論　文

スライス計算効率化のためのプログラム依存グラフの節点集約法

大畑　文明[†]　横森　励士[†]　西松　顯[†,††]　井上　克郎[†,†††]

Node Merging Method of Program Dependence Graph for Efficient Slice Computation

Fumiaki OHATA[†], Reishi YOKOMORI[†], Akira NISHIMATSU[†,††], and Katsuro INOUE[†,†††]

あらまし　プログラム依存グラフ（PDG）とはプログラム文間の依存関係を表す有向グラフであり，プログラムスライスに利用される．これまで提案されている多くの PDG 構築法は，スライスの精度向上を目標としてきた．しかし，大規模ソフトウェアにスライスを適用する場合，その精度だけでなく抽出の効率を考慮する必要がある．

本研究では，依存情報が文単位で保持される従来手法に対し，より大きな粒度で保持させることで効率向上を行ない，精度にも留意した PDG 構築手法を提案する．また，実際に提案手法の評価を行い，空間コスト 10 – 40%，時間コスト 5 – 60%の削減を得た．

キーワード　プログラムスライス，プログラム依存グラフ，効率，精度

1. まえがき

プログラムスライス（*Program Slice*）は Weiser [6] により提案され，プログラム保守，プログラム理解，プログラム統合 などに利用される．プログラムスライスは，大きく**静的スライス**（*Static Slice*）[6]，**動的スライス**（*Dynamic Slice*）[4] に分けられる．前者は入力データを与えない静的解析に基づき，入力データのすべての可能性を考慮した文間の依存関係が抽出される．後者は特定の入力データが与えられる動的解析に基づき，その入力データによる実行系列間の依存関係が抽出される．本論文では静的スライスに着目し，以降，静的スライスを単にスライスと呼ぶ．

効率に関して，我々は以下の 2 点に着目している．

（1）　依存解析のコスト

一般に，スライス抽出に必要となるプログラムの依存解析は多くの時間，空間を消費する．例えば，28,000 行の C プログラムに対し 2 時間を要したとされる報告もある [8]．我々はスライスを用いた対話形式のプログラム解析ツールを求めており，依存解析のコストの増大はその実現の妨げとなる．

（2）　依存解析の精度

スライス抽出はプログラム文間の依存関係に基づいて行なわれるが，依存関係の抽出精度を高めるにはその解析コストの増大は避けられない．加えて，現在主流となっているポインタ，配列を含む言語で記述されたプログラムの解析はさらに困難となる．

プログラムを解析する場合，一般にこの 2 者（コスト – 精度）はトレードオフの関係にあり，このトレードオフに関して多くの研究がなされている [2], [7]．本研究では，プログラム依存グラフ（PDG）を用いてコストと精度のトレードオフを考える．PDG の構築はスライス抽出に必要不可欠なものであり，その中でも，プログラム文間の依存関係把握のためのデータフロー計算に多くの時間，空間を要する．

解析効率を高めるには様々な手法が考えられる．その 1 つとして，通常各文の依存情報は PDG の対応する 1 節点に保持させるが，複数文の依存情報を 1 つの PDG 節点に保持させる節点集約による手法が考えら

[†] 大阪大学大学院 基礎工学研究科，豊中市
Graduate School of Engineering Science, Osaka University, Toyonaka-shi, 560-8531 Japan
[††] アールスリーインスティテュート，大阪市
R3 institute, Osaka-shi, 541-0057 Japan
[†††] 奈良先端科学技術大学院大学 情報科学研究科，生駒市
Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma-shi, 630-0101 Japan

れる．この節点集約を用いることにより，PDG 節点数は減少し解析コストの削減が期待できる．

本論文では，節点集約によるコスト削減手法を目的別に 2 つ提案する．一つは精度の低下を抑えた空間コスト，時間コスト削減手法（手法 1），もう一つは空間コストが若干増加するが精度の低下のない時間コスト削減手法（手法 2）である．また，実際に我々が開発したスライスシステムに追加実装を行ないその有効性を評価した．いくつかのサンプルプログラムに対する検証の結果，空間コスト 10 – 40%，時間コスト 5 – 60%の削減が得られた．

以降，2. ではプログラムスライスについて紹介する．3. では節点集約について述べ，4. で手法 1 を，5. で手法 2 をそれぞれ提案する．6. で提案手法の評価を行ない，7. でまとめと今後の課題について述べる．

2. プログラムスライス

スライス計算にはさまざまな手法が存在するが，本研究ではプログラム依存グラフによるスライス抽出技法 [3] に着目する．以下，各フェーズを簡単に述べる．ただし，ソースプログラムの構文解析，意味解析により**コントロールフローグラフ**（*Control Flow Graph*, *CFG*）[1] が得られているものとする．なお，本論文で対象とする言語はポインタのない手続き型言語とする．ポインタを含むプログラムに対しては，既に提案されているポインタ解析手法 [5] を併用することで適用可能である．

