数理モデル化と問題解決 25－1
（1999．7．16）

プログラムスライシングのVRMLへの導入とその改良

丸山 博史[†,††] 荒木 啓二郎[†]

†九州大学 ††NECソフトウェア九州

プログラムスライシング技術は，プログラムの各部分の依存関係を解析し，それをグラフにてモデル化し，ある視点に関連する部分的なプログラムを求めるという各種問題に有効なものである．また，三次元物体を表示するための言語VRML(Virtual Reality Modeling Language)は，非常に有望であるが，まだツール等による支援は充分ではないため，アニメーション等の動的な部分の開発の作業が困難になってきた．この作業に関し，着目する部分に関連する対象群を表すVRML部分プログラムを取り出すことで，困難さを軽減できる．本研究では，その部分プログラム抽出の問題を，スライシング技術における従来のグラフによるモデル化をVRMLという新規応用分野に対し適用することで解決した．また，より効果的な適用として，コストが低くなるように従来のスライシング手法を改良した．以上を支援ツールとして実装し，有効性を実際に確認した．

Program Slicing Introduction into VRML and Its Refinement

HIROSHI MARUYAMA[†,††] and KEIJIRO ARAKI[†]

†Kyushu University ††NEC Software Kyushu, Ltd.

There is program slicing technique, which analizes dependency of each part of the program, and models it in a form of graph, then extracts a relevant partial program for a designated focus. It is very useful in many kinds of problems. VRML(Virtual Reality Modeling Language) is a language with a promising effectiveness in describing 3-dimensional objects. But currently, dynamic features developments such as animation can not be supported well, thus the developments becomes difficult. Extracting relevant partial 3-dimensional objects description for a focused area in the developments, the difficulty can be reduced. In order to solve such extraction problem, we applied the typical modeling with the graph into the new area of VRML, then indicated VRML slicing. Moreover, we proposed more effective slicing with lower cost. We implemented a VRML slicing support tool and evaluated its effectiveness.

1. はじめに

プログラムスライシング技術[1]はプログラム内の命令間の関係把握を容易にするものであり，スライシング基準と呼ばれるものにより与えられたある着目点に対し，関連部分をスライスと呼ばれる部分プログラムとして抽出する．そこでは，プログラムの各部分が節点，解析した依存関係が枝となる有向グラフでプログラムをモデル化し，ある着目点から到達可能な節点集合がスライスとなる．ここでは，対象とするプログラムの文脈を解析してスライスを求める静的スライスを扱う．また，静的な解析により構築するグラフに対し，ある実行に関し使用されなかったことを判定できた依存関係の枝を除去後に求める最近のハイブリッドスライス[2]と呼ばれるものも扱う．スライス導出では，いかに本質的に関連性のある部分のみを，低コストで求められるかが重要になる．

各種三次元物体を含んだ空間をインターネット上で構築できる言語VRML(Virtual Reality Modeling Language)[3][4]の第二版によるプログラムは，VRML対応のビューアにて実行し表示され，ユーザはあたかもその三次元空間内に自らが存在し，行動しているかのように，状態をいろいろ変化させることが出来る．VRMLプログラム開発において，静的な三次元物体作成は，CADのようなツールによって効果的に支援されるようになってきた．しかし，未だ動的部分の開発では，人手による高度に知的な作業が多く，特にそこで重要であるテスト，デバッグといった作業やプログラムの理解を困難にしている．

