

(19) 世界知的所有権機関
国際事務局(43) 国際公開日
2008 年 10 月 9 日 (09.10.2008)

PCT

(10) 国際公開番号
WO 2008/120367 A1(51) 国際特許分類:
G06F 9/45 (2006.01)

(21) 国際出願番号: PCT/JP2007/056916

(22) 国際出願日: 2007 年 3 月 29 日 (29.03.2007)

(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).

(74) 代理人: 伊東 忠彦 (ITO, 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 3 2 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, BH, 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, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

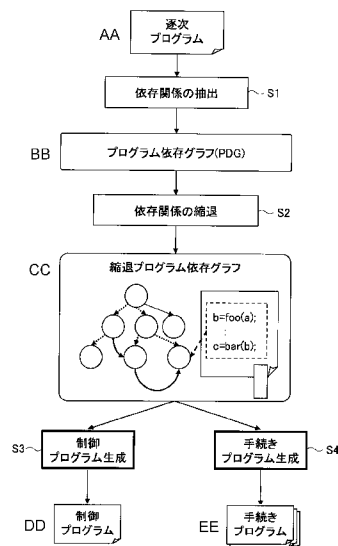
(84) 指定国 (表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), ヨーロッパ (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE,

[続葉有]

(54) Title: PARALLELIZATION PROGRAM GENERATING METHOD, PARALLELIZATION PROGRAM GENERATOR, AND PARALLELIZATION PROGRAM GENERATING PROGRAM

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

[図1]



AA... SEQUENTIAL PROGRAM
 S1... EXTRACT DEPENDENCE RELATION
 BB... PROGRAM DEPENDENCE GRAPH (PDG)
 S2... DEGENERATE DEPENDENCE RELATION
 CC... DEGENERATE PROGRAM DEPENDENCE GRAPH
 S3... GENERATE CONTROL PROGRAM
 S4... GENERATE PROCEDURE PROGRAM
 DD... CONTROL PROGRAM
 EE... PROCEDURE PROGRAM

(57) Abstract: A parallelization program generating method includes the steps of calculating the order in which the vertices of a degenerate program dependence graph are executed, integrating vertex matrices to be executed in order without including any of branch or merger out of the vertices to which the execution order is given into basic blocks, generating the procedure equivalent to each of the vertices of the degenerate program dependence graph, and placing an instruction to execute the subsequent procedure after an instruction to meet the preceding procedure about a procedure having a dependence relation across the basic blocks. The parallelization program generating method further includes the steps of generating an instruction to register the dependence relation of the subsequent procedure to the preceding procedure about a procedure having a dependence relation within the same basic block and generating an instruction to instruct direct data transfer from the procedure to the procedure across both of data transfer within the same block and data transfer across the basic blocks to generate a procedure control program for controlling the execution of the procedures.

(57) 要約: 並列化プログラム生成方法は、縮退プログラム依存グラフの頂点の実行順序を計算し、実行順序を与えられた複数の頂点のうちで分岐及び合流の何れも含まずに順番に実行される頂点列を基本ブロックとして纏め、縮退プログラム依存グラフの頂点の各々に相当する手続きを生成し、基本ブロック間をまたいでの依存関係がある手続きについては先行手続きを待ち合わせる命令の後に後続手続きを実行する命令を配置し、同一の基本ブロック内部で依存関係がある手続きについては先行手続きに対する後続手続きの依存関係を登録する命令を生成し、同一の基本ブロック内部でのデータ転送及び基本ブロック間をまたいでのデータ転送の両方について手続きから手続きへの直接のデータ転送を指示する命令を生成するようにして、手続きの実行を制御する手続き制御プログラムを生成する各段階を含む。

WO 2008/120367 A1



IS, IT, LT, LU, LV, MC, MT, 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

明 細 書

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

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十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。

[0008] 従って、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(並列化プログラム)を生成する方法を提供することが必要になる。但し、このようにして生成する並列化プログラムにおいては、以下に説明するように、スレッド間の依存関係に基づく待ち時間の発生という問題について考慮する必要がある。

[0009] 並列化プログラムの各スレッドの実行を制御する方式としては、例えば、手続を非同期の遠隔呼び出しとして呼び出すことにより並列にスレッドを実行する方式、手続に実行開始するメッセージを送信することにより並列にスレッドを実行する方式、スレッド間で共有メモリを利用して入出力変数の受け渡しを行なうことにより並列にスレッドを実行する方式等が考えられる。しかしこれらの方式では、第1の手続(スレッド)の実行結果を利用する第2の手続がある場合、第1の手続の終了を待つ命令とそれに続く第2の手続を実行する命令とを、他の手続の実行に要する時間などを見積もって、プログラム中の適当な場所に配置しておくことになる。この場合、第1の手続が予想以上に早く終了した場合などに、第2の手続を実行するまでに、不必要な待ち時間が発生してしまう。

[0010] 図1は、無駄な待ち時間の発生について説明するための図である。図1において、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1(各スレッドに対応する手続の実行及び終了待ちを制御するプログラム)を実行する。図1の例では、まずプロセッサ0から、プロセッサ1乃至プロセッサ3に対して手続A乃至Cの実行を順番に要求する(start A()~start C())。その後プロセッサ0は、手続Aの終了を待って(wait A())、手続Aの実行結果を利用する手続Dの実行を要求する(start D())。その後、手続Bの終了を待って(wait B())、手続Bの実行結果を利用する手続Eの実行を要求する(start E())。更にその後、手続Cの終了を待って(wait C())、手続Cの実行結果を利用する手続Fの実行を要求する(start

F())。

- [0011] この例では、手続Cが終了してから手続Fの実行を要求するまでに待ち時間が発生している。これは、スレッド制御プログラム中において、手続Bの終了待ち合わせ (wait B())と手続Eの実行要求 (start E())が、手続Cの終了待ち合わせ (wait C())と手続Fの実行要求 (start F())よりも前に配置されているからである。このような命令配置順のために、手続Bが終了しないと、手続Cの終了待ち合わせ及び手続Fの実行要求が実行されないことになる。
- [0012] このような命令配置は、手続Bが手続Cよりも早く実行が終了するであろうという見積もりに基づくものである。手続Cの方が手続Bよりも早く終了することが分かっていたならば、手続Cの終了待ち合わせ及び手続Fの実行要求を、手続Bの終了待ち合わせ及び手続Eの実行要求よりも前に配置することが考えられる。しかし実際には、手続の実行にかかる時間は処理データの内容等にも依存するので、終了時間を正確に見積もることは不可能である。従って、単純な遠隔手続呼び出し、共有メモリによるスレッド、メッセージ送信等の上記方式では、図1に示すような待ち時間を無くすことはできない。
- [0013] 本願の出願人は、依存関係待ち合わせ付き非同期遠隔手続呼び出し方式として、並列化プログラムの各スレッドの実行を制御する際に、各手続毎に他の手続に対する依存関係を実行条件として指定し、各手続をプロセッサ毎の実行キューに投入し、実行条件が満たされた手続を実行していくという方式を既に提案している。このような方式を、依存関係待ち合わせ付き非同期遠隔手続呼び出し方式と呼ぶ。
- [0014] 図2は、依存関係待ち合わせ付き非同期遠隔手続呼び出し方式による手続実行の制御について説明するための図である。図2において、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2(各スレッドに対応する手続の実行及び依存関係を制御するプログラム)を実行する。この際プロセッサ0は、手続呼び出しプログラム3を実行することにより、スレッド制御プログラム2に規定される各手続を各プロセッサ毎のキューを用いて管理する。
- [0015] 図2の例では、まず制御プログラム2の命令start A()に従って、プロセッサ1の実行キュー4に手続Aが投入される。また制御プログラム2の命令start B()に従って、プロ

セッサ2の実行キュー5に手続Bが投入される。更に制御プログラム2の命令start C()に従って、プロセッサ3の実行キュー6に手続Cが投入される。

[0016] 同様に、制御プログラム2の命令start D()、start E()、及びstart F()に従って、実行キュー4乃至6にそれぞれ手続D、E、及びFが投入される。またスレッド制御プログラム2中のdep(x, y, ...)は依存関係を指定する命令であり、手続Xの依存先が手続Y、...であることを示す。即ち、手続Xを実行するためには、手続Y、...の実行が終了している必要があることを示す。制御プログラム2の命令dep(D, A)に従って、プロセッサ1の実行キュー4中の手続Dに対して、依存先の手続がAであることが登録される。また制御プログラム2の命令dep(E, A, B)に従って、プロセッサ2の実行キュー5中の手続Eに対して、依存先の手続がA及びBであることが登録される。更に、制御プログラム2の命令dep(F, A, C)に従って、プロセッサ3の実行キュー6中の手続Fに対して、依存先の手続がA及びCであることが登録される。

[0017] このようにして各プロセッサ毎に設けた実行キューに投入されている手続を、キューの順番に従って対応するプロセッサで実行する。この際、依存先が登録されていない手続(図2においてNULLで示されている手続)については無条件に実行し、依存先が登録されている手続については、依存先の手続の終了を検出してから実行する。このようにプロセッサ毎にキューを設け、実行条件が満たされたキュー内の手続(実行可能手続)から順番に実行していくことで、図1に示したような待ち時間を無くすることができる。

[0018] 以上説明したように、上記の依存関係待ち合わせ付き非同期遠隔手続呼び出し方式を用いれば、並列化プログラムの実行時における不要な待ち合わせ時間の発生を防ぐことができる。従って、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並列化プログラムを生成する際には、上記の依存関係待ち合わせ付き非同期遠隔手続呼び出し方式に適用可能な並列化プログラムを生成することが望ましい。

[0019] 本願の出願人は、依存関係待ち合わせ付き非同期遠隔手続呼び出し方式に適用可能な並列化プログラム生成方法を既に提案している。この並列化プログラム生成方法では、まずプログラムの実行順序関係を計算し、分岐(IF、GOTO、LOOP等)

や合流を含まない順番に実行される頂点の列である基本ブロックを求める。そして、同一の基本ブロック内部で依存関係がある手続きの実行については、依存関係待ち合わせ付き非同期遠隔手続呼び出しにより手続きを実行する。また、異なる基本ブロックをまたいでの手続き間の依存関係については、先行手続きの終了待ち合わせを行ってから、後続手続きを実行するようにしている。このような構成とすることで、複雑な制御の依存関係が存在する基本ブロック間については、手続きの実行を待ち合わせにより実現することで制御プログラムの生成を容易なものとし、実行順が固定である同一基本ブロック内については、依存関係待ち合わせ付き非同期遠隔手続呼び出しにより無駄な待ち合わせ時間をなくすることができる。

[0020] 上記並列化プログラム生成方法では、異なる基本ブロックをまたいだプロセッサ間でのデータ転送については、常に制御プロセッサ(図2のプロセッサ0)又は制御プロセッサの制御下にあるデータ転送ユニットを介してのデータ転送となっていた。即ち、手続きを実行する第1のプロセッサから制御プロセッサ(又はデータ転送ユニット)にデータをまず転送し、その後、制御プロセッサ(又はデータ転送ユニット)から手続きを実行する第2のプロセッサにこのデータを転送していた。これは、元の逐次プログラムの条件判定の結果に応じてデータ転送の対象が異なったり、手続きの実行順序に依存関係が存在したりするので、制御プロセッサにより一元的に動作を管理することが、正しいデータ転送実現のための比較的容易な解決方法だからである。しかしこのように常に制御プロセッサがデータ転送に介在するのでは、プログラム処理が非効率的になり処理の実行に余計な時間がかかる。従って、このような基本ブロックをまたいでのデータ転送についても、制御プロセッサを介さずに直接に手続き実行プロセッサ間で行うようにすることが、並列化プログラムの処理が効率的になり好ましい。

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:David W. Wall. Limits of Instruction-Level Parallelism. Proceedings of the fourth international conference on Architectural support for programming languages pp. 176-188 May. 1991.

非特許文献2:S. Horwitz, J. Prins, and T. Reps, "Integrating non-interfering versio

ns of programs," ACM Transactions on Programming Languages and Systems, vol. 11, no. 3, pp. 345-387, 1989.

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

非特許文献4: 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.

非特許文献5: 中田育男著: "コンパイラの構成と最適化", 朝倉書店, 1999

発明の開示

発明が解決しようとする課題

- [0021] 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、マルチプロセッサ上で効果的に動作する非投機的かつ依存関係待ち合わせに基づく並列化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

- [0022] 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、文と文と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縮退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実行順序を算出し、該実行順序を与えられた複数の頂点のうちで分岐及び合流の何れも含まず順番に実行される頂点列を基本ブロックとして纏め、該縮退プログラム依存グラフの該頂点の各々に相当する手続きを生成し、該基本ブロック間をまたいだ依存関係がある手続きについては先行手続きの出力データ転送を待ち合わせる命令の後に後続手続きを実行する命令を配置し、同一の基本ブロック内部で依存関係がある手続きについては先行手続きの出力データ転送に対する後続手続きの依存関係を登録する命令を生成し、および同一の基本ブロック内部でのデータ転送及び基本ブロック間をまたいでデータ転送それぞれについて

ては手続きから手続きへの直接のデータ転送を指示する命令および該データ転送の先行手続きに対する依存関係を登録する命令を生成して、該手続きの実行を制御する手続き制御プログラムを生成することを特徴とする。

[0023] 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とを含み、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縮退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該頂点の実行順序を算出し、該実行順序を与えられた該複数の頂点のうちで分岐及び合流の何れも含まずに順番に実行される頂点列を基本ブロックとして纏め、該縮退プログラム依存グラフの頂点の各々に相当する手続きを生成し、該基本ブロック間をまたいだ依存関係がある手続きについては先行手続きの出力データ転送を待ち合わせる命令の後に後続手続きを実行する命令を配置し、同一の基本ブロック内部で依存関係がある手続きについては先行手続きの出力データ転送に対する後続手続きの依存関係を登録する命令を生成し、および同一の基本ブロック内部でのデータ転送及び基本ブロック間をまたいでデータ転送の両方について手続きから手続きへの直接のデータ転送を指示する命令および該データ転送の先行手続きに対する依存関係を登録する命令を生成して、該手続きの実行を制御する手続き制御プログラムを生成することを特徴とする。

[0024] 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、文と文と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縮退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実行順序を算出し、該実行順序を与えられた複数の頂点のうちで分岐及び合流の何れも含まずに順番に実行される頂点列を基本ブロ

ックとして纏め、該縮退プログラム依存グラフの該頂点の各々に相当する手続きを生成し、該基本ブロック間をまたいだ依存関係がある手続きについては先行手続きの出力データ転送を待ち合わせる命令の後に後続手続きを実行する命令を配置し、同一の基本ブロック内部で依存関係がある手続きについては先行手続きの出力データ転送に対する後続手続きの依存関係を登録する命令を生成し、および同一の基本ブロック内部でのデータ転送及び基本ブロック間をまたいでデータ転送それぞれについては手続きから手続きへの直接のデータ転送を指示する命令および該データ転送の先行手続きに対する依存関係を登録する命令を生成して、該手続きの実行を制御する手続き制御プログラムを生成することを計算機に実行させるコードを含むことを特徴とする。

発明の効果

[0025] 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。

[0026] また、異なる基本ブロックをまたいでの手続き間の依存関係については、先行手続きの終了待ち合わせを行ってから、後続手続きを実行するようにする。また同一の基本ブロック内部で依存関係がある手続きの実行については、依存関係待ち合わせ付き非同期遠隔手続呼び出しにより手続きを実行する。即ち、基本ブロック間をまたいでの依存関係がある手続きについては先行手続きの出力データ転送を待ち合わせる命令の後に後続手続きを実行する命令を配置して、この命令の配置順により依存関係を非明示的に規定して、依存関係を満たすように手続き制御する。また同一の基本ブロック内部で依存関係がある手続きについては後続手続きの先行手続きから後続手続きへのデータ転送に対して依存関係を明示的に登録する命令を生成するようにして、依存関係を満たすように手続き制御する。このような構成とすることで、複雑な制御の依存関係が存在する基本ブロック間については、手続きの実行を待ち合

わせにより実現することで制御プログラムの生成を容易なものとし、実行順が固定である同一基本ブロック内については、依存関係待ち合わせ付き非同期遠隔手続呼び出しにより無駄な待ち合わせ時間をなくすることができる。

また、同一の基本ブロック内部でのデータ転送及び基本ブロック間をまたいでデータ転送の両方について手続から手続への直接のデータ転送を指示する命令を生成するようにしたので、制御プロセッサを介さずに直接に手続実行プロセッサ間でデータ転送でき、並列化プログラムの処理を効率化することができる。

図面の簡単な説明

[0027] [図1]無駄な待ち時間の発生について説明するための図である。

[図2]依存関係待ち合わせ付き非同期遠隔手続呼び出し方式による手続実行の制御について説明するための図である。

[図3]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4]手続きプログラム生成方法の概要を示す図である。

[図5]図4の手続きプログラム生成方法により生成される手続きプログラムを示す図である。

[図6]第1の実施例により手続き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7]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8]頂点 v 以下の制御の流れを再構成する処理(図7のステップS2)を示すフローチャートである。

[図9]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10]逆依存及び出力依存を求める処理(図9のステップS4)を示すフローチャートである。

[図11]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12]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13]逆依存の追加処理を示すフローチャートである。

[図14]出力依存の追加処理を示すフローチャートである。

[図15]逆依存及び出力依存を求める処理(図9のステップS5)を示すフローチャート

である。

[図16]全域木を説明するための図である。

[図17]全域木を模式的に示す図である。

[図18]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19]図18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20]基本ブロックを抽出する処理のフローチャートを示す図である。

[図21]プロセッサ毎に変数を生成する処理と依存関係を抽出する処理のフローチャートを示す図である。

[図22]制御プログラムを生成する処理のフローチャートを示す図である。

[図23]基本ブロックの集合 B' の要素 B_i 以下の手続き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24]手続き制御プログラムの構造を示す図である。

[図25]第2の実施例による手続き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26]手続き毎に変数を生成する処理のフローチャートを示す図である。

[図27]第2の実施例における基本ブロックの集合 B' の要素 B_i 以下の手続き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28]第3の実施例による手続き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29]変数を生成する処理のフローチャートを示す図である。