Phase 1: 定義，参照変数の抽出

プログラムの各文で定義，参照される変数を抽出する．このフェーズはプログラム文の 1 回の走査で終わる．

Phase 2: 依存関係解析

Phase 1 の結果を元に，プログラム文間の，次に示す 2 つの依存関係を抽出する．

ソースプログラム中の 2 文 s，t に関して，以下の条件を満たすとき，s から t の間に**制御依存関係**（*Control Dependence*, *CD*）[3] が存在するという．

（1） s は条件文（節） かつ，
（2） t の実行は s の判定結果に依存する

この関係を $CD(s,t)$ と表す．また，以下の条件を満たすとき，s から t の間に変数 v に関する**データ依存関係**（*Data Dependence*, *DD*）[3] が存在するという．

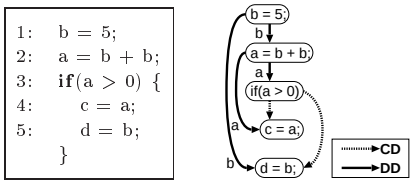


図 1 サンプルプログラム および その PDG
Fig. 1 Sample program and PDG

（1） s は v を定義する かつ，
（2） t は v を参照する かつ，
（3） s から t への 1 つ以上の実行経路の中に，v の再定義が起こらない経路が少なくとも 1 つ存在する

この関係を $DD(s,v,t)$ と表す．制御依存関係の計算は Phase 1 の結果および CFG から容易に行なうことができる．しかし，データ依存関係の条件（3）の判定には，手続き呼び出しやポインタ 等の解析が必要不可欠であり，繰り返し文，再帰呼び出しによる再帰的な依存関係も存在するため，スライス抽出過程の中でも最も計算量を必要とする．

Phase 3: プログラム依存グラフの構築

Phase 2 で抽出された依存関係を利用し，プログラム依存グラフを構築する．**プログラム依存グラフ**（*Program Dependence Graph*, *PDG*）[3] は，プログラムに存在する文を節点，文間の依存関係（データ依存関係，制御依存関係）を辺で表現した有向グラフである．図 1 にサンプルプログラムおよびその PDG を示す．PDG は Phase 2 で抽出された依存関係から容易に構築可能である．

Phase 4: スライスの抽出

スライス基準に対するスライスを抽出する．**スライス基準**（*Slicing Criterion*）は対 $<s, v>$ で示され．s は文，v は s で定義もしくは参照される変数を表す．スライス基準 $<s, v>$ に対するスライスとは，s に対応した PDG 中の節点 V_s から，逆方向に制御依存辺およびデータ依存辺を経て推移的に到達可能な節点集合に対応する文の集合をいう．図 2 に図 1 のスライス基準 < 文 5, b> に対するスライス (太枠部，文 1，2，3，5) を示す．スライスの抽出は PDG を辿るのみであるため，依存関係解析に比べそれに要する計算量はわずかである．

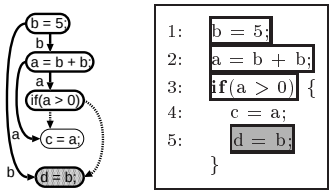


図 2　図 1 のスライス基準 < 文 5, b> のスライス
Fig. 2　Slice of slicing criterion (5, b) for Fig.1

3. 節点集約

3.1 方　　針

通常 Phase 2 に最も手間がかかる．我々は，この依存関係解析の手間を小さくするための手法として，**節点集約**（*Node Merging*）（以降，単に集約と呼ぶ）を提案する．本手法を用いることにより，情報の共有による情報量（空間コスト）削減や，節点数の減少による計算量（時間コスト）削減が期待できる．

集約の方針を次に示す．なお，**非集約 PDG**（*Non-merged PDG*）とは従来手法で用いられる集約のないプログラム依存グラフである．一方，**集約 PDG**（*Merged PDG*）とは今回提案する集約が行われたプログラム依存グラフである．また，各 PDG を用いて抽出されるスライスをそれぞれ，**非集約スライス**（*Non-merged Slice*），**集約スライス**（*Merged Slice*）と呼ぶ．

（1） 集約は Phase 1 および Phase 2 と独立したフェーズで行ない，Phase 1，Phase 2 の変更は最小限に抑える．

（2） Phase 1 および CFG から得られる情報のみを用いて集約を行ない Phase 2 に渡す．

（3） 方針 (1)，(2) に関連するが，集約は後述する集約条件が成り立つ連続文に限定する．また，手続き呼び出し文およびそれを内包する制御文は集約対象としない．

（4） 同じスライス基準に対し，非集約 PDG を用いて抽出されたスライス A には含まれなかった文が，集約 PDG を用いて抽出されたスライス B に含まれること（精度の低下）があるが，スライス A に含まれる文は必ずスライス B に含まれる．

