(12) 特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2007 年9 月20 日 (20.09.2007)

PCT

(10) 国際公開番号
WO 2007/105309 A1

(51) 国際特許分類:
G06F 9/45 (2006.01)

(21) 国際出願番号: PCT/JP2006/305016

(22) 国際出願日: 2006 年3 月14 日 (14.03.2006)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(71) 出願人 (米国を除く全ての指定国について): 富士通株式会社 (FUJITSU LIMITED) [JP/JP]; 〒2118588 神奈川県川崎市中原区上小田中４丁目１番１号 Kanagawa (JP).

(72) 発明者; および

(75) 発明者/出願人 (米国についてのみ): 伊藤 真紀子 (ITO, Makiko) [JP/JP]; 〒2118588 神奈川県川崎市中原区上小田中４丁目１番１号 富士通株式会社内 Kanagawa (JP). 三宅 英雄 (MIYAKE, Hideo) [JP/JP]; 〒2130012 神奈川県川崎市高津区坂戸３丁目２番１号 富士通エルエスアイテクノロジ株式会社内 Kanagawa (JP). 須賀 敦浩 (SUGA, Atsuhiro) [JP/JP]; 〒2118588 神奈川県川崎市中原区上小田中４丁目１番１号 富士通株式会社内 Kanagawa (JP).

(74) 代理人: 伊東 忠彦 (ITOH, Tadahiko); 〒1506032 東京都渋谷区恵比寿４丁目２０番３号 恵比寿ガーデンプレイスタワー３２階 Tokyo (JP).

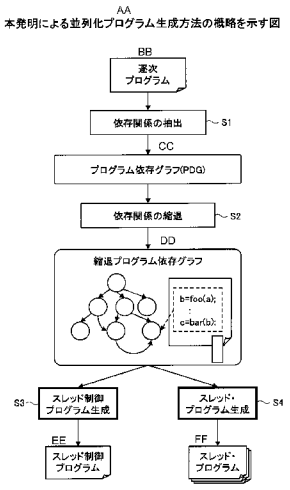
(81) 指定国 (表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) 指定国 (表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY,

[続葉有]

(54) Title: PARALLELED PROGRAM GENERATION PROGRAM, PARALLELED PROGRAM GENERATION DEVICE, AND PARALLELED PROGRAM GENERATION METHOD

(54) 発明の名称: 並列化プログラム生成プログラム、並列化プログラム生成装置、及び並列化プログラム生成方法



AA SCHEMATIC DIAGRAM OF PARALLELED PROGRAM GENERATING METHOD OF THE PRESENT INVENTION
BB SEQUENTIAL PROGRAM
S1 EXTRACTION OF DEPENDENCY
CC PROGRAM DEPENDENT GRAPH (PDG)
S2 DEGENERATION OF DEPENDENCY
DD DEGENERATED PROGRAM DEPENDENT GRAPH
S3 GENERATION OF THREAD CONTROL PROGRAM
S4 GENERATION OF THREAD PROGRAM
EE THREAD CONTROL PROGRAM
FF THREAD PROGRAM

(57) **Abstract:** A paralleled program generation program is characterized in including codes to make a computer execute the steps of inputting a sequential program with each sentence constructing the sequential program as the top; generating a program dependent graph with sides defined between the tops of the sentences; generating a degenerated program dependent graph in which the tops of the program dependent graph are merged to reduce the number of the tops; generating a thread program corresponding to each of the tops of the degenerated program dependent graph; and generating a thread control program to control the start-up and synchronization of the thread program, respectively.

(57) 要約: 並列化プログラム生成プログラムは、逐次プログラムを入力として、逐次プログラムを構成する各文を頂点として有するとともに、文と文の間の関係を頂点間の辺として有するプログラム依存グラフを生成し、プログラム依存グラフの頂点同士を融合することにより頂点の数を減少させた縮退プログラム依存グラフを生成し、縮退プログラム依存グラフの頂点の各々に相当するスレッド・プログラムを生成し、スレッド・プログラムの起動及び同期を制御するスレッド制御プログラムを生成する各段階を計算機に実行させるコードを含むことを特徴とする。

WO 2007/105309 A1

KG, KZ, MD, RU, TJ, TM), ヨーロッパ (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

添付公開書類:
— 国際調査報告書

２文字コード及び他の略語については、定期発行される各*PCT*ガゼットの巻頭に掲載されている「コードと略語のガイダンスノート」を参照。

明 細 書

並列化プログラム生成プログラム、並列化プログラム生成装置、及び並列化プログラム生成方法

技術分野

[0001] 本発明は、一般にプログラム生成方法、装置、及びプログラムに関し、詳しくは並列化プログラム生成方法、装置、及びプログラムに関する。

背景技術

[0002] 近年、シングル・プロセッサでのプログラム性能には限界があることが知られてきた。従来、性能を上げるためには、プロセッサの動作周波数を高くすることで単位時間あたりの処理量を増やす方法と、命令を並列に実行することで同時に実行できる処理を増やす方法とがとられてきた。

[0003] しかし動作周波数を高くすると消費電力が大きくなるという問題があるとともに、動作周波数の向上には物理的な限界があるという問題がある。また、命令レベルの並列性は高々2～4程度であり(非特許文献1)、投機的な実行などを導入することにより多少並列性を上げることはできるが、それにも限界があることが知られている。

[0004] そこで、命令レベルよりも大きな粒度でプログラムを並列化し、複数のプロセッサにて実行することにより、処理性能を向上させる方法が注目されている。しかしながら、制御による分岐が多い逐次プログラムを効果的な並列プログラムへ変換する画一的な方法は、これまでのところ知られていない。

[0005] 逐次プログラムを分割して複数のプロセッサ上で並列に実行するプログラムを生成する手法として、ループに着目したデータ・レベル並列化という方法と、制御に着目した投機的なスレッド実行という方法が知られている。

[0006] 特許文献1では、ループの中におけるデータの依存関係を解析し、配列を分割して、ループの処理を複数のプロセッサで実行させる。この手法は、数値計算等の規則的なループの処理が多い場合に有効である。

[0007] また特許文献2は、逐次プログラムにおける分岐に着目して、投機的なスレッド実行に置換する手法を示す。この手法では、制御の流れに基づいてプログラムを並列化

するので、プログラムの潜在的な並列性を充分に抽出できているとはいえない。また、投機的スレッド実行機構を持たないマルチプロセッサにおいては予測失敗時のロールバックのコストが大きいので、分岐予測ヒット率が低いアプリケーションにはこの手法は適さない。

特許文献1:特許第3028821号公報

特許文献2:特許第3641997号公報

非特許文献1:S. Horwitz, J. Prins, and T. Reps, "Integrating non-interfering versions of programs," ACM Transactions on Programming Languages and Systems, vol. 11, no. 3, pp. 345-387, 1989.

非特許文献2:Jeanne Ferrante, Karl J. Ottenstein, Joe D. Warren, "The Program Dependence Graph and Its Use in Optimizatio," ACM Transactions on Programming Languages and Systems, pp. 319-419, vol. 9 no. 3, July 1987.

非特許文献3:Susan Horwitz, Jan Prins, Thomas Reps, "On the adequacy of program dependence graphs for representing programs," Proceedings of the 15th Annual ACM Symposium on the Principles of Programming Languages, pp. 146-157, Jan., 1988.

発明の開示

発明が解決しようとする課題

[0008] 以上を鑑みて、本発明は、大規模なソフトウェアを対象として、逐次プログラムを並列化することにより、マルチプロセッサ上で効果的に動作する非投機的なマルチ・スレッド・プログラムを生成する方法、装置、及びプログラムを提供することを目的とする。

課題を解決するための手段

[0009] 並列化プログラム生成プログラムは、逐次プログラムを入力として、該逐次プログラムを構成する各文を頂点として有するとともに、文と文の間の関係を該頂点間の辺として有するプログラム依存グラフを生成し、該プログラム依存グラフの該頂点同士を融合することにより該頂点の数を減少させた縮退プログラム依存グラフを生成し、該縮退プログラム依存グラフの頂点の各々に相当するスレッド・プログラムを生成し、該

スレッド・プログラムの起動及び同期を制御するスレッド制御プログラムを生成する各段階を計算機に実行させるコードを含むことを特徴とする。

[0010] 並列化プログラム生成装置は、逐次プログラムと並列化プログラム生成プログラムとを格納するメモリと、該メモリに格納された該並列化プログラム生成プログラムを実行することで該メモリに格納された該逐次プログラムから並列化プログラムを生成する演算処理ユニットを含み、該演算処理ユニットは、該並列化プログラム生成プログラムを実行することにより、該逐次プログラムを構成する各文を頂点として有するとともに文と文の間の関係を該頂点間の辺として有するプログラム依存グラフを生成し、該プログラム依存グラフの該頂点同士を融合することにより該頂点の数を減少させた縮退プログラム依存グラフを生成し、該縮退プログラム依存グラフの頂点の各々に相当するスレッド・プログラムを生成し、該スレッド・プログラムの起動及び同期を制御するスレッド制御プログラムを生成することを特徴とする。

[0011] 並列化プログラム生成方法は、逐次プログラムを入力として、該逐次プログラムを構成する各文を頂点として有するとともに、文と文の間の関係を該頂点間の辺として有するプログラム依存グラフを生成し、該プログラム依存グラフの該頂点同士を融合することにより該頂点の数を減少させた縮退プログラム依存グラフを生成し、該縮退プログラム依存グラフの頂点の各々に相当するスレッド・プログラムを生成し、該スレッド・プログラムの起動及び同期を制御するスレッド制御プログラムを生成する各段階を含むことを特徴とする。

発明の効果

[0012] 本発明の少なくとも1つの実施例によれば、制御の流れグラフではなく、制御の依存関係を示すグラフであるプログラム依存グラフに基づいて並列化プログラムを生成するので、制御の流れ(分岐)を超えたプログラムの並列性を抽出することができる。また、プログラム依存グラフを縮退してグラフの規模を削減することで、その後の並列化プログラム生成処理の効率化及び最適化が可能になるとともに、大きな粒度での並列化を実現することができる。

図面の簡単な説明

[0013] [図1]本発明による並列化プログラム生成方法の概略を示す図である。

[図2]スレッド・プログラム生成方法の概要を示す図である。

[図3]図2のスレッド・プログラム生成方法により生成されるスレッド・プログラムを示す図である。

[図4]スレッド制御プログラムの生成方法を示すフローチャートである。

[図5]頂点間の実行順序関係を決定する方法を示すフローチャートである。

[図6]頂点v以下の制御の流れを再構成する処理(図5のステップS2)を示すフローチャートである。

[図7]Regionの実行順序関係を計算する処理を示すフローチャートである。

[図8]逆依存及び出力依存を求める処理(図7のステップS4)を示すフローチャートである。

[図9]着目領域を越える変数参照を抽出する処理を示すフローチャートである。

[図10]着目領域を越える変数代入を抽出する処理を示すフローチャートである。

[図11]逆依存の追加処理を示すフローチャートである。

[図12]出力依存の追加処理を示すフローチャートである。

[図13]逆依存及び出力依存を求める処理(図7のステップS5)を示すフローチャートである。

[図14]全域木を説明するための図である。

[図15]全域木を模式的に示す図である。

[図16]全域木間の順序関係を計算する処理を示すフローチャートである。

[図17]図16の処理による逆依存辺の追加について説明する図である。

[図18]頂点間の実行順序関係を決定する方法の変形例を示すフローチャートである。

[図19]頂点vp以下のスレッド制御プログラムを生成する処理を示すフローチャートである。

[図20](a)は入力逐次プログラムの部分を示す図、(b)は対応する縮退プログラム依存グラフを示す図である。

[図21]図20の縮退プログラム依存グラフから第1の実施例に従い生成されるスレッド制御プログラムを示す図である。

[図22]以上のスレッド制御プログラムの動作をスレッド・プログラムの実行とともに示す模式図である。

[図23]図20の縮退プログラム依存グラフから第2の実施例に従い生成されるスレッド制御プログラムを示す図である。

[図24]以上のスレッド制御プログラムの動作をスレッド・プログラムの実行とともに示す模式図である。

[図25]図20の縮退プログラム依存グラフから第3の実施例に従い生成されるスレッド制御プログラムを示す図である。

[図26]以上のスレッド制御プログラムの動作をスレッド・プログラムの実行とともに示す模式図である。

[図27]本発明による並列化プログラム生成方法を実行する装置の構成を示す図である。

符号の説明

[0014] 10 入力変数の受信部分

11 変数宣言部分

12 プログラム本体部分

13 出力変数の送信部分

21, 22 全域木

31 出力依存辺

32, 33 逆依存辺

510 コンピュータ

511 CPU

512 RAM

513 ROM

514 二次記憶装置

515 可換媒体記憶装置

516 インターフェース

520 ディスプレイ装置

521 キーボード

522 マウス

523 通信装置

発明を実施するための最良の形態

[0015] 以下に、本発明の並列化プログラム生成方法の概略及び実施例を添付の図面を用いて詳細に説明する。

[0016] 図1は、本発明による並列化プログラム生成方法の概略を示す図である。

[0017] ステップS1で逐次プログラムからプログラム依存グラフ(PDG:Program Dependence Graph)を生成する。次に、ステップS2で、スレッドとして他のプロセッサエレメントで実行するに適した処理量となるまで依存関係を縮退することにより、スレッドを頂点とする縮退プログラム依存グラフを作成する。ステップS3で、作成した縮退プログラム依存グラフから、非投機的にスレッドの起動と同期を制御するスレッド制御プログラムを生成する。またステップS4で、縮退プログラム依存グラフから、その各頂点に相当するスレッド・プログラムを生成する。

[0018] まず逐次プログラムからプログラム依存グラフを生成する処理(図1のステップS1)について説明する。

[0019] プログラム依存グラフとは、例えば非特許文献1乃至3等に説明されるように、プログラムの文を頂点とし、文と文の間の関係を辺で表現したグラフである。非特許文献1乃至3に記載されるプログラム依存グラフは、次のような頂点集合Vと辺集合Eの組で表現されるものであり、逐次プログラムを解析することにより生成できる。

[0020] [V:頂点集合]

エントリ:プログラムの開始ポイントを表す。

[0021] 初期定義:プログラム開始時の初期値の定義を表す。

[0022] プリディケート: If-then-elseまたはwhile-loopの条件判定を表す。

[0023] 代入文:プログラムの代入文を表す。

[0024] 最終使用:プログラム終了時の変数の参照を表す。

[0025] [E:辺集合]

[制御依存辺: $v \rightarrow_{c}^{L} w$]プリディケート頂点vに対して、その条件判定結果により、

頂点wに到達するか否かが決まることを表す。Lは条件判定のフラグを表し、L=Tのときは条件判定結果が真の場合に頂点wを実行し、L=Fのときは結果が偽の場合に頂点wを実行する。

[0026] ［データ依存辺］

［ループ独立フロー依存辺: $v \rightarrow_{li}^{x} w$］頂点vで代入された変数xの値を、頂点wで参照するような場合のデータ依存関係を表す。ここでは、ループを繰り越さない場合のみを表す。

[0027] ［ループ繰り越しフロー依存辺: $v \rightarrow_{lc(L)}^{x} w$］頂点vで代入された変数xの値を、頂点wで参照するような場合のデータ依存関係を表す。ループLを繰り越す場合を表す。

[0028] ［定義順序関係: $v \rightarrow_{do(u)}^{x} w$］頂点v及び頂点wが変数xの値を代入し、頂点uで参照するような場合の、頂点vと頂点wの順序関係を表す。制御の流れによっては、v, wの順に実行される可能性がある場合に、その実行順序を表すものである。

[0029] 以下において、縮退プログラム依存グラフを作成する処理（図1のステップS2）について説明する。

[0030] 上記のような一般的なプログラム依存グラフでは、文または代入式を頂点としたグラフとなっている。文または代入式を頂点とした場合、大規模なソフトウェアではグラフの頂点数が数千～数万となってしまう。一般的に、コンパイラのグラフを用いた最適化の問題の計算量は、グラフの規模に対して指数関数的に増大することが知られている。したがって、例えば数個の手続きなどを対象とした頂点数が数十程度のグラフの場合には、解析が可能であるが、現実的な規模のソフトウェア全体に対する最適化は困難といえる。

[0031] そこで、プログラム依存グラフの頂点数及び辺数を低減すべく、プログラム依存グラフの依存関係を縮退して頂点を融合し、粗粒度のプログラム依存グラフを作成する。依存関係を縮退することによりグラフの規模を1/10～1/100とすることで、現実的な時間にて、プログラムの最適化を可能にする。

[0032] 依存関係の縮退は、次のような方法で、縮退可能な依存関係及び頂点の集合を求め、依存関係を削除して頂点を1つの頂点に融合することにより実行される。

[0033] 1．構文規則に基づく縮退

一般にプログラム依存グラフから等価な逐次プログラムの制御の流れを再構成することは、困難と言われている。これは、制御の依存関係のみの表現となっているため、依存関係を満足する制御の流れは一意に決定できない上に、グラフを変形するような最適化を行なった場合、依存関係を満足するような制御の流れが存在しないような場合も出てくるためである。

[0034] しかし、表現するプログラムの制御構造を、if文、while文、及び、代入文に限定し、プログラム依存グラフの制御依存部分グラフ(頂点と制御依存辺のみで構成される部分グラフ)の形が木構造となる場合は、プログラムの制御の流れを再構成できることが知られている(非特許文献1)。そこで、プログラムにおけるif文、while文でない制御文に対して、入り口と出口がそれぞれ1つとなるようなプログラムのブロックを求める。ブロック全体とブロック内部の依存関係を1つの頂点に縮退することで、安全に制御の流れを再構成可能な範囲の縮退プログラム依存グラフを作成する。

[0035] 2. 結合度に基づく縮退

プログラム依存グラフを探索して、頂点間の結合の強さを求める。結合度は、データ依存辺とその大きさ、及び、制御依存辺、処理の大きさから計算されるものとする。ある結合度以上の頂点に対して、縮約可能な条件を満足する場合は、頂点を結合し依存関係を縮約する。ここで、次の2つ条件を満たすときに、頂点を結合しての縮約が可能となる。