[図30]入力逐次プログラムの部分及び対応する縮退プログラム依存グラフを示す図である。

[図31]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

- [0028] 10 入力変数の引数受信部分
- 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 全城木

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

[0029] 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。

[0030] 図3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。

[0031] 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、手続き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、手続き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、非投機的に手続きの起動と同期を制御する手続き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手続きプログラムを生成する。

[0032] 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3のステップS1)について説明する。

[0033] プログラム依存グラフとは、例えば非特許文献2乃至4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2乃至4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。

[0034] [V:頂点集合]

エントリ:プログラムの開始ポイントを表す。

[0035] 初期定義:プログラム開始時の初期値の定義を表す。

[0036] プリディケート: If-then-elseまたはwhile-loopの条件判定を表す。

[0037] 代入文:プログラムの代入文を表す。

[0038] 最終使用:プログラム終了時の変数の参照を表す。

[0039] [E:辺集合]

[制御依存辺: $v \xrightarrow{L}_c w$] 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。

[0040] [データ依存辺]

[ループ独立フロー依存辺: $v \xrightarrow{L}_{li}^x w$] 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、ループを繰り返さない場合のみを表す。

[0041] [ループ繰り返しフロー依存辺: $v \xrightarrow{L}_{lc(L)}^x w$] 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を繰り返す場合を表す。

[0042] [定義順序関係: $v \xrightarrow{do(u)}^x w$] 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。制御の流れによつては、v, w, u, あるいは、v, uの順に実行される可能性がある場合に、v, wの実行順序を表すものである。

[0043] 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3のステップS2)について説明する。

[0044] 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。

[0045] 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。

[0046] 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。

[0047] 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。

[0048] 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2)。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。

[0049] 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。

[0050] 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。

[0051] 2) 頂点間のデータ依存パスに外部の頂点が含まれない。

[0052] 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。

[0053] [V:頂点集合]

エントリ:プログラムの開始ポイントを表す。

[0054] 初期定義:プログラム開始時の初期値の定義を表す。

[0055] プリディケート: If-then-elseまたはwhile-loopの条件判定を表す。

[0056] 文の集合: プログラムを構成する文の集合を表す。

[0057] 最終使用:プログラム終了時の変数の参照を表す。

[0058] [E:辺集合]

[制御依存辺: $v \xrightarrow[c]{L} w$] 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。

[0059] [データ依存辺]

[ループ独立フロー依存辺: $v \xrightarrow[l]{x} w$] 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。ここでは、ループを繰り返さない場合のみを表す。

[0060] [ループ繰り返しフロー依存辺: $v \xrightarrow[lc(L)]{x} w$] 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を繰り返す場合を表す。

[0061] [定義順序関係: $v \xrightarrow{do(u)}{x} w$] 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, u, あるいは、v, uの順に実行される可能性がある場合に、v, wの実行順序を表すものである。

[0062] 以下において、手続き制御プログラムを生成する処理(図3のステップS3)及び手続きプログラムを生成する処理(図3のステップS4)について説明する。

[0063] まず手続き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する、1つの手続きプログラムを1つの頂点に対して生成する。また、制御の流れより手続き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。

[0064] 図4は、手続きプログラム生成方法の概要を示す図である。図5は、図4の手続きプログラム生成方法により生成される手続きプログラムを示す図である。

[0065] 図4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、入力変数を引数として受信するためのプログラム部分を生成する。これにより、図5に示す入力変数の引数受信部分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5に示す変数宣言部分11が生成される。

[0066] 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5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返すためのプログラム部分を生成する。これにより、図5に示す出力変数のセット部分13が生成される。

[0067] このように、手続き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。また、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。

[0068] 次に手続き制御プログラムの生成について説明する。非特許文献2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、プログラムの実行順序関係を計算し、基本ブロックを求める。基本ブロックとは、分岐(IF、GOTO、LOOP等)や合流を含まない順番に実行される頂点の列のことを言う。各中間節点が表す制御構造と子頂点が表す「手続き」の呼び出しを行なうプログラムを生成することで、並列プログラムを生成することができる。「手続き」を実行する上で必要となる入力データのデータ転送および出力結果のデータ転送とそれらの待ち合わせを行なうコードも生成する。基本ブロック内の手続き呼び出しおよびデータ転送の依存関係に関しては、依存関係待ち合わせのメカニズムを用いて制御する。

[0069] 本発明によるプログラムの実行は、図2に示されるのと同様に、手続要求側プロセッサ(制御プロセッサ)により制御プログラムを実行し、この制御プログラムが呼び出す各手続きの手続きプログラムを各手続実行側プロセッサにより実行する。各手続きプログラムは、前述のように、頂点が表す文／文の集合を実行する手続きである。手続き呼び出し及び依存関係の登録と手続きの実行とについては、図2に示した仕組みと同様であり、制御プロセッサにおいて手続き呼び出しプログラム3が管理する各プロセッサ毎のキューに手続きと依存関係とを登録し、実行可能状態となった手続きを順次実行していく。

[0070] 手続きの入力データは、制御プロセッサから実行するプロセッサに転送する。但し、先行する手続きの結果を利用する場合には、手続きを実行したプロセッサから後続の手続きを実行するプロセッサに対して直接データを転送することとする。これは、元の逐次プログラムにおける条件判定に相当する、制御プログラム上の条件判定結果によっては、複数データのうち適切なデータを選択することが必要となる場合がある。このようなデータ選択は、データ転送の依存関係として制御することになる。

[0071] 実行するマルチプロセッサシステムが分散メモリの場合、手続きへの入出力データをプロセッサ上のメモリ領域に転送し、プロセッサがその領域を用いて計算を行ない、結果を他の適切な領域に転送することになる。この場合のデータ変数の割り付け方としては、1)プロセッサ毎に使用する可能性のある変数の複製領域を作成する方式と、2)手続き毎に使用する可能性のある変数の複製領域を作成する方式とが考えられる。プロセッサ毎に変数の複製領域を作成する方式では、同一のプロセッサが実

行する第1の手続きと第2の手続きとが同一の変数 x を使用する場合、このプロセッサのメモリ領域には1つの変数 x の領域しか設けない。また手続き毎に変数の複製領域を作成する方式では、同一のプロセッサが実行する第1の手続きと第2の手続きとが同一の変数 x を使用する場合、このプロセッサのメモリ領域には第1の手続きの変数 x の領域と第2の手続きの変数 x の領域とをそれぞれ別個に設ける。

[0072] また更に、両方式の組み合わせとして、3) プロセッサ毎に使用する可能性のある変数の複製領域を作成し、逆依存関係又は出力依存関係による待ち合わせを削減できる場合は、手続き毎の異なる領域を作成する方式が考えられる。これら第1乃至第3の方式において、同一の変数に対する複製領域の各々について、異なる名前を付けて区別する。

[0073] 上記、1)乃至3)の何れの方式を用いるかに応じて、変数間の依存関係をどのように扱うかが異なってくる。1)及び3)の方式の場合には、変数間の逆依存関係／出力依存関係を抽出し、プロセッサ毎に変数の複製領域を作成することで解消される逆依存関係／出力依存関係と、プロセッサ毎に変数の複製領域を作成することでは解消されない逆依存関係／出力依存関係とを区別して、解消されないものについては手当てする必要がある。それに加え、フロー依存関係及び定義順序関係についても手当てする必要がある。また2)の方式の場合には、手続き毎に変数の複製領域を作成することにより、逆依存関係／出力依存関係については解消されることになるので、データ依存関係のうちのフロー依存関係及び定義順序関係についてのみ手当てが必要となる。

[0074] ここで逆依存関係は、第1の命令においてある変数の値が使われた後に第2の命令においてその変数が定義される可能性がある場合に対応し、第1の命令から第2の命令に逆依存関係が存在すると言う。また出力依存関係は、第1の命令においてある変数の値が定義された後に第2の命令においてその変数が定義される可能性がある場合に対応し、第1の命令から第2の命令に出力依存関係が存在すると言う。何れの関係においても、第2の命令を実行してから第1の命令を実行するように実行順序を逆にはできない。

[0075] 逆依存関係或いは出力依存関係にある頂点 v , w については、当該変数のデータ

転送に関して適切な待ち合わせを行なう必要がある。これについては後ほど詳細に説明する。

[0076] 以下に、本発明の実施例について詳細に説明する。第1乃至第3の実施例は、それぞれ上記の第1乃至第3の方式に対応する。また第4乃至第6の実施例は、それぞれ第1乃至第3の実施例に対して、定義順序関係に関するデータ転送を高速化する変形を加えたものである。第1乃至第6の実施例について、以下に順番に説明する。

[0077] 図6は、第1の実施例により手続き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を計算し、求めた実行順序(制御の流れ)から基本ブロックを抽出する。縮退したプログラム依存グラフは、データ及び制御の依存関係のみを表現したグラフであって頂点間の実行順序は明示されていないので、これから適切な制御の流れを再構成する必要がある。そこで、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、各中間節点の子頂点の実行順序を計算する。この結果、頂点間の半順序関係を求めることができる。この実行順序関係を用いて、制御プログラムを生成することとなる。またその課程において、逆依存関係、出力依存関係が抽出される。更に、求めた実行順序(制御の流れ)から、基本ブロックを抽出する。

[0078] 次にステップS2で、変数の生成と依存関係抽出を行う。本実施例では、プロセッサ毎に変数を生成し、それら変数についての依存関係を抽出する。

[0079] 次にステップS3で、制御プログラムの変数と初期値代入文を生成する。ここで変数としては、データの受け渡しを行うための変数を生成する。

[0080] 次にステップS4で、S1で求めた実行順序順に制御依存部分グラフを探索し、制御プログラムを生成する。プリディケート頂点については、その頂点が表す制御構造を生成する。そして、制御構造の本文として、当該頂点の下位の部分木の制御プログラムを生成する。基本ブロックについては依存関係に基づく非同期遠隔手続きおよびデータ転送を行う文を生成する。これについては以下に詳細に説明する。

[0081] 更にステップS5で、手続きの実行結果の待ち合わせを行う文を生成する。

[0082] 図7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7の処理は、図6のステップS1の前半部分に相当する。図7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。

[0083] 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。

[0084] 図8は、頂点 v 以下の制御の流れを再構成する処理(図7のステップS2)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及び頂点 v である。

[0085] ステップS1で、 $\text{Region}(v, T) = \{u \mid u \in V, v \xrightarrow{c}^T u \in E\}$ 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 $\text{Region}(v, T)$ とは、頂点 u の集合であって、頂点 v から頂点 u への $L=F$ 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、 $v \xrightarrow{c}^T u$ は $L=F$ の制御依存辺を示すものである。

[0086] ステップS2で、 $\text{Region}(v, T)$ の実行順序関係を計算する。ステップS3で、 $\text{Region}(v, F) = \{u \mid u \in V, v \xrightarrow{c}^F u \in E\}$ 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 $\text{Region}(v, F)$ とは、頂点 u の集合であって、頂点 v から頂点 u への $L=F$ 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。

[0087] 図9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8のステップS2及びステップS4の各々に対応する。図9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。

[0088] 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。

[0089] 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

n)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。

[0090] 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り返す。

[0091] 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。この際、各頂点の各PE(プロセッサエレメント)への割り付けも行われる。

[0092] 図10は、逆依存及び出力依存を求める処理(図9のステップS4)を示すフローチャートである。図10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。

[0093] 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。

[0094] 図11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11の処理は図10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。

[0095] 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返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。

- [0096] 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。
- [0097] 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。
- [0098] 図12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12の処理は図10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。
- [0099] 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。
- [0100] 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。
- [0101] 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。
- [0102] 図13は、逆依存の追加処理を示すフローチャートである。図13の処理は図10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。
- [0103] 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。

- [0104] 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。
- [0105] 図14は、出力依存の追加処理を示すフローチャートである。図14の処理は図10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。
- [0106] 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。
- [0107] 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。
- [0108] 図15は、逆依存及び出力依存を求める処理(図9のステップS5)を示すフローチャートである。図15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。
- [0109] 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、
- $$\text{Span}(v, x) = \{v\} \cup \{u \mid v \xrightarrow{x} u \in E\}$$
- と定義される。図16は、全域木を説明するための図である。図16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17は、全域木を模式的に示す図である。全域木 $\text{Span}(v_i, x)$ 及び全域木 $\text{Span}(v_j, x)$ が、データ依存グラフとして図17に示されるように構成される。
- [0110] 図15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 $\text{Span}(h_0, x)$ 及び $\text{Span}(h_1, x)$ 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 $\text{Span}(h_0, x)$ 及び $\text{Span}(h_1, x)$ について、 $\text{Span}(h_0, x)$ に含まれる頂点と $\text{Span}(h_1, x)$ に含まれる頂点との間に辺(依存関係)がないことを言う。

[0111] 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。

[0112] 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。

[0113] 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。

[0114] 以上の処理により、2つの全域木 $\text{Span}(h_0, x)$ 及び $\text{Span}(h_1, x)$ 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。

[0115] 図18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18の処理は、図15のステップS6及びステップS11に相当する。図18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。

[0116] ステップS1で、着目領域内の各辺 e (頂点 $v \rightarrow$ 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。

- [0117] ステップS3で、 $V_a \leftarrow \{u \mid v \in \text{Span}(u, x)\}$ とするとともに、 $V_b \leftarrow \{u \mid w \in \text{Span}(u, x)\}$ 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。
- [0118] 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、 $\text{Span}(v_a, x)$ の頂点であって $\text{Span}(v_b, x)$ 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。
- [0119] ステップS7で、 $v_c \rightarrow v_b$ 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。
- [0120] 図19は、図18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19には、頂点 v の変数 x に関する全域木 $\text{Span}(v, x)$ と頂点 w の変数 x に関する全域木 $\text{Span}(w, x)$ 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 $\text{Span}(v_a, x)$ (即ち $\text{Span}(v, x)$)の各頂点 v_c (即ち v 、25、26)に対して、全域木 $\text{Span}(v_b, x)$ (即ち $\text{Span}(w, x)$)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。
- [0121] 図20は、基本ブロックを抽出する処理のフローチャートを示す図である。図20に示す処理は、図6のステップS1の後半部分の処理に相当する。図20の処理の入力は、実行順序関係が決定された縮退したプログラム依存グラフである。
- [0122] 求めた制御の流れの順に頂点を探索し、頂点の種類に応じた処理を行なう。以下の説明において B は基本ブロックの集合であり、 B_i は i 番目の基本ブロックである。また v は現在の頂点(着目頂点)であり、 u は現在の頂点の1つ前の頂点である。
- [0123] まずステップS2で、最初的基本ブロック B_0 を空集合として生成する。次にステップS2で、 u をエン트리頂点(プログラムの開始ポイント)として、 v をエン트리頂点の次の頂点とする。ステップS4で、現在の頂点 v が最終頂点であるか否かを判断する。最終頂点である場合には、処理を終了して基本ブロックの集合 B が生成される。
- [0124] 現在の頂点 v が最終頂点でない場合には、ステップS5に進み、現在の頂点 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。プリディケート頂点である場合には、ステップS6に進み、 i をインクリメントしてから B_i の要素を v とすることで、新たなプリディケートのみの基本ブロック B_i を形成する。その後ステップS7で、更に i をインクリメントして、新たな空集合の基本ブロック B_i を形成する。

[0125] 現在の頂点 v がプリディケート頂点でない場合(S5でNoの場合)には、ステップS8で、現在の頂点 v と1つ前の頂点 u とが、同一のプリディケート頂点からの制御依存関係を有し、且つその制御依存関係が同一の条件判定フラグに基づくものであるか否かを判定する。この判定結果がNOとなるのは、例えば u と v とが、IF文の内部と外部とに対応する場合や、IF文のTHEN節とELSE節とに対応する場合等である。即ち、ステップS8においては、同一の条件判定に応じて双方共に実行される2つの頂点であるか否かが判定されている。

[0126] ステップS8の判定がYESの場合には、ステップS9で、現在の基本ブロックに現在の頂点 v を追加する。ステップS8の判定がNOの場合には、ステップS10で、 i をインクリメントして新たな空集合の基本ブロック B_i を形成する。その後ステップS11で、この新たに生成された基本ブロック B_i に現在の頂点 v を追加する。その後ステップS12で u と v とをそれぞれ次の頂点に更新し、ステップS4に戻り以降の処理を繰り返す。

[0127] 以上の処理により、分岐(IF、GOTO、LOOP等)や合流を含まない順番に実行される頂点の列である各基本ブロック B_i を生成し、これらの基本ブロックを要素とする基本ブロックの集合 B を生成することができる。分岐や合流を含まない頂点の列とは、固定の1つの実行順に従い順番に実行される頂点の列のことである。図20のフローチャートから分かるように、各プリディケート頂点は単独で1つの基本ブロック B_i を構成し、プリディケート頂点でない1つの基本ブロック B_i には、途中で分岐も合流もなく固定の1つの実行順に従い順番に実行される頂点の列が含まれることになる。

[0128] 本発明では、異なる基本ブロックをまたいでの手続き間の依存関係については、先行手続きの出力データ転送の終了待ち合わせを行ってから、後続手続きを実行するようにする。また同一の基本ブロック内部で依存関係がある手続きの実行については、依存関係待ち合わせ付き非同期遠隔呼び出しにより手続きを実行する。即ち、基本ブロック間をまたいでの依存関係がある手続きについては先行手続きの出力

データ転送を待ち合わせる命令の後に後続手続きを実行する命令を配置することにより、依存関係を満たすように手続き制御する。また同一の基本ブロック内部で依存関係がある手続きについては後続手続きの先行手続きの出力データ転送への依存関係を明示的に登録する命令を生成するようにして、依存関係を満たすように手続き制御する。このような構成とすることで、複雑な制御の依存関係が存在する基本ブロック間については、手続きの実行を待ち合わせにより実現することで制御プログラムの生成を容易なものとし、実行順が固定である同一基本ブロック内については、依存関係待ち合わせ付き非同期遠隔手続き呼び出しにより無駄な待ち合わせ時間をなくすることができる。