（5） 局所依存関係という，データ依存関係，制御依存関係を組み合わせた依存関係を新たに定義し節点集約に利用する．局所依存関係については次節で述べる．

3.2 局所依存関係

以下の条件のいずれかを満たす場合，文 s は文 t に対し変数 α に関する**局所依存関係**（*Local Dependence*, LD）が成り立つといい，$LD(s, \alpha, t)$ と表す．なお，前述のとおり，局所依存関係はデータ依存関係および制御依存関係を組み合せたものであり，変数 α を定義する文 s およびそれを参照する文 t 間には，α の再定義が起らない経路が少なくとも 1 つ存在することが前提となる．

（1） 文 s で変数 α が定義され，文 t で変数 α が参照される．(図 3(a))

（2） 変数 α を参照し変数 β を定義する文 u が存在し，かつ文 s が α を定義し文 t が β を参照する．(図 3(b))

（3） 変数 α を参照する条件文（繰り返し文）u が存在し，文 t はその分岐節（繰り返し節）であり（$CD(u, t)$），かつ文 s は変数 α を定義する．(図 3(c)) なお，このような変数 α を，文 t を**支配**（*Control*）する変数という．

（4） 文 s と文 t は同じ文であり，かつ文 t は変数 α を定義する．(図 3(d))

局所依存関係の抽出は Phase 1 および CFG から得られる情報を用いることで可能であり，一段の間接依存まで調べている．多段の間接依存を認めることも可能であるが，計算コストの増大が予想されるため行なっていない．また，局所依存関係は後述する集約可否判定にのみ用いられ，PDG 辺として存在することはない．

3.3 節点集約

集約はプログラムの 2 つ以上の直結する文を対象としており，ここでは連続する 2 文の集約に関してのみ述べる．この手法を繰り返し適用することで 3 文以上の集約もできる．なお，離れた 2 文にも同様の方法は適用可能であるが，Phase 2 – 4 の実装への変更が必要であり対象としていない．

集約条件

となりあう 2 文 x, y について，文 x は変数 $X_1, X_2, \cdots, X_m$ を参照，文 y は変数 $Y_1, Y_2, \cdots, Y_n$ を参照しているとする．いま変数集合 $V \equiv \{X_1, X_2, \cdots, X_m, Y_1, Y_2, \cdots, Y_n\}$ に属する各変数 v に関して，$LD(s, v, x) \wedge LD(s, v, y)$ を満たす文 s がそれぞれ存在する場合，あるいは文 s が存在しないような変数がある場合でもその個数が $limit$ 個以下であ

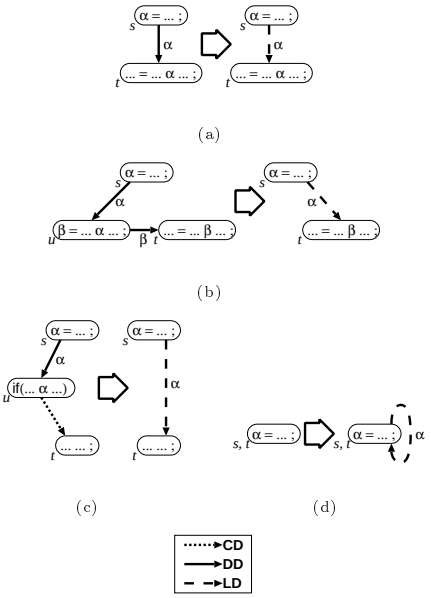
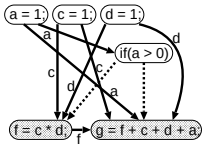


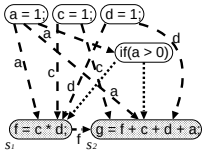
図 3　局所依存関係
Fig. 3　Local Dependence Relation

```
1:   a = 1;
2:   b = 1;
3:   c = 1;
4:   d = 1;
5:   if(a > 0) {
6:       e = b;
7:       f = c * d;
8:       g = f + c + d + a;
     }
9:   h = g;
10:  i = e;
```

(a) プログラム



(b) DD, CD による依存グラフ（PDG の一部）



(c) LD, CD による依存グラフ

図 4　節点集約
Fig. 4　Merging Nodes

れば，文 x，y を 1 節点に集約する．

集約例

図 4(a) のプログラムの文 7，8 に対する $limit = 0$ での集約を例に挙げる．図 4(b) は DD，CD による依存グラフ，図 4(c) は LD，CD による依存グラフである（ただしいずれのグラフも，文 7，8 に関連する依存辺およびその辺に接する節点からなる部分グラフである）．変数 c，d は文 7 で，変数 f，c，d，a は文 8 で参照されている．変数 c に関して $LD(3, c, 7) \wedge LD(3, c, 8)$，変数 d に関しても同様に $LD(4, d, 7) \wedge LD(4, d, 8)$ が成り立つ．変数 f，a それぞれについても，$LD(7, f, 7) \wedge LD(7, f, 8)$，$LD(1, a, 7) \wedge LD(1, a, 8)$ が成り立つ．それゆえ，2 文 7，8 を 1 節点に集約する．