[0036] 1)プログラム依存グラフに対応するCFG(Control Flow Graph:制御フローグラフ)上で頂点集合外から頂点集合内への分岐は頂点集合の先頭頂点へのみであり、頂点集合内から頂点集合外への分岐は頂点集合の最後の頂点のみである。

[0037] 2)頂点間のデータ依存パスに外部の頂点が含まれない。

[0038] 以上のようにして、「構文規則に基づく縮退」又は「結合度に基づく縮退」により、頂点数が大幅に削減された縮退プログラム依存グラフを生成することができる。縮退プログラム依存グラフは、次の要素から構成される。

[0039] [V:頂点集合]

エントリ:プログラムの開始ポイントを表す。

[0040] 初期定義:プログラム開始時の初期値の定義を表す。

[0041] プリディケート: If-then-elseまたはwhile-loopの条件判定を表す。

[0042] 文の集合: プログラムを構成する文の集合を表す。

[0043] 最終使用:プログラム終了時の変数の参照を表す。

[0044] [E:辺集合]

[制御依存辺: $v \rightarrow_{c}^{L} w$]プリディケート頂点vに対して、その条件判定結果により、頂点wに到達するか否かが決まることを表す。Lは条件判定のフラグを表し、L=Tのときは条件判定結果が真の場合に頂点wを実行し、L=Fのときは結果が偽の場合に頂点wを実行する。

[0045] [データ依存辺]

[ループ独立フロー依存辺: $v \rightarrow_{li}^{x} w$]頂点vで代入された変数xの値を、頂点wで参照するような場合のデータ依存関係を表す。ここでは、ループを繰り越さない場合のみを表す。

[0046] [ループ繰り越しフロー依存辺: $v \rightarrow_{lc(L)}^{x} w$]頂点vで代入された変数xの値を、頂点wで参照するような場合のデータ依存関係を表す。ループLを繰り越す場合を表す。

[0047] [定義順序関係: $v \rightarrow_{do(u)}^{x} w$]頂点v及び頂点wが変数xの値を代入し、頂点uで参照するような場合の、頂点vと頂点wの順序関係を表す。制御の流れによっては、v, wの順に実行される可能性がある場合に、その実行順序を表すものである。

[0048] 以下において、スレッド制御プログラムを生成する処理(図1のステップS3)及びスレッド・プログラムを生成する処理(図1のステップS4)について説明する。

[0049] まずスレッド・プログラムの生成について説明する。上記のようにして生成された縮退プログラム依存グラフの頂点は、入力逐次プログラムの文の部分集合であって、文の間の制御の流れの情報を有している。従って、着目する1つの頂点へのデータフロー入力辺が表す変数を入力とし、データフロー出力辺が表す変数を出力として、1つのスレッド・プログラムを1つの頂点に対して生成する。また、制御の流れよりスレッド・プログラムの本文を、また、本文の実行に必要な局所変数をそれぞれ生成する。

[0050] 図2は、スレッド・プログラム生成方法の概要を示す図である。図3は、図2のスレッド・プログラム生成方法により生成されるスレッド・プログラムを示す図である。

[0051] 図2のステップS1において、着目頂点についてデータフロー入力辺が表す変数を

入力として、入力変数の受信のためのプログラム部分を生成する。これにより、図3に示す入力変数の受信部分10が生成される。ステップS2において必要な変数を探索する。更にステップS3において、探索により見つかった変数について変数宣言を生成する。これにより、図3に示す変数宣言部分11が生成される。

[0052] ステップS4において、着目頂点の文の間の制御の流れの情報に基づいて、プログラムの本文を生成する。これにより、図3に示すプログラム本体部分12が生成される。ステップS5において、着目頂点のデータフロー出力辺が表す変数を出力として、出力変数の送信のためのプログラム部分を生成する。これにより、図3に示す出力変数の送信部分13が生成される。

[0053] 次にスレッド制御プログラムの生成について説明する。非特許文献1に記載される技術に基づいて、縮退したプログラム依存グラフから制御の流れを安全に再構成することができる。具体的には、縮退したプログラム依存グラフの制御依存部分木について、各中間節点に対して、子頂点の実行順序を求める(スケジューリング)。その結果より、各中間節点が表示する制御構造と子頂点が表すスレッド・プログラムの呼出を行なうプログラムを生成する。スレッド・プログラムを呼び出す際に、入力データ及び出力データの送受信と待ち合わせを行なうコードも生成する。このようなスレッド制御プログラムの生成については、以下の実施例において詳細に説明する。

[0054] 以下に、本発明の実施例について詳細に説明する。第1の実施例は、非同期遠隔手続き呼び出しによる並列化プログラムの実現に関する。

[0055] スレッド・プログラムとしては、頂点が表す文/文の集合を実行する手続きとする。従って、入力変数を手続きの引数とし、出力変数を復帰値あるいは、出力変数を格納するアドレスを引数として受け取るような手続きを作成する。

[0056] 次に、頂点の部分プログラムが使用、定義する変数で、入力の変数以外を求め、変数の宣言を生成する。部分プログラムを出力し、最後に、復帰値として出力の変数の値を返すreturn文、あるいは、引数として受け取ったアドレスに対して、出力する変数の値を代入する文を生成する。

[0057] スレッド制御プログラムとしては、頂点の手続きを非同期の遠隔呼び出しとして呼び出すものとする。また、後続頂点の表す手続きを呼び出すときには、先行する手続き

呼び出しを待ち合わせるものとする。最後に、プログラムの実行結果(最終使用の変数の値)を保証するために、最終使用の変数を出力する手続き待ち合わせる。

[0058] なお、手続きの終了待ち合わせを制御するために、手続き呼び出しの状態を表す識別子を用いることとする。手続き呼び出しの状態として、"未実行"、"呼び出し中"、"待ち合わせ済"の3つの状態を考えるものとする。

[0059] 図4は、スレッド制御プログラムの生成方法を示すフローチャートである。まずステップS1で、頂点間の実行順序関係の計算を行う。一般的に、プログラム依存グラフは、頂点間の依存関係のみを表現したグラフであり、頂点間の実行順序は明示されない。しかし、プログラムとして表現するためには、依存関係(制御依存、データ依存)を満足する実行順序を求めてやる必要がある。ここでは、依存関係の満足するために必要な、逆依存関係、出力依存関係を求め、実行順番を決定する。

[0060] 次にステップS2で、変数と初期値代入文を生成する。ここで変数としては、次の2種類の変数を生成する。1つは、プログラムの計算に必要な変数であり、初期定義頂点及び最終使用頂点が表現する変数を生成するとともに、プログラム依存グラフのデータ依存辺が表現する変数を生成する。もう1つは、プログラムの制御に必要な変数であり、各文/文の集合からなる頂点の遠隔手続き呼び出しを識別する変数を生成する。その初期値は「未実行」とする。

[0061] 次にステップS3で、頂点vp以下のスレッド制御プログラムを生成する。これについては以下に詳細に説明する。更にステップS4で、終了値の待ち合わせを生成する。

[0062] 図5は、頂点間の実行順序関係を決定する方法を示すフローチャートである。図5の処理は、図4のステップS1に相当する。図5に示す処理の入力は縮退したプログラム依存グラフPDGであり、出力は縮退したプログラム依存グラフPDG及びその制御の流れである。

[0063] ステップS1で、縮退したプログラム依存グラフPDGのエントリ頂点(プログラムの開始ポイント)をvとする。ステップS2で、頂点v以下の制御の流れを再構成する。以上で処理を終了する。

[0064] 図6は、頂点v以下の制御の流れを再構成する処理(図5のステップS2)を示すフローチャートである。図6の処理の入力は、縮退したプログラム依存グラフPDG及び頂

点vである。

[0065] ステップS1で、Region(v, T) = {u | u ∈ V, $v \rightarrow_c^T u$ ∈ E}が空集合であるか否かを判断する。空集合であれば処理を終了し、空集合でなければステップS2に進む。ここでRegion(v, T)とは、頂点uの集合であって、頂点vから頂点uへのTrueの制御依存関係が存在するものである。ここでVは頂点集合、Eは辺集合、$v \rightarrow_c^T u$はTrueの制御依存辺を示すものである。

[0066] ステップS2で、Region(v, T)の実行順序関係を計算する。ステップS3で、Region(v, F) = {u | u ∈ V, $v \rightarrow_c^F u$ ∈ E}が空集合であるか否かを判断する。空集合であれば処理を終了し、空集合でなければステップS4に進む。ここでRegion(v, F)とは、頂点uの集合であって、頂点vから頂点uへのFalseの制御依存関係が存在するものである。以上で処理を終了する。

[0067] 図7は、Regionの実行順序関係を計算する処理を示すフローチャートである。この処理は、図6のステップS2及びステップS4の各々に対応する。図7の処理の入力は、縮退したプログラム依存グラフPDG及びV'(着目Region)である。

[0068] ステップS1で、着目領域V'の各頂点vについて、ステップS2乃至S3の処理を繰り返すループを開始する。ステップS2で、vがプレディケート頂点(If-then-else又はwhile-loopの条件判定を表す頂点)であるか否かを判断する。vがプレディケート頂点である場合のみ、ステップS3を実行する。ステップS3で、頂点v以下の実行順序関係を計算する。

[0069] 次に、ステップS4で、逆依存及び出力依存を求める。ここでは制御の流れに起因するデータ依存関係(逆依存、出力依存)を抽出する。具体的には、着目領域(Region)を越えるデータ依存関係から、着目領域内の逆依存及び出力依存を表出する。

[0070] 次に、ステップS5で、逆依存及び出力依存を求める。ここでは着目領域(Region)内の実行順序を決定する。即ち、実行順序が一意に定まらないRegion内頂点の集合について適切な実行順序制約を決定する。具体的には、求められた逆依存関係や出力依存関係などによる実行順序制約をもとに、Region内の逆依存関係や出力依存関係を明らかにして、実行順序を決定する。実行順序が任意となる場合は、実行順序を仮定して逆依存関係、出力依存関係を求め、矛盾が起きない実行順序が得ら

れるまで試行を繰返す。

[0071] 最後にステップS6でスケジューリングを行う。即ち、上で求めた実行順次関係に基づいて頂点の実行順を決定する。これは、半順序関係の成立するグラフのスケジューリングという一般的な問題に帰着できる。従って、トポロジカル・ソートや、頂点の実行時間の概算を重みとしたリスト・スケジューリングなどのよく知られたスケジューリング手法を適用することができる。

[0072] 図8は、逆依存及び出力依存を求める処理(図7のステップS4)を示すフローチャートである。図8の処理の入力は、縮退したプログラム依存グラフPDG及びV'(着目Region)である。

[0073] ステップS1で、着目領域V'を越える変数参照を抽出してV_{def}とする。ステップS2で、着目領域V'を越える変数代入を抽出してV_{use}とする。ステップS3で、V_{use}及びV'に基づいて逆依存辺を追加する。ステップS4で、V_{def}及びV'に基づいて出力依存辺を追加する。以上で処理を終了する。

[0074] 図9は、着目領域を越える変数参照を抽出する処理を示すフローチャートである。図9の処理は図8のステップS1に相当し、縮退したプログラム依存グラフPDG及びV'(着目Region)を入力とする。

[0075] ステップS1で、頂点の集合V_{use}を空にする。ステップS2で、着目領域V'内の各フロー依存辺について以降の処理を繰り返すループを開始する。ここでフロー依存辺としては、ループ独立フロー依存辺とループ繰り越しフロー依存辺とを含む。ステップS3で、フロー依存辺eの依存元頂点をuとするとともに、辺eの依存先頂点をvとする。

[0076] ループ繰り越しフロー依存辺である場合には、ステップS4で、依存先頂点vが着目領域V'に含まれるという条件が満たされるか否かを判定する。またループ独立フロー依存辺である場合には、ステップS5で、依存元頂点uが着目領域V'に含まれず且つ依存先頂点vが着目領域V'に含まれるという条件が満たされるか否かを判定する。この判定結果がyesの場合のみ、ステップS6を実行する。ステップS6で、頂点の集合V_{use}に依存先頂点vを追加する。

[0077] 最後に、ステップS7で、頂点の集合V_{use}を値として返す。以上で処理を終了する。

[0078] 図10は、着目領域を越える変数代入を抽出する処理を示すフローチャートである。

図10の処理は図8のステップS2に相当し、縮退したプログラム依存グラフPDG及びV'(着目Region)を入力とする。

[0079] ステップS1で、頂点の集合V_{def}を空にする。ステップS2で、着目領域V'内の各フロー依存辺について以降の処理を繰り返すループを開始する。ここでフロー依存辺としては、ループ独立フロー依存辺とループ繰り越しフロー依存辺とを含む。ステップS3で、フロー依存辺eの依存元頂点をuとするとともに、辺eの依存先頂点をvとする。

[0080] ループ繰り越しフロー依存辺である場合には、ステップS4で、依存先頂点vが着目領域V'に含まれるという条件が満たされるか否かを判定する。またループ独立フロー依存辺である場合には、ステップS5で、依存元頂点uが着目領域V'に含まれ且つ依存先頂点vが着目領域V'に含まれないという条件が満たされるか否かを判定する。何れかの判定結果がyesの場合のみ、ステップS6を実行する。ステップS6で、頂点の集合V_{def}に依存先頂点vを追加する。

[0081] 最後に、ステップS7で、頂点の集合V_{def}を値として返す。以上で処理を終了する。

[0082] 図11は、逆依存の追加処理を示すフローチャートである。図11の処理は図8のステップS3に相当し、縮退したプログラム依存グラフPDG、V'(着目Region)、及び頂点集合V_{use}を入力とする。

[0083] ステップS1で、頂点集合V_{use}の各頂点vに対して以降の処理を繰り返すループを開始する。ステップS2で、頂点vで使用する各変数xに対して以降の処理を繰り返すループを開始する。ステップS3で、着目領域V'の各頂点uに対して以降の処理を繰り返すループを開始する。

[0084] ステップS4で、頂点uが変数xを定義するか否かを判定する。判定結果がyesの場合のみ、ステップS5を実行する。ステップS5において、vからuへの逆依存辺を追加する。以上で処理を終了する。

[0085] 図12は、出力依存の追加処理を示すフローチャートである。図12の処理は図8のステップS4に相当し、縮退したプログラム依存グラフPDG、V'(着目Region)、及び頂点集合V_{def}を入力とする。

[0086] ステップS1で、頂点集合V_{def}の各頂点uに対して以降の処理を繰り返すループを開始する。ステップS2で、頂点uで使用する各変数xに対して以降の処理を繰り返す

ループを開始する。ステップS3で、着目領域V'の各頂点vに対して以降の処理を繰り返すループを開始する。

[0087] ステップS4で、頂点vが変数xを定義するか否かを判定する。判定結果がyesの場合のみ、ステップS5を実行する。ステップS5において、vからuへの出力依存辺を追加する。以上で処理を終了する。

[0088] 図13は、逆依存及び出力依存を求める処理(図7のステップS5)を示すフローチャートである。図13の処理の入力は、縮退したプログラム依存グラフPDG及びV'(着目Region)である。

[0089] ステップS1で、着目領域内の全域木を求めSとする。変数xを定義する頂点vとその変数xを使用するRegionR内の頂点との集合として、頂点vの変数xに関する全域木が、

$$Span(v, x) = \{v\} \cup \{u \mid v \xrightarrow[li]{x} u \in E_R\}$$

と定義される。図14は、全域木を説明するための図である。図14に示されるプログラム依存グラフにおいて、頂点v_iにおいて変数xが定義され、2つの頂点v1及びv2が変数xを使用する。この場合、頂点v_i、v1、及びv2で全域木21を形成する。また頂点v_jにおいて変数xが定義され、2つの頂点v3及びv4が変数xを使用する。この場合、頂点v_j、v3、及びv4で全域木22を形成する。図15は、全域木を模式的に示す図である。全域木$Span(v_i, x)$及び全域木$Span(v_j, x)$が、データ依存グラフとして図15に示されるように構成される。

[0090] 図13に戻り、ステップS2で、実行順が未決定である2つの任意の全域木を順次選択して以降の処理を繰り返すループが開始される。ステップS3で、着目領域に閉路がなく、同一変数xに対する独立した全域木$Span(h_0,x)$及び$Span(h_1,x)$が存在するか否かを判定する。ここで、「独立した」とは、2つの全域木 $Span(h_0,x)$及び$Span(h_1,x)$について、$Span(h_0,x)$に含まれる頂点と$Span(h_1,x)$に含まれる頂点との間に辺(依存関係)がないことを言う。

[0091] ステップS4でR(Region)のオリジナルをスタックに退避させる。ステップS5で、$h_0 \rightarrow h_1$の出力依存辺を追加し、推移閉包を求める。ステップS6で、全域木間の順序関係を計算する。

[0092] ステップS7で、R(Region)に閉路が存在するか否かを判定する。存在しない場合には、以降の処理ステップS8～ステップS11をスキップする。存在する場合には、ステップS8に進む。ステップS8で、スタックが空か否かを判断する。空の場合にはエラー終了する。空でない場合には、ステップS9で、Rのオリジナルをスタックから取り出す。

[0093] 以上の処理は、頂点h_0からh_1への出力依存関係をグラフに追加したときに、巡回グラフとならない場合には追加した依存関係を確定させ、巡回グラフになった場合には元のグラフに戻すことに相当する。元のグラフに戻した後は、以降に示すように、頂点h_1からh_0への出力依存関係をグラフに追加する。即ち、ステップS10で、$h_1 \rightarrow h_0$の出力依存辺を追加し、推移閉包を求める。ステップS11で、全域木間の順序関係を計算する。

[0094] 以上の処理により、2つの全域木 Span(h_0,x)及びSpan(h_1,x)に対する実行順序が決定する。更に、実行順が未決定である2つの任意の全域木を順次選択して同様の処理を繰り返し、全ての全域木間の順序関係が決定されたところで終了する。