- [0129] 以上により、基本ブロックを抽出することができる。即ち、図6のステップS1の後半部分の処理が実行される。
- [0130] 図21は、プロセッサ毎に変数を生成する処理と依存関係を抽出する処理のフローチャートを示す図である。図21に示す処理は、図6のステップS2の処理に相当する。
- [0131] ステップS1で、縮約したプログラム依存グラフの各頂点について以降の処理を繰り返すループを開始する。
- [0132] ステップS2で、着目頂点がプログラム・ブロック頂点の場合、その頂点の手続きを実行するプロセッサに対して、既に変数を作成済みか否かを判定する。実行するプロセッサについて変数が作成済みの場合はステップS4に進む。実行するプロセッサについて変数が未作成の場合は、ステップS3で変数を作成し、その後ステップS4に進む。
- [0133] ステップS4で、変数の名前を付けかえる。即ち、例えばプロセッサPE1に変数xを作成してある場合、この変数xがプロセッサPE1の変数xであることを示すような変数名(例えばPE1__x)に変更する。
- [0134] 以上の処理を、縮約したプログラム依存グラフの各頂点について実行する。その後、ステップS5で、逆／出力依存関係を抽出する。なお、逆依存関係及び出力依存関係は、図6のステップS1で既に求められている。このステップS5では、上記の変数名変更により依存関係が解消された逆依存関係及び出力依存関係を削除することで逆依存関係と出力依存関係を求めてもよい。

[0135] 変数 x に関する逆依存関係 $v \rightarrow_{\text{anti}}^x w$ を削除する条件は、

$$PE(v) \neq PE(w)$$

かつ

$$\neg \exists u \in V \ w \rightarrow_f^x u \in E \text{ かつ } PE(v) = PE(u)$$

である。ここで $PE(v)$ は頂点 v が実行されるプロセッサ PE を表し、上記第1の条件では、逆依存関係にある頂点 v と w が異なるプロセッサ PE に割り付けられていることを示している。この場合、プロセッサ毎に変数が異なるので、このような逆依存関係は削除できる可能性がある。もし逆に、頂点 v と w が同一プロセッサ PE_i に対して割り付けられているとすると、そのプロセッサ PE_i の変数 x (例えば PE_i_x)に関して逆依存関係が解消されないので、当該逆依存関係を削除することはできない。即ち、頂点 v の処理が終了するまで、頂点 w の実行を待ち合わせる必要がある。

[0136] また上記第2の条件では、頂点 w で代入された値が頂点 u で参照されることを想定している。ここで頂点 v と頂点 u が同一のプロセッサ PE_i に割り付けられている場合、頂点 w で代入された変数 x の値が、頂点 u で参照するためにこのプロセッサ PE_i の変数 x に転送されることとなる。頂点 v もプロセッサ PE_i 上の変数 x を参照するため、頂点 v の処理が終わるまで、頂点 w からのデータ転送を待ち合わせる必要がある。

[0137] また変数 x に関する出力依存関係 $v \rightarrow_{\text{output}}^x w$ を削除する条件は、

$$PE(v) \neq PE(w)$$

かつ

$$\neg \exists u \in V \ w \rightarrow_f^x u \in E \text{ } PE(v) = PE(u)$$

である。上記第1の条件では、頂点 v と w が異なるプロセッサ PE に割り付けられていることを示している。もし逆に、頂点 v と w が同一プロセッサ PE_i に対して割り付けられているとすると、 PE_i の変数 x に関して出力依存関係が解消されない。頂点 v の結果を後続の頂点が利用するので、それらのデータ転送が完了するまで、頂点 w の実行を待ち合わせる必要がある。

[0138] また上記第2の条件では、頂点 w で代入された値が、頂点 u で参照されることを想定する。ここで頂点 v と頂点 u が同一のプロセッサ PE_i に割り付けられているとすると、頂点 w で代入された値が PE_i の変数 x に転送されることとなる。頂点 v の結果を後続の頂

点が利用するため、それらのデータ転送が完了するまで、頂点wからのデータ転送を待ち合わせる必要がある。なお、定義順序関係に相当する場合は、定義順序関係として扱い、出力依存関係は削除するものとする。

[0139] 以上により、変数作成及び依存関係抽出を実行することができる。即ち、図6のステップS2の処理が実行される。

[0140] 図22は、制御プログラムを生成する処理のフローチャートを示す図である。図22に示す処理は、図6のステップS4(及びS5)の処理に相当する。図22の処理の入力は、実行順序関係が決定された縮退したプログラム依存グラフ及び基本ブロックの集合Bである。

[0141] ステップS1において、各初期定義頂点について以降の処理を繰り返すループを開始する。ここで初期定義頂点とは、変数の初期値が定まっている頂点のことをいう。

[0142] ステップS2で、出力フロー依存辺に対応するデータ転送を生成する。即ち、初期定義頂点からプログラム・ブロック頂点へ向かうフロー依存辺について、データ転送を行なう文を生成する。これは、初期のデータ転送を実行するためのものである。

[0143] 各初期定義頂点について以上の処理が繰り返し実行されると、その後、ステップS3において、実行開始を指示する文を生成する。

[0144] ステップS4で、プログラムの先頭を表すエントリ頂点 v_{Entry} の直下の子頂点vを要素とする基本ブロックの集合をB'とする。ステップS5において、B'の各要素 B_i について、iの昇順に以降の処理を繰り返すループを開始する。ステップS6で、 B_i についての手続き制御プログラムを生成する。

[0145] 図23は、基本ブロックの集合B'の要素 B_i 以下の手続き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23の処理は、図22のステップS6に相当する。図23に示す処理の入力は縮退したプログラム依存グラフPDG及び基本ブロック要素 B_i である。

[0146] 図23のステップS1で、基本ブロック B_i の要素(頂点)の種類を判定する。基本ブロック B_i の要素である頂点の種類を判定することによって、基本ブロック B_i がプログラム・ブロックの集合であるか、プレディケート頂点であるかが分かる。

[0147] ステップS1の判定の結果、基本ブロック B_i がプログラム・ブロックの集合の場合は、

基本ブロック B_i に属する頂点の手続きを呼び出す文とその間の依存関係を登録する文とを生成することとなる。具体的には、まずステップS2において、基本ブロック B_i の先行手続きの出力データに対する待ち合わせを生成する。この際、ブロック外からブロック内へのフロー依存関係に関して、データ転送の終了待ち合わせを生成する。また同時に、定義順序関係及び逆依存関係、出力依存関係に関しても、手続きあるいはデータ転送の終了待ち合わせを生成する。これは、メモリ上の同一変数に対して、データが読み書きされる順を保証するための待ち合わせである。ここでは、次の5種類の辺について待ち合わせを生成する。

1) B_i の頂点 w へのループ繰越フロー依存辺: $v \rightarrow_{lc(L)} w \quad w \in B_i$

頂点 $v \rightarrow$ 頂点 w へのデータ転送について待ち合わせを生成する。

2) $B_x (i \neq x)$ の頂点 v から B_i の頂点 w へのループ独立フロー依存辺: $v \rightarrow_{li} w \quad u \in B_x \quad w \in B_i (i \neq x)$

頂点 $v \rightarrow$ 頂点 w へのデータ転送について待ち合わせを生成する。

3) B_i の頂点 w への定義順序関係: $v \rightarrow_{do(u)} w \quad w \in B_i$

頂点 $v \rightarrow$ 頂点 t へのデータ転送について待ち合わせを生成する。

4) $B_x (i \neq x)$ の頂点 v から B_i の頂点 w への逆依存関係: $v \rightarrow_{anti} w \quad v \in B_x \quad w \in B_i (i \neq x)$

4-1) $PE(v)=PE(w)$ の場合

頂点 v の手続き呼び出しについて待ち合わせを生成する。

4-2) $\exists u \in V \quad w \rightarrow_f^x u \in E$ かつ $PE(v)=PE(u)$ の場合

頂点 v の手続き呼び出しについて待ち合わせを生成する。

5) $B_x (i \neq x)$ の頂点 v から B_i の頂点 w への出力依存関係: $v \rightarrow_{output} w \quad v \in B_x \quad w \in B_i (i \neq x)$

5-1) $PE(v)=PE(w)$ の場合

頂点 v から全ての頂点 u への変数 x に関するデータ転送($\forall e = (v \rightarrow_f^x u) \in E$)について待ち合わせを生成する。

5-2) $\exists u \in V \quad w \rightarrow_f^x u \in E$ かつ $PE(v)=PE(u)$ の場合

頂点 v から全ての頂点 t への変数 x に関するデータ転送($\forall e = (v \rightarrow_f^x t) \in E$)について待ち合わせを生成する。

[0148] 即ち、ループ繰越フロー依存辺、ループ独立フロー依存辺、及び定義順序関係に

については無条件に待ち合わせを生成し、逆依存関係及び出力依存関係については上記に示される場合についてのみ待ち合わせを生成する。逆依存関係及び出力依存関係について、上記に示される場合以外は、前述のように削除されている。

[0149] 次にステップS3で、基本ブロック B_i の各頂点 v について、実行順序の順番で以降の処理を繰り返すループを開始する。ステップS4で、頂点 v の非同期遠隔手続き呼び出しを生成する。

[0150] ステップS5-1で、基本ブロック B_i に属する頂点から頂点 v へのループ独立フロー依存関係に関して依存関係を登録する文を生成する。

[0151] ステップS5-2で、基本ブロック B_i に属する頂点 v から他のプロセッサへのデータ転送指示を行う文、及び先行する手続き呼び出しに対する当該データ転送動作の依存関係登録を行う文を生成する。同一プロセッサ内の頂点に対しては、データ転送が不要なのでこの処理は行なわない。なお、基本ブロックを越えないデータ転送であるか基本ブロックを越えるデータ転送であるかに関わらず、制御プロセッサを介することなく後続手続きを実行するプロセッサに直接にデータ転送するように、データ転送指示を生成する。

[0152] 更にステップS5-3で、逆依存関係／出力依存関係に基づく依存関係を登録する文を生成する。具体的には、次の2種類の辺について依存関係を登録する

1) B_i の頂点 v から B_i の頂点 w への逆依存関係: $v \rightarrow_{\text{anti}}^x w, w \in B_i$

1-1) $PE(v)=PE(w)$ の場合

頂点 v の手続き呼び出しから、頂点 w の手続き呼び出しについて依存関係を登録する。

1-2) $\exists u \in V, w \rightarrow_f^x u \in E$ かつ $PE(v)=PE(u)$ の場合

頂点 v の手続き呼び出しから、頂点 $w \rightarrow$ 頂点 u のデータ転送について依存関係を登録する。

2) B_i の頂点 v から B_i の頂点 w への出力依存関係: $v \rightarrow_{\text{output}}^x w, w \in B_i$

2-1) $PE(v)=PE(w)$ の場合

頂点 v から全ての頂点 u への変数 x に関するデータ転送($\forall e=(v \rightarrow_f^x u) \in E$)から、頂点 w の手続き呼び出しについて依存関係を登録する。

2-2) $\exists u \in V \ w \xrightarrow[f]{x} u \in E$ かつ $PE(v)=PE(u)$ の場合

頂点 v から全ての頂点 t への変数 x に関するデータ転送 ($\forall e = (v \xrightarrow[f]{x} u) \in E$) から、頂点 $w \rightarrow$ 頂点 u のデータ転送について依存関係を登録する。

- [0153] 基本ブロック B_i の全ての頂点 v についてこれらの処理を繰り返した後に、ステップ S6 で、実行開始を指示する文を生成する。
- [0154] ステップ S1 の判定の結果、基本ブロック B_i がプリディケート頂点 v の場合は、頂点 v の表す制御構造を生成する。まずステップ S7 で、基本ブロック B_i の要素 v の先行手続きに対する待ち合わせを生成する。即ち、条件式で参照する変数の値を確定するために、入力フロー依存辺について、先行する手続き呼び出しを待ち合わせる文を生成する。ここでは、当該頂点の外のループを繰り越すフロー依存辺と、当該頂点へのループ独立フロー依存辺との2種類のデータ依存入力辺について、出力元頂点の手続き終了待ち合わせを生成する。
- [0155] 次にステップ S8 で、頂点 v のプレディケートの種類を判定する。プレディケートがループである場合には、ステップ S9 に進む。プレディケートが if 文である場合には、ステップ S14 に進む。
- [0156] ステップ S8 の判定結果がループを示す場合には、ステップ S9 において、入力逐次プログラムにおいて相当する for 文或いは while 文を生成する。次にステップ S10 において、頂点 v への $L=T$ の制御依存関係がある頂点 u を要素とする基本ブロックの集合を B' とする。ステップ S11 において、 B' の各要素 B_i について、 i の昇順に以降の処理を繰り返すループを開始する。ステップ S12 で、 B_i についての手続き制御プログラムを生成する。このステップ S12 は入れ子構造となっており、 B_i についてステップ S12 を実行することは、この B_i について図 22 全体のフローチャートを実行することに相当する。
- [0157] ループの終了後、ステップ S13 で、頂点 v へのループを繰り越す先行手続きの終了待ち合わせを生成する。これは、ループを繰り越して条件を判定するので、本文の末尾に、条件式への入力データ待ち合わせ (自ループを繰り越す入力フロー依存辺) を行なう文を追加するものである。
- [0158] ステップ S8 の判定結果が if 文を示す場合には、ステップ S14 において、if 文を生成

する。次にステップS15で、then節を生成する。ステップS16で、頂点 v への $L=T$ の制御依存関係がある頂点 u を要素とする基本ブロックの集合を B' とする。ステップS17において、 B' の各要素 B_i について、 i の昇順に以降の処理を繰り返すループを開始する。ステップS18で、 B_i についての手続き制御プログラムを生成する。このステップS18は入れ子構造となっており、 B_i についてステップS18を実行することは、この B_i について図22全体のフローチャートを実行することに相当する。なおステップS17及びS18で生成された文が、then節の本文を構成することになる。

[0159] 次にステップS19で、頂点 v への $L=F$ の制御依存関係がある頂点 u を要素とする基本ブロックの集合を B' とする。ステップS20で、基本ブロックの集合 B' が空集合であるか否かを判定し、空集合の場合には処理を終了する。基本ブロックの集合 B' が空集合でない場合、ステップS21で、else節を生成する。ステップS22で、 B' の各要素 B_i について、 i の昇順に以降の処理を繰り返すループを開始する。ステップS23で、 B_i についての手続き制御プログラムを生成する。このステップS23は入れ子構造となっており、 B_i についてステップS23を実行することは、この B_i について図22全体のフローチャートを実行することに相当する。なおステップS22及びS23で生成された文が、else節の本文を構成することになる。

[0160] 以上の処理を実行することで、基本ブロック B_i 以下の手続き制御プログラムが生成される。図24は、第1の実施例の場合の手続き制御プログラムの構造を示す図である。

[0161] 図24に示されるように、本発明の第1の実施例の場合の制御プログラムは、変数の宣言初期化部分41、プレディケートへの入力データ待合わせ部分42、プレディケートの制御構造の生成部分43、基本ブロックへの依存関係待ち合わせ部分44、基本ブロック内のスレッド起動と依存関係登録部分45、及び、手続き及びデータ転送の待ち合わせ終了処理部分46を含む。基本ブロック内のスレッド起動と依存関係登録部分45では、非同期遠隔手続き呼び出しの起動指示、手続きの出力データの転送指示、依存関係の登録、手続きのディスパッチ(実行開始)を行う。

[0162] なおプログラム・ブロックは手続きとして呼び出されることとなる。ここでは、分散メモリを想定しているため、入力データは予め実行するプロセッサ上に転送されているも

のとする。そのため、入出力変数のためのデータ領域は予め用意する。また、実行結果は、実行するプロセッサ上に格納し、必要とされるプロセッサへ適宜その値を転送するものとする(このデータ転送は制御プログラムにて制御する)。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、適切なアドレスに出力する変数の値を代入する文を生成する。

[0163] 以下に、本発明の第2の実施例を説明する。この第2の実施例は、手続き毎に使用する可能性のある変数の複製領域を作成する方式に対応する。以下においては、第2の実施例と第1の実施例とが相違する部分について主に説明する。特に説明のない部分については、第2の実施例と第1の実施例とは基本的に同様である。

[0164] 図25は、第2の実施例による手続き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、変数の生成を行う。即ち、各プログラム・ブロック頂点で読み書きする変数について、当該頂点を実行するプロセッサ上に頂点毎(手続き毎)の変数を生成する。更に、生成した変数を利用するために名前を付け替える。このように、手続き毎に変数を生成することによって、逆依存関係／出力依存関係を削減でき、実行順序関係の求め方に自由度が上がる。これを考慮して、ステップS1及びステップS2の順序を、第1の実施例とは入れ替えてある。

[0165] 次にステップS2で、頂点間の実行順序関係を計算し、求めた実行順序(制御の流れ)から基本ブロックを抽出する。この処理は、第1の実施例において説明した図6のステップS1と同様の処理である。なお上記のように手続き毎に変数を作成することにより、全ての逆依存関係及び出力依存関係が削除されることになる。従って、逆依存関係及び出力依存関係については抽出する必要がない。

[0166] 次にステップS3で、制御プログラムの変数と初期値代入文を生成する。この際、静的単一代入形式(非特許文献5、320頁)に変換することで、並列性を向上されることも考えられる。ここで変数としては、データの受け渡しを行うための変数を生成する。

[0167] 次にステップS4で、S2で求めた実行順序順に制御依存部分グラフを探索し、制御プログラムを生成する。プリディケート頂点については、その頂点が表す制御構造を生成する。そして、制御構造の本文として、当該頂点の下位の部分木の制御プログラ

ムを生成する。基本ブロックについては依存関係に基づく非同期遠隔手続きおよびデータ転送を行う文を生成する。この処理は、第1の実施例の図22に示す処理と同様である。但し、基本ブロック B_i についての手続き制御プログラムを生成する段階(図22のステップS6に対応する処理)の内容が、第1の実施例の場合と異なる。