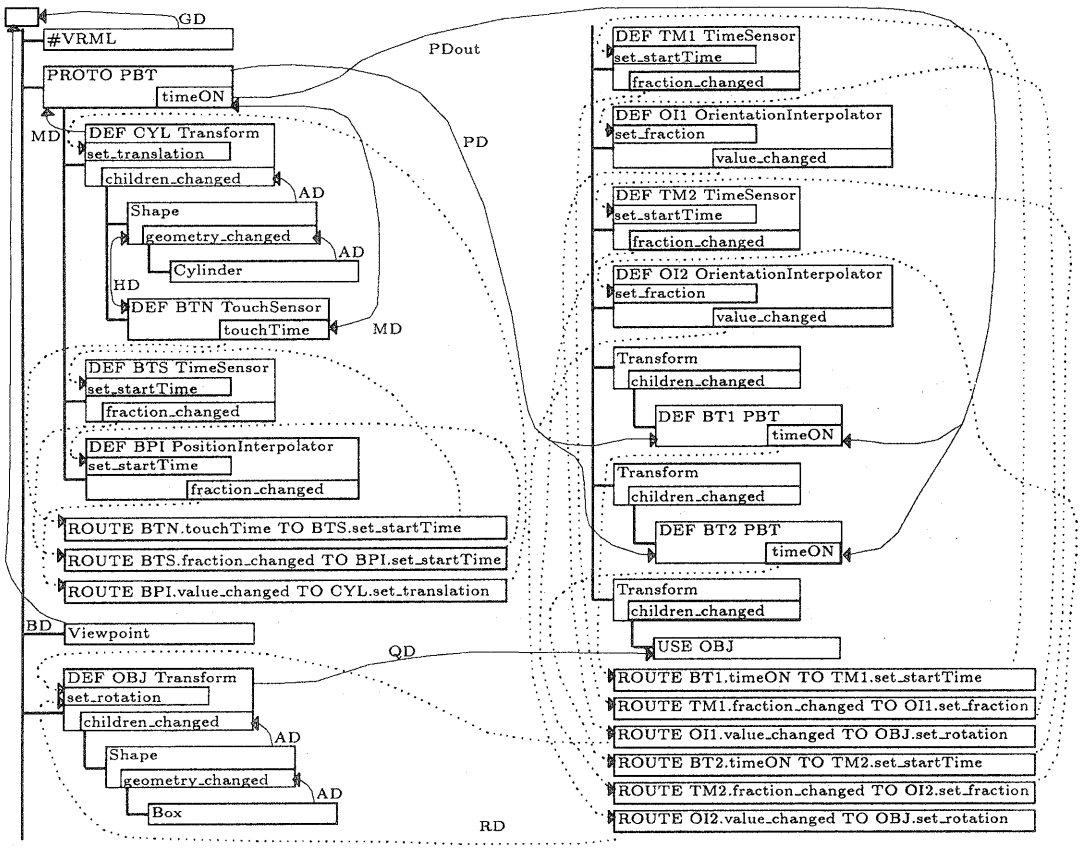

図 1 プログラム例に対する静的 VDG
Fig. 1 Static VDG for the sample program

作業に関係する部分のみをプログラム全体から小さい部分として取り出し，対象領域の概念を簡潔にすることで，この作業を支援することを考える．この部分プログラム抽出の問題の解決には，スライシング技術が適している．そこで VRML に対し，従来と同様なグラフによるモデル化およびスライシングを適用し，さらに，より効果的な適用のために改良したスライシング手法を以降で提示する．支援ツールを作成し，実験した結果も述べる．

2. VRML スライシングの方法

VRML の静的スライシングにおける基準は，任意のノード内の任意の属性から構成され，それは複数で構成されても良いとする．VRML のハイブリッドスライシングにおける基準には，さらに着目する動作の情報も含める．これは，ある実行の状況を一意に表す情報であるとする．一般のスライシングでは，変数値への影響のみを考えたが，VRML では，ビューアでの見え方を重視したいので，着目している属性の値が変わらなくとも実行時にビューアで見た際の見え方に関して影響する場合も含めることにする．ここで影響と表現しているものは，VRML における依存関係である．VRML に共通する特徴を多く持っているので，並行動作を行うオブジェクト指向言語に関するプログラムスライシングの研究[5]を参考にした．そこで提示している依存関係を一部流用し，独自に必要と思われるものを追加し，新しい依存関係の体系を構築した．

VRML プログラムを実際に実行せず，その文脈を解析し前述の依存関係を抽出するモデル化の結果のグラフを，静的な VRML 依存関係グラフ (静的 VDG) と呼ぶこととする．それに対し，VRML プログラムを実際に実行した際，影響が実際に伝わらなかった事実が確認された依存関係を，静的 VDG から削除して求まるものを動的 VDG とする．このモデル化後，基準で指示された着目点から，依存関係を逆向きにたど

表 1　例題の VRML プログラムに対する各種スライシング結果
Table 1　Each slicing result for the sample VRML program

基準における着目点	静的スライス	HS(両ボタン操作時)	HS(片方ボタン操作時)	HS(ボタン操作なし)
視点 (ViewPoint)	4 (64, 38)	4 (73, 47)	4 (79, 53)	4 (91, 71)
左ボタン	32 (101, 98)	32 (110, 107)	32 (116, 113)	24 (115, 112)
中央の立方体	63 (142, 142)	63 (151, 151)	49 (137, 134)	13 (99, 79)

(HS はハイブリッドスライスを意味する)