[0095] 図16は、全域木間の順序関係を計算する処理を示すフローチャートである。図16の処理は、図13のステップS6及びステップS11に相当する。図16の処理の入力は、縮退したプログラム依存グラフPDG及びV'(着目Region)である。

[0096] ステップS1で、着目領域内の各辺e(頂点v→頂点w)について以降の処理を繰り返すループを開始する。ステップS2で、頂点wで定義され、頂点vで参照される各変数xについて以降の処理を繰り返すループを開始する。

[0097] ステップS3で、$V_a \leftarrow \{ u \mid v \in \mathrm{Span}(u, x) \}$とするとともに、$V_b \leftarrow \{ u \mid w \in \mathrm{Span}(u, x) \}$とする。これは、頂点vを要素として含む変数xに関する全域木における変数xを定義する頂点の集合を求めるとともに、頂点wを要素として含む変数xに関する全域木における変数xを定義する頂点の集合を求めることである。

[0098] ステップS4で、V_aの各頂点v_aについて以降の処理を繰り返すループを開始する。ステップS5で、V_bの各頂点v_bについて以降の処理を繰り返すループを開始する。更にステップS6で、Span(v_a, x)の頂点であってSpan(v_b, x)の頂点でない各頂点v_cについて以降の処理を繰り返すループを開始する。

[0099] ステップS7で、vc→vbがE(辺集合)に含まれるか否かを判定する。判定結果がyesの場合のみステップS8を実行する。ステップS8で、$v_c \rightarrow v_b$の逆依存辺を追加し、推移閉包を求める。以降、各ループの処理を繰り返す。

[0100] 図17は、図16の処理による逆依存辺の追加について説明する図である。図17には、頂点vの変数xに関する全域木Span(v,x)と頂点wの変数xに関する全域木Span(w,x)とが示される。頂点vを要素として含む変数xに対する全域木Span(v_a, x)(即ちSpan(v,x))の各頂点v_c(即ちv、25、26)に対して、全域木Span(v_b, x)(即ちSpan(w,x))のヘッドv_b(変数を定義している頂点w)への逆依存辺32、33を追加する。

[0101] 図18は、頂点間の実行順序関係を決定する方法の変形例を示すフローチャートである。図18のフローチャートに示す処理を、図5のフローチャートに示す処理の代わりに用いてもよい。即ち、頂点間の実行順序関係を決定する処理において、前段階のステップS0として、SSA(静的単一代入形式)を適用する処理を実行してもよい。即ち、縮退プログラム依存グラフを静的単一代入形式に変換してもよい。この場合、図7に示すステップS7の処理(逆依存、出力依存を求め着目領域内の実行順序を決定する処理:図13のフローチャート)を省略することができる。

[0102] 図19は、頂点vp以下のスレッド制御プログラムを生成する処理を示すフローチャートである。図19の処理は、図4のステップS3に相当する。図19に示す処理の入力は縮退したプログラム依存グラフPDG及び頂点v_pである。

[0103] ここで頂点v_pとは着目頂点である。また以下の説明において、頂点v_p以下の頂点(子頂点)とは、縮退プログラム依存グラフの制御依存辺に着目したときの、部分グラフ(木構造になる)における頂点v_p以下の頂点を指すものとする。言い換えれば、頂点v_pから頂点uへの制御依存辺によるパス(道)が存在するような、頂点uの集合を指す。

[0104] ステップS1で、v_pのデータ依存関係(vpの子頂点以下からの自ループへのループ繰り越しフロー依存関係を除く)に関する先行頂点の実行終了待ち合わせを生成する。本実施例では非同期の手続き呼び出しを利用するため、手続きの実行結果を利用する場合は、先行する手続き呼び出しの終了を待ち合わせる必要がある。先行頂点の手続き呼び出しを表す識別子を用いて、先行頂点の状態が「呼び出し中」の場合は終了を待ち合わせるというコードを生成する。待ち合わせ後は、状態を「待ち合

わせ済」とするコードを生成する。

[0105] ステップS2でv_pの種類を判定する。頂点v_pの種類によって以下のようなコードを生成する。

[0106] ステップS2の判定の結果、v_pが文（又は文の集合）頂点の場合は、ステップS3で、頂点v_pの処理を呼び出すコードを生成する。即ち、非同期で頂点に相当する遠隔手続き呼び出しを行なうコードを生成する。また当該手続き呼び出しを表す識別子に対して、その状態を「呼び出し中」とするコードを生成する。

[0107] ステップS2の判定の結果、v_pがループのプリディケート頂点の場合は、ステップS4で、forプリディケートであるかwhileプリディケートであるかに応じて、for文或いはwhile文を生成する。ステップS5で、v_pの子頂点以下の実行終了待ち合わせを生成する。即ち、ループの最初において、前に実行したループ本文の処理が実行中であれば、終了待ち合わせを行なうコードを生成する。

[0108] ステップS6で、v_pのL＝T（条件判定結果が真）の場合の制御依存子頂点v_cを、求めた実行順序順に探索して、ステップS7の処理を繰り返すループを開始する。ステップS7で、v_c以下のスレッド制御プログラムを生成する。即ち、子頂点v_cが表すプログラムを上記生成したfor／while文ループの本文として生成する。このステップS6の処理は、図19のフローチャートの処理に対応し、処理が入れ子構造となっている。

[0109] ステップS8で、v_pのループ繰越フロー依存入力辺に関して自ループ内の先行頂点の実行終了待ち合わせを生成する。即ち、次のループへ向けて、頂点v_p実行に必要なデータを待ち合わせるため、v_pの先行頂点の実行終了を待ち合わせるコードを生成する。

ステップS2の判定の結果、v_pがif文のプリディケート頂点の場合は、ステップS9でif文を生成する。次にステップS10で、then節を生成する。

[0110] 次にステップS11で、v_pのL＝T（条件判定結果が真）の場合の制御依存子頂点v_cを、求めた実行順序順に探索して、ステップS12の処理を繰り返すループを開始する。ステップS12で、v_c以下のスレッド制御プログラムを生成する。即ち、子頂点v_cが表すプログラムを上記生成したthen節の本文として生成する。このステップS12の処理は、図19のフローチャートの処理に対応し、処理が入れ子構造となっている。

[0111] ステップS13で、L＝F（条件判定結果が偽）の制御依存出力辺が存在するか否かを判定する。存在しない場合には処理を終了し、存在する場合にはステップS14に進む。

[0112] ステップS14で、else節を生成する。ステップS15で、v_pのL＝F（条件判定結果が偽）の場合の制御依存子頂点v_cを、求めた実行順序順に探索して、ステップS16の処理を繰り返すループを開始する。ステップS16で、v_c以下のスレッド制御プログラムを生成する。即ち、子頂点v_cが表すプログラムを上記生成したelse節の本文として生成する。このステップS16の処理は、図19のフローチャートの処理に対応し、処理が入れ子構造となっている。

[0113] 以上の処理を実行することで、頂点vp以下のスレッド制御プログラムが生成される。以下に、第1の実施例により生成されたスレッド・プログラム及びスレッド制御プログラムについて、その構成及び動作を具体的な例を用いて説明する。

[0114] 図20は、（a）入力逐次プログラムの部分及び（b）対応する縮退プログラム依存グラフを示す図である。図20（a）に示す入力逐次プログラムからプログラム依存グラフを生成し、頂点を結合して縮退することにより、（b）に示す縮退プログラム依存グラフが生成される。頂点v_0からv_5が存在し、頂点v_1は縮退により文の集合となっている。

[0115] 図21は、図20の縮退プログラム依存グラフから第1の実施例に従い生成されるスレッド制御プログラムである。最初に変数の宣言があり、使用する変数x, y, a, b, pを宣言するとともに、手続を表す変数v1, v3, v4, v5を"未実行"に設定する（変数宣言&設定41）。

[0116] 図20（a）に示す逐次プログラムのif文の中は、（b）に示す縮退プログラム依存グラフの頂点v_1に対応する。図21において、この頂点v_1に対応する手続v1が手続呼び出し文42で呼び出される。続いて、代入文43において、対応する手続識別変数v1が"呼び出し中"に設定される。

[0117] 図20（a）に示す逐次プログラムのwhile文の中では、変数a, b, x, yが使用されるので、これらの変数を待ち合わせる必要がある。図21の終了待ち合わせ文44において、手続v1の終了を待ち合わせ、変数x, yの値が手続の返し値として返される。続いて、代入文45において、対応する手続識別変数v1が"待ち合わせ済み"に設定

される。また手続v4が呼び出し中である場合には、終了待ち合わせ文46において、手続v4の終了を待ち合わせ、変数bの値が手続の返し値として返される。

[0118] その後、手続呼び出し文47で手続v3を呼び出し、手続呼び出し文48で手続v4を呼び出す。これらの手続は、図20(b)に示される頂点v_3及び頂点v_4に対応する。

[0119] 更に、終了待ち合わせ文49で手続v3を待ち合わせて変数aの値を獲得し、while文の先頭に戻る。獲得したaの値を用いて条件判定を行ない、結果が偽であれば、ループを抜ける。その後、終了待ち合わせ文50で手続v4を待ち合わせて変数bの値を獲得する。最後に、変数a, bを用いて手続呼び出し文51で手続v5を呼び出して、終了待ち合わせ文52で変数xの値を獲得する。

[0120] 図22は、以上のスレッド制御プログラムの動作をスレッド・プログラムの実行とともに示す模式図である。図22では、プロセッサ0乃至プロセッサ3の4つのプロセッサが用いられる。プロセッサ0でスレッド制御プログラムを実行する。while文の条件が成立すると、手続v1は実行されなかったので、ただちに、手続v3が最初に呼び出されスレッド・プログラム61がプロセッサ1により実行される。またそれに続いて手続v4が呼び出されスレッド・プログラム62がプロセッサ2により実行される。これらは図21の手続呼び出し文47での手続v3呼び出し、手続呼び出し文48での手続v4呼び出しに対応する。whileループの末尾にて、条件判定に用いる変数aの値を待ち合わせるべく、手続v3が待ち合わされる。これは図21の手続き待ち合わせ文49に対応する。

[0121] また再度while文の条件が成立すると、while文の2度目のループにおいて、先のループで呼び出しを行なった手続v4の待ち合わせが行なわれる。これは、図21の手続き待ち合わせ文46に対応する。手続v3が呼び出されスレッド・プログラム63(プログラムコードは61と同一)がプロセッサ1により実行される。またそれに続いて手続v4が呼び出されスレッド・プログラム64(プログラムコードは62と同一)がプロセッサ2により実行される。また、再度、手続v3の待ち合わせが行なわれる。

[0122] while文終了後に、手続v4を待ち合わせて、手続v5が呼び出されスレッド・プログラム65がプロセッサ3により実行される。これは、それぞれ、図21の終了待ち合わせ文50での待ち合わせと、手続呼び出し文51での手続v5呼び出しに対応する。最後に、手続v5の終了を待ち合わせ、その返り値を取得して、プログラムを終了する。

[0123] 以下に、本発明の第2の実施例として、メッセージ通信による並列化プログラムの実現について説明する。基本的に、スレッド・プログラムとスレッド制御プログラムの生成の仕方は第1の実施例と同様であり、如何にスレッド間のやりとりを実現するかが、手続呼び出しとメッセージ通信とで異なるだけである。

[0124] スレッド・プログラムは、注目する頂点へのフロー依存辺が表す入力変数を受信するコードと、実行の開始を指示するメッセージを受信するコードと、頂点が表す文/文の集合と、当該頂点からのフロー依存辺が表す出力変数を送信するコード、実行の完了を表すメッセージを送信するコードから構成される。スレッド制御プログラムは、プリディケート頂点で表現される分岐及びループの制御を行なうコードと、文/文の集合を表す頂点に対して、入力変数と手続きの実行開始を指示するメッセージを送信し、出力変数と終了を通知するメッセージを受信するコードから構成される。出力変数及び終了通知の受信は、後続の頂点で必要となった時点で行なうものとする。最後に、プログラムの実行結果(最終使用の変数の値)を保証するために、変数及び実行完了のメッセージを受信する。終了通知の待ち合わせを制御するために、頂点の状態を表す識別子を用い、"未実行"、"実行中"、"待ち合わせ済"の3つのいずれかの状態を持つものとする。

[0125] 図23は、図20の縮退プログラム依存グラフから第2の実施例に従い生成されるスレッド制御プログラムである。最初に変数の宣言があり、使用する変数x, y, a, b, pを宣言するとともに、手続を表す変数v1, v3, v4, v5を"未実行"に設定する(変数宣言&設定71)。

[0126] 図20(a)に示す逐次プログラムのif文の中は、(b)に示す縮退プログラム依存グラフの頂点v_1に対応する。図23において、この頂点v_1に対応する手続v1にメッセージ送信文72で実行の開始を指示する。続いて、代入文73において、対応する手続識別変数v1が"実行中"に設定される。

[0127] 図20(a)に示す逐次プログラムのwhile文の中では、変数a, b, x, yが使用されるので、これらの変数を待ち合わせる必要がある。図23の終了通知メッセージ受信文74において、手続v1からの終了通知メッセージを待ち合わせ、変数xの値が通知される。続いて、代入文75において、対応する手続識別変数v1が"待ち合わせ済み"に

設定される。また手続v4が実行中である場合には、終了通知メッセージ受信文76において、手続v4からの終了通知メッセージを待ち合わせ、変数bの値が通知される。

[0128] その後、メッセージ送信文77で手続v3に実行開始を指示し、メッセージ送信文78で手続v4に実行開始を指示する。これらの手続は、図20(b)に示される頂点v_3及び頂点v_4に対応する。更に、終了通知メッセージ受信文79で手続v3からの終了通知メッセージを待ち合わせ、while文の先頭に戻る。受信した変数aの値を用いて条件判定を行ない、結果が偽であれば、ループを抜ける。その後、終了通知メッセージ受信文80で手続v4からの終了通知メッセージを待ち合わせる。最後に、変数a, bを用いてメッセージ送信文81で手続v5に実行開始を指示して、終了通知メッセージ受信文82で変数xの値を獲得する。

[0129] 図24は、以上のスレッド制御プログラムの動作をスレッド・プログラムの実行とともに示す模式図である。図24では、プロセッサ0乃至プロセッサ3の4つのプロセッサが用いられる。プロセッサ0でスレッド制御プログラムを実行する。while文の条件が成立すると、if文が成立しなかったため、ただちに、実行開始メッセージが必要な変数x, aとともに手続v3に送信され、スレッド・プログラム91がプロセッサ1により実行される。またそれに続いて実行開始メッセージが必要な変数y, bとともに手続v4に送信され、スレッド・プログラム92がプロセッサ2により実行される。これらは図23におけるメッセージ送信文77での手続v3への通信、メッセージ送信文78での手続v4への通信に対応する。whileループの末尾にて、条件判定に用いる変数aが受信される。これは図23の手続き待ち合わせ文79に対応する。

[0130] また再度while文の条件が成立すると、while文の2度目のループにおいて、手続v4の変数が受信される。これは図23のメッセージ受信文76に対応する。実行開始メッセージが手続v3に送信され、スレッド・プログラム93(プログラムコードは91と同一)がプロセッサ1により実行される。またそれに続いて実行開始メッセージが手続v4に送信され、スレッド・プログラム94(プログラムコードは92と同一)がプロセッサ2により実行される。また、再度、手続v3の待ち合わせが行なわれる。

[0131] while文終了後に、手続v4の変数bが受信され、実行開始メッセージが必要な変数a, bとともに手続v5に送信され、スレッド・プログラム95がプロセッサ3により実行され

る。これは、それぞれ、図23のメッセージ受信文70での待ち合わせと、メッセージ送信文81での手続v5への通信に対応する。最後に、手続v5の変数xを受信して、プログラムを終了する。

[0132] 以下に、本発明の第3の実施例として、マルチ・スレッド・ライブラリによる並列化プログラムの実現について説明する。基本的に、スレッド・プログラムとスレッド制御プログラムの生成の仕方は第1の実施例と同様であり、如何にスレッド間のやりとりを実現するかが、手続呼び出しとマルチ・スレッド・ライブラリとで異なるだけである。

[0133] スレッド・プログラムは、スレッドとして実行され、共有メモリを利用して入出力変数の受け渡しを行なう。そのため、入出力となる共有変数をロックすることとなる。頂点の部分プログラムを実行し、最後に、共有変数のロックを解除し、スレッドを終了する。

[0134] 具体的には、スレッドの手続きを生成する。入出力変数の受け渡し方法に、共有メモリを利用するため、入出力変数をロックするコードを生成する。次に、頂点の部分プログラムが使用、定義する変数で、入力の変数以外を求め、変数の宣言を生成する。部分プログラムを出力し、最後に、共有変数のロックを解除するコードと、スレッドを終了するコードを生成する。

[0135] スレッド制御プログラムは、頂点の手続きをスレッドとして呼び出すものとする。また、後続頂点の実行を呼び出すときには、先行するスレッドを待ち合わせるものとする。最後に、プログラムの実行結果(最終使用の変数の値)を保証するために、その値を出力するスレッドを待ち合わせる。待ち合わせを制御するために、スレッドの実行状態を表す識別子を用い、"未実行"、"実行中"、"待ち合わせ済"の3つのいずれかの状態を持つものとする。

[0136] 図25は、図20の縮退プログラム依存グラフから第3の実施例に従い生成されるスレッド制御プログラムである。最初に変数の宣言があり、使用する変数x, y, a, b, pを宣言するとともに、スレッドを表す変数v1, v3, v4, v5を"未実行"に設定する(変数宣言&設定101)。

[0137] 図20(a)に示す逐次プログラムのif文の中は、(b)に示す縮退プログラム依存グラフの頂点v_1に対応する。図25において、この頂点v_1に対応するスレッドv1がスレッド生成文102で生成される。続いて、代入文103において、対応するスレッド識別変数

v1が"実行中"に設定される。

[0138] 図20(a)に示す逐次プログラムのwhile文の中では、変数a, b, x, yが使用されるので、これらの変数を待ち合わせる必要がある。図25の終了待ち合わせ文104において、スレッドv1の終了を待ち合わせ、変数x, yの値が共有メモリを利用して受け渡される。続いて、代入文105において、対応するスレッド識別変数v1が"待ち合わせ済み"に設定される。またスレッドv4が実行中である場合には、終了待ち合わせ文106において、スレッドv4の終了を待ち合わせ、変数bの値が共有メモリを利用して受け渡される。