[0168] 更にステップS5で、手続きの終了の待ち合わせを行う文を生成する。

[0169] 図26は、手続き毎に変数を生成する処理のフローチャートを示す図である。図26に示す処理は、図25のステップS1の処理に相当する。

[0170] ステップS1で、縮約したプログラム依存グラフの各頂点 v について以降の処理を繰り返すループを開始する。

[0171] ステップS2で、着目頂点 v がプログラム・ブロック頂点の場合、その頂点の手続きを実行するプロセッサに対して、既にその頂点(手続き)に対応する変数を作成済みか否かを判定する。実行するプロセッサについて変数が作成済みの場合はステップS4に進む。実行するプロセッサについて変数が未作成の場合は、ステップS3で変数を作成し、その後ステップS4に進む。

[0172] ステップS4で、変数の名前を付けかえる。即ち、例えばプロセッサPE1に手続きP1の変数 x を作成してある場合、この変数 x がプロセッサPE1の手続きP1の変数 x であることを示すような変数名(例えばPE1__P1__ x)に変更する。以上の処理を、縮約したプログラム依存グラフの各頂点について実行する。その後、ステップS5で、依存関係を抽出する。

[0173] 前述のように、第2の実施例における図25のステップS4の制御プログラム生成処理は、第1の実施例の図22に示す処理と同様である。但し、基本ブロック B_i についての手続き制御プログラムを生成する段階(図22のステップS6に対応する処理)の内容が、第1の実施例の場合と異なる。

[0174] 図27は、第2の実施例における基本ブロックの集合 B' の要素 B_i 以下の手続き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27に示す処理の入力は縮退したプログラム依存グラフPDG及び基本ブロック要素 B_i である。以下において、図27のフローチャートと図23のフローチャートとで同一の部分については説明を省略し、異なる部分についてのみ説明する。

[0175] 図27のステップS2では、基本ブロック B_i の先行手続きに対する待ち合わせを生成するが、逆依存関係及び出力依存関係は考慮する必要がない。従って、ここでは次の3種類の辺について待ち合わせを生成する。

1) B_i の頂点 w へのループ繰越フロー依存辺: $u \xrightarrow{lc(L)} w, w \in B_i$

頂点 $v \rightarrow$ 頂点 w へのデータ転送について待ち合わせを生成する。

2) $B_x (i \neq x)$ の頂点 u から B_i の頂点 w へのループ独立フロー依存辺: $u \xrightarrow{li} w, u \in B_x, w \in B_i (i \neq x)$

頂点 $v \rightarrow$ 頂点 w へのデータ転送について待ち合わせを生成する。

3) B_i の頂点 w への定義順序関係: $u \xrightarrow{do(t)} w, w \in B_i$

頂点 $v \rightarrow$ 頂点 t へのデータ転送について待ち合わせを生成する。

[0176] また図27に示す第2の実施例では、逆依存関係及び出力依存関係は考慮する必要がないので、図23に示すステップS5-3に相当する処理は実行されない。即ち、逆依存関係／出力依存関係に基づく依存関係を登録する文を生成する必要はない。

[0177] 以上のようにして、第2の実施例における制御プログラムを生成することができる。第2の実施例の場合の手続き制御プログラムの構造は、第1の実施例の場合の手続き制御プログラムの構造と同様である。また手続きの生成についても、第1の実施例の場合と同様である。

[0178] 以下に、本発明の第3の実施例を説明する。この第3の実施例は、プロセッサ毎に使用する可能性のある変数の複製領域を作成し、逆依存関係又は出力依存関係による待ち合わせを削減できる場合は、手続き毎の異なる領域を作成する方式に対応する。

[0179] 図28は、第3の実施例による手続き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。

[0180] まずステップS1で、頂点間の実行順序関係を計算し、求めた実行順序(制御の流れ)から基本ブロックを抽出する。縮退したプログラム依存グラフは、データ及び制御の依存関係のみを表現したグラフであって頂点間の実行順序は明示されていないので、これから適切な制御の流れを再構成する必要がある。そこで、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、各中間節点の子頂点の実行順序を計算する。この結果、頂点間の半順序関係を求めることができる。この実行順序関係を用いて、制御プログラムを生成することとなる。またその課程において、逆依存関係、出力依存関係が抽出される。更に、求めた実行順序(制御の流れ)から、基本ブロックを抽出する。この処理は、第1の実施例において説明した図6のステップS1と同一の処理である。

- [0181] 次にステップS2で、変数の生成を行う。即ち、各プログラム・ブロック頂点で読み書きする変数について、当該頂点を実行するプロセッサ上に変数を生成する。更に、生成した変数を利用するために名前を付け替える。本実施例では、まず最初にプロセッサ毎に使用する可能性のある変数の複製領域を作成し、その後逆依存関係及び出力依存関係をチェックし、逆依存関係又は出力依存関係による待ち合わせを削減できる場合には手続き毎に異なる変数の複製領域を作成する。
- [0182] 次にステップS3で、制御プログラムの変数と初期値代入文を生成する。この際、静的単一代入形式(非特許文献5、320頁)に変換することで、並列性を向上されることも考えられる。ここで変数としては、データの受け渡しを行うための変数を生成する。
- [0183] 次にステップS4で、S2で求めた実行順序順に制御依存部分グラフを探索し、制御プログラムを生成する。プリディケート頂点については、その頂点が表す制御構造を生成する。そして、制御構造の本文として、当該頂点の下位の部分木の制御プログラムを生成する。基本ブロックについては依存関係に基づく非同期遠隔手続きおよびデータ転送を行う文を生成する。この処理は、第1の実施例の図22に示す処理と同様である。但し、基本ブロック B_i についての手続き制御プログラムを生成する段階(図22のステップS6に対応する処理)については、第2の実施例の場合と同一の処理を実行する。
- [0184] 更にステップS5で、手続きの終了の待ち合わせを行う文を生成する。
- [0185] 図29は、変数を生成する処理のフローチャートを示す図である。図29に示す処理は、図28のステップS2の処理に相当する。
- [0186] ステップS1で、縮約したプログラム依存グラフの各頂点 v について以降の処理を繰り返すループを開始する。

- [0187] ステップS2で、着目頂点 v がプログラム・ブロック頂点の場合、その頂点の手続きを実行するプロセッサに対して、既にその頂点(手続き)に対応する変数を作成済みか否かを判定する。実行するプロセッサについて変数が作成済みの場合はステップS4に進む。実行するプロセッサについて変数が未作成の場合は、ステップS3で変数を作成し、その後ステップS4に進む。
- [0188] ステップS4で、変数の名前を付けかえる。即ち、例えばプロセッサPE1に変数 x を作成してある場合、この変数 x がプロセッサPE1の変数 x であることを示すような変数名(例えばPE1__ x)に変更する。以上の処理を、縮約したプログラム依存グラフの各頂点について実行する。
- [0189] その後、ステップS5で、全ての逆依存関係及び出力依存関係を探索して、各依存関係に対して以下の処理を繰り返し実行する。
- [0190] ステップS6で、着目している依存関係(逆依存関係又は出力依存関係)が、ステップS4の変数名変更により解消されているか否かを判断する。この依存関係が解消されているか否かの判断は、図21のステップS5での依存関係の解消及び削除の判断と同様である。依存関係が解消されているものについては、その依存関係を削除する。依存関係が解消されていない場合は、ステップS7で、処理を実行するプロセッサ上に、着目依存関係に対応する手続きの変数を複製して作成し、その後ステップS8に進む。
- [0191] ステップS8で、変数の名前を付けかえる。即ち、例えばプロセッサPE1に手続きP1の変数 x 及び手続きP2の変数 x を作成してある場合、これらの変数の変数名を例えばPE1__P1__ x 及びPE1__P2__ x のように各プロセッサ及び各手続きに固有の名前に変更する。以上の処理を、各逆依存関係及び出力依存関係について実行する。
- [0192] 前述のように、第3の実施例における図28のステップS4の制御プログラム生成処理は、第1の実施例の図22に示す処理と同様である。但し、基本ブロック B_i についての手続き制御プログラムを生成する段階(図22のステップS6に対応する処理)の内容が、第2の実施例の場合と同一の処理、即ち図27に示すフローチャートの処理となる。即ち、逆依存関係及び出力依存関係は考慮する必要がないので、逆依存関係及

び出力依存関係についての待ち合わせ及び依存関係の登録処理は実行されない。

[0193] 以上のようにして、第3の実施例における制御プログラムを生成することができる。

第3の実施例の場合の手続き制御プログラムの構造は、第1の実施例の場合の手続き制御プログラムの構造と同様である。また手続きの生成についても、第1の実施例の場合と同様である。

[0194] 以下に本発明の第4乃至第6の実施例について説明する。これら第4乃至第6の実施例は、それぞれ第1乃至第3の実施例に対して、定義順序関係に関するデータ転送を高速化するように修正を加えたものである。

[0195] 図30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30(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、適宜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。

[0196] 頂点vで定義されたxの値と、頂点wで定義されたxの値の何れかが頂点uで使われる可能性があるとき、頂点vのxから頂点wのxに頂点uに関する定義順序 (def-order dependence) の関係があるという。頂点vの手続きで求めた変数xの値と、頂点wの手続きで求めた変数xの値が、それぞれ頂点uに転送されることとなる。頂点vの実行時点では、条件式(if(p))の判定結果が不明なため、どちらの値が頂点uで使用されるのかは未定である。そこで、頂点vの結果を投機的に頂点uに対して転送し、条件判定の結果、値を上書きすることが判明した時点で、先行する転送をキャンセルする。その上で、頂点wの結果を頂点uに対して転送する。これは、明示的にデータ転送キャンセルの指示を生成する方法、又は投機実行が正しくなかったことが判明した時点でデータ転送キャンセルし正しいデータ転送を開始するマルチ・プロセッサ向け並列プログラム実行装置を利用する方法、の何れかの方法を用いて実行することができる。以下の説明では、明示的にデータ転送キャンセルの指示を生成する方法を用いた例について説明する。

[0197] 以下に本発明の第4の実施例を説明する。第4の実施例は、第1の実施例と比較して、基本ブロックB_iについての手続き制御プログラムを生成する処理(図23)におけるステップS2の処理内容のみが異なる。他の処理については、第4の実施例と第1の

実施例とは同一である。

[0198] 第1の実施例では、基本ブロック B_i についての手続き制御プログラムを生成する処理(図23)のステップS2において、定義順序関係について待ち合わせを生成していた。それに対して第4の実施例では、基本ブロック B_i についての手続き制御プログラムを生成する処理(図23)のステップS2において、 B_i の頂点 w への定義順序関係: $u \rightarrow do(t)w \ w \in B_i$ については、データ転送 $u \rightarrow f t$ のキャンセルを生成する(明示的にキャンセルする)。即ち、定義順序関係については、待ち合わせではなく、先行するデータ転送をキャンセルする(ライブラリで実現する場合は、基本ブロック内の上書きするデータ転送指示によりキャンセルされるので、ここでのキャンセルも不要である)。

[0199] このようにして第4の実施例では、第1の実施例に対して定義順序関係に関するデータ転送のキャンセル指示を追加することにより、処理をより高速化することが可能となる。

[0200] 以下に本発明の第5の実施例を説明する。第5の実施例は、第2の実施例と比較して、基本ブロック B_i についての手続き制御プログラムを生成する処理(図23)におけるステップS2の処理内容のみが異なる。他の処理については、第5の実施例と第2の実施例とは同一である。

[0201] 第2の実施例では、基本ブロック B_i についての手続き制御プログラムを生成する処理(図23)のステップS2において、定義順序関係について待ち合わせを生成していた。それに対して第5の実施例では、基本ブロック B_i についての手続き制御プログラムを生成する処理(図23)のステップS2において、 B_i の頂点 w への定義順序関係: $u \rightarrow do(t)w \ w \in B_i$ については、データ転送 $u \rightarrow f t$ のキャンセルを生成する(明示的にキャンセルする)。即ち、定義順序関係については、待ち合わせではなく、先行するデータ転送をキャンセルする(ライブラリで実現する場合は、基本ブロック内の上書きするデータ転送指示によりキャンセルされるので、ここでのキャンセルも不要である)。

[0202] このようにして第5の実施例では、第2の実施例に対して定義順序関係に関するデータ転送のキャンセル指示を追加することにより、処理をより高速化することが可能となる。

[0203] 以下に本発明の第6の実施例を説明する。第6の実施例は、第3の実施例と比較し

て、基本ブロック B_i についての手続き制御プログラムを生成する処理(図23)におけるステップS2の処理内容のみが異なる。他の処理については、第6の実施例と第3の実施例とは同一である。

[0204] 第3の実施例では、基本ブロック B_i についての手続き制御プログラムを生成する処理(図23)のステップS2において、定義順序関係について待ち合わせを生成していた。それに対して第6の実施例では、基本ブロック B_i についての手続き制御プログラムを生成する処理(図23)のステップS2において、 B_i の頂点 w への定義順序関係: $u \rightarrow do(t)w \ w \in B_i$ については、データ転送 $u \rightarrow f \ t$ のキャンセルを生成する(明示的にキャンセルする)。即ち、定義順序関係については、待ち合わせではなく、先行するデータ転送をキャンセルする(ライブラリで実現する場合は、基本ブロック内の上書きするデータ転送指示によりキャンセルされるので、ここでのキャンセルも不要である)。

[0205] このようにして第6の実施例では、第3の実施例に対して定義順序関係に関するデータ転送のキャンセル指示を追加することにより、処理をより高速化することが可能となる。

[0206] 図31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。

[0207] 図31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31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。

[0208] 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。

- [0209] 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。
- [0210] 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。
- [0211] 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。
- [0212] 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、文と文と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縮退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実行順序を算出し、
- 該実行順序を与えられた複数の頂点のうちで分岐及び合流の何れも含まず順番に実行される頂点列を基本ブロックとして纏め、
- 該縮退プログラム依存グラフの該頂点の各々に相当する手続きを生成し、
- 該基本ブロック間をまたいだ依存関係がある手続きについては先行手続きの出力データ転送を待ち合わせる命令の後に後続手続きを実行する命令を配置し、同一の基本ブロック内部で依存関係がある手続きについては先行手続きの出力データ転送に対する後続手続きの依存関係を登録する命令を生成し、および同一の基本ブロック内部でのデータ転送及び基本ブロック間をまたいでデータ転送それぞれについては手続きから手続きへの直接のデータ転送を指示する命令および該データ転送の先行手続きに対する依存関係を登録する命令を生成して、該手続きの実行を制御する手続き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方法。
- [2] 該手続き制御プログラムを生成するときに、該手続きを実行する各プロセッサ毎に変数を生成するように該手続き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方法。
- [3] 該手続き制御プログラムを生成するときに、該手続き毎に変数を生成するように該手続き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方法。
- [4] 該手続き制御プログラムを生成するときに、該手続きを実行する各プロセッサ毎に変数を生成し、更に各手続き毎に変数を生成することにより該依存関係を解消することが可能な変数については該手続き毎に変数を生成するように該手続き制御プログ

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方法。

- [5] 該手続き制御プログラムを生成するときに、定義順序関係について先行するデータ転送をキャンセルする指示を生成するように該手続き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方法。

- [6] 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とを含み、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縮退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該頂点の実行順序を算出し、

該実行順序を与えられた該複数の頂点のうちで分岐及び合流の何れも含まずに順番に実行される頂点列を基本ブロックとして纏め、

該縮退プログラム依存グラフの頂点の各々に相当する手続きを生成し、

該基本ブロック間をまたいだ依存関係がある手続きについては先行手続きの出力データ転送を待ち合わせる命令の後に後続手続きを実行する命令を配置し、同一の基本ブロック内部で依存関係がある手続きについては先行手続きの出力データ転送に対する後続手続きの依存関係を登録する命令を生成し、および同一の基本ブロック内部でのデータ転送及び基本ブロック間をまたいでデータ転送の両方について手続きから手続きへの直接のデータ転送を指示する命令および該データ転送の先行手続きに対する依存関係を登録する命令を生成して、該手続きの実行を制御する手続き制御プログラムを生成する

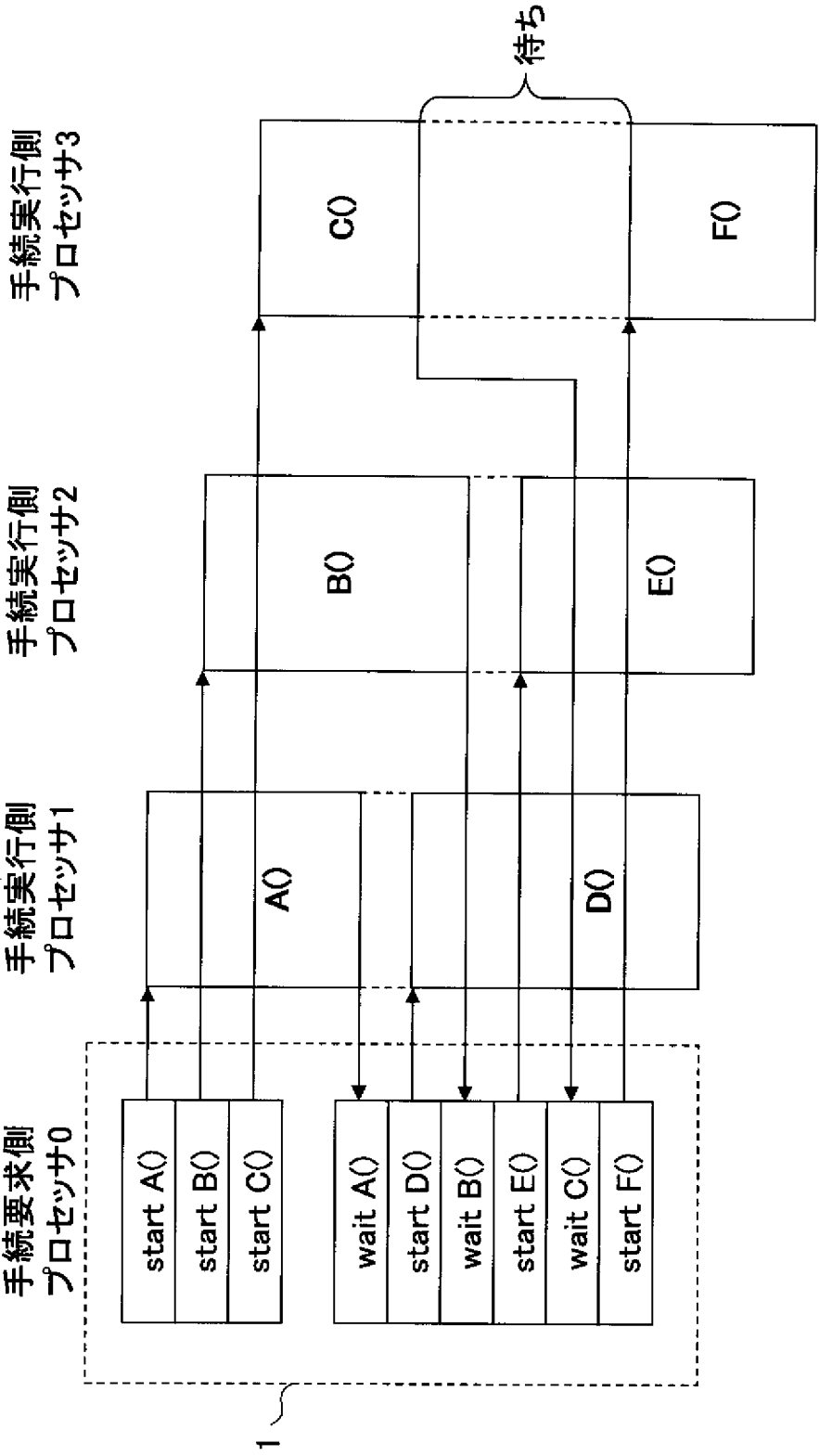
ことを特徴とする並列化プログラム生成装置。

- [7] 該演算処理ユニットは、該手続きを実行する各プロセッサ毎に変数を生成するように該手続き制御プログラムを生成することを特徴とする請求項6記載の並列化プログラ

