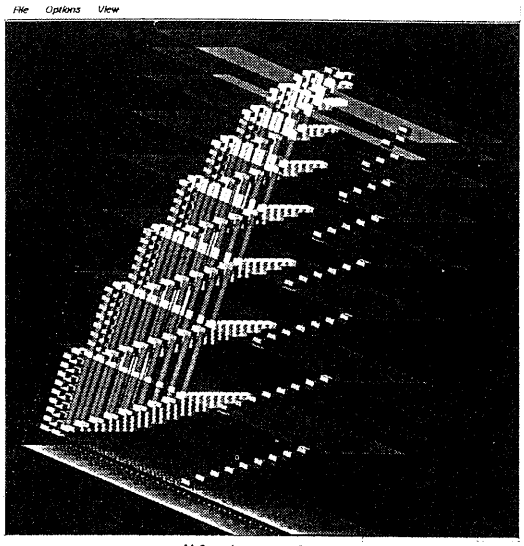


life time poles

図 6: 図 4のデータ依存の 3 次元表示

[1]本稿では述べないがプログラム構造ビューを利用することにより、並列化不十分な箇所が容易にわかる。

[2]本稿では白黒印刷のために観測が難しいが、実際には容易に観測される。



life time poles

図 7: 図 5のデータ依存の 3 次元表示

4.2 表示例 2

図 8のプログラムは行列の積計算を表している。図 9は配列 A, B, TEMP と変数 S が、どのようにアクセスされるかを示している。(a) はこれらの変数や配列の寿命を、(b) は逆依存を、それぞれ 3 次元表示したものである。

(b) の左側に注目してみると、ユーザは、そこに存在するポールによって、配列 TEMP と変数 S の逆依存が並列化を妨げているという、重要な情報を得ることができる。

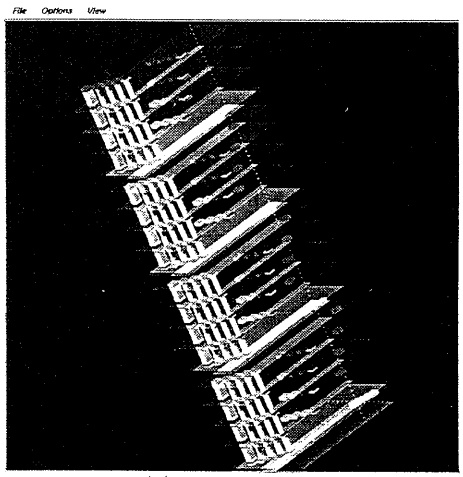
そこで、表示例 1 と同様に、配列 TEMP と変数 S をスカラー・エクスパンションしてみると、逆依存の存在を表すポールがなくなり、[3]サンプルプログラム 4 が並列化可能であることがわかる。

さらに、プログラム 2 をプロセッサ数 2 で実行していると考え、ストリップ・マイニング [2] で並列化を試みる。ストリップ・マイニングした後のデータ依存の状態を表示している図 10を見ると、配列 S へのアクセスを示すキューブが、階段のように並んでおり、それぞれの要素が、短い間しかアクセスされていないことがわかる。また、配列 TEMP の 4 分の 1 の要素の寿命が、プログラム実行の最初の 4 分の 1 で終っていることから、配列 TEMP と 変数 S が効率よく利用されていないこともわかる。メモリーを効率的に利用するために、先ほどとは逆の順で、ストリップ・マイニングを先に、スカラー・エクスパンションを後に適用してみた。その結果が図 11である。**物体**とデータのマップの状態から、メモリーの効率的利用がなされたことを確認することができる。

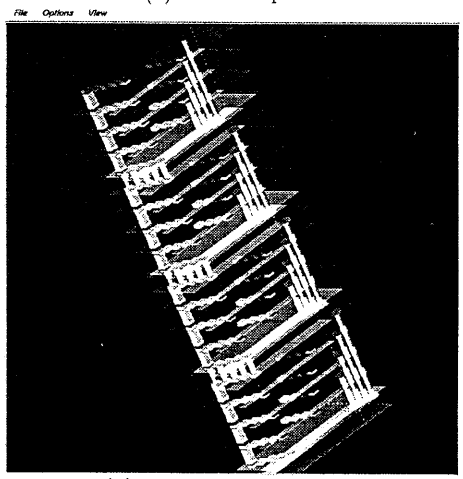
[3]紙面の都合で図は表示していない

```
DO I=1,4
  DO J=1,4
    TEMP[J]=A[I,J]
  EndDO
  DO J=1,4
    S=0
    DO K=1,4
      S=S+TEMP[K]*B[K,J]
    EndDO
    A[I,J]=S
  EndDO
EndDO
```