ここでは説明のため依存グラフを用いたが，集約対象を直結する文に限定していることで $LD(s, \alpha, t)$ の文 s に該当する文を把握する必要はなく（集約対象文で同一識別子の変数が参照されている場合，その変数を定義した文は同じである．ただし，集約対象文内でその変数が再定義される可能性があるが，その把握は容易である），集約判定は変数の集合演算に置き換えることができる．[9]

集約の制御

集約の程度は $limit$ 値（$limit \geq 0$）により制御され，$limit$ を大きくすることで集約は進み，$limit$ を小さくすることで集約は抑えられる．コストと精度はトレードオフの関係にあり，目的に応じた $limit$ 値の選定が必要となる．本論文では，その中でも特徴的な $limit$ 値に着目し，先に述べた節点集約手法を拡張した 2 つの集約方法を提案する．

- 精度の低下を抑えた時間コスト，空間コスト削減のための，依存関係の局所性を利用した節点集約法（手法 1，$limit \ll \infty$）
- 空間コストは若干増加するが精度の低下のない時間コスト削減のための，節点分解をともなう節点集約法（手法 2，$limit = \infty$）

その他の $limit$ 値に関しては，特性を見い出せなかったため今回は扱っていない．また $limit \ll \infty$ の具体的な値として，6. では 0, 1, 2 を評価対象としている．

3. 4　集約アルゴリズム

集約の判定には，集約対象文に関する以下のような情報が必要であるが，

- 定義，参照される変数名
- 分岐節（繰り返し節）内の文かどうか
- となりあう文の情報

これらの情報は Phase 1 および CFG から取得できる．

3. 3 では連続文の集約判定を例に挙げて説明したが，集約はプログラム構造に依存するため，連続文以外にも，ブロック，分岐文，繰り返し文の集約アルゴリズム（集約判定 および 集約手続き）を考えた [9]．これ

らは我々が開発したシステムに Phase 1.5（Phase 1 – 2 間）として実装されている．

4. 依存関係の局所性を利用した節点集約法（手法 1）

本節では，依存関係の局所性を利用した節点集約法について述べる．本手法により，精度の低下を抑えた時間コスト，空間コストの削減を得ることができる．詳細は [9] で述べられており，ここでは主要部分のみ紹介する．

4.1 依存関係の局所性

依存関係の局所性（*Dependency Locality*）とは，直観的には対象となる複数文が持つ依存関係の類似性を表したものであり，$limit \ll \infty$ で集約可能な文にその性質が現れる．複数文を 1 節点に集約することを考えたとき，集約対象文に関するすべての依存関係を集約節点に委譲させる必要があるが，単純に連続文を集約した PDG をスライス抽出に利用するとスライスの精度は著しく低下してしまう．しかし，$limit = 0$ のとき集約可能な 2 文 s，t は**共に同じ文集合 $\mathcal{U}$ に局所依存している**ため，言いかえると，同一の文集合 $\mathcal{U}$ に依存する文 s，t を集約しているため，集約による精度の低下を抑えることができる．

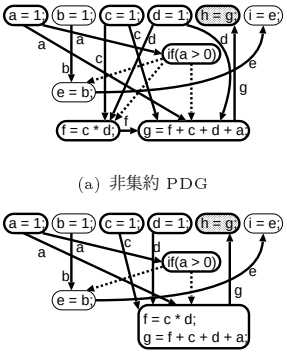
依存関係の局所性による集約の例

図 4(a) のサンプルプログラムに対する，非集約 PDG，集約 PDG（文 7，8 を $limit = 0$ で集約）を図 5(a)，(b) にそれぞれ示す．スライス基準 < 文 9, g> に対するスライスを非集約 PDG および集約 PDG を用いて抽出したとき，文 7，8 が依存関係の局所性を有することから，非集約 PDG によるスライス（文 1, 3，4，5，7，8，9）と同じスライスが集約 PDG でも抽出されていることが分かる．

$limit = 0$ の集約で精度の低下が起こる状況は 2 つ存在する．1 つは，文 s，t のいずれかのみ非集約スライスに含まれるようなスライス基準に対し集約スライスを抽出するときである．このとき，集約スライスは文 s，t いずれも含むことになる．もう 1 つは，集約節点で参照のみされる変数をスライス基準としたときである．こちら関しては 4.2 で述べる．

4.2 支 配 表

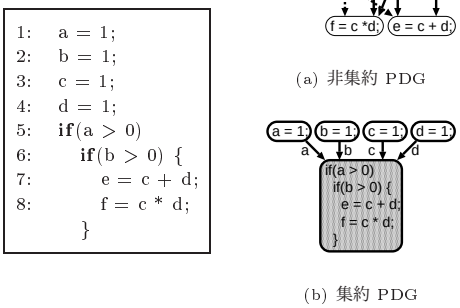
$limit \ll \infty$ の集約により，スライスの精度低下を抑えた集約を行なうことができる．しかし集約節点で参照のみされる変数をスライス基準としたとき，得られ