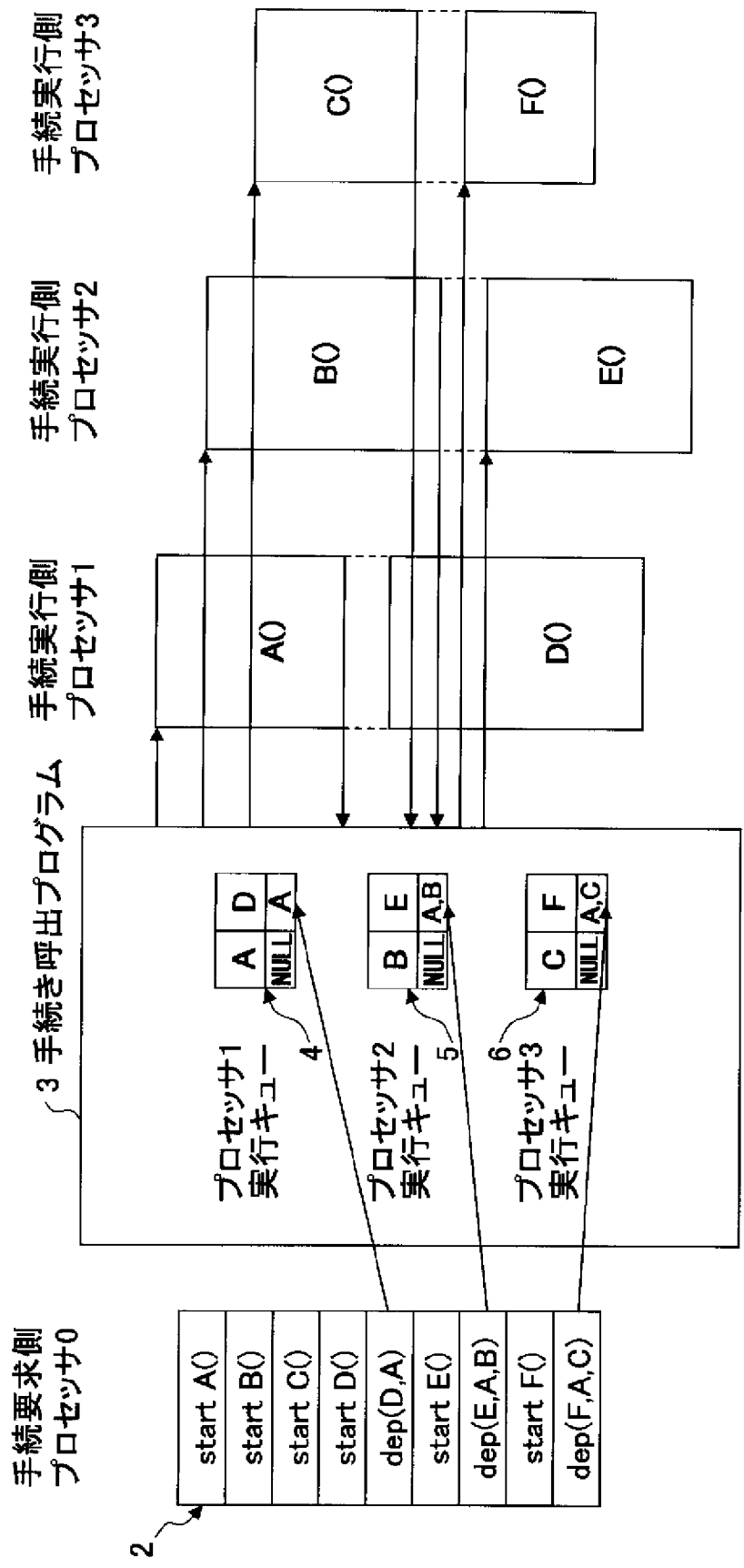
ラム生成装置。

- [8] 該演算処理ユニットは、該手続き毎に変数を生成するように該手続き制御プログラムを生成することを特徴とする請求項6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、該手続きを実行する各プロセッサ毎に変数を生成し、更に該手続き毎に変数を生成することにより該依存関係を解消することが可能な変数については各手続き毎に変数を生成するように該手続き制御プログラムを生成することを特徴とする請求項6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、文と文と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縮退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実行順序を算出し、該実行順序を与えられた複数の頂点のうちで分岐及び合流の何れも含まず順番に実行される頂点列を基本ブロックとして纏め、該縮退プログラム依存グラフの該頂点の各々に相当する手続きを生成し、該基本ブロック間をまたいだ依存関係がある手続きについては先行手続きの出力データ転送を待ち合わせる命令の後に後続手続きを実行する命令を配置し、同一の基本ブロック内部で依存関係がある手続きについては先行手続きの出力データ転送に対する後続手続きの依存関係を登録する命令を生成し、および同一の基本ブロック内部でのデータ転送及び基本ブロック間をまたいでデータ転送それぞれについては手続きから手続きへの直接のデータ転送を指示する命令および該データ転送の先行手続きに対する依存関係を登録する命令を生成して、該手続きの実行を制御する手続き制御プログラムを生成すること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。

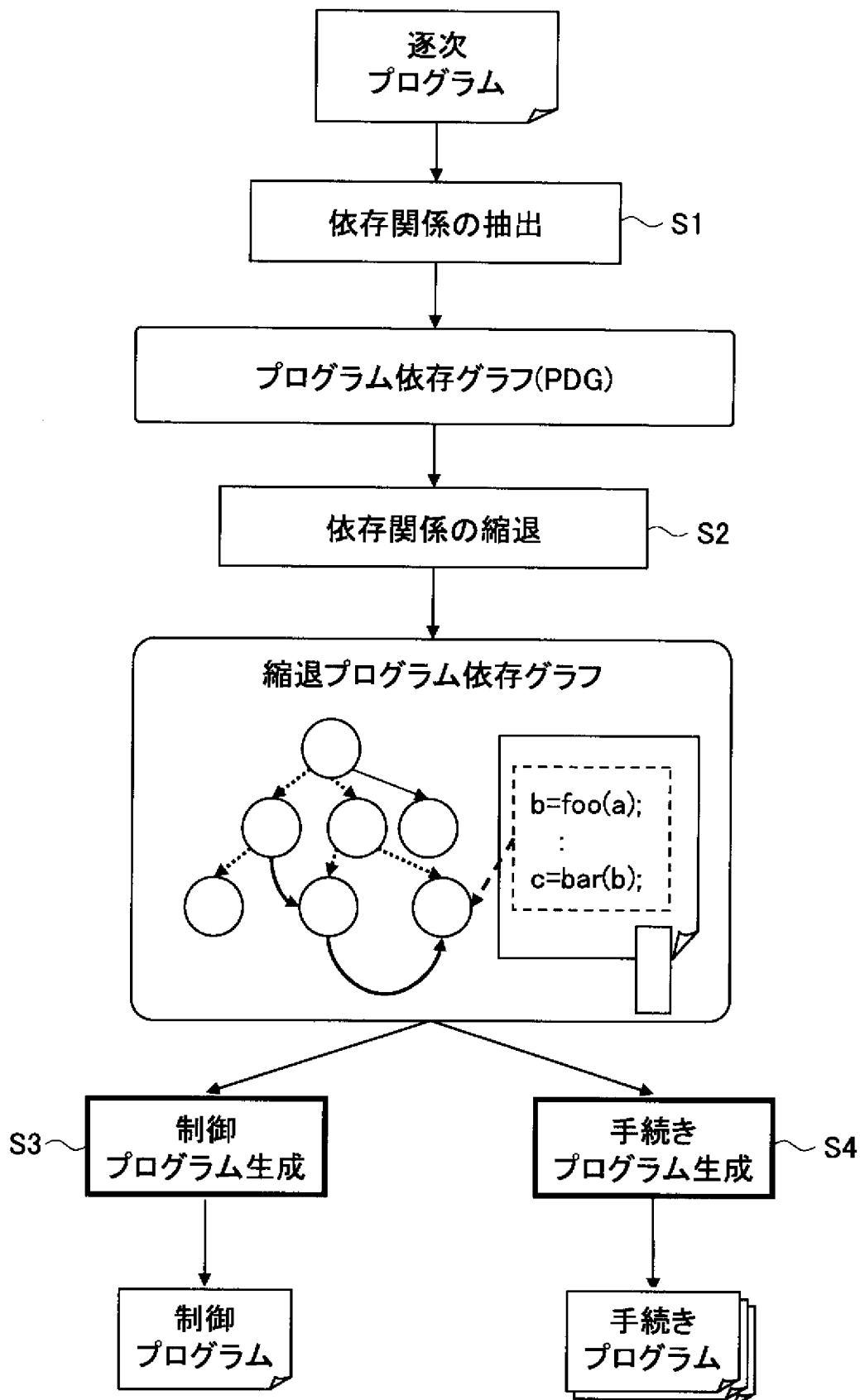
[図1]



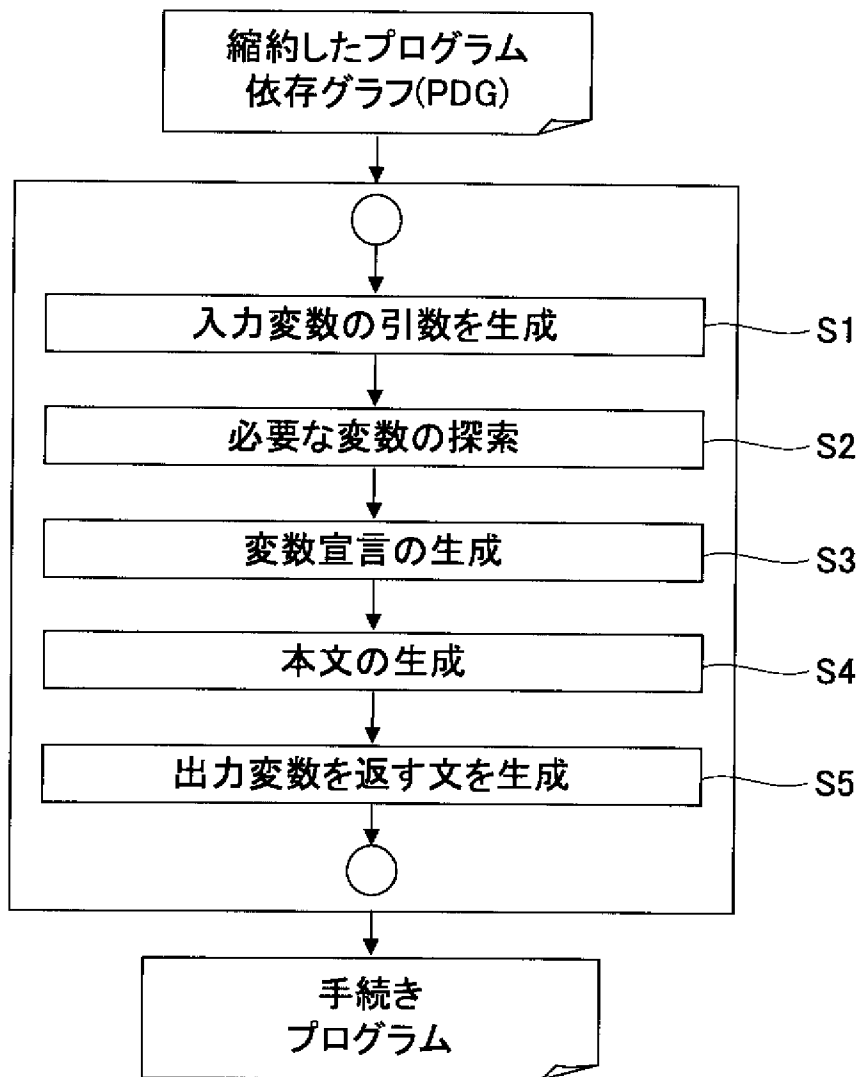
[図2]



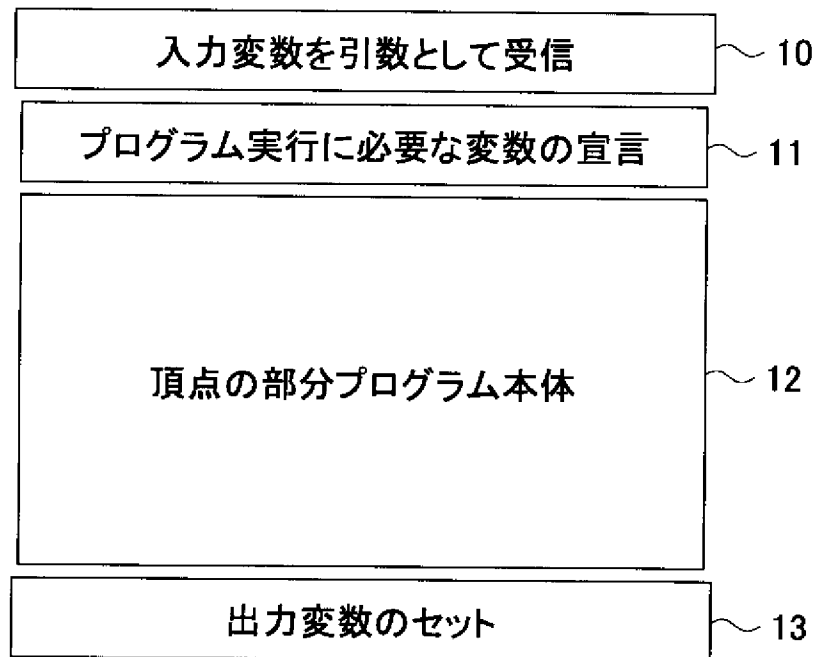
[図3]



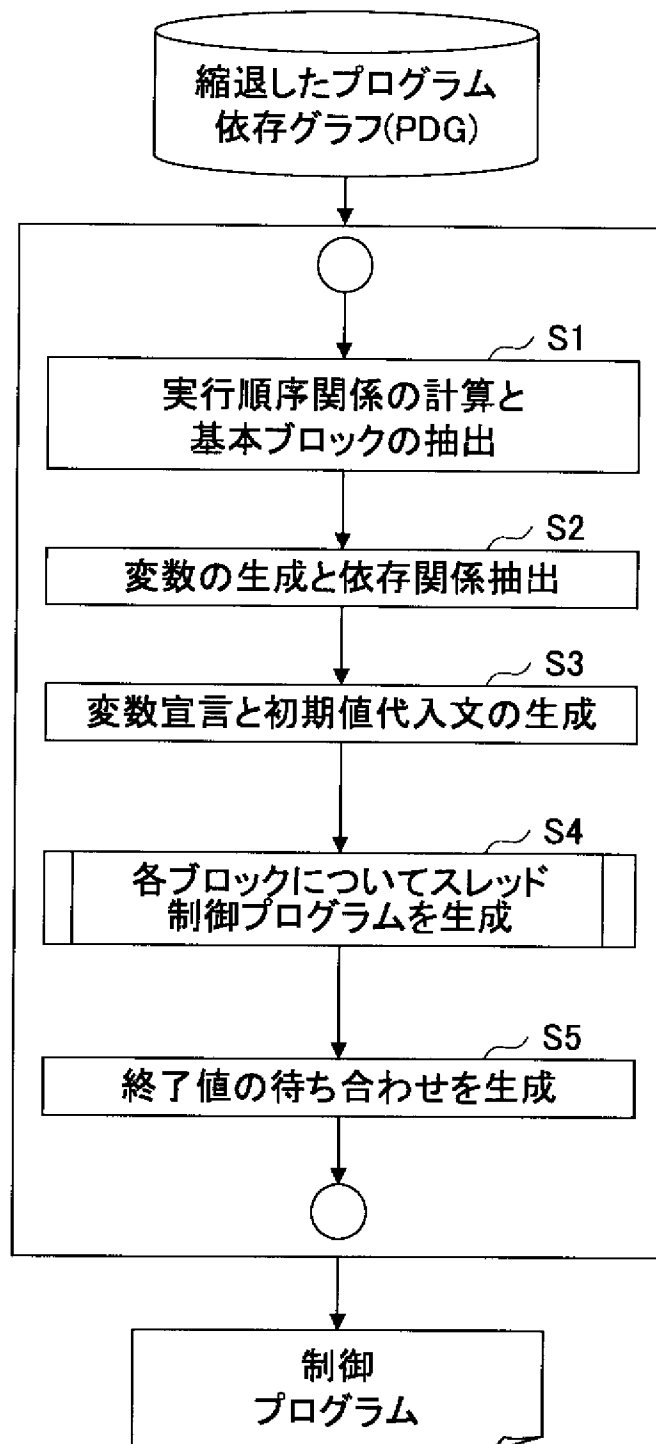
[図4]