[0139] その後、スレッド生成文107でスレッドv3を生成し、スレッド生成文108でスレッドv4を生成する。これらのスレッドは、図20(b)に示される頂点v_3及び頂点v_4に対応する。

[0140] 更に、終了待ち合わせ文109でスレッドv3を待ち合わせて変数aの値を獲得し、while文の先頭に戻る。獲得したaの値を用いて条件判定を行ない、結果が偽であれば、ループを抜ける。その後、終了待ち合わせ文50でスレッドv4を待ち合わせて変数bの値を獲得する。最後に、変数a, bを用いてスレッド生成文51でスレッドv5を生成し、終了待ち合わせ文52で変数xの値を獲得する。

[0141] 図26は、以上のスレッド制御プログラムの動作をスレッド・プログラムの実行とともに示す模式図である。図26では、1つのプロセッサ0が用いられる。プロセッサ0でスレッド制御プログラムを実行するとともに、各スレッド・プログラムを実行する。while文の条件が成立すると、手続v1は実行されなかったので、ただちに、スレッドv3が最初に生成されスレッド・プログラム121が実行される。またそれに続いてスレッドv4が生成されスレッド・プログラム122が実行される。これらは図25のスレッド生成文107でのスレッドv3生成及びスレッド生成文108でのスレッドv4生成に対応する。whileループの末尾にて、条件判定に用いる変数aの値を獲得すべく、スレッドv3の終了が待ち合わされる。これは図25の待ち合わせ文109に対応する。

[0142] また再度while文の条件が成立すると、while文の2度目のループにおいて、先のループで生成したスレッドv4の待ち合わせが行なわれる。これは、図25の待ち合わせ文106に対応する。スレッドv3が生成されスレッド・プログラム123(プログラムコー

ドは121と同一）が実行される。またそれに続いてスレッドv4が生成されスレッド・プログラム124（プログラムコードは122と同一）が実行される。再度、スレッドv3の待ち合わせが行なわれる。

[0143] while文終了後に、スレッドv4を待ち合わせて、スレッドv5が生成されスレッド・プログラム125が実行される。これは、それぞれ、図25の待ち合わせ文110での待ち合わせと、スレッド生成文111でのスレッドv5生成に対応する。最後に、スレッドv5を待ち合わせ、その結果を取得して、プログラムを終了する。

[0144] なお、スレッドの生成及び合流の代わりに、セマフォなどを用いたスレッド間同期機構を用いて、同等の制御を行なうことも考えられる。

[0145] 図27は、本発明による並列化プログラム生成方法を実行する装置の構成を示す図である。

[0146] 図27に示されるように、本発明による並列化プログラム生成方法を実行する装置は、例えばパーソナルコンピュータやエンジニアリングワークステーション等のコンピュータにより実現される。図27の装置は、コンピュータ510と、コンピュータ510に接続されるディスプレイ装置520、通信装置523、及び入力装置よりなる。入力装置は、例えばキーボード521及びマウス522を含む。コンピュータ510は、CPU511、RAM512、ROM513、ハードディスク等の二次記憶装置514、可換媒体記憶装置515、及びインターフェース516を含む。

[0147] キーボード521及びマウス522は、ユーザとのインターフェースを提供するものであり、コンピュータ510を操作するための各種コマンドや要求されたデータに対するユーザ応答等が入力される。ディスプレイ装置520は、コンピュータ510で処理された結果等を表示すると共に、コンピュータ510を操作する際にユーザとの対話を可能にするために様々なデータ表示を行う。通信装置523は、遠隔地との通信を行なうためのものであり、例えばモデムやネットワークインターフェース等よりなる。

[0148] 本発明による並列化プログラム生成方法は、コンピュータ510が実行可能なコンピュータプログラムとして提供される。このコンピュータプログラムは、可換媒体記憶装置515に装着可能な記憶媒体Mに記憶されており、記憶媒体Mから可換媒体記憶装置515を介して、RAM512或いは二次記憶装置514にロードされる。或いは、こ

のコンピュータプログラムは、遠隔地にある記憶媒体（図示せず）に記憶されており、この記憶媒体から通信装置523及びインターフェース516を介して、RAM512或いは二次記憶装置514にロードされる。

[0149] キーボード521及び／又はマウス522を介してユーザからプログラム実行指示があると、CPU511は、記憶媒体M、遠隔地記憶媒体、或いは二次記憶装置514からプログラムをRAM512にロードする。CPU511は、RAM512の空き記憶空間をワークエリアとして使用して、RAM512にロードされたプログラムを実行し、適宜ユーザと対話しながら処理を進める。なおROM513は、コンピュータ510の基本動作を制御するための制御プログラムが格納されている。

[0150] 上記コンピュータプログラム（並列化プログラム生成プログラム即ち並列化プログラム生成コンパイラ）を実行することにより、コンピュータ510が、上記各実施例で説明されたように並列化プログラム生成方法を実行する。

[0151] 以上、本発明を実施例に基づいて説明したが、本発明は上記実施例に限定されるものではなく、特許請求の範囲に記載の範囲内で様々な変形が可能である。

請求の範囲

[1] 逐次プログラムを入力として、該逐次プログラムを構成する各文を頂点として有するとともに、文と文の間の関係を該頂点間の辺として有するプログラム依存グラフを生成し、

該プログラム依存グラフの該頂点同士を融合することにより該頂点の数を減少させた縮退プログラム依存グラフを生成し、

該縮退プログラム依存グラフの頂点の各々に相当するスレッド・プログラムを生成し、

該スレッド・プログラムの起動及び同期を制御するスレッド制御プログラムを生成する

各段階を計算機に実行させるコードを含むことを特徴とする並列化プログラム生成プログラム。

[2] スレッド制御プログラムを生成する段階は、

該縮退プログラム依存グラフの頂点間の実行順序関係を計算し、

該計算された実行順序関係順に該縮退プログラム依存グラフの該頂点を探索しながら該縮退プログラム依存グラフの各頂点の種類に応じて該スレッド制御プログラムを生成する

各段階を含むことを特徴とする請求項1記載の並列化プログラム生成プログラム。

[3] 該実行順序関係を計算する段階は、該縮退プログラム依存グラフを静的単一代入形式に変換する段階を含むことを特徴とする請求項2記載の並列化プログラム生成プログラム。

[4] 該縮退プログラム依存グラフの各頂点が表す文又は文の集合を実行する手続として該スレッド・プログラムを生成し、該手続を非同期の遠隔呼び出しとして呼び出すとともに先行頂点の手続を待ち合わせてから後続頂点の手続を呼び出すように制御するスレッド制御プログラムを生成することを特徴とする請求項1記載の並列化プログラム生成プログラム。

[5] 該縮退プログラム依存グラフの各頂点が表す文又は文の集合を実行する手続として該スレッド・プログラムを生成し、該手続に実行開始するメッセージを送信するとも

に先行頂点の手続を待ち合わせてから後続頂点の手続に実行指示するように制御するスレッド制御プログラムを生成することを特徴とする請求項1記載の並列化プログラム生成プログラム。

[6] 該縮退プログラム依存グラフの各頂点が表す文又は文の集合を実行するスレッドとして該スレッド・プログラムを生成し、共有メモリを介して入出力変数を受け渡し該スレッドを生成するとともに先行頂点のスレッドを待ち合わせてから後続頂点のスレッドを生成するように制御するスレッド制御プログラムを生成することを特徴とする請求項1記載の並列化プログラム生成プログラム。

[7] 逐次プログラムと並列化プログラム生成プログラムとを格納するメモリと、

該メモリに格納された該並列化プログラム生成プログラムを実行することで該メモリに格納された該逐次プログラムから並列化プログラムを生成する演算処理ユニットを含み、該演算処理ユニットは、該並列化プログラム生成プログラムを実行することにより、該逐次プログラムを構成する各文を頂点として有するとともに文と文の間の関係を該頂点間の辺として有するプログラム依存グラフを生成し、該プログラム依存グラフの該頂点同士を融合することにより該頂点の数を減少させた縮退プログラム依存グラフを生成し、該縮退プログラム依存グラフの頂点の各々に相当するスレッド・プログラムを生成し、該スレッド・プログラムの起動及び同期を制御するスレッド制御プログラムを生成することを特徴とする並列化プログラム生成装置。

[8] 該演算処理ユニットは該スレッド制御プログラムを生成する際に、該縮退プログラム依存グラフの頂点間の実行順序関係を計算し、該計算された実行順序関係順に該縮退プログラム依存グラフの該頂点を探索しながら該縮退プログラム依存グラフの各頂点の種類に応じて該スレッド制御プログラムを生成することを特徴とする請求項7記載の並列化プログラム生成装置。

[9] 該演算処理ユニットは該実行順序関係を計算する際に、該縮退プログラム依存グラフを静的単一代入形式に変換することを特徴とする請求項8記載の並列化プログラム生成装置。

[10] 逐次プログラムを入力として、該逐次プログラムを構成する各文を頂点として有するとともに、文と文の間の関係を該頂点間の辺として有するプログラム依存グラフを生成

し、

該プログラム依存グラフの該頂点同士を融合することにより該頂点の数を減少させた縮退プログラム依存グラフを生成し、

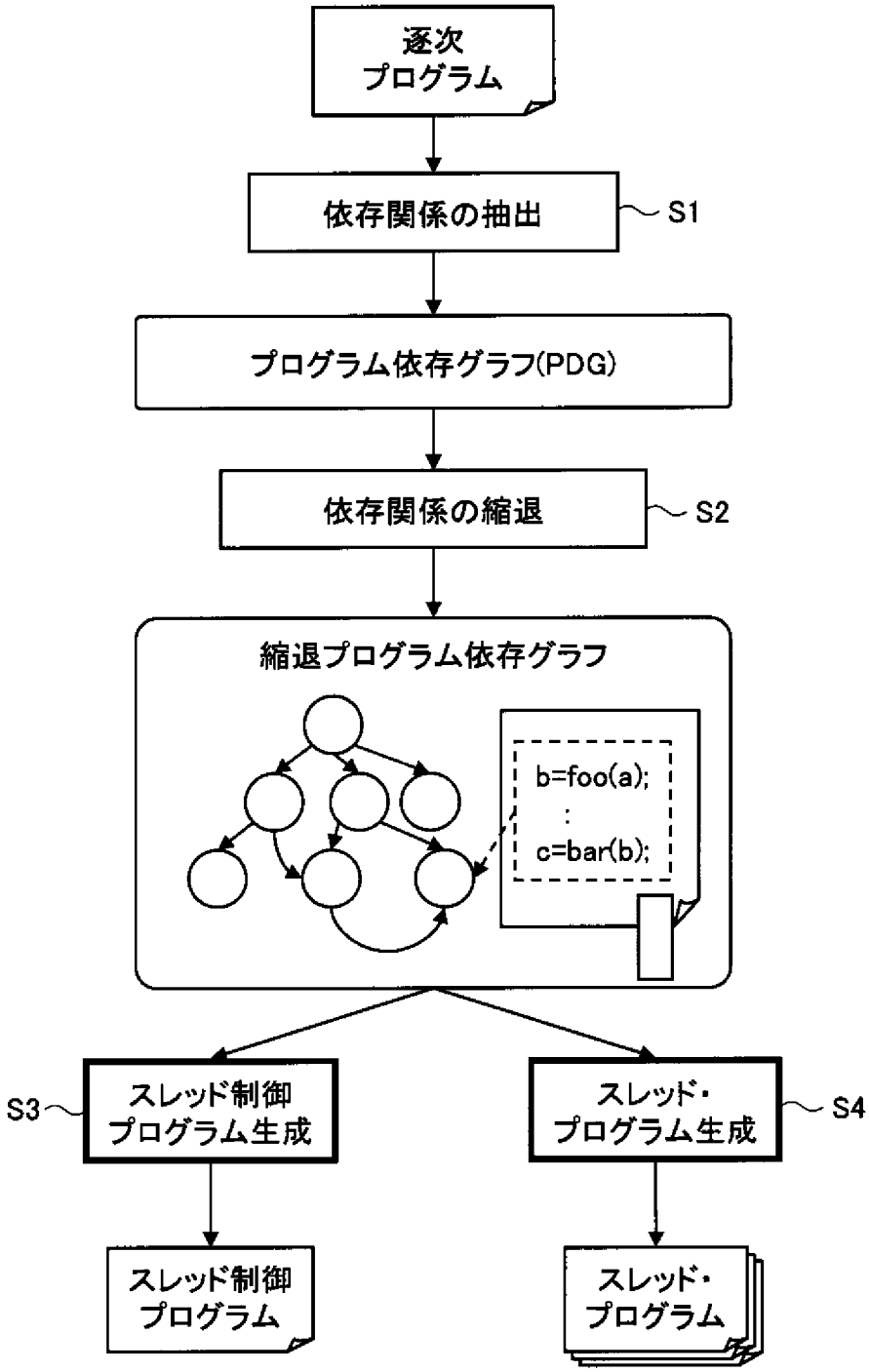
該縮退プログラム依存グラフの頂点の各々に相当するスレッド・プログラムを生成し、

該スレッド・プログラムの起動及び同期を制御するスレッド制御プログラムを生成する

各段階を含むことを特徴とする並列化プログラム生成方法。

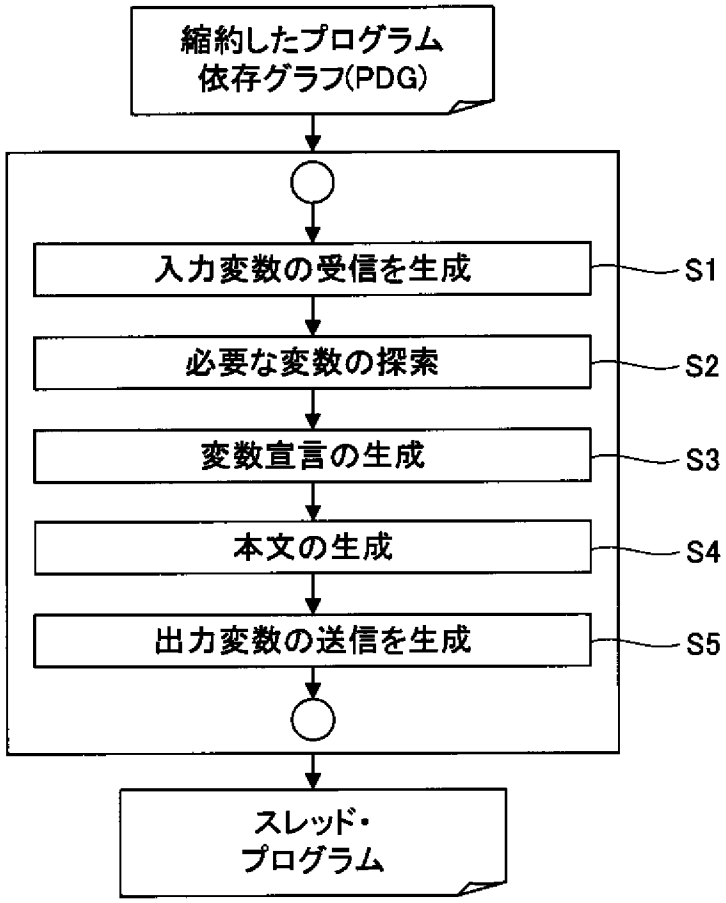
[図1]

本発明による並列化プログラム生成方法の概略を示す図



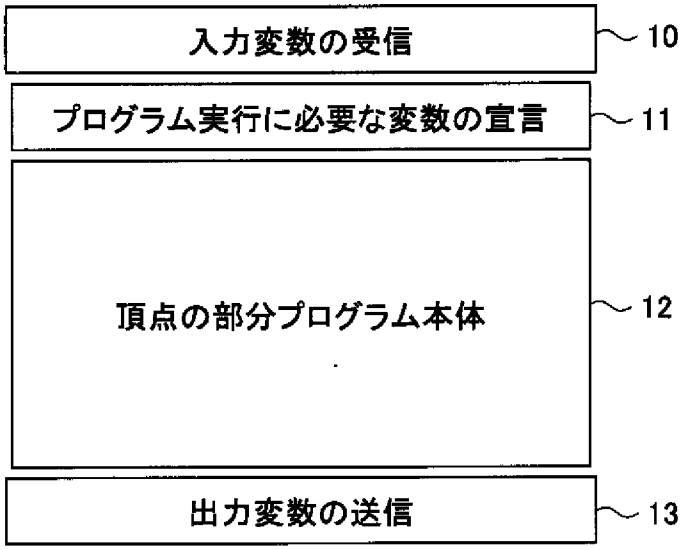
[図2]