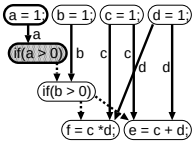


(a) 非集約 PDG

(b) 集約 PDG

図 5 スライス基準 < 文 9, g> のスライス
Fig. 5 Slice of <9, g>





(a) 非集約 PDG

(b) 集約 PDG

図 6 集約節点からのスライス
Fig. 6 Slice of merged node

るスライスの精度が著しく低下する可能性がある（集約節点で定義される変数をスライス基準としたときや，PDG 探索中に集約節点に到達したときには，このような問題は生じない）．集約により集約前節点間に存在した制御依存に関する情報は失われ，定義変数は当然であるが参照変数であっても，すべての参照変数に依存していると解釈せざるを得ない．例えば，図 6 では集約によりスライスサイズが 2 から 8 に増加している．

この問題を解決するため，同一集約節点に存在する参照変数間の支配関係を記憶する支配表を集約節点に持たせた．**支配関係**（*Control Relation*）とは，変数 α が条件節で参照され変数 β が対応する述部で参照されるとき α〜β 間に存在する関係をいう．また，α が β を支配するともいう．前述のとおり，集約節点内の

支配 \ 被支配	a	b	c	d
a	√	√		
b		√	√	√
c			√	
d				√

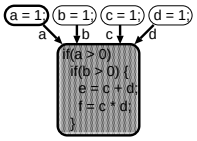


図 7　支配表と集約節点からのスライス
Fig. 7　Control table and slice of merged node

制御依存に関する情報は破棄されるため，この支配関係でその役割を補う．また，**支配表**（*Control Table*）とは n 行 n 列（n は各集約節点の参照変数の数）の表であり，支配関係の存在する組には √で，存在しない組は空欄で表す．支配表は集約時に逐次更新され，集約節点の参照のみされる変数 a がスライス基準となったとき，支配表を参照することで変数 a が真に依存している参照変数を把握できる．

図 7 に，図 6 の集約節点の支配表およびそれに基づき抽出された集約節点（網掛部）の変数 a のスライス（太枠部）を示す．支配表から集約節点の変数 a はそれ自身にのみ依存していることが分かり，スライスサイズは 8 から 5 に減少している．

5. 節点分解をともなう節点集約法（手法 2）

4. では，依存関係の局所性を有する $limit \ll \infty$ 集約による，精度の低下を抑えた節点集約法を提案した．依存関係の局所性を利用した集約では，空間コスト，時間コストの削減を行うことができる．一方 $limit = \infty$ とすると，コスト削減は十分に期待できるが，精度の大幅な低下は避けられない．そこで，$limit = \infty$ の集約による PDG を構築後，その集約節点を分解する手法が考えられる．この手法は，節点の分解を考慮する必要があるため空間コストが若干増加するが，精度の低下のない時間コスト削減を行うことができる．また $limit = \infty$ では局所依存関係を抽出する必要はなく，集約可能な文（手続き呼び出し文およびそれを内包する制御文以外）はすべて集約される．

5.1 節 点 分 解

節点分解（*Node Decomposition*）とは，Phase 1.5 を経て構築された集約 PDG の集約節点を分解し，その内部の依存関係のみ解析することで非集約 PDG を再構成することをいう．

ここで，データ依存関係を以下の 2 つに分類する．

（1） 大域的な依存関係 ... 手続きをまたぐ依存関係や再帰的な依存関係などを指し，その抽出に要する解析範囲が大きい

（2） 局所的な依存関係 ... 手続き内で閉じた依存関係や非再帰的な依存関係などを指し，その抽出に要する解析範囲が小さい

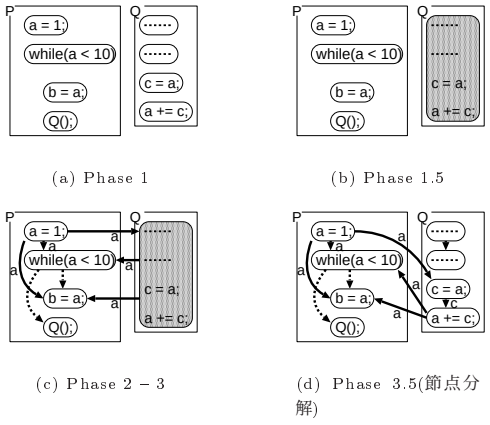
前述のとおり，スライス抽出過程の中でも最も時間計算量を要するのはデータ依存関係の抽出であるが，それは大域的な依存関係の存在によるものが多い．そこで，$limit = \infty$ での集約により大域的な依存関係の抽出対象となる節点を可能な限り減らし，その抽出コストを削減させる．一方，集約により 1 つの集約節点にまとめられていた節点間の依存関係は局所的な依存関係に該当し，分解時にそれら節点間の依存関係解析のみ行うことで抽出可能である．