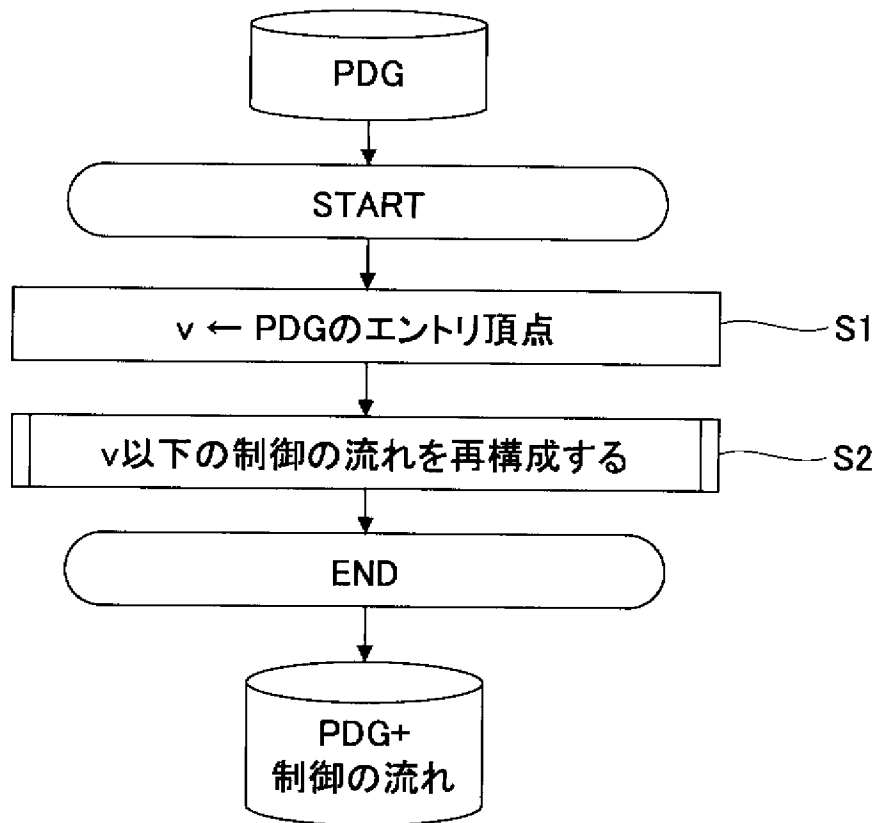
[図5]



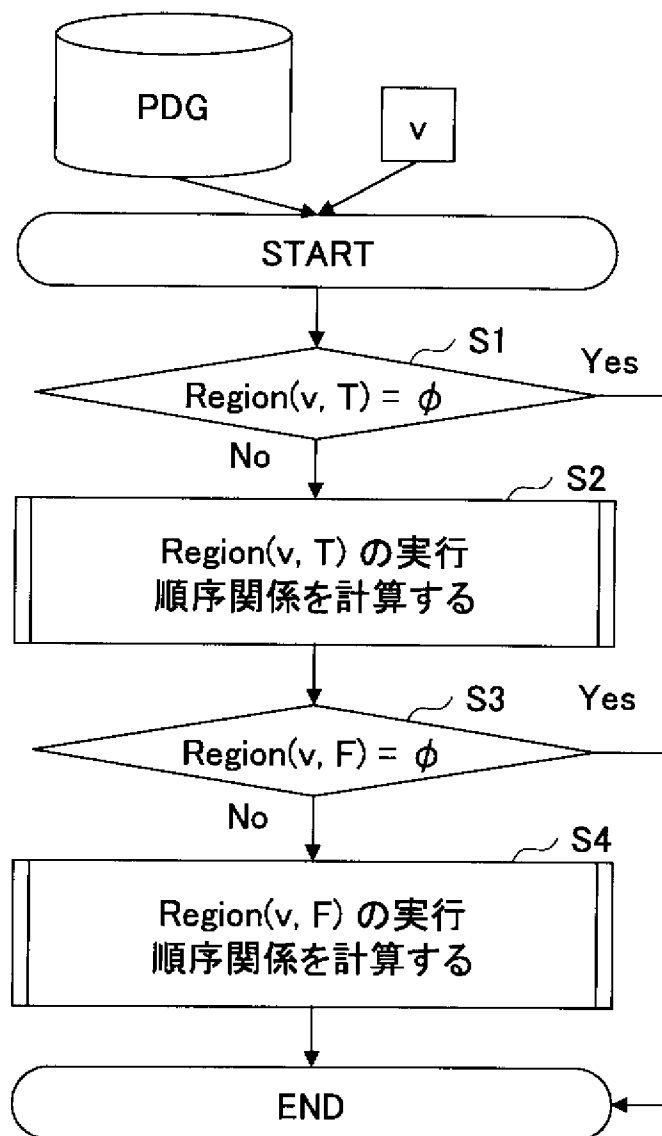
[図6]



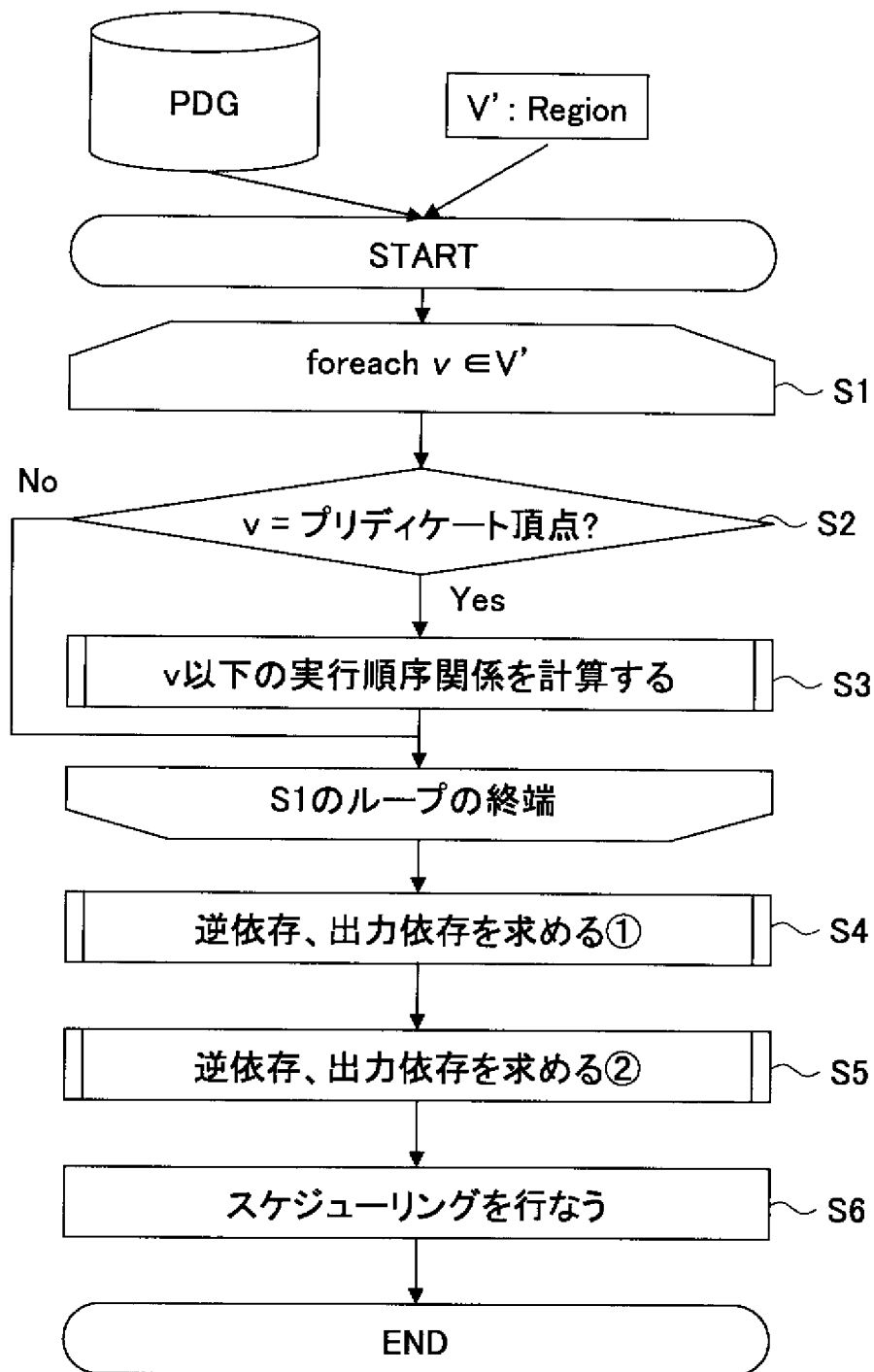
[図7]



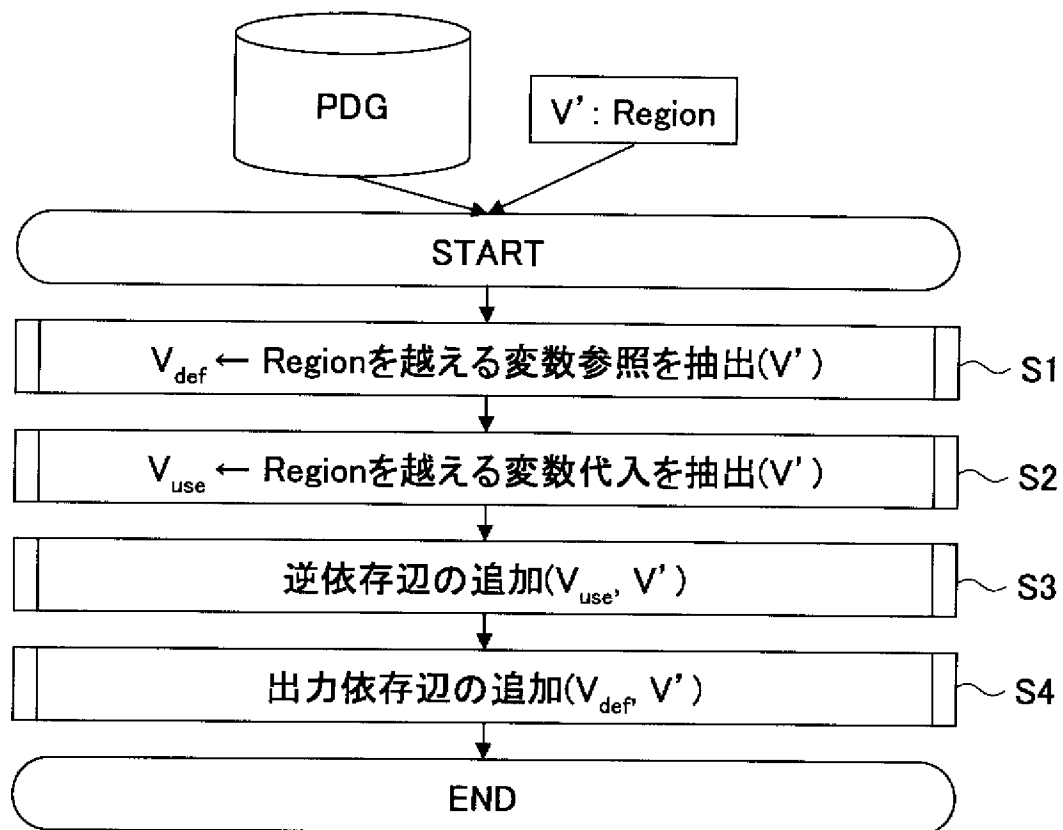
[図8]



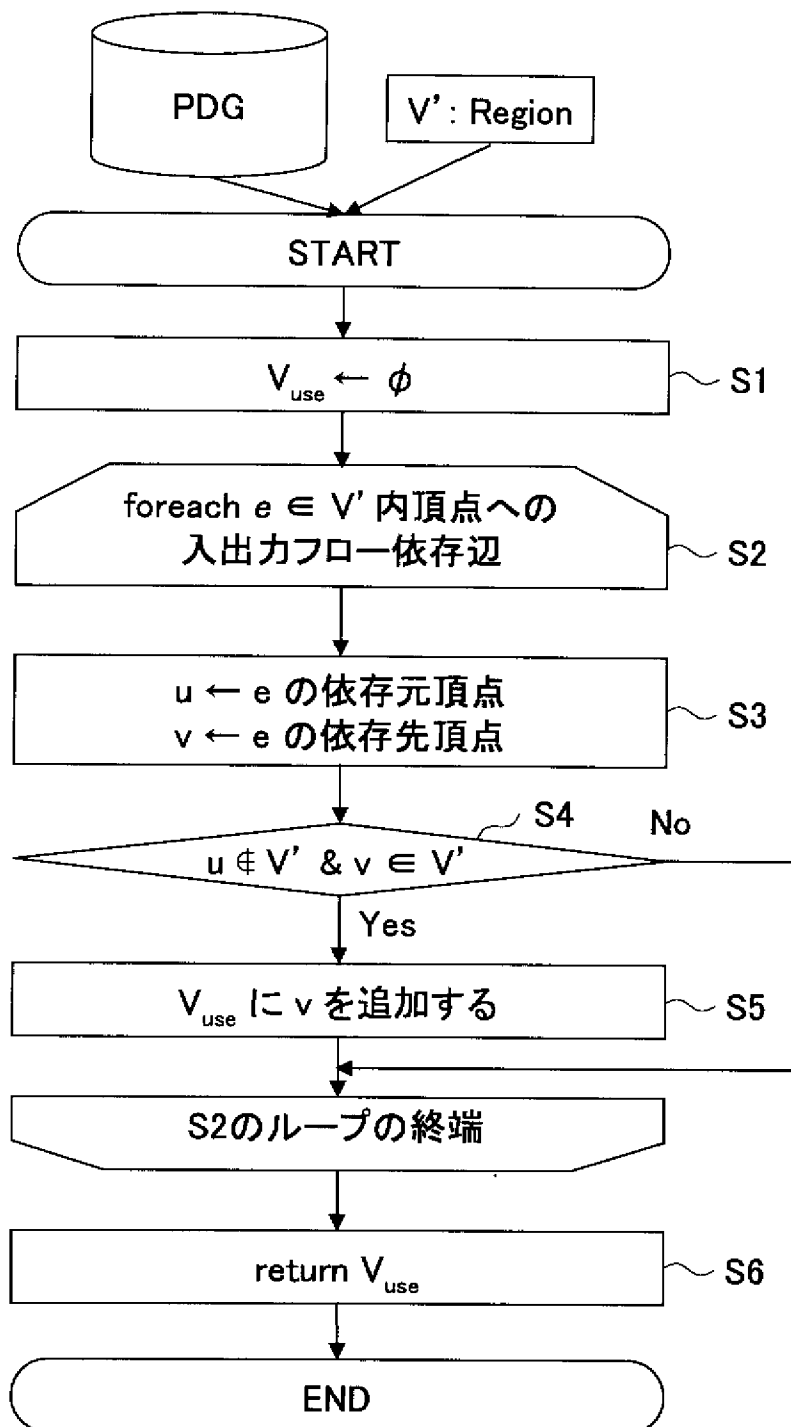
[図9]



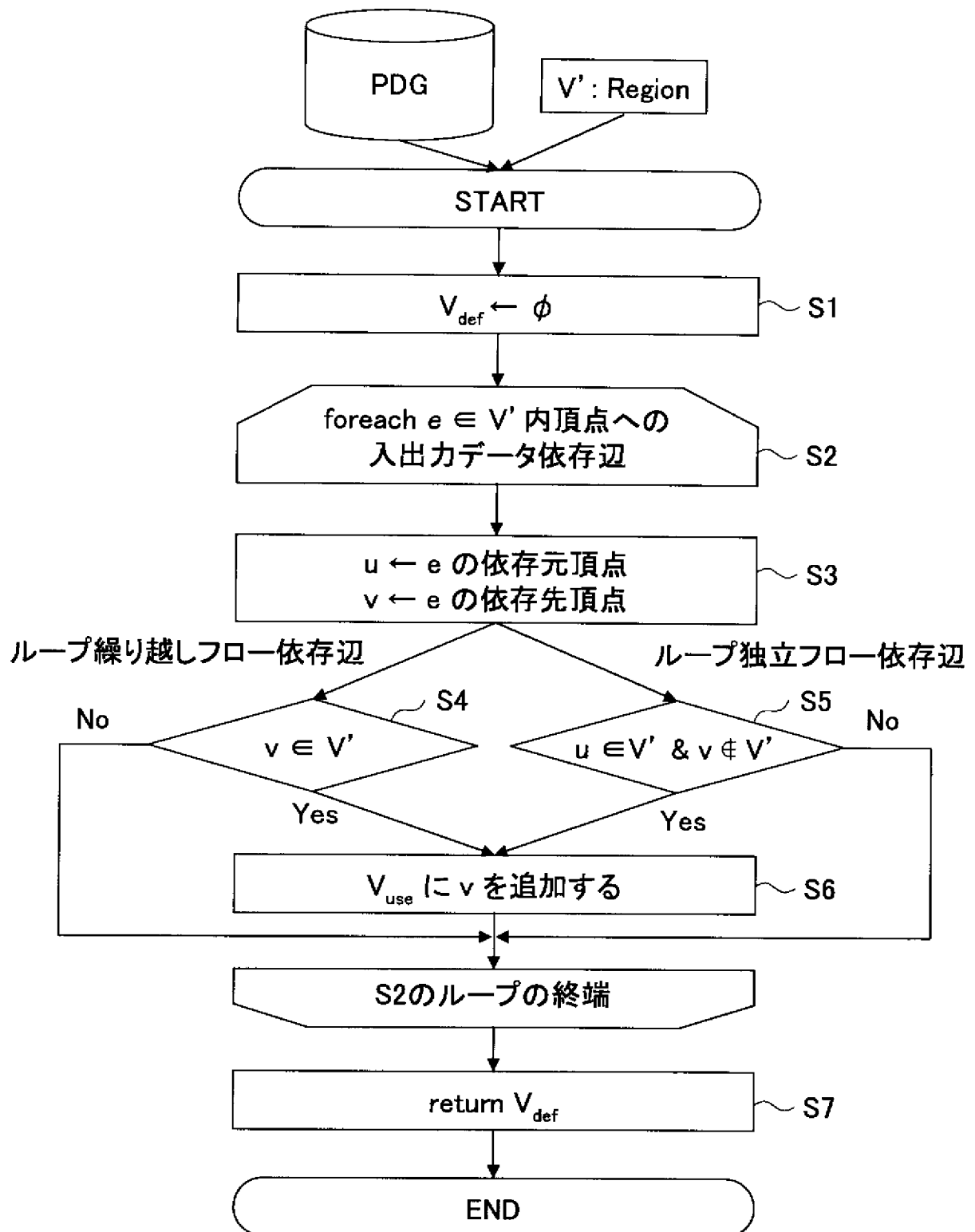
[図10]