スレッド・プログラム生成方法の概要を示す図



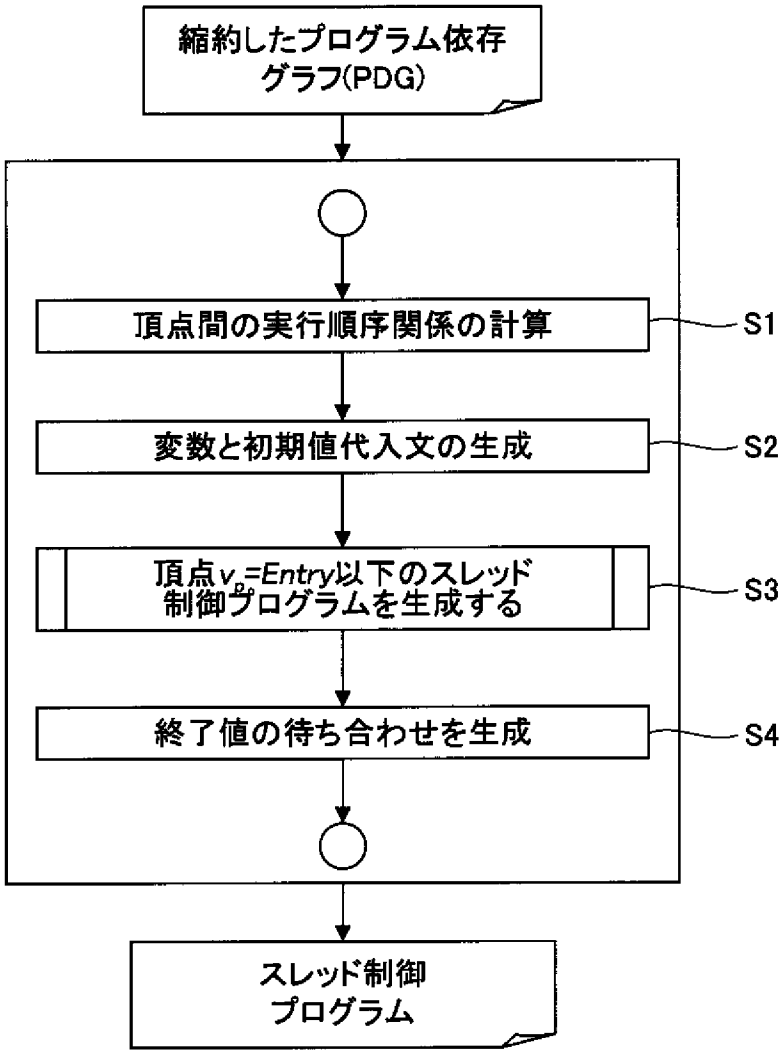
[図3]

図2のスレッド・プログラム生成方法により
生成されるスレッド・プログラムを示す図



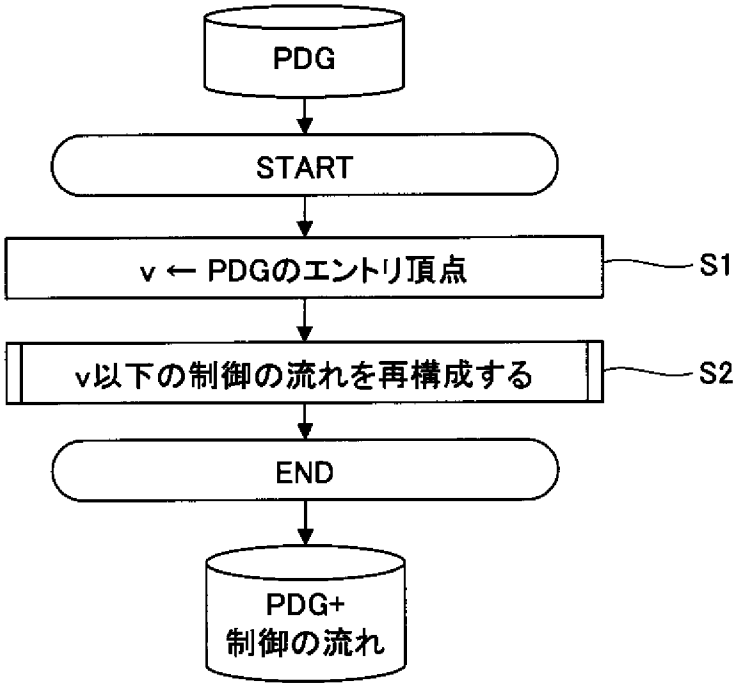
[図4]

スレッド制御プログラムの生成方法を示すフローチャート



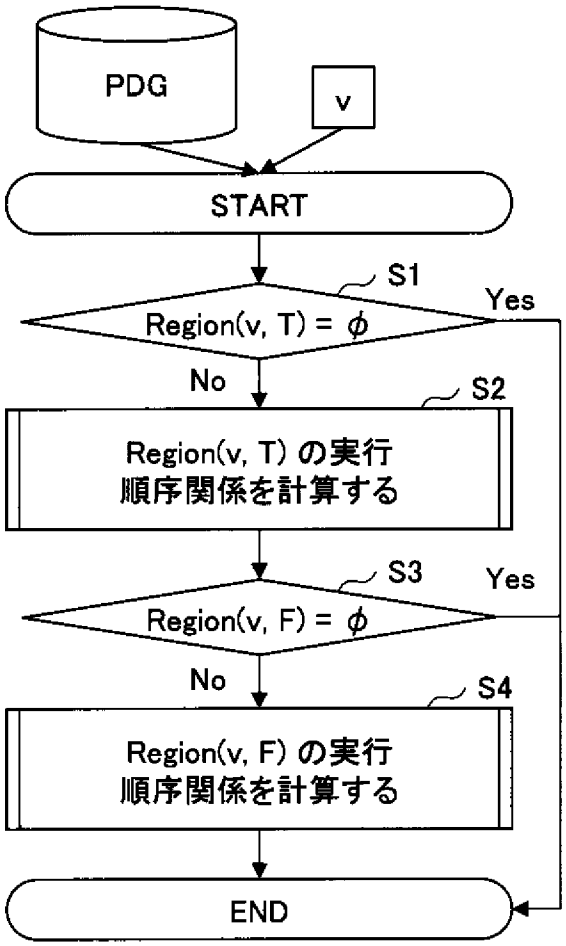
[図5]

頂点間の実行順序関係を決定する方法を示すフローチャート



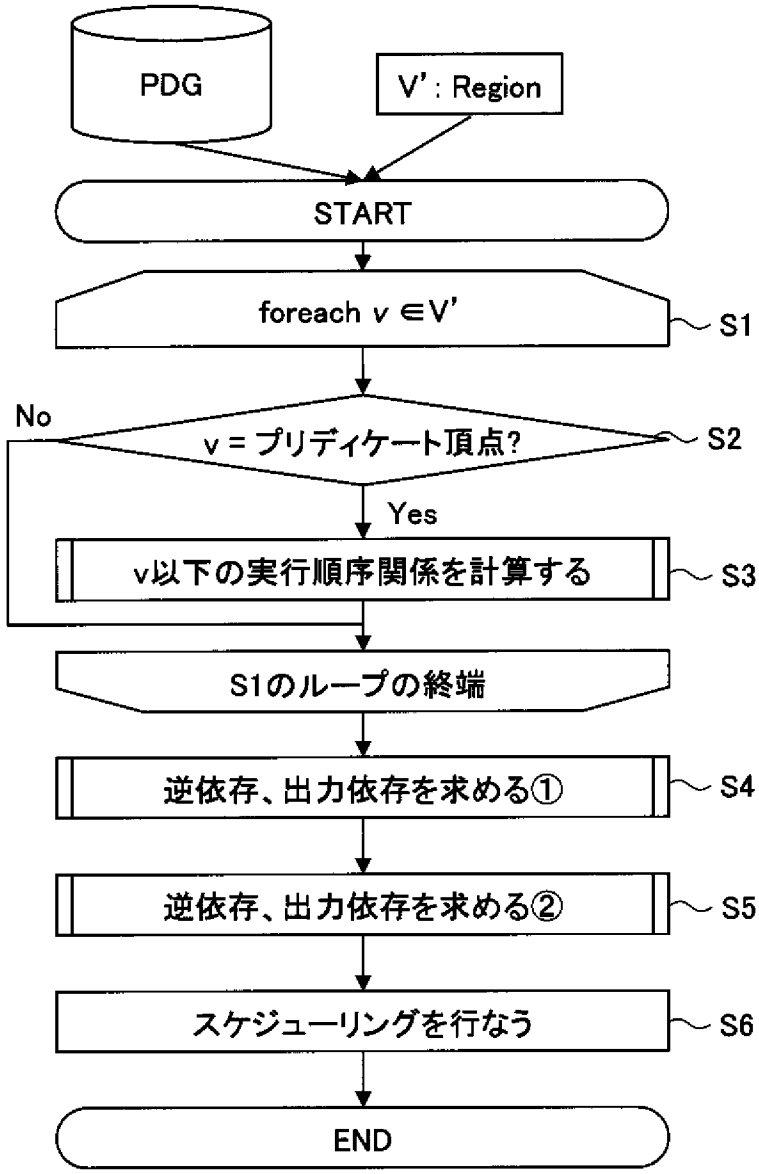
[図6]

頂点v以下の制御の流れを再構成する処理
（図5のステップS2）を示すフローチャート



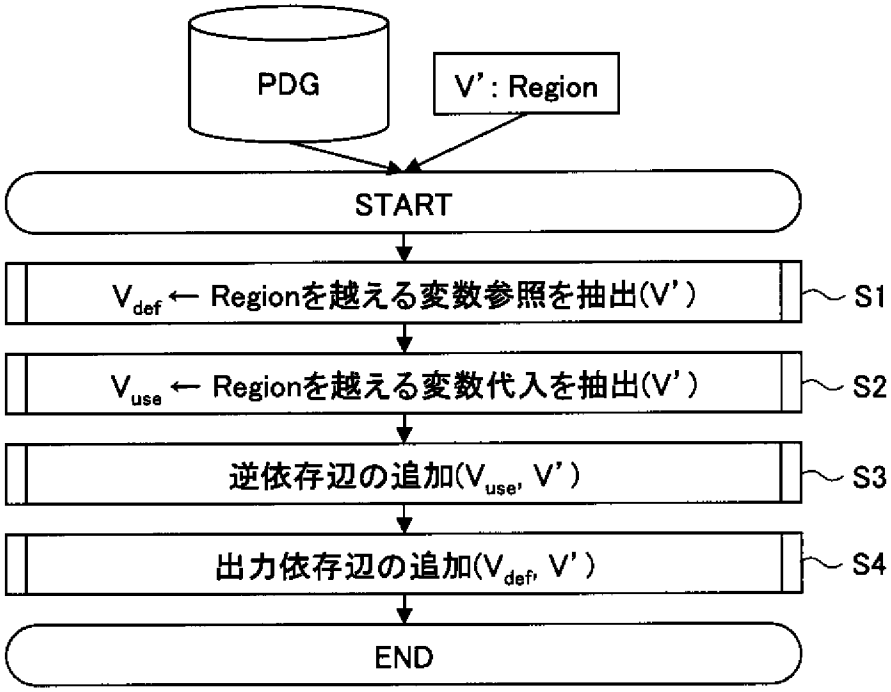
[図7]

Regionの実行順序関係を計算する処理を示すフローチャート



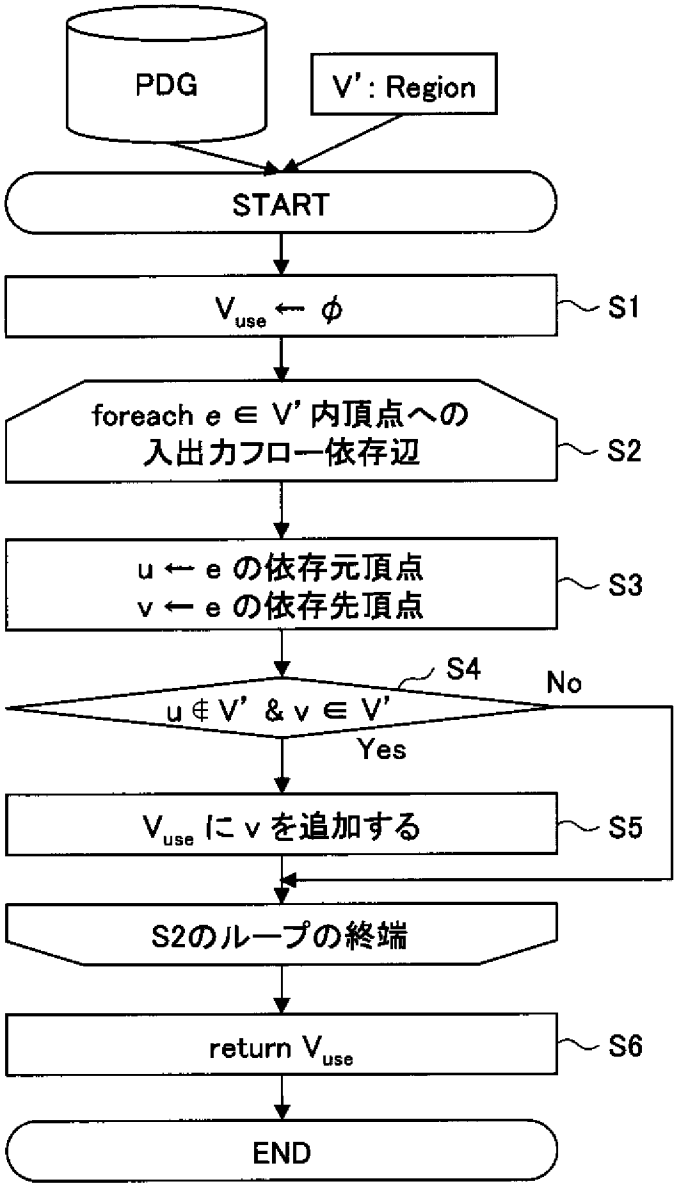
[図8]

逆依存及び出力依存を求める処理(図7のステップS4)を示すフローチャート



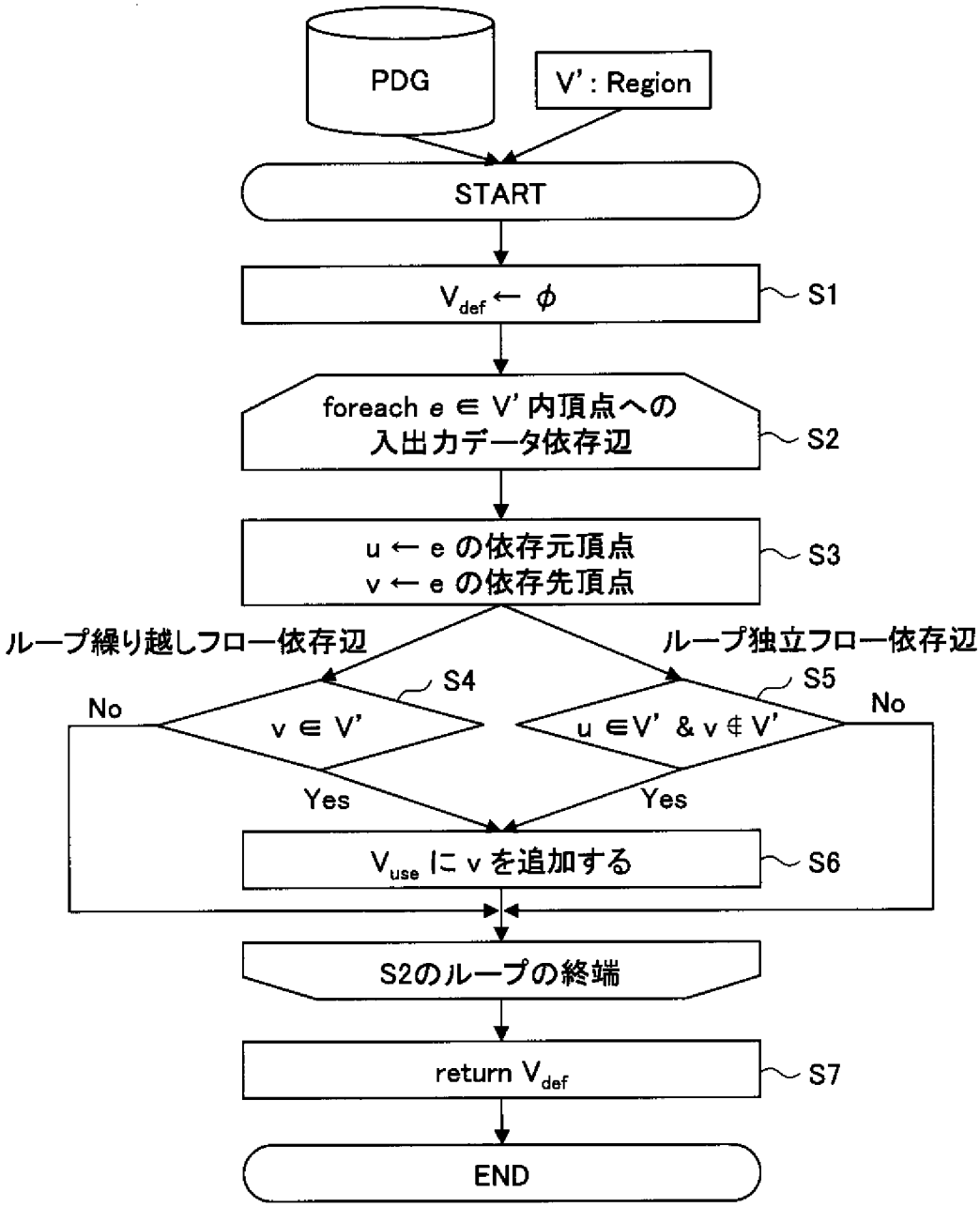
[図9]