ることで，静的，動的 VDG から，それぞれ静的，ハイブリッドスライスが求められる．

ここでは，ある VRML プログラムを例に，実際に VDG を構築し，スライシングをおこなう．この例は，中央にある立方体を 2 つのボタンで回転させるものである．さらに，上から立方体を見た景色を上方に表示する．それぞれボタンは，ある 1 つのプロトタイプの別々のインスタンスとして存在する．各ボタンはクリックされると一瞬下がった後に上へ戻る．この例の実行では，**図 3** の左上のような画面が表示される．この VRML プログラム例に対する静的 VDG を**図 1** に示す．ここでは，各依存関係を矢印付きの線で示しているが，親ノードから子ノードへの AD は太線のシーングラフの関係で示されるので省略しており，RD は特別に矢印付きの破線で示している．

スライス例 1　まず，左のボタン (ノード名は BT1) を基準とした静的スライスは，ボタンのプロトタイプ，左ボタンのインスタンスから成り，スライスの規模は元のプログラムの約 46%になったこのスライスを実行すると，図 3 の右上のような画面となり，左側の 1 つのボタン要素しか表示されないが，左のボタンをクリックするとボタン部分が上下に動く．

スライス例 2　中央の立方体 (図 1 の左側最下のノード「Box」) を基準とした静的スライスは，主にボタンのプロトタイプ，両ボタンのインスタンス，さらに立方体関連の ROUTE 文から成り，90%になった

スライス例 3　中央の立方体という同じ基準で，左ボタンのみをクリックした場合のハイブリッドスライスを求める．プログラムの動作履歴を参照し，図 1 の静的 VDG から，右下の 3 つの ROUTE 文が中継している RD を除去できる．その結果得られる動的 VDG から，主にプロトタイプ，左ボタンのインスタンス，立方体，それらを関連付ける ROUTE 文から成るスライスを，70%の規模で導出できた

スライス例 4　さらに同じ基準に対し，左右どちらのボタンもクリックしない場合におけるハイブリッドスライスを求める．その場合，全 RD を除去でき，わずか約 19%となった

3. スライシングの改良

まず，人的コストを減らせるように改良する．ハイブリッドスライシングに関し，VRML においても，文献 2) のような，コードの付与による実行情報取得を

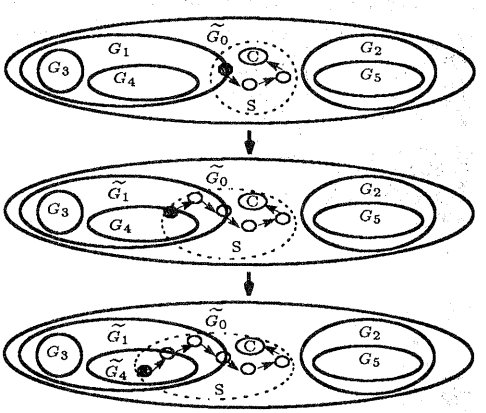

図 2　遅延解析によるスライシングの概念
Fig. 2　The notion of slicing with delayed analysis

採用することにした．実行されるスライス対象プログラムの適切な位置にその地点が実行されたかどうかを調べるための情報取得用のコードを埋め込む必要がある．本研究ではその文献での手法と異なり，そのコード埋め込み地点の決定や，そこへ埋め込む作業は，人手によらず自動的に行えるようにした．有効性を考慮し ROUTE 文をその埋め込み地点に選んだ．

さらに計算コストを減らせるような改良を考える．プログラムをいくつかの部分に分割し，実際に解析が必要になるまで解析するタイミングを遅らせるように改良する．**図 2** では，各 G_i がプログラム中のあるモジュールにて，まだ依存関係の解析をおこなっていない部分とし，内部の依存関係を解析し $\widetilde{G}_i(V_i, E_i)$ を得るとする．この図の例では，プログラム中で G_2，G_3，G_5 に対応する部分の依存関係を解析せずに着目点 C に関するスライス S を求められたため，解析に関するコストを減らすことができたことを示している．ここでは，解析を遅らせる対象となる単位となる部分として，プロトタイプ定義の部分を採用する．あるプロトタイプに対応するインスタンスの記述部がスライス対象として操作段階で初めて抽出された際に，そのプロトタイプ内の依存関係解析を行うようにする．