節点分解をともなう節点集約の例

図 8(a) は Phase 1 終了後の手続き P，Q を表しており，Phase 1.5 の $limit = \infty$ の節点集約により図 8(b) となる．その後 Phase 2 – 3 により図 8(c) の集約 PDG が構築される．手続き Q は手続き P から呼び出されており，かつその呼び出し文は**while** 文の繰り返し節に存在するため，手続き Q 内の依存関係解析を少なくとも 2 回行わなければならない．しかし，手続き Q 内を $limit = \infty$ で集約することでデータ依存解析の対象節点が減り，手続き P から手続き Q におよぶ大域的な依存関係解析のコスト削減が得られる．最後に集約節点内の局所的な依存関係解析を行い，図 8(d) の非集約 PDG が再構成される．ここでは説明のため比較的単純な例を用いたが，手続き Q がより大規模である場合や，再帰呼び出し経路に含まれている場合，集約によるコスト削減の効果はより大きなものとなる．

5.2 分解アルゴリズム

$limit = \infty$ の節点集約により構築された集約 PDG に対し，各集約節点内部で分解アルゴリズムを適用し非集約 PDG を再構成する．分解アルゴリズムは，Phase 2 の依存関係解析アルゴリズムとほぼ同じものとなっている．異なるのは，集約前節点に関する依存関係は対応する集約節点が代わりに保持しているため，集約節点内部の依存関係解析後，集約節点に関する依存関係を分解後（集約前）の節点に分散（図 8(c) – 図 8(d)）させなければならない点である．依存関係の分散が終わると，これらの依存関係を元に非集約 PDG の再構成を行う．分解アルゴリズムは Phase 3.5（Phase 3 – 4 間）として実装されている．



(a) Phase 1　(b) Phase 1.5

(c) Phase 2 – 3　(d) Phase 3.5(節点分解)

図 8　節点分解
Fig. 8　Node decomposition

6. 評　　価

6.1 実　　験

提案手法を我々が開発したスライスシステム [10] 上に実現した．依存関係解析アルゴリズムは [11] による．実験は以下に示す 5 種類の PDG の比較を行った．

N　集約なし

L_0　依存関係の局所性を利用した節点集約（手法 1, $limit = 0$）

L_1　依存関係の局所性を利用した節点集約（手法 1, $limit = 1$）

L_2　依存関係の局所性を利用した節点集約（手法 1, $limit = 2$）

C　節点分解をともなう節点集約（手法 2）

今回使用したテストプログラムを表 1 に，PDG 節点数を表 2 に（括弧内は集約節点数を表す），PDG 辺数を表 3 に，支配表のセル数を表 4 に，節点数，辺数，支配表セル数の単純和を表 5 に，PDG 構築までの時間 (Phase 1 – 3 の合計時間，集約，分解を行う場合は Phase 1.5，Phase 3.5 もそれぞれ含まれる) を表 6 に，平均スライスサイズ (テストプログラムの各手続きで最後に参照される変数をスライス基準として選び，それらのスライスに含まれる文数の平均) を表 7 に示す．

6.2 考　　察

空間コスト（PDG サイズ）

局所依存関係による節点集約に関して，PDG 節点数は 10 – 40%（表 2），PDG 辺数は 10 – 20%（表 3）の削減が得られた．L_0 – L_2 には今回新たに定義した支配表が導入されているが（表 4），支配表の各セルは真偽の 2 値のみ保持し，辺，節点に必要な情報量に比較すると十分に小さい．節点分解をともなう節点集約では，表 2 に示すように集約前節点の保存のため節点数が増加している（N および L_1 と比較すると，それぞれ約 10%，約 40%の増加となっている）．

時間コスト（PDG 構築時間）

表 1　テストプログラム
Table 1　Sample programs

	行	手続き	内容
P_1	333	14	チケット予約
P_2	429	18	酒屋問題
P_3	449	30	小計算問題の集合
P_4	831	22	ソーティング

表 2　PDG 節点数
Table 2　Number of nodes

	N	L_0	L_1	L_2	C
P_1	169	119(26)	114(27)	103(34)	180(11)
		(−29.59%)	(−32.54%)	(−39.05%)	(+6.51%)
P_2	211	166(22)	153(27)	136(28)	231(20)
		(−21.33%)	(−27.49%)	(−35.55%)	(+9.48%)
P_3	243	199(19)	187(24)	177(28)	270(27)
		(−18.11%)	(−23.05%)	(−27.16%)	(+11.11%)
P_4	503	459(124)	419(150)	409(165)	547(44)
		(−8.75%)	(−16.70%)	(−18.69%)	(+8.75%)

表 3　PDG 辺数
Table 3　Number of edges

	N	L_0	L_1	L_2	C
P_1	935	833	817	774	935
		(−10.91%)	(−12.62%)	(−17.22%)	
P_2	1487	1387	1336	1290	1487
		(−6.72%)	(−10.15%)	(−13.25%)	
P_3	1092	980	951	912	1092
		(−10.26%)	(−12.91%)	(−16.48%)	
P_4	3360	3135	2791	2750	3360
		(−6.70%)	(−16.93%)	(−18.15%)	