着目領域を越える変数参照を抽出する処理を示すフローチャート



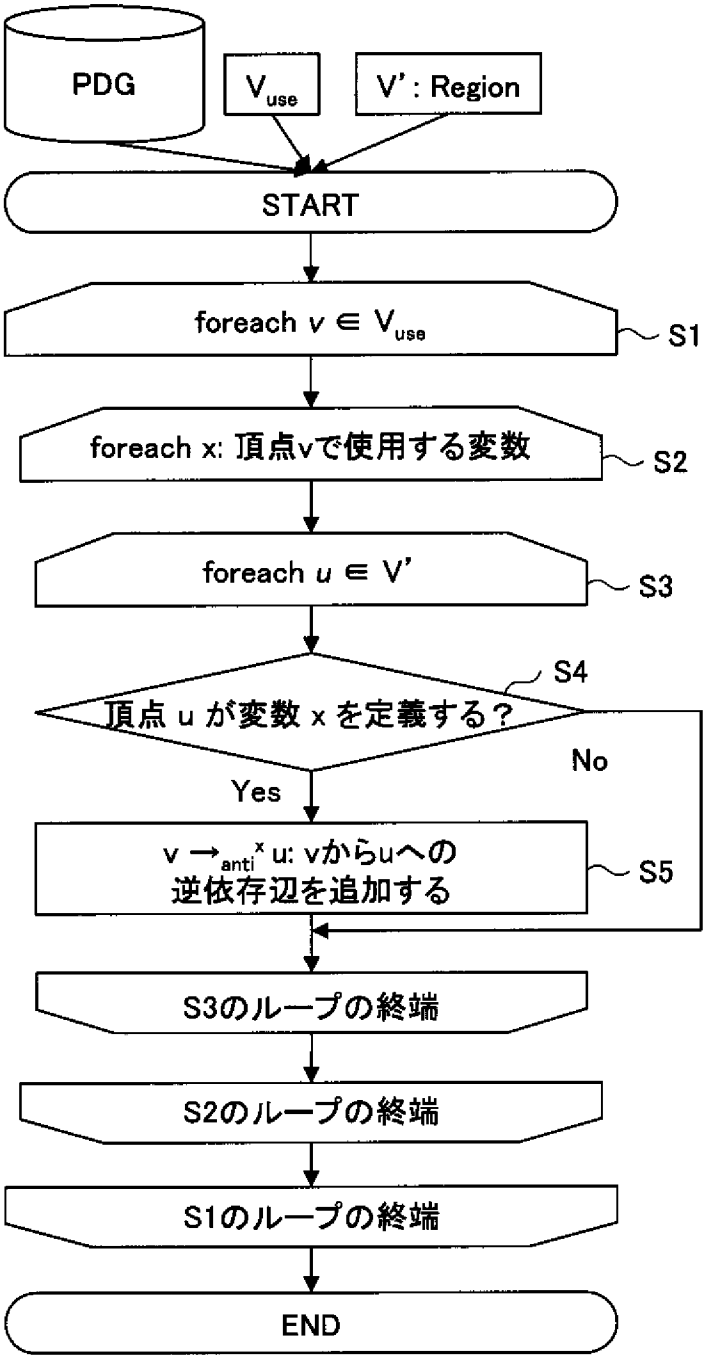
[図10]

着目領域を越える変数代入を抽出する処理を示すフローチャート



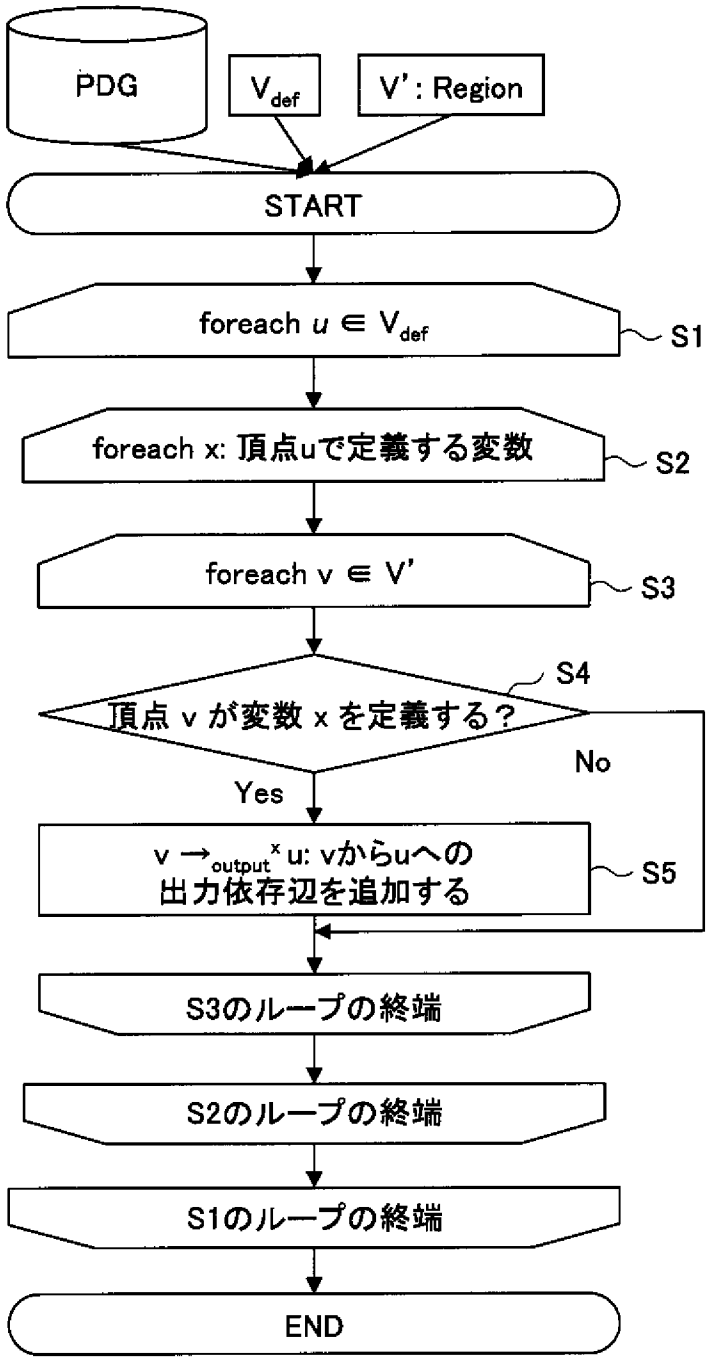
[図11]

逆依存の追加処理を示すフローチャート



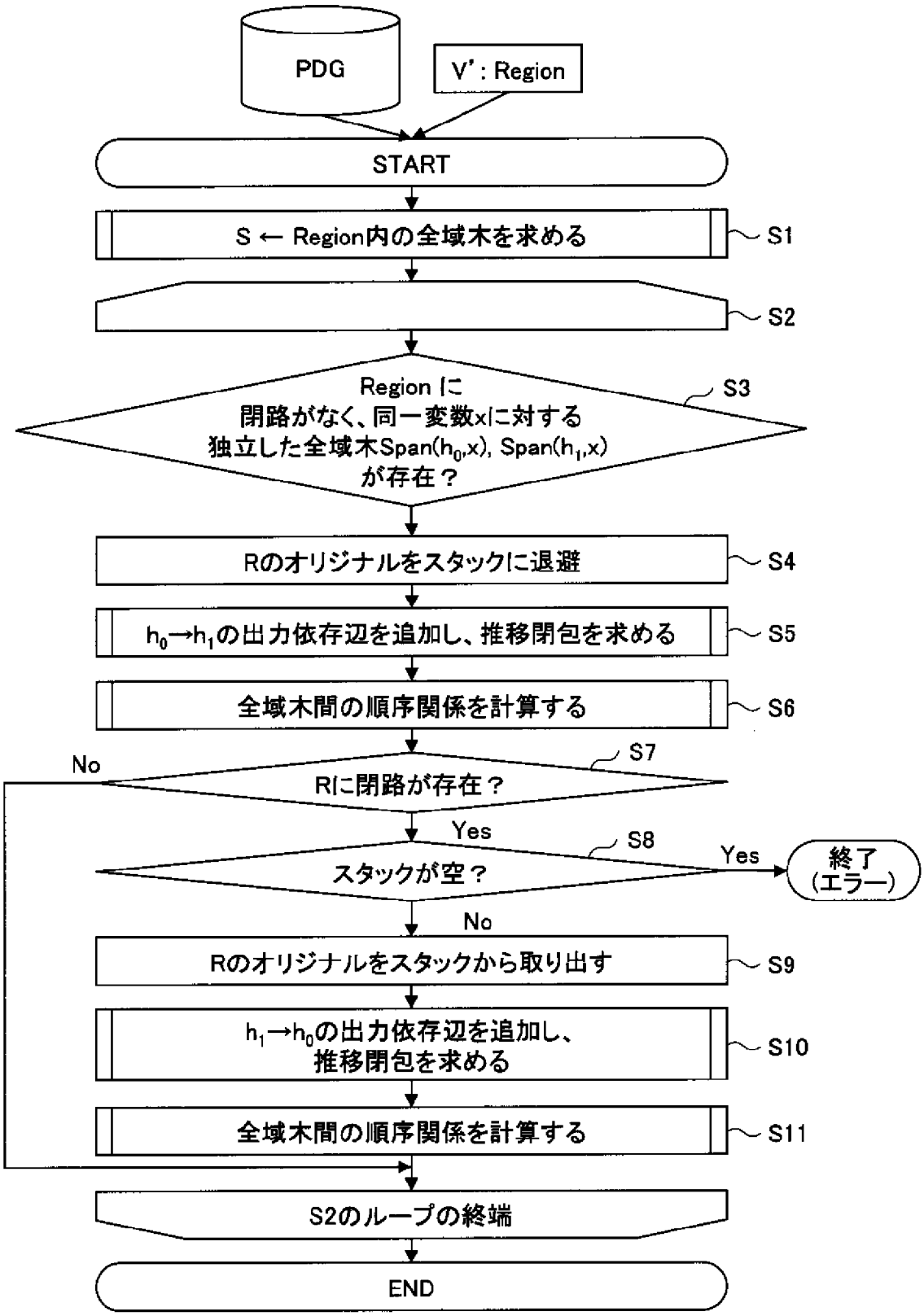
[図12]

出力依存の追加処理を示すフローチャート



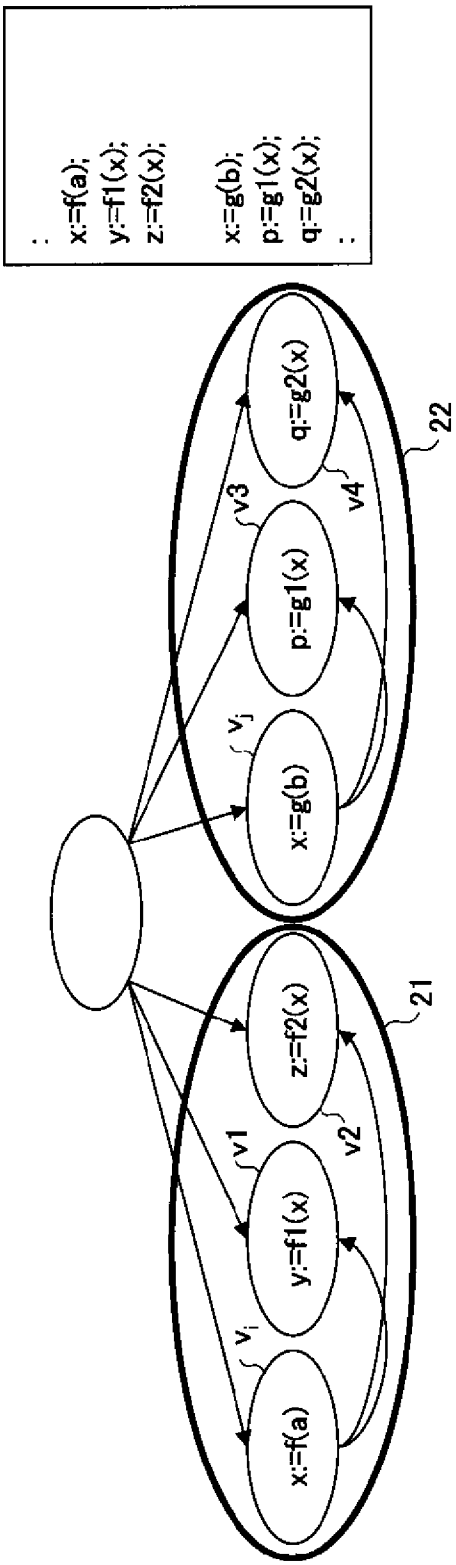
[図13]

逆依存及び出力依存を求める処理(図7のステップS5)を示すフローチャート



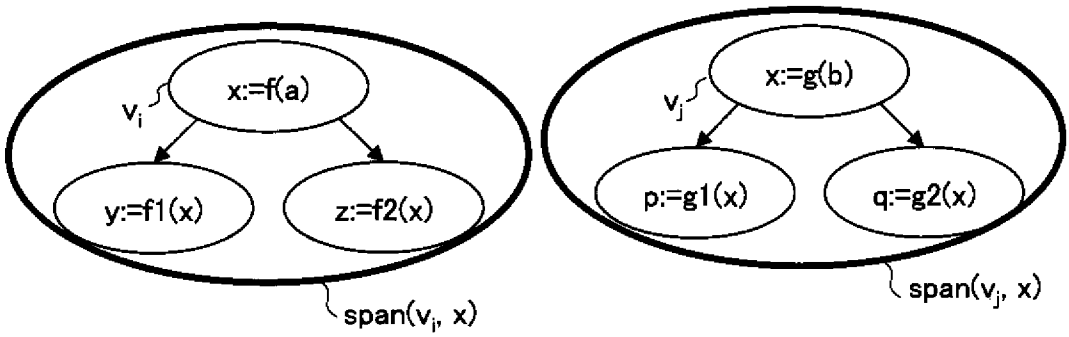
[図14]

全域木を説明するための図



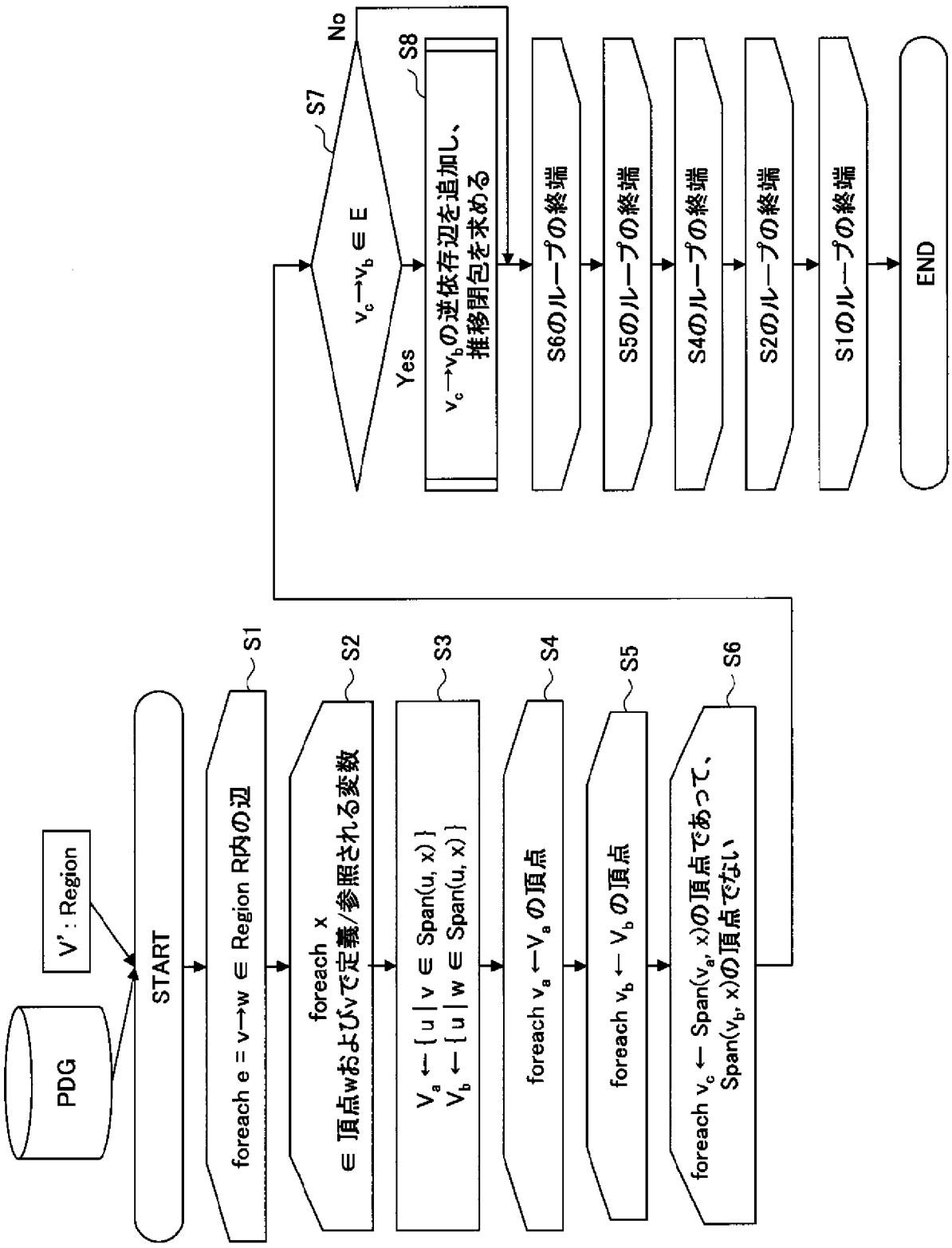
[図15]

全域木を模式的に示す図



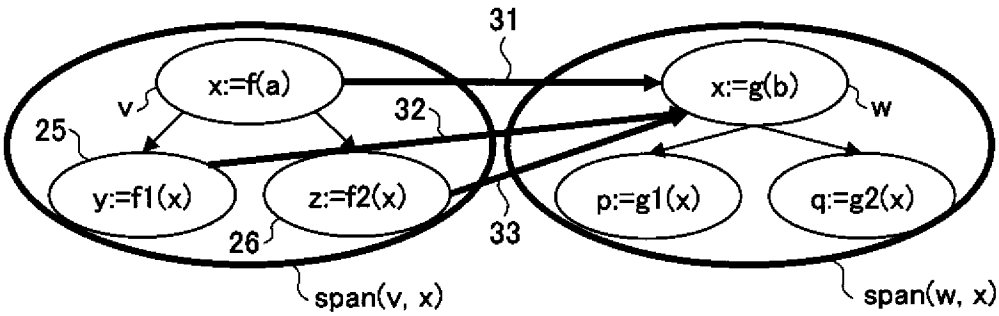
[図16]