4. ツールの試作およびスライシングの評価

静的スライシング，ハイブリッドスライシング，さらにそれぞれにて遅延解析も行えるスライシングのツールを作成した．これを使用し，実際に前述のスライス

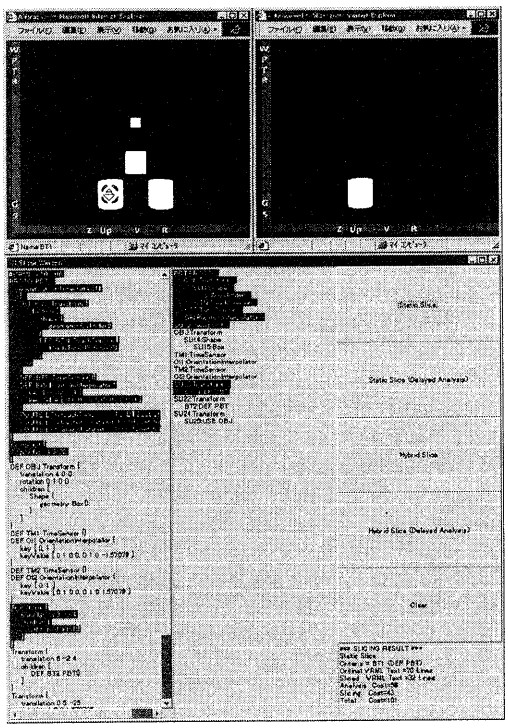

図 3 作成したスライシングツールの実行画面
Fig. 3 An execution screen of our slicing tool

表 2 VRML プログラムの例題に対するスライシング結果
Table 2 Slicing result for various sample VRML programs

規模 (L)	SS/L	HS/L	HS/SS
70	90%	70%	78%
151	55%	33%	60%
205	60%	49%	82%
平均	68%	51%	73%

例 1 を行った場合の画面例を図 3 に示す．本ツールにより，スライシングにおける基準の指示や，結果としてのスライスを把握することが容易になった．

本ツールを使って，同じスライシング基準に対し，それぞれのスライシング法によるスライスの大きさ，およびコストを比較する．コストの単位は，1 つの依存関係を抽出もしくは削除した場合，その 1 つをたどる場合，スライスとして節点を 1 つ抽出した場合，以上の全てをそれぞれ 1 としている．**表 1** に例題プログラムに対する各種スライシング結果を示す．各欄の内容は，「抽出されたスライスの行数 (従来の一括解析によるコスト，遅延解析によるコスト)」である．この結果，ある特定の実行に関するスライスを求める場合は，ハイブリッドスライスが静的スライスよりも小さくなり得ることが確認できた．また，遅延解析のスライシングにおいて 20%から 40%程度のコストを削減できる場合があることも確認できた．

スライシングにより着目するプログラムの規模を小さくできる場合，実行時にビューアのパフォーマンスが向上し，プログラム自身の可読性も向上し，スライス以外の部分についての知識が少なくとも作業が可能となり，より小さい簡潔な環境でプログラミング作業が可能になる点で，作業効率が向上できる．スライスサイズに関し，前述のプログラム例も含め，同程度の複雑さで規模 (行数を L とする) の異なる 3 つの例にて，代表的な対象に関する静的スライス (行数は SS)，および同じ対象に関するハイブリッドスライス (行数は HS) を求めた結果を**表 2** に示す．一般にハイブリッドスライスは静的スライスの 7 割程度になり，それらスライスの規模は元のプログラムの半分程度になることもあることが分かった．さらに現場において，3000 行程度プログラムから担当する部分を 500 行程度で取り出せた等，現場でも役立つこともを確認した．

5. おわりに

我々は VRML プログラム開発における課題を支援するために，三次元対象群から，着目している部分に関連する対象群を表す部分プログラムを取り出す問題に着目し．その問題解決のために従来からのグラフによるモデル化を VRML という新規応用分野に対し適用した．また，より効果的な適用として改良したスライシング手法も提示した．実際に支援ツールを作成し実験した結果，有効性を確認できた．

今後は，本ツールを強化していく他，三次元対象群から一部の対象群を取り出すという，本研究の持つ一般性，および VRML の持つオブジェクト指向性から，他の各種分野，各種言語への応用も考えたい．

参考文献

1) 下村隆夫: プログラムスライシング技術と応用，共立出版 (1995).
2) R. Gupta, and M. L. Soffa: Hybrid Slicing: An Approach for Refining Static Slices Using Dynamic Information, *Proceedings of the 3rd International Symposium on the Foundation of Software Engineering*, pp.29-40 (1995).
3) The Virtual Reality Modeling Language Specification, *ISO/IEC DIS 14772-1* (1997).
4) Curtis Beeson: An Object-Oriented Approach To VRML Development, *Proceedings of VRML97* (1997).
5) J. Zhao, J. Cheng, and K. Ushijima: Static Slicing of Concurrent Object-Oriented Programs, *Proceedings of the 20th IEEE Anuual International Computer Software and Applications Conference*, pp.312-320 (1996).