表 4　支配表セル数
Table 4　Number of cells of control tables

	N	L_0	L_1	L_2	C
P_1	0	59	87	176	0
P_2	0	81	125	183	0
P_3	0	26	45	102	0
P_4	0	150	212	237	0

表 5　節点，辺数，支配表セル数の和
Table 5　Sum of nodes, edges and cells of control tables

	N	L_0	L_1	L_2	C
P_1	1104	1011	1018	1053	915
P_2	1698	1634	1614	1609	1704
P_3	1335	1205	1183	1191	1081
P_4	3863	3744	3442	3396	1811

表 6　PDG 構築時間 [ms]
Table 6　PDG construction time[ms]

	N	L_0	L_1	L_2	C
P_1	102.53	80.789	80.573	74.48	69.258
		(−21.21%)	(−21.42%)	(−27.38%)	(−32.45%)
P_2	191.01	161.792	157.557	150.603	129.192
		(−15.30%)	(−17.51%)	(−21.15%)	(−32.36%)
P_3	187.84	161.494	156.074	147.365	157.712
		(−14.03%)	(−16.91%)	(−21.55%)	(−16.04%)
P_4	739.172	708.597	607.979	585.679	281.638
		(−4.14%)	(−17.75%)	(−20.77%)	(−61.90%)

Celeron-450MHz-128MB(FreeBSD)

表 7　平均スライスサイズ [文]
Table 7　Average slice size[# statements]

	N	L_0	L_1	L_2	C
P_1	94.21	95.21	95.21	95.21	94.21
		(+1.06%)	(+1.06%)	(+1.06%)	
P_2	143.22	145.00	147.44	148.28	143.22
		(+1.24%)	(+2.95%)	(+3.53%)	
P_3	46.80	48.27	48.27	52.50	46.80
		(+3.13%)	(+3.13%)	(+12.18%)	
P_4	173.43	178.52	178.57	178.57	173.43
		(+2.94%)	(+2.97%)	(+2.97%)	

依存関係の局所性を利用した節点集約に関して，解析時間は 5 – 30%の削減が得られた（表 6）．また，集約に要する時間はいずれのプログラムに対しても 10[ms] 以下であった．

節点分解をともなう節点集約に関して，解析時間は 15 – 60%の削減が得られた（表 6）．$limit = \infty$ の集約により，解析時間の大半を占める大域的な依存関係の解析時間が削減されたためである．一方，局所的な依存関係の解析に要する計算コストは少なく，分解に要する時間は P_4 で 70[ms] であった．

精度（スライスサイズ）

非集約スライスと比べ集約スライスのサイズは多少大きくなる．しかし，依存関係の局所性を有する文のみ集約するため，スライスの精度が大きく低下することはない．表 7 で示されているように，L_0，L_1 とも 1 – 3%程度のスライスサイズ増加で抑えられている．

これらの実験結果から，依存関係の局所性を利用した節点集約では $limit = 1$ が最も有効であると考えられる．

limit 値が 0，1，2 と大きくなるにつれ，解析時間の短縮は期待できるが反対にスライスサイズは増加する．L_0 と L_1 間ではスライスサイズに影響をほとんど及ぼすことなく解析時間の短縮が得られているが，L_2 では P_4 のスライスサイズの大幅な増加が確認されている．一方，節点数，辺数，支配表セル数の単純和（表 5）を考えたとき，$limit = 1$ まで減少傾向であったものが $limit = 2$ で増加傾向に変化している．これら精度，時間コスト，空間コストの 3 点を考慮すると，$limit = 1$ を妥当な値として導き出すことができる．

今回は実装の言語制約のため，比較的規模の小さいプログラムに対する実験のみであった．残念ながら，現時点では存在するすべてのプログラムに対して $limit = 1$ が有効であると断定することはできないが，現在開発中のシステムに本手法を適用することで大規模プログラムに対する検証も行いたいと考えている．

6.3 関連研究

プログラムスライスに関してさまざまな研究がなされている．解析コストとスライス精度とのトレードオフに関するものとしては [2], [7] がある．これらは，ユーザが解析コストとスライスの精度を操作できるシステムを開発しているが，解析方法全体を変更することでトレードオフを考慮している．

[8] では完全に相互依存する節点のみの集約を行なっている．我々の手法は，相互依存しなくとも同じ依存関係を持つ節点であれば集約を行なう．また，*limit* 値を変更することで解析コストとスライス精度のトレードオフが制御可能である．