全域木間の順序関係を計算する処理を示すフローチャート



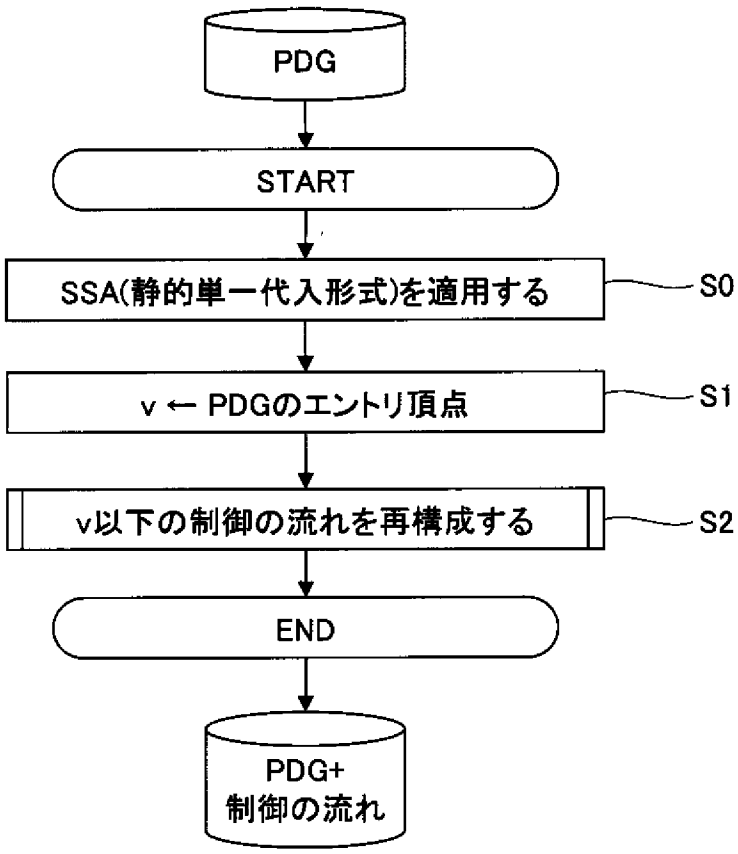
[図17]

図16の処理による逆依存辺の追加について説明する図



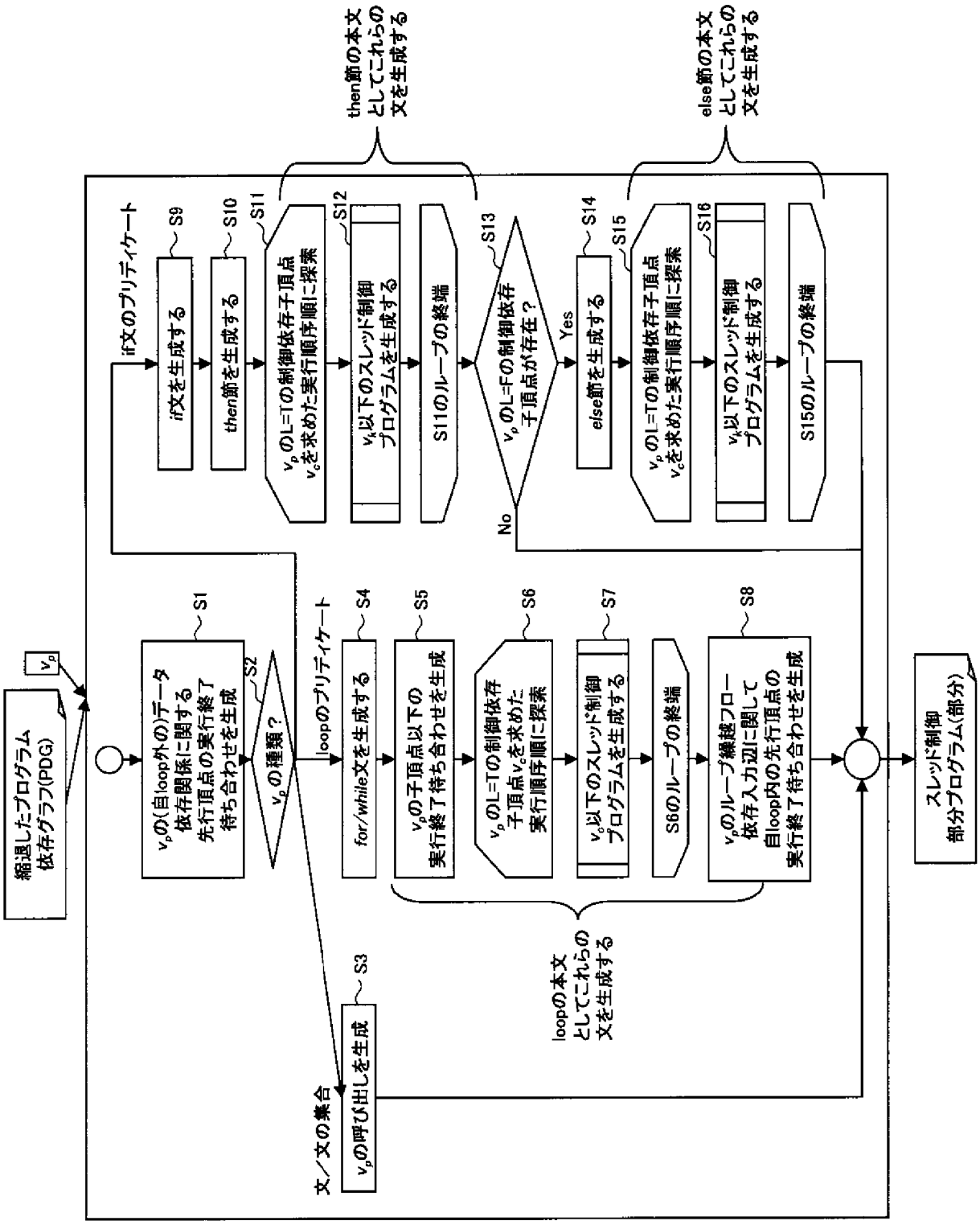
[図18]

頂点間の実行順序関係を決定する方法の変形例を示すフローチャート



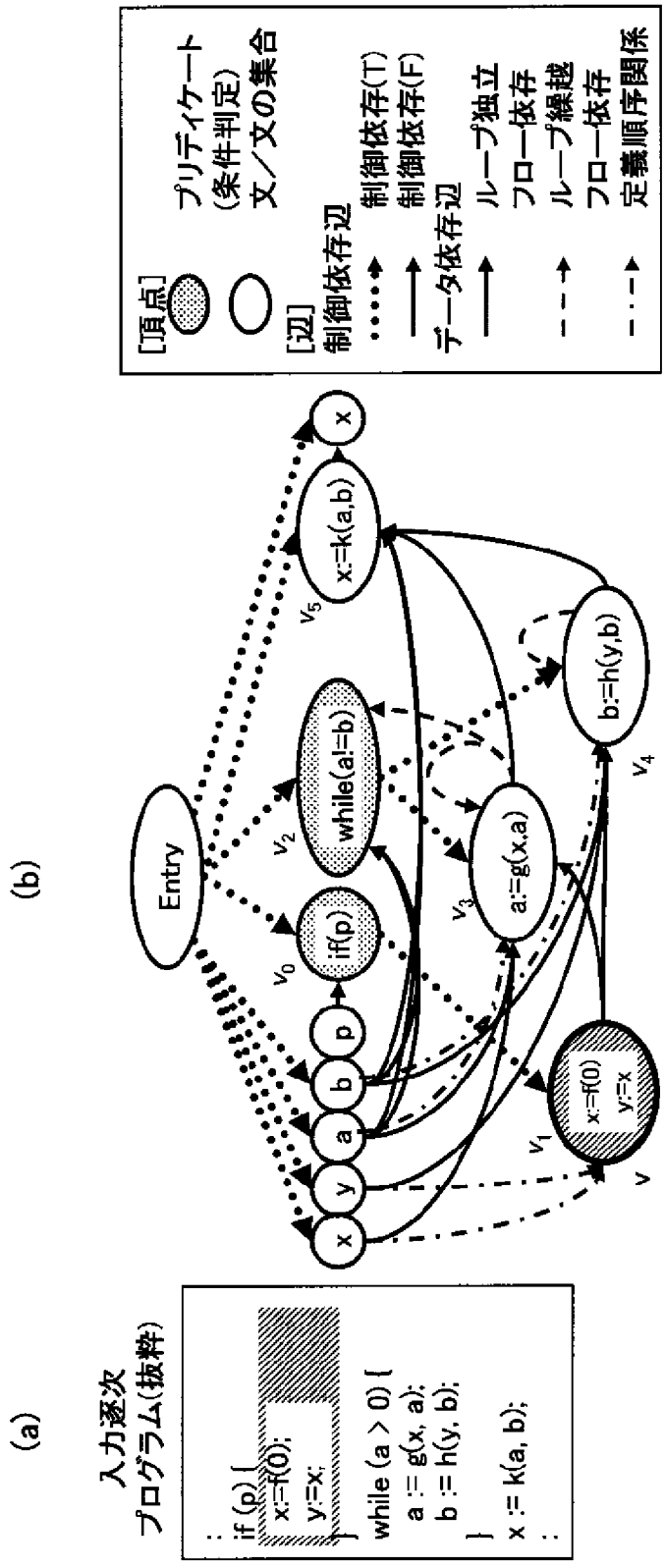
[図19]

頂点vp以下のスレッド制御プログラムを生成する処理を示すフローチャート



[図20]

(a)は入力逐次プログラムの部分を示す図、
(b)は対応する縮退プログラム依存グラフを示す図



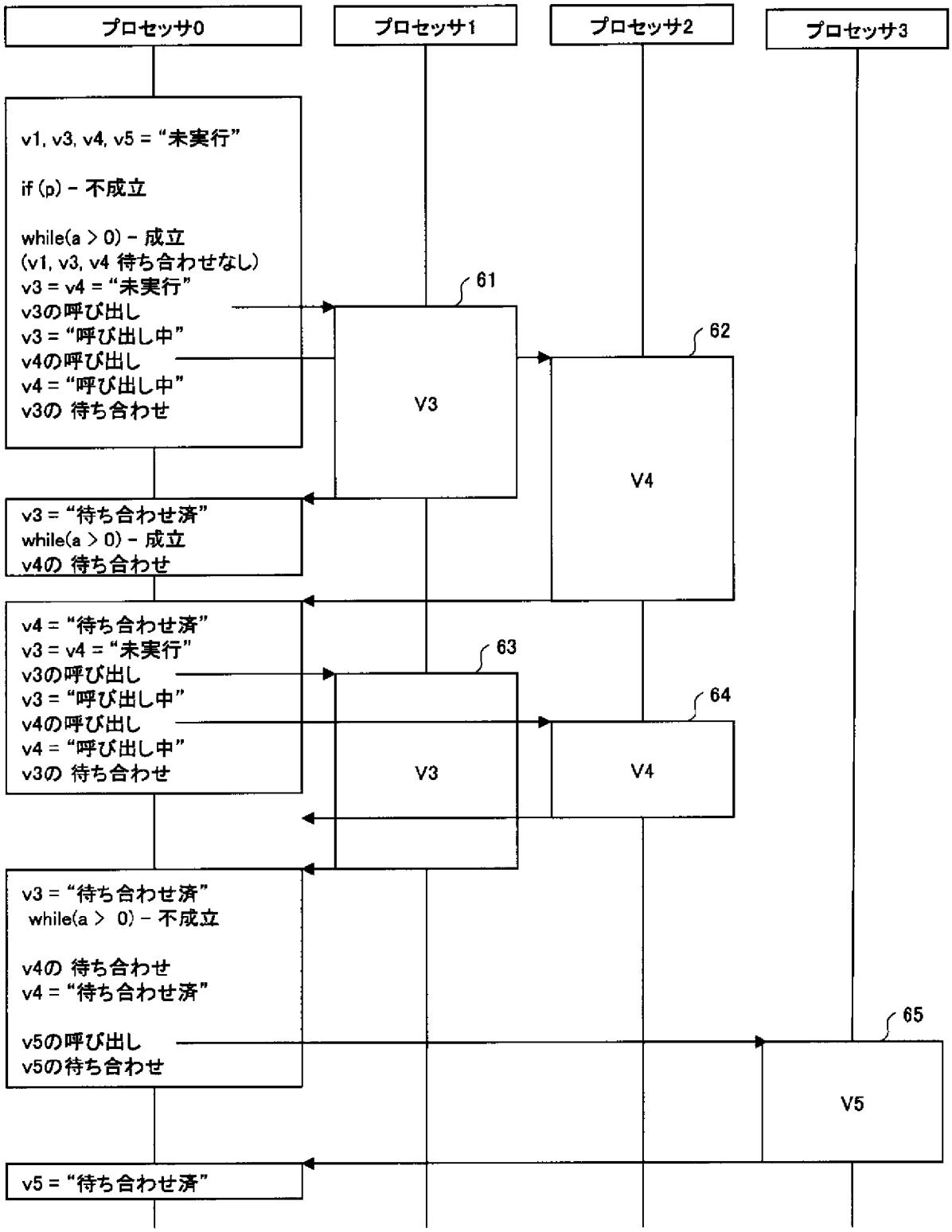
[図21]

図２０の縮退プログラム依存グラフから
第１の実施例に従い生成されるスレッド制御プログラムを示す図

```
// 変数の宣言
var v1, v3, v4, v5 = “未実行” ～ 41
var x, y, a, b, p;
: (途中略)
if(p) {
  thread_v1_start() ～ 42
  v1 = “呼び出し中”～ 43
}
while(a > 0) {
  if (v1 = “呼び出し中”) {
    (x, y) ← thread_v1_wait() ～ 44
    v1 = “待ち合わせ済” ～ 45
  }
  if (v4 = “呼び出し中”) {
    b ← thread_v4_wait() ～ 46
    v4 = “待ち合わせ済”
  }
  v3 = v4 = “未実行”
  thread_v3_start(x, a); ～ 47
  v3 = “呼び出し中”
  thread_v4_start(y, b); ～ 48
  v4  = “呼び出し中”
  if (v3 = “呼び出し中”) {
    a ← thread_v3_wait()
    v3 = “待ち合わせ済”
  }
}
if (v3 = “呼び出し中”) {
  a ← thread_v3_wait() ～ 49
  v3 = “待ち合わせ済”
}
if (v4 = “呼び出し中”) {
  b ← thread_v4_wait() ～ 50
  v4 = “待ち合わせ済”
}
thread_v5_start(a, b); ～ 51
v5  = “呼び出し中”
if (v5 = “呼び出し中”) {
  x ← thread_v5_wait() ～ 52
  v5 = “待ち合わせ済”
}
```

[図22]

以上のスレッド制御プログラムの動作を
スレッド・プログラムの実行とともに示す模式図

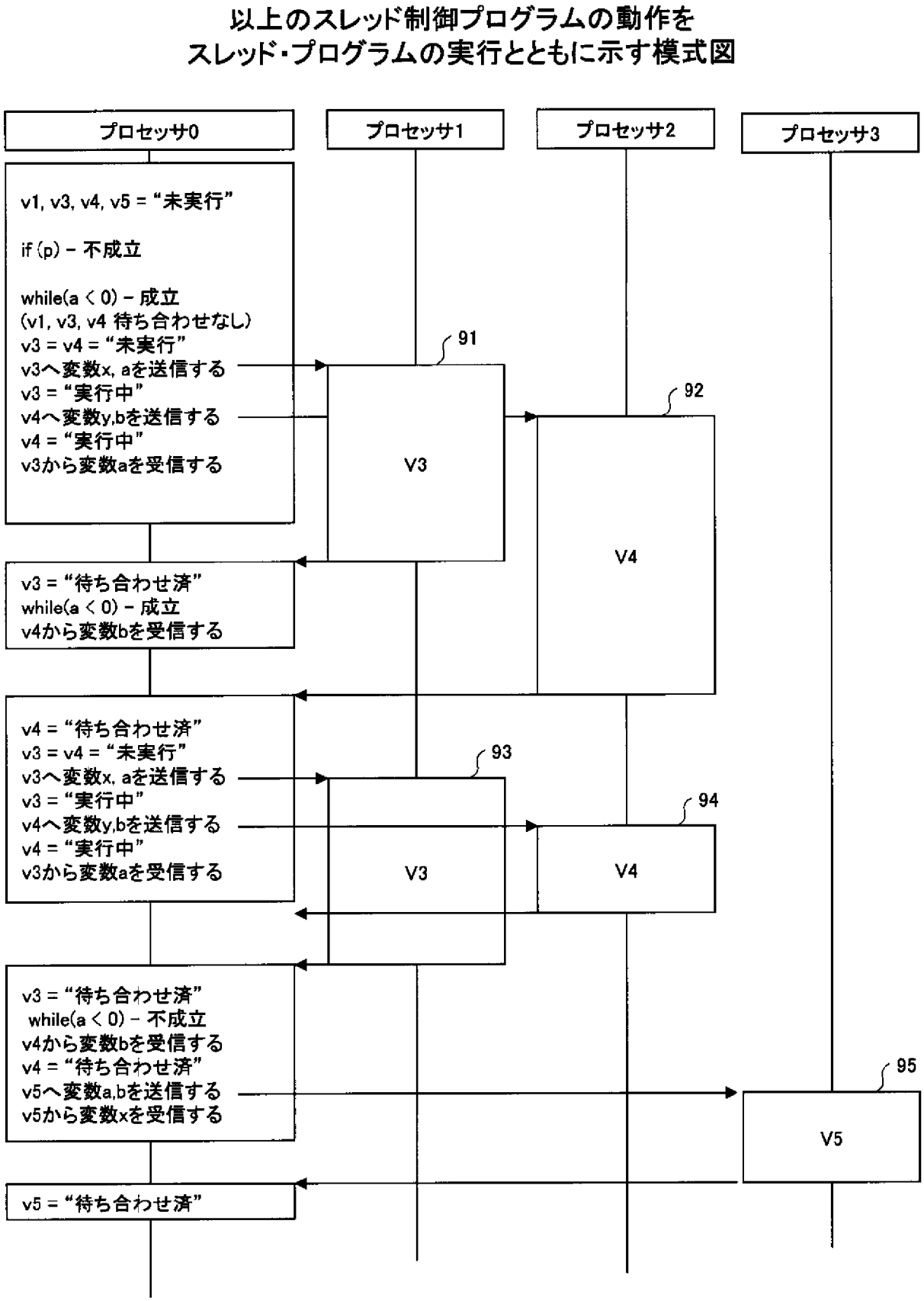


[図23]