図 8: サンプルプログラム 4



(a) life time poles



(b) anti dependence poles

図 9: サンプルプログラム 4 の 3 次元データ依存表示

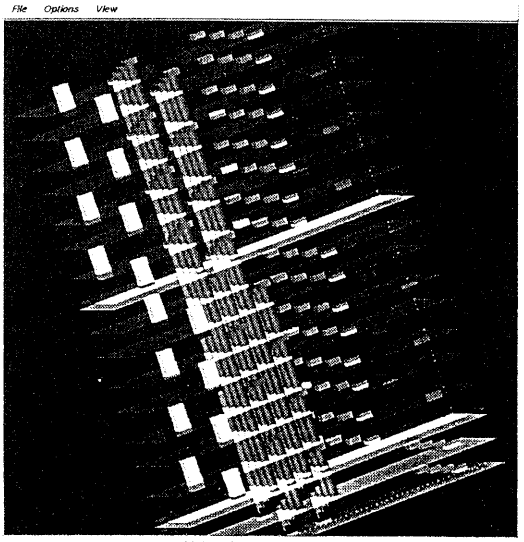


図 10: サンプルプログラム4にスカラ・エクスパンション、ストリップ・マイニングの順に適用した結果

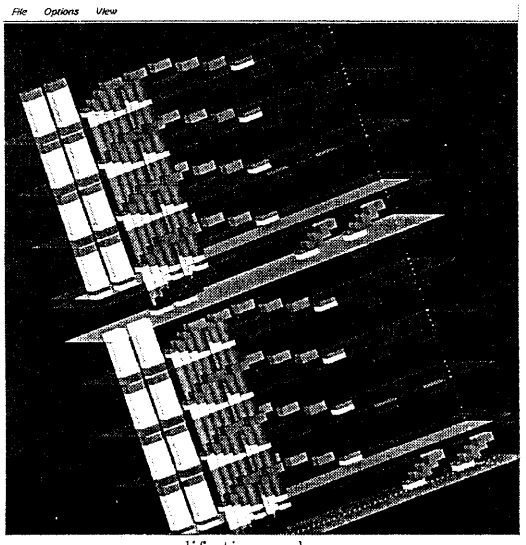


図 11: サンプルプログラム4にストリップ・マイニング、スカラ・エクスパンションの順に適用した結果

5 おわりに

本稿では、3次元情報視覚化の有効性を示すことを目的として、データ依存を3次元的に視覚化する法方を提案した。われわれが構築中の並列化支援システム NaraView が有する機能である、データ依存の3次元表示を通して、並列化に必要な情報を得られることを示した。

われわれは今後の課題として、ビュジュアル・インターフェイスを使って、ユーザが与えられたソース・プログラムを変更する機能を付け加えることを考えている。例えば、ビュー 上で、冗長な依存が検出された場合、依存を示すポール を切ることによって、ソース・プログラムを実際に書き換えることなく、依存を消去する。この操作によって、並列性をみつけることができれば、NaraView が自動的にソース・プログラムを変更する。

また、NaraView に見られるような3次元表示法の、他の情報表示への汎用性を考慮してみたい。

参考文献

[1] Banerjee, U. "Loop Transformations for Restructuring Compilers: the foundations", Kluwer Academic Publishers, 1993.

[2] Polychronopoulos, C.D. "Parallel Programming and Compilers", Kluewer Academic Press, 1988.

[3] Polychronopoulos, C.D. et al. "Parafrase-2: An environment for parallelizing, partitioning, synchronizing, and scheduling programs on multiprocessors", Proceedings of the 1989 International Conference on Parallel Processing, 1989.

[4] Fairchild, K.M., Poltrock, S.E. and Furnas,G.W. "SemNet: Three-Dementional Graph Representation of Large Knowledge Base, Cognitive resentation And Its Applications For Human-Computer Interaction(Guindon, R.(ed.))", Lawrence Erlbaum Associates, 1988, pp.201-133.

[5] Mackinlay, J.D., Robertson, G.G. and Card, S.K. "The perspective wall : detail and context smoothly integrated" Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'91), pp.173-179, ACM Press, 1991.

[6] Robertson, G.G., Mackinlay, J.D. and Card,S.K. "Cone Trees: Animated 3D visualization of hierarchical information", Proceedings of the AMC Conference on Human Factors in Computing Systems (CHI'91),pp.189-194, ACM Press, 1991.

[7] 笹倉万里子、木和田智子、城和貴、荒木啓二郎 "並列化支援視覚化システム NaraView におけるプログラム情報の3次元表示法" 並列処理シンポジウム JSPP'96, pp.267-274. 1996.