7. まとめと今後の課題

節点集約による PDG 構築コスト削減手法を 2 つ提案し，その有効性を実験により評価した．依存関係の局所性を利用した節点集約法は，スライスサイズが大きくなる副作用が存在するが，その増加はわずかであり，空間コスト，時間コストの削減が可能である．コストと精度のトレードオフは *limit* 値により制御可能で，実験結果から $limit = 1$ が最も有効であると思われる．節点分解をともなう節点集約法は，空間コストが増加するが精度の低下の無い時間コスト削減を行うことができる．これらの手法はそれぞれ特徴を持っており，ユーザの要求に応じた使い分けが可能である．例えば，限られた計算機資源で大規模プログラムを解析する場合，依存関係の局所性を利用した節点集約法が有効であろう．

本研究をふまえ，Java を対象言語とする新しいスライスシステムおよびプログラム理解ツールの開発を行っている．また，ソフトウェア開発工程でのスライスシステムの有効性をこのシステムを用いて評価する

予定である．

謝辞　本研究は，一部文部省科学研究費補助金特定領域研究 (A)(2)(課題番号:10139223) の補助を受けている．

文　献

[1] Aho, A.V., Sethi, S. and Ullman, J.D., “Compilers : Principles, Techniques, and Tools”, Addison-Wesley, 1986.

[2] D. C. Atkison, W. G. Griswold, “The design of whole-program analysis tools,” in Proceedings of the 18th International Conference on Software Engineering, pp.16–27, Berlin, Germany, 1996.

[3] Horwitz, S. and Reps, T., “The use of program dependence graphs in software engineering,” in Proceedings of the 14th International Conference on Software Engineering, pp.392–411, Melbourne, Australia, 1992.

[4] Korel, B. and Rilling, J., “Dynamic program slicing methods”. In *Information and Software Technology Special Issue on Program Slicing*, vol.40, pages 647–659, 1998.

[5] M. Shapiro, and S. Horwitz, “Fast and accurate flow-insensitive point-to analysis,” in 24th Annual ACM SIGACT-SIGPLAN Symposium on the Principles of Programming Languages, 1997.

[6] M. Weiser, “Program slicing,” in Proceedings of the 5th International Conference on Software Engineering, pp.439–449, San Diego, USA, 1981.

[7] P. Tonella, G. Antoniol, R. Fiutem and E. Merlo, “Variable precision reaching definitions analysis for software maintenance,” in Proceedings of the Euromicro Working Conference on Software Maintenance and Reengineering, pp.60–67, Berlin, Germany, 1997.

[8] The Wisconsin Program-Slicing Tool 1.0, Reference Manual. Computer Sciences Department, University of Wisconsin-Madison, 1997.

[9] 大畑, 西松, 井上, “依存関係の局所性を利用したプログラム依存グラフの効率的な構築法,” 情処学研報, 99-SE-122, pp.17–24, 1999.

[10] 佐藤, 飯田, 井上, “プログラムの依存解析に基づくデバッグ支援ツールの試作,” 情処学論, vol.37, no.4, pp.536–545, 1996.

[11] 植田, 練, 井上, 鳥居, “再帰を含むプログラムのスライス計算法,” 信学誌, vol.J78-D-I, no.1, pp.11–22, 1995.

（平成年月日受付，月日再受付）

大畑　文明

平 10 阪大・基礎工・情報中退．平 12 同大大学院修士課程了．現在同大学院博士課程在学中．プログラムスライスの研究に従事．

横森　励士

平 11 阪大・基礎工・情報卒．平 13 同大大学院修士課程了．現在同大学院博士課程在学中．プログラムスライスの研究に従事．

西松　　顯

平 9 阪大・基礎工・情報卒．平 11 同大大学院修士課程了．現在アールスリーインスティテュート勤務．在学中，プログラムスライスの研究に従事．

井上　克郎　（正員）

昭 54 阪大・基礎工・情報卒．昭 59 同大大学院博士課程了．同年同大・基礎工・情報・助手．昭 59〜昭 61 ハワイ大マノア校・情報工学科・助教授．平 1 阪大・基礎工・情報・講師．平 3 同学科・助教授．平 7 同学科・教授．工博．ソフトウェア工学の研究に従事．

Fig.1
サンプルプログラム および その PDG
Sample program and PDG
Fig.2
図 1 のスライス基準 < 文 5, b> のスライス
Slice of slicing criterion (5, b) for Fig.1
Fig.3
局所依存関係
Local Dependence Relation
Fig.4
節点集約
Merging Nodes
Fig.5
スライス基準 < 文 9, g> のスライス
Slice of <9, g>
Fig.6
集約節点からのスライス
Slice of merged node
Fig.7
支配表と集約節点からのスライス
Control table and slice of merged node
Fig.8
節点分解
Node decomposition
Table.1
テストプログラム
Sample programs
Table.2
PDG 節点数
Number of nodes
Table.3
PDG 辺数
Number of edges
Table.4
支配表セル数
Number of cells of control tables
Table.5
節点，辺数，支配表セル数の和
Sum of nodes, edges and cells of control tables
Table.6
PDG 構築時間 [ms]
PDG construction time[ms]
Table.7
平均スライスサイズ [文]
Average slice size[# statements]