図20の縮退プログラム依存グラフから
第2の実施例に従い生成されるスレッド制御プログラムを示す図

```
// 変数の宣言
var v1, v3, v4, v5 = “未実行” ～ 71
var x, y, a, b, p;
: (途中略)
if(p) {
  msg_snd(thread_v1, 1) ～ 72
  v1 = “実行中” ～ 73
}
while(a > 0) {
  if (v1 = “実行中”) {
    x ←msg_rcv(thread_v1) ～ 74
    v1 = “待ち合わせ済” ～ 75
  }
  if (v4 = “実行中”) {
    b←msg_rcv(thread_v4) ～ 76
    v4 = “待ち合わせ済”
  }
  v3 = v4 = “未実行”
  msg_snd(thread_v3, (x, a)) ～ 77
  v3 = “実行中”
  msg_snd(thread_v4, (y, b)) ～ 78
  v4  = “実行中”
  if (v3 = “実行中”) {
    a ←msg_rcv(thread_v3)
    v3 = “待ち合わせ済”
  }
}
if (v3 = “実行中”) {
  msg_rcv(thread_v3) ～ 79
  v3 = “待ち合わせ済”
}
if (v4 = “実行中”) {
  msg_rcv(thread_v4) ～ 80
  v4 = “待ち合わせ済”
}
msg_snd(thread_v5, (a, b)); ～ 81
v5  = “実行中”
if (v5 = “実行中”) {
   x←msv_rcv(thread_v5) ～ 82
   v5 = “待ち合わせ済”
}
```

[図24]

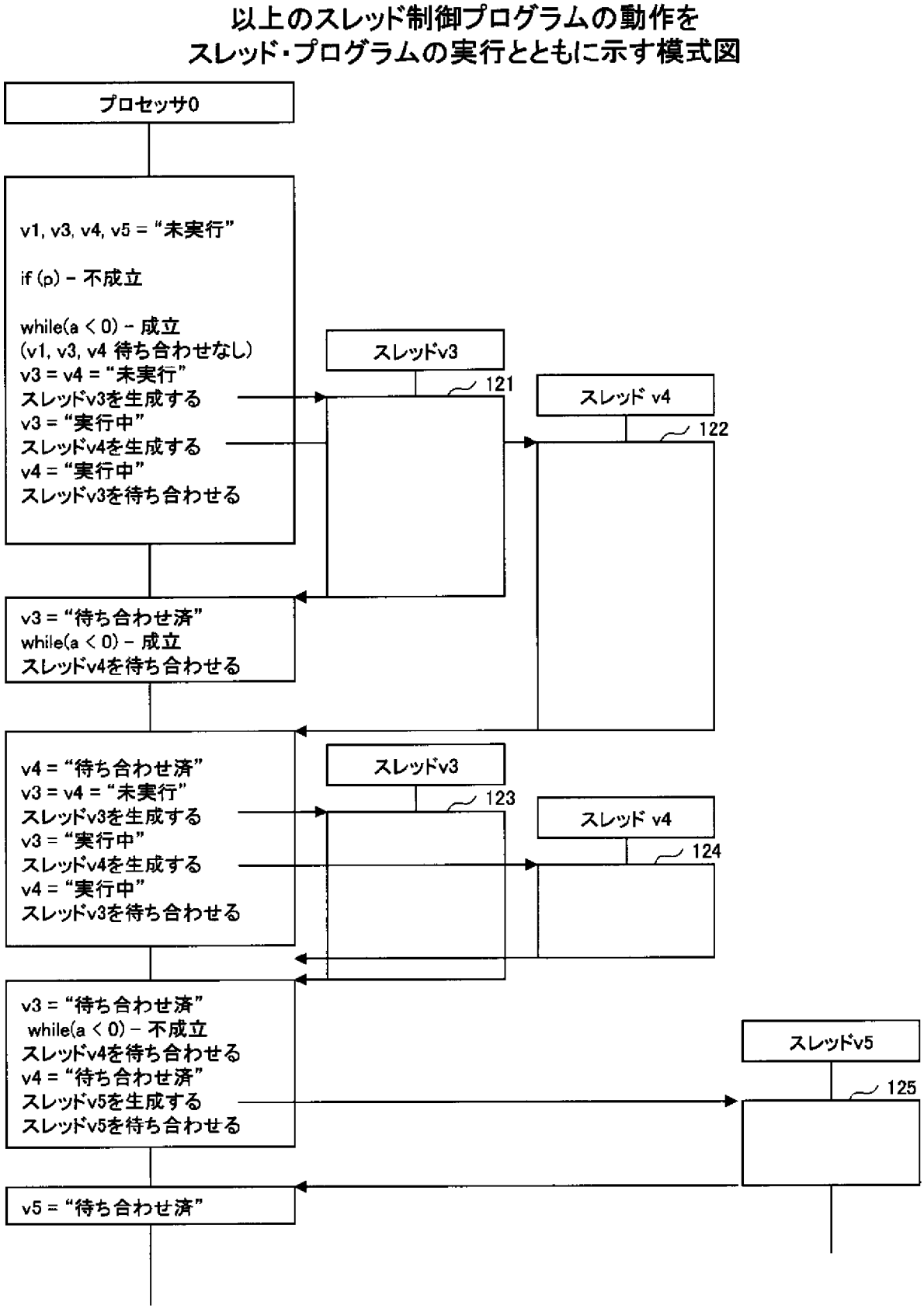

以上のスレッド制御プログラムの動作を
スレッド・プログラムの実行とともに示す模式図
プロセッサ0
プロセッサ1
プロセッサ2
プロセッサ3
v1, v3, v4, v5 = “未実行”
if (p) – 不成立
while(a < 0) – 成立
(v1, v3, v4 待ち合わせなし)
v3 = v4 = “未実行”
v3へ変数x, aを送信する
v3 = “実行中”
v4へ変数y,bを送信する
v4 = “実行中”
v3から変数aを受信する
91
92
V3
V4
v3 = “待ち合わせ済”
while(a < 0) – 成立
v4から変数bを受信する
v4 = “待ち合わせ済”
v3 = v4 = “未実行”
v3へ変数x, aを送信する
v3 = “実行中”
v4へ変数y,bを送信する
v4 = “実行中”
v3から変数aを受信する
93
94
V3
V4
v3 = “待ち合わせ済”
while(a < 0) – 不成立
v4から変数bを受信する
v4 = “待ち合わせ済”
v5へ変数a,bを送信する
v5から変数xを受信する
95
V5
v5 = “待ち合わせ済”

[図25]

図20の縮退プログラム依存グラフから
第3の実施例に従い生成されるスレッド制御プログラムを示す図

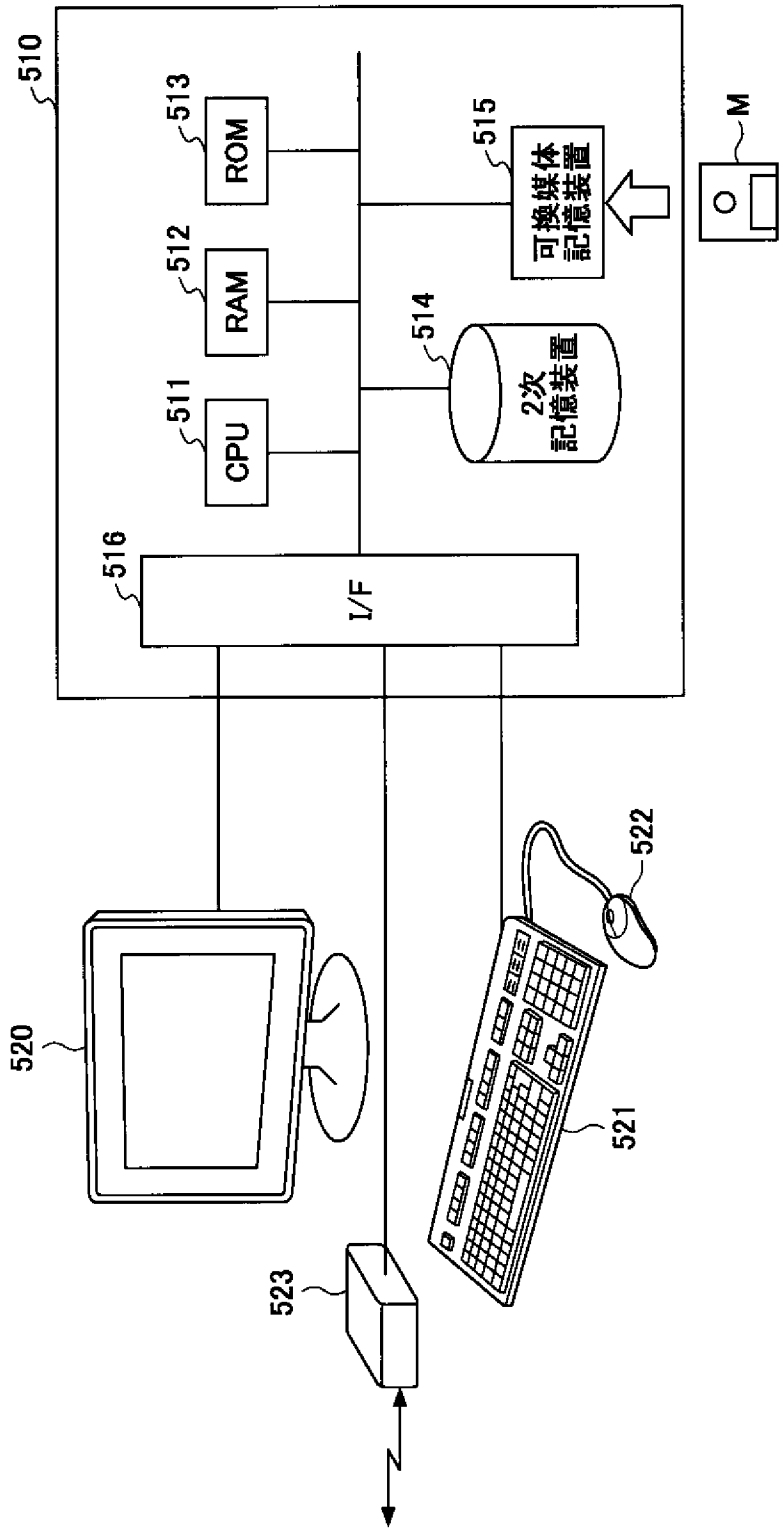
```
// 変数の宣言
var v1, v3, v4, v5 = “未実行” ～ 101
var x, y, a, b, p;
:(途中略)
if(p) {
  スレッドv1を生成 ～ 102
  v1 = “実行中” ～ 103
}
while(a > 0) {
  if (v1 = “実行中”) {
    x ← スレッドv1の待ち合わせ ～ 104
    v1 = “待ち合わせ済” ～ 105
  }
  if (v4 = “実行中”) {
    b ← スレッドv4の待ち合わせ ～ 106
    v4 = “待ち合わせ済”
  }
  v3 = v4 = “未実行”
  スレッドv3を生成(x, a) ～ 107
  v3 = “実行中”
  スレッドv4を生成(y, b) ～ 108
  v4  = “実行中”
  if (v3 = “実行中”) {
    a ← スレッドv3の待ち合わせ
    v3 = “待ち合わせ済”
  }
}
if (v3 = “実行中”) {
  a ← スレッドv3の待ち合わせ ～ 109
  v3 = “待ち合わせ済”
}
if (v4 = “実行中”) {
  b ← スレッドv4の待ち合わせ ～ 110
  v4 = “待ち合わせ済”
}
スレッドv5を生成(a, b) ～ 111
v5 = “実行中”
if (v5 = “実行中”) {
   x←スレッドv5の待ち合わせ ～ 112
   v5 = “待ち合わせ済”
}
```

[図26]


以上のスレッド制御プログラムの動作を
スレッド・プログラムの実行とともに示す模式図
プロセッサ0
v1, v3, v4, v5 = “未実行”
if (p) – 不成立
while(a < 0) – 成立
(v1, v3, v4 待ち合わせなし)
v3 = v4 = “未実行”
スレッドv3を生成する
v3 = “実行中”
スレッドv4を生成する
v4 = “実行中”
スレッドv3を待ち合わせる
スレッドv3
121
スレッド v4
122
v3 = “待ち合わせ済”
while(a < 0) – 成立
スレッドv4を待ち合わせる
v4 = “待ち合わせ済”
v3 = v4 = “未実行”
スレッドv3を生成する
v3 = “実行中”
スレッドv4を生成する
v4 = “実行中”
スレッドv3を待ち合わせる
スレッドv3
123
スレッド v4
124
v3 = “待ち合わせ済”
while(a < 0) – 不成立
スレッドv4を待ち合わせる
v4 = “待ち合わせ済”
スレッドv5を生成する
スレッドv5を待ち合わせる
スレッドv5
125
v5 = “待ち合わせ済”

[図27]

本発明による並列化プログラム生成方法を実行する装置の構成を示す図



INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2006/305016

A. CLASSIFICATION OF SUBJECT MATTER
G06F9/45(2006.01)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F9/45(2006.01)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Jitsuyo Shinan Koho 1922-1996 Jitsuyo Shinan Toroku Koho 1996-2006
Kokai Jitsuyo Shinan Koho 1971-2006 Toroku Jitsuyo Shinan Koho 1994-2006

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	Hironori KASAHARA, "Heiretsu Shori Software", The Transactions of the Institute of Electrical Engineers of Japan C, 20 November, 1993 (20.11.93), Vol.113-C, No.11, pages 919 to 927	1-10
Y	Tsuneo NAKANISHI et al., "CDP^2 Algorithm - Data Bunkatsu Graph Jo deno Togoteki Data Program Bunkatsu Algorithm -, Information Processing Society of Japan Kenkyu Hokoku (95-HPC-57), 25 August, 1995 (25.08.95), Vol.95, No.81, pages 139 to 144	1-10

☒ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
15 May, 2006 (15.05.06)	23 May, 2006 (23.05.06)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2006/305016

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	Satoshi UCHIDA et al., "Izon Graph no Henkei niyoru Heiretsuka Shori no Seigosei", FIT2002 (Forum on Information Technology) Ippan Koen Ronbunshu separate Vol.1, 13 September, 2002 (13.09.02), pages 7 to 8	1-10
Y	Yuichi ITO et al., "Seigyo Izon no Kanwa o Koryo shita Heiretsusei Chushutsu Shuho", Information Processing Society of Japan Kenkyu Hokoku (2001-ARC-144), 27 July, 2001 (27.07.01), Vol.2001, No.76, pages 87 to 92	3,9
Y	Yang Wei et al., "Guard Tsuki PDG o Mochiita Meirei Level Heiretsu Architecture no Tameno Saitekika Compiler no Jitsugen", Information Processing Society of Japan Dai 58 Kai (Heisei 11 Nen Zenki) Zenkoku Taikai Koen Ronbunshu (1), 11 March, 1999 (11.03.99), 1-269 to 1-270	3,9

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ２００６／３０５０１６

Ａ．　発明の属する分野の分類（国際特許分類（ＩＰＣ））

Int.Cl.　*G06F9/45(2006.01)*

Ｂ．　調査を行った分野

調査を行った最小限資料（国際特許分類（ＩＰＣ））

Int.Cl.　*G06F9/45(2006.01)*

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報	１９２２－１９９６年
日本国公開実用新案公報	１９７１－２００６年
日本国実用新案登録公報	１９９６－２００６年
日本国登録実用新案公報	１９９４－２００６年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

Ｃ．　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
Y	笠原博徳，並列処理ソフトウェア，電気学会論文誌Ｃ，１９９３．１１．２０，Ｖｏｌ．１１３－Ｃ，Ｎｏ．１１，ｐ９１９－９２７	1-10
Y	中西恒夫　外２名，ＣＤＰ^{2}アルゴリズムーデータ分割グラフ上での統合的データ・プログラム分割アルゴリズムー，情報処理学会研究報告（９５－ＨＰＣ－５７），１９９５．０８．２５，Ｖｏｌ．９５，Ｎｏ．８１，ｐ１３９－１４４	1-10

☑　Ｃ欄の続きにも文献が列挙されている。

☐　パテントファミリーに関する別紙を参照。

＊　引用文献のカテゴリー

「Ａ」特に関連のある文献ではなく、一般的技術水準を示すもの

「Ｅ」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの

「Ｌ」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）

「Ｏ」口頭による開示、使用、展示等に言及する文献

「Ｐ」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献

「Ｔ」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの

「Ｘ」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの

「Ｙ」特に関連のある文献であって、当該文献と他の１以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの

「＆」同一パテントファミリー文献

国際調査を完了した日	国際調査報告の発送日
１５．０５．２００６	２３．０５．２００６

国際調査機関の名称及びあて先

日本国特許庁（ＩＳＡ／ＪＰ）

郵便番号１００－８９１５

東京都千代田区霞が関三丁目４番３号

特許庁審査官（権限のある職員）　５Ｂ　９３６６

漆原　孝治

電話番号　０３－３５８１－１１０１　内線　３５４５

Ｃ（続き）．　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
Ｙ	内田智士　外２名，依存グラフの変形による並列化処理の整合性，ＦＩＴ２００２（情報科学技術フォーラム）一般講演論文集　第１分冊，２００２．０９．１３，ｐ７－８	1-10
Ｙ	伊藤佑一　外３名，制御依存の緩和を考慮した並列性抽出手法，情報処理学会研究報告（２００１－ＡＲＣ－１４４），２００１．０７．２７，Ｖｏｌ．２００１，Ｎｏ．７６，ｐ８７－９２	3,9
Ｙ	楊巍　外３名，ガード付きＰＤＧを用いた命令レベル並列アーキテクチャのための最適化コンパイラの実現，情報処理学会第５８回（平成１１年前記）全国大会講演論文集（１），１９９９．０３．１１，１－２６９～１－２７０	3,9