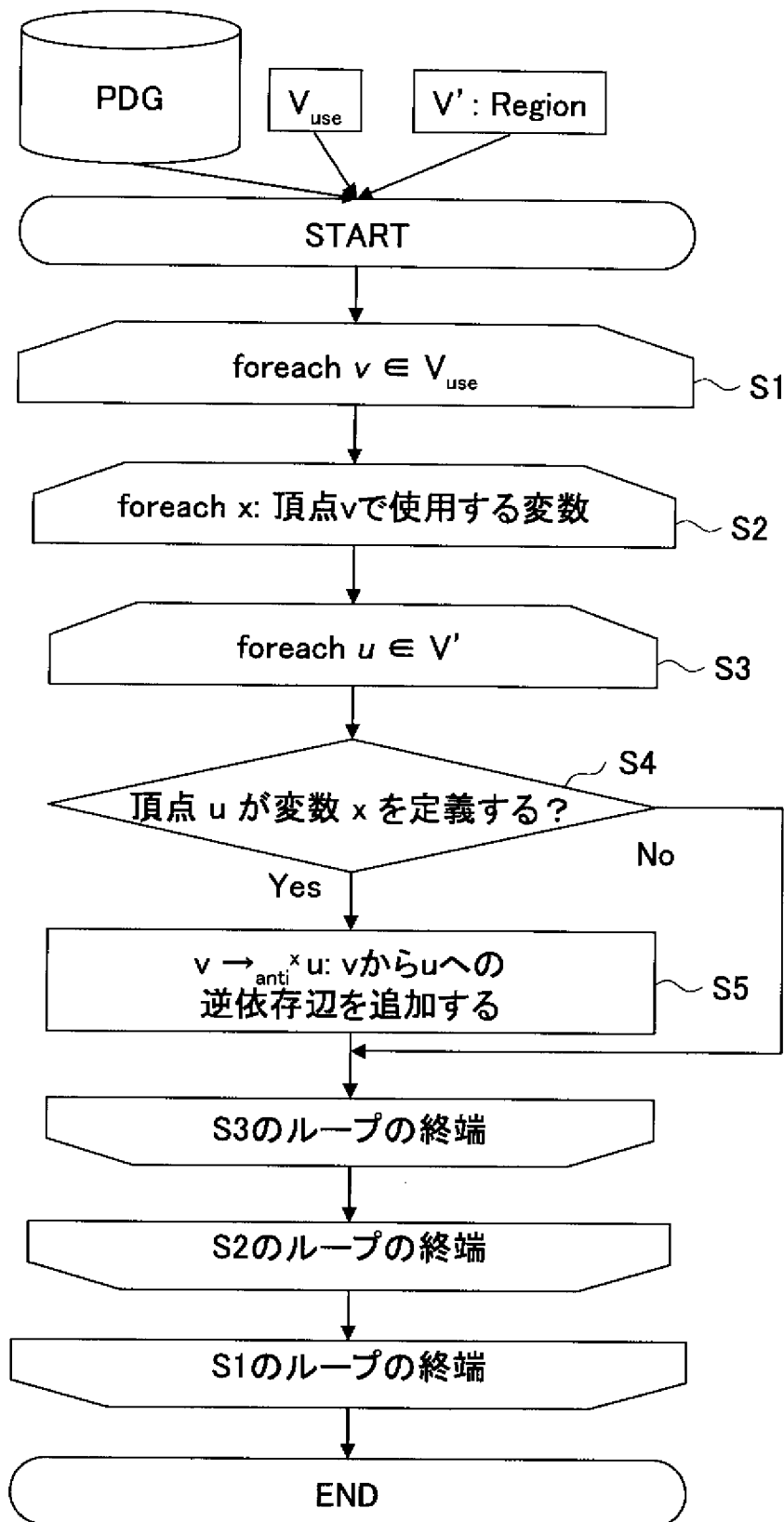
[図11]



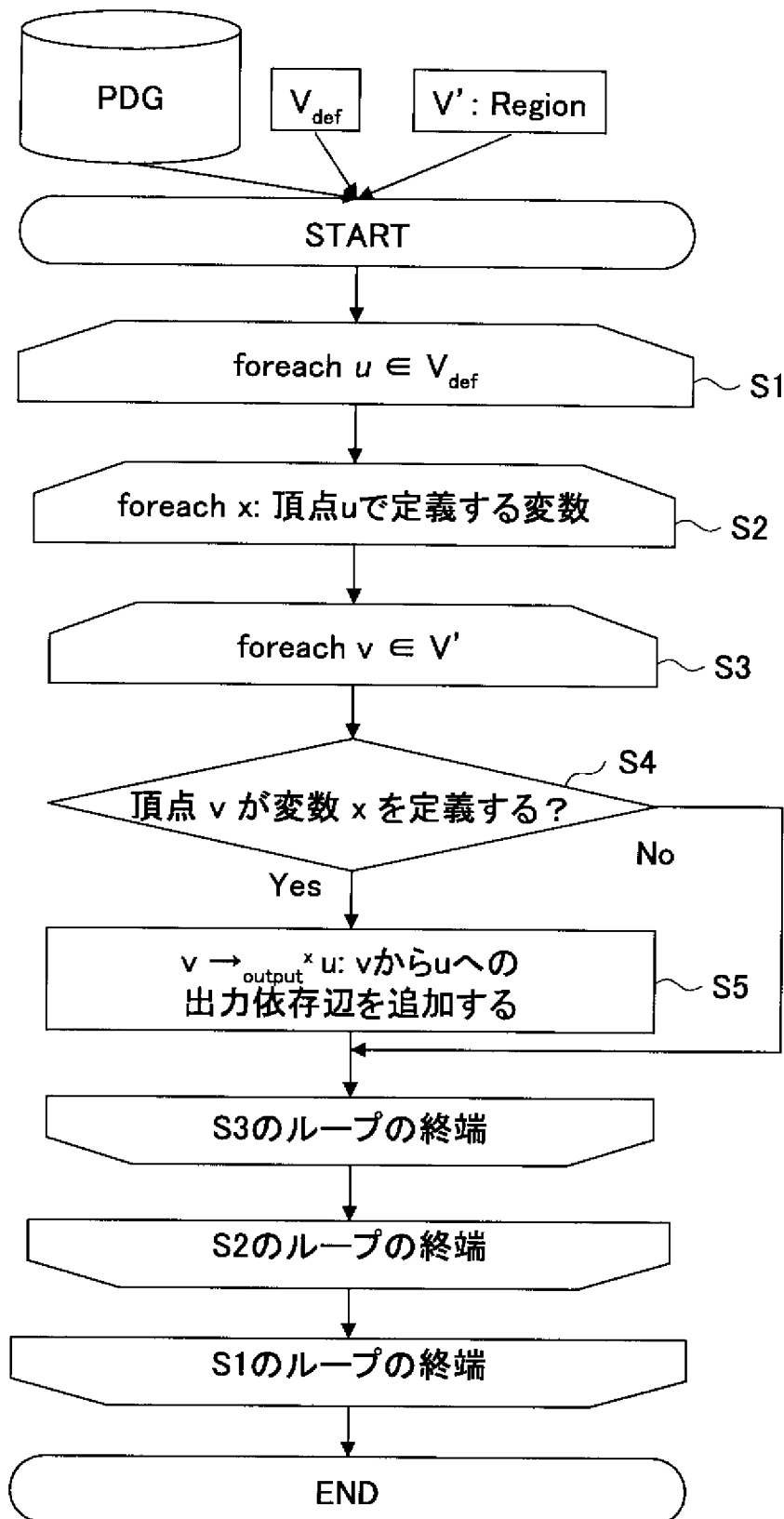
[図12]



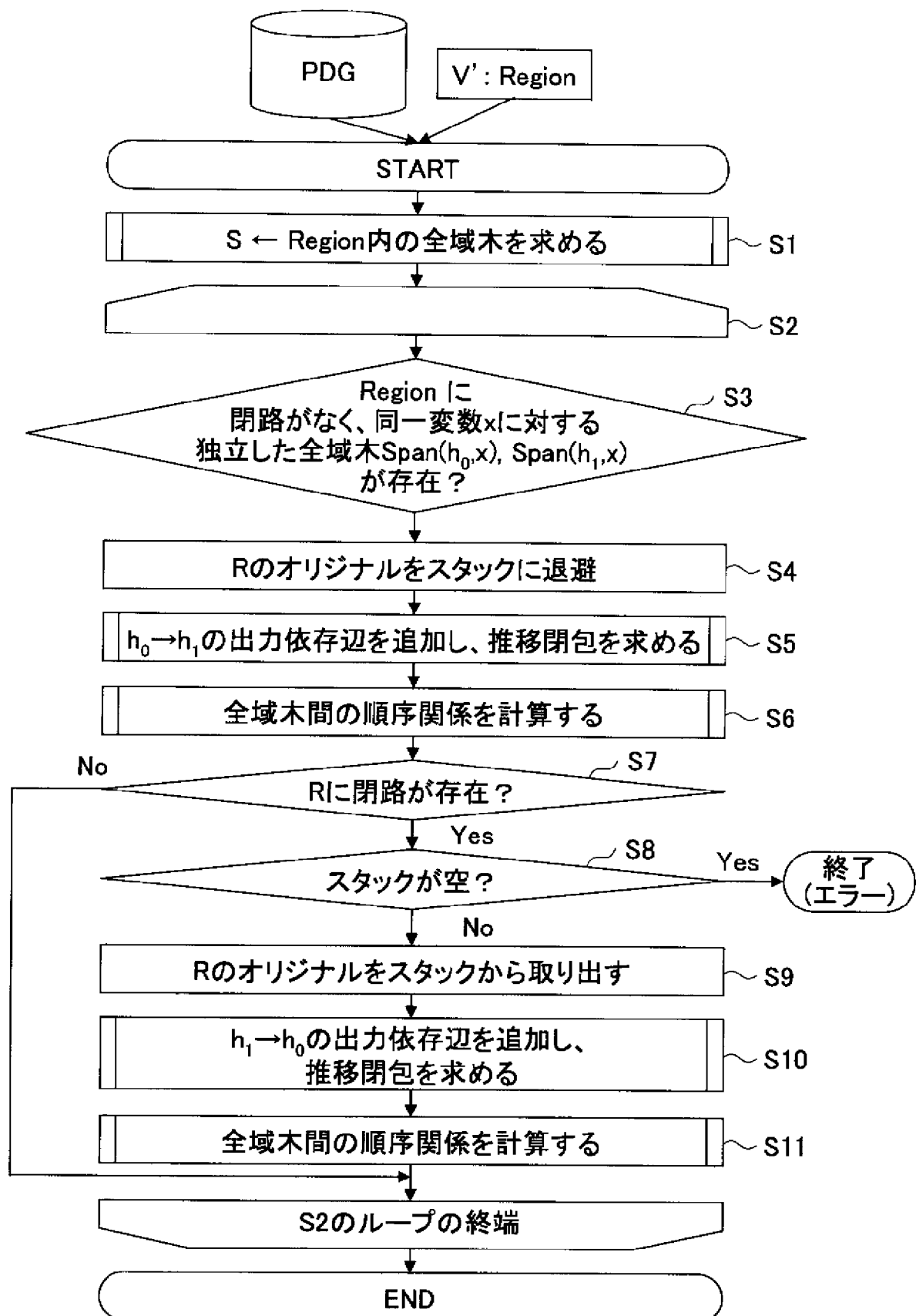
[図13]



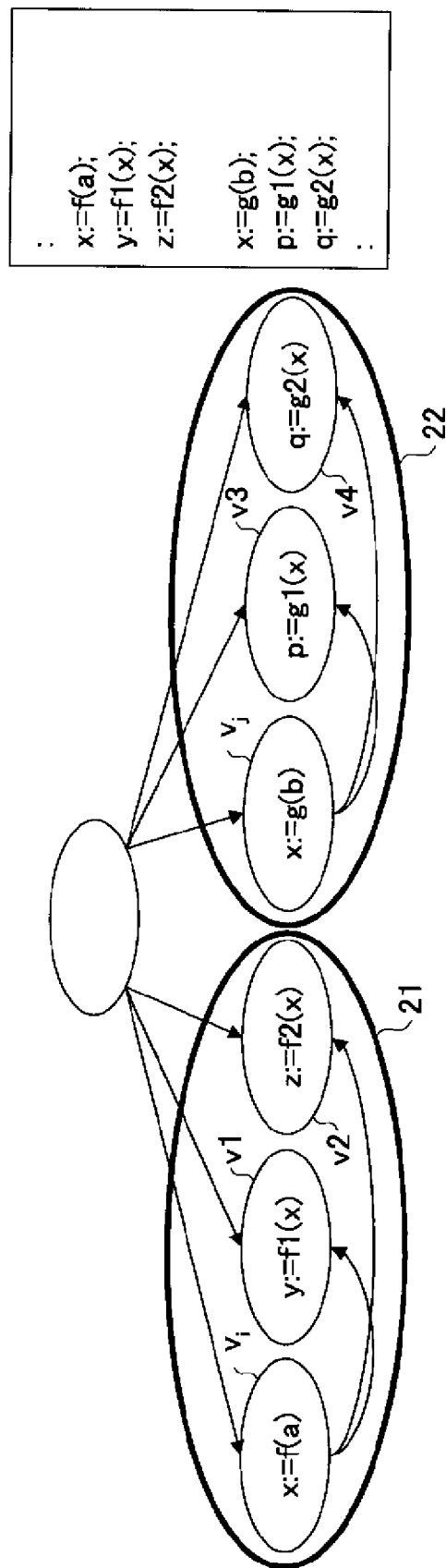
[図14]



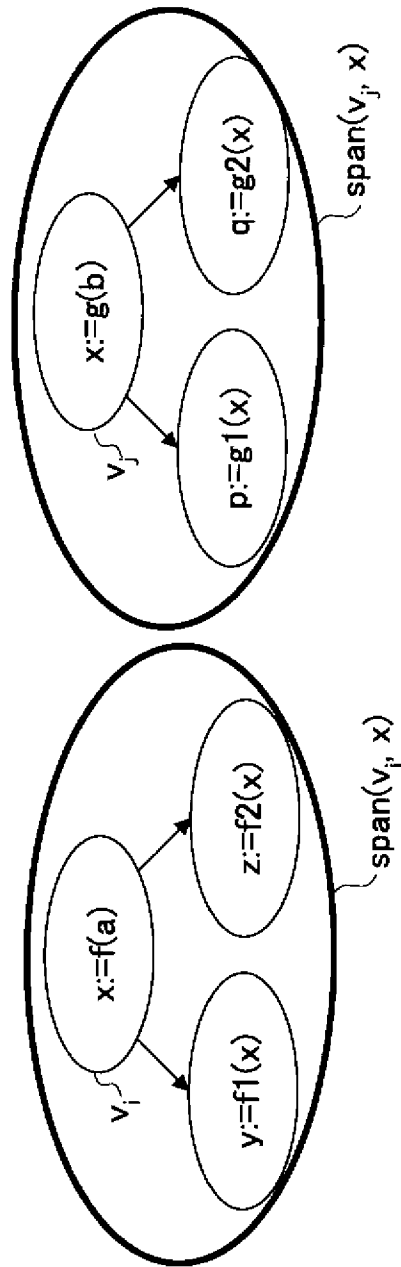
[図15]



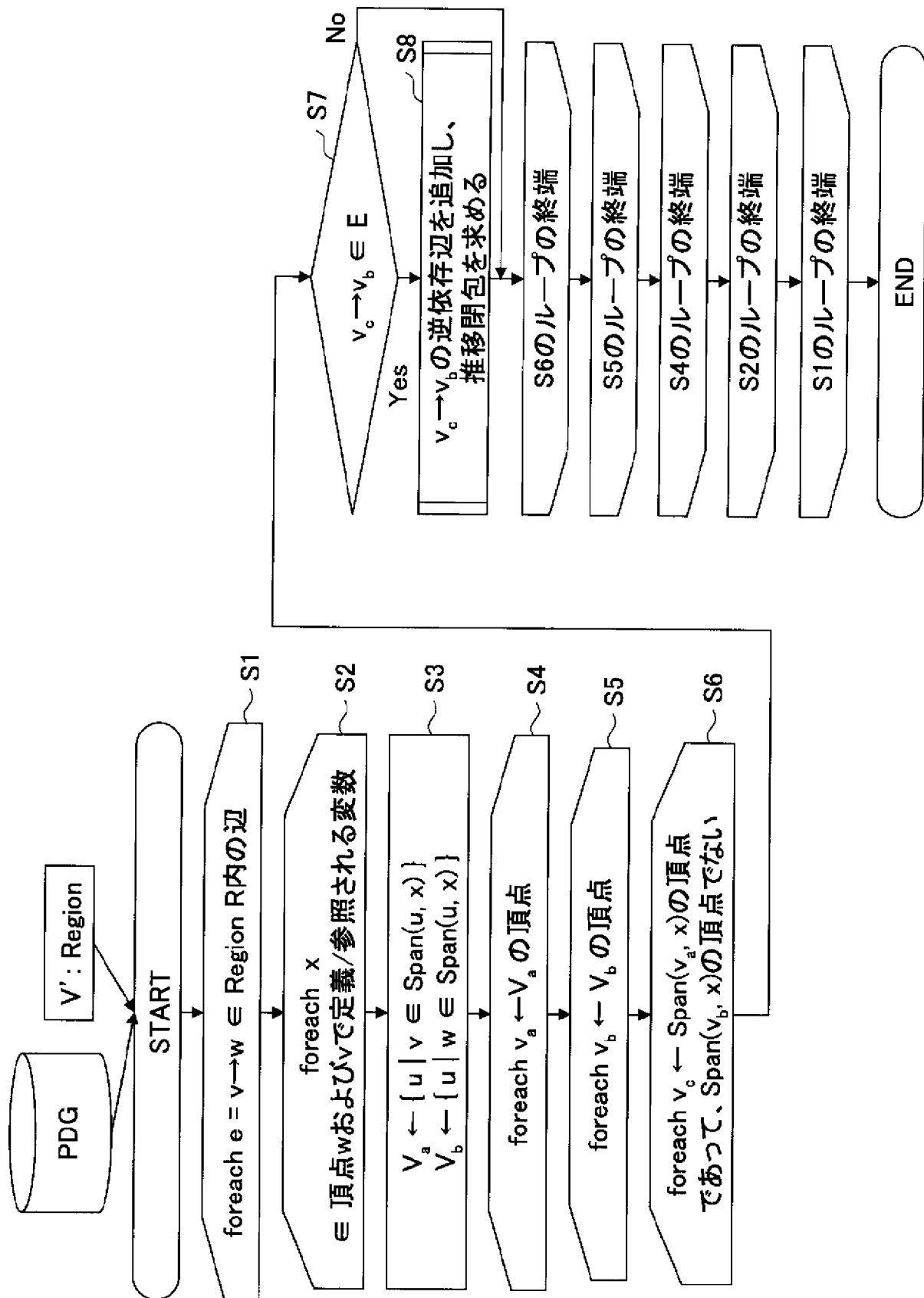
[図16]



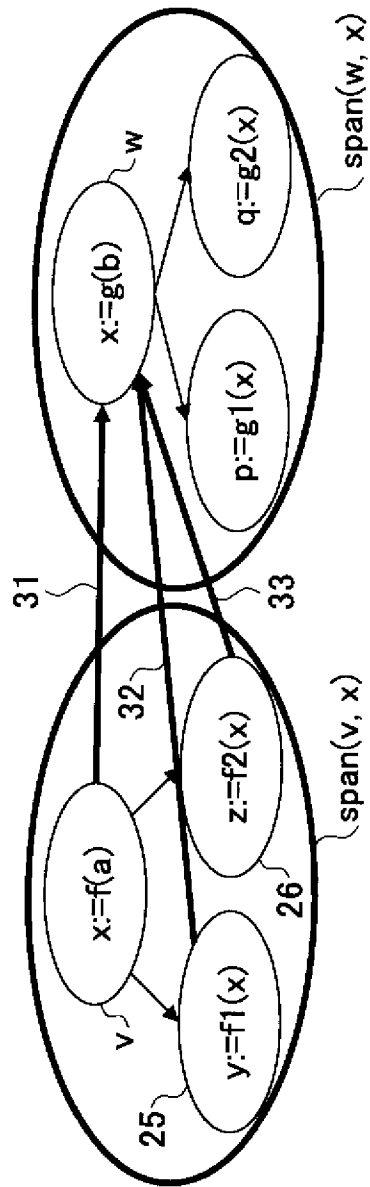
[図17]



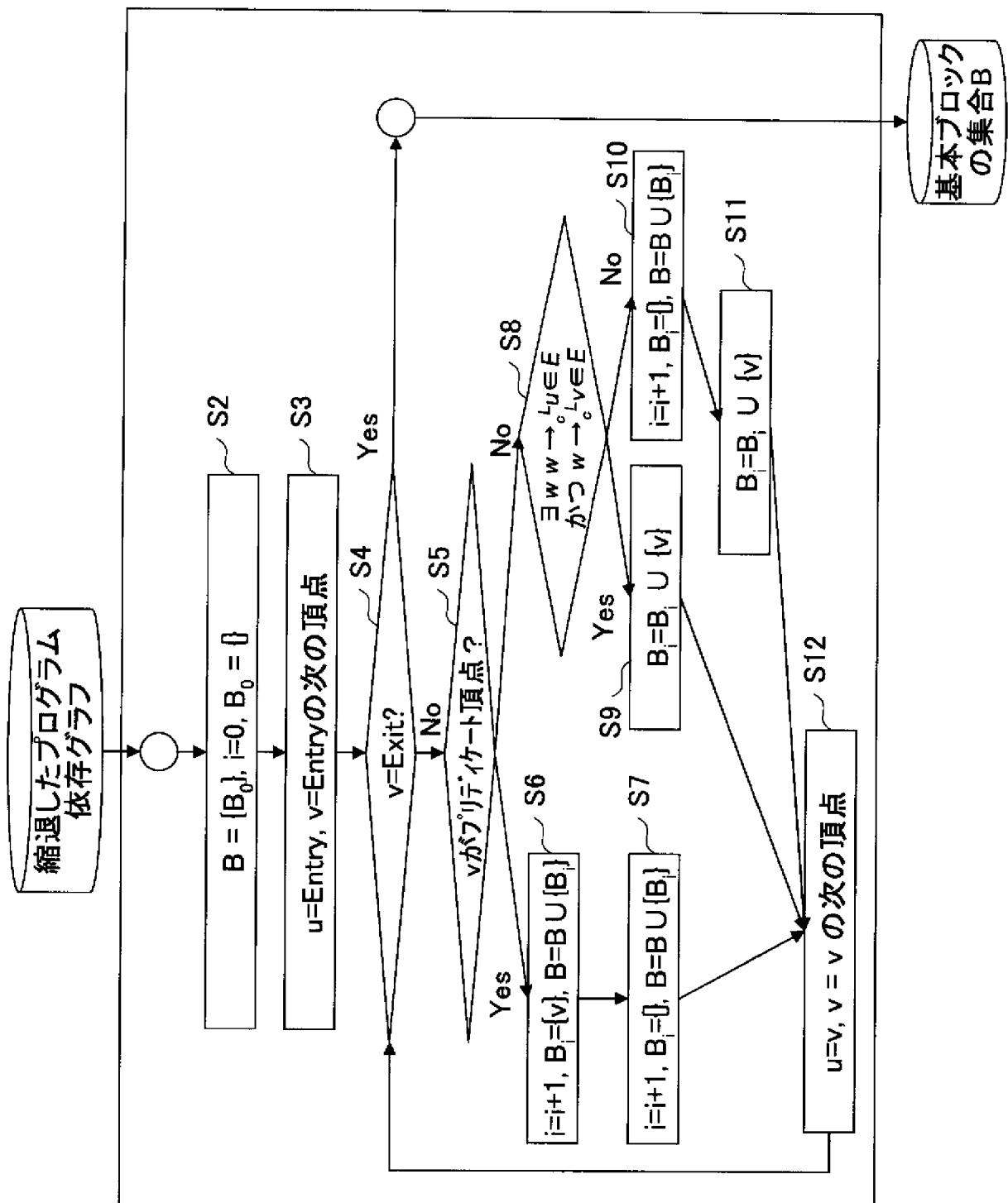
[図18]



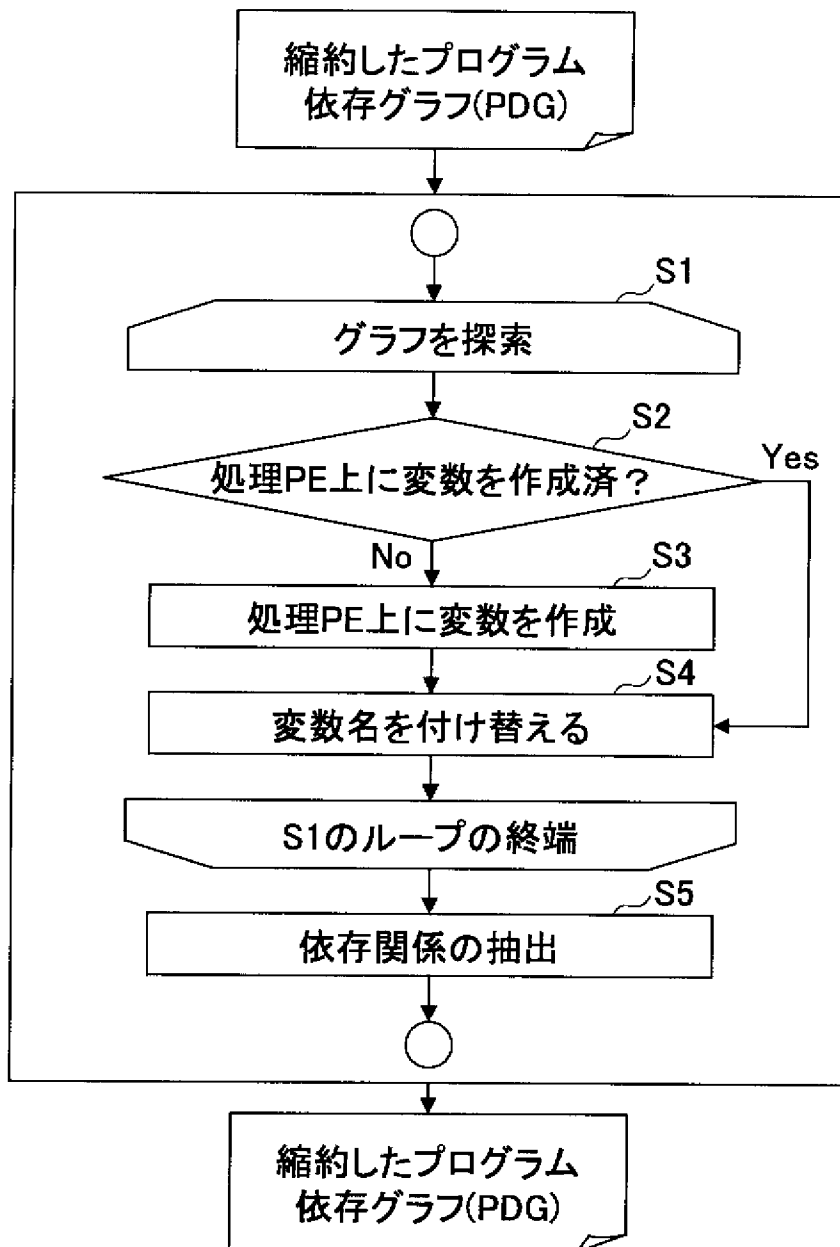
[図19]



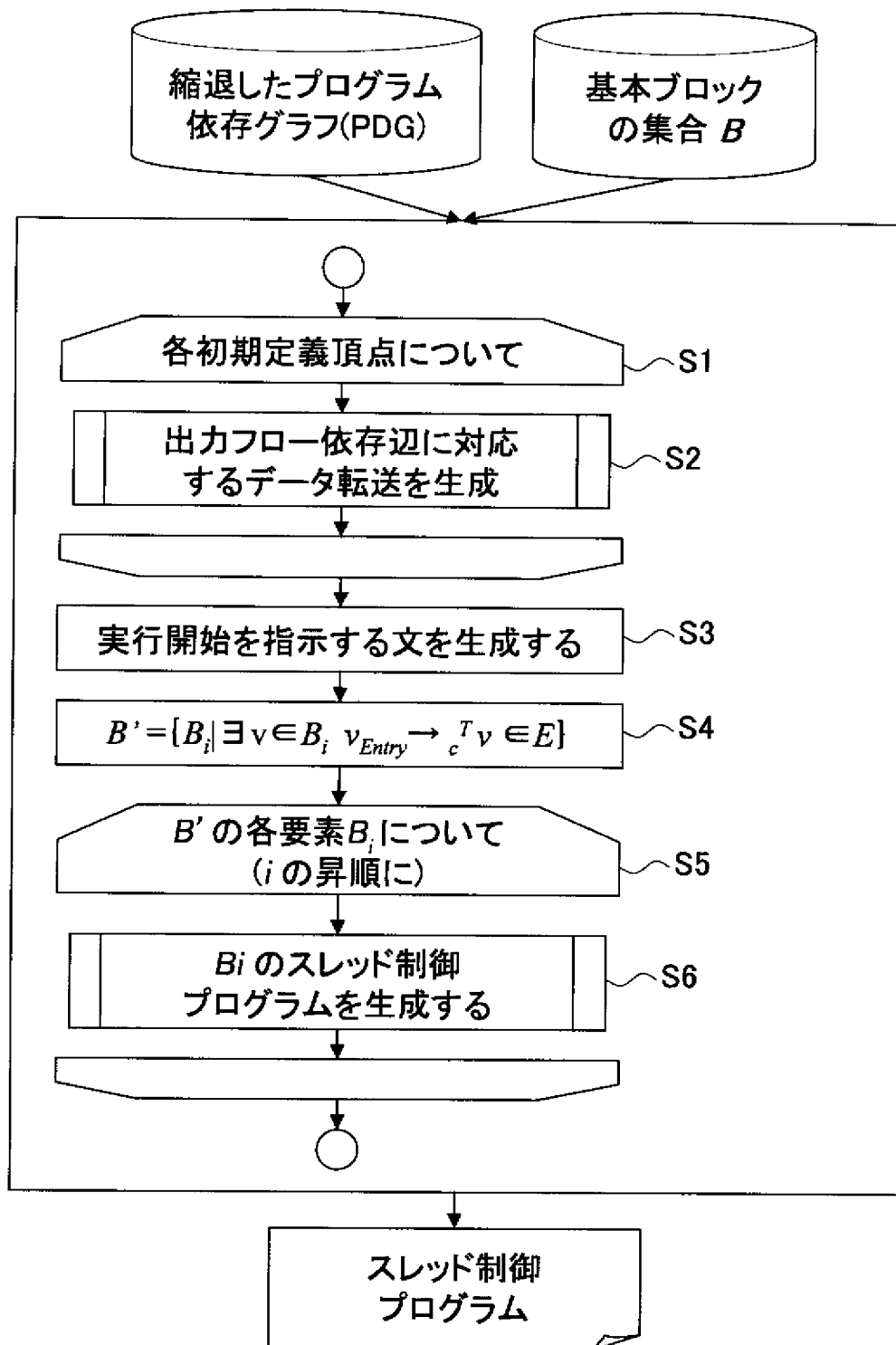
[図20]



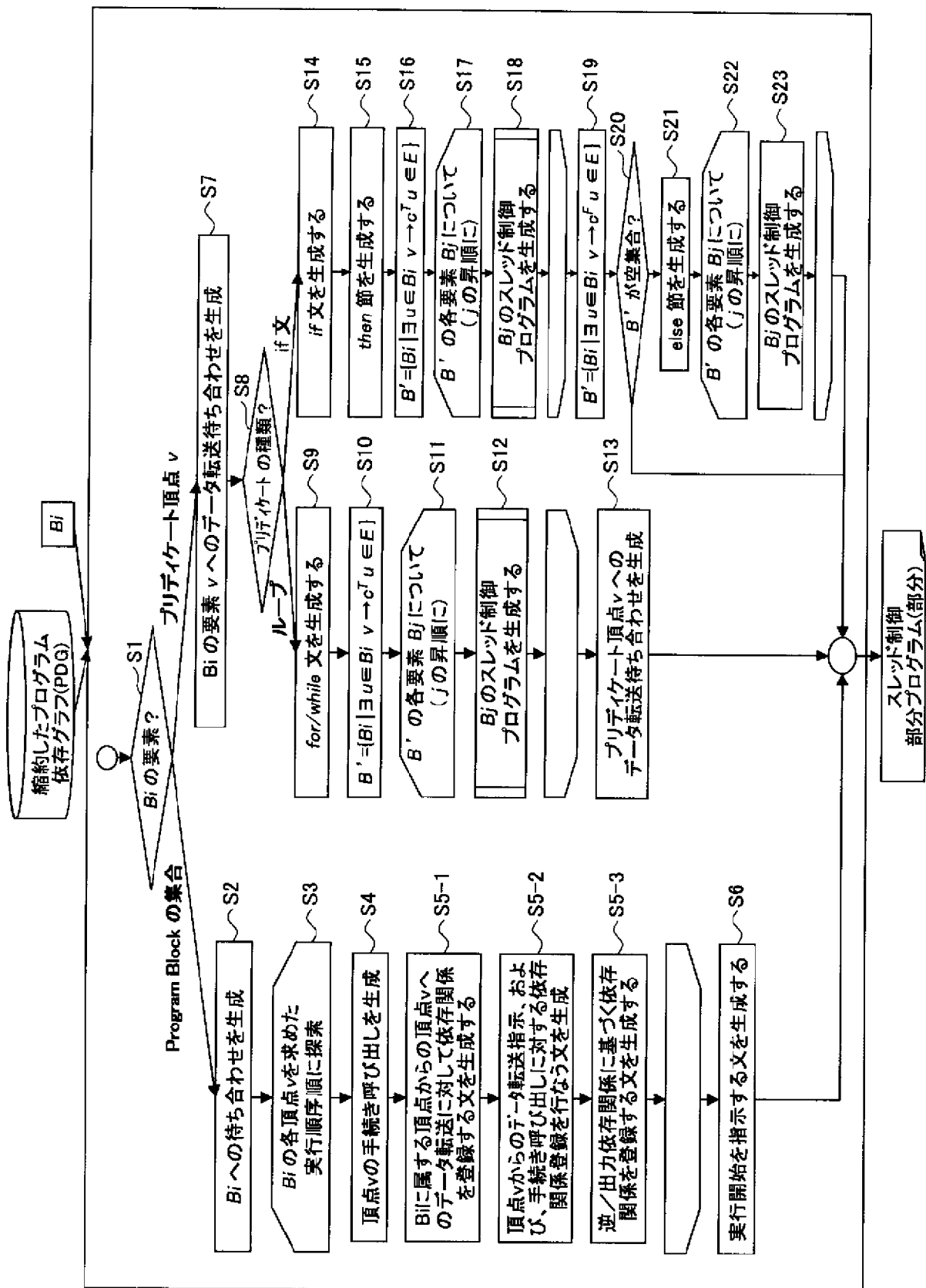
[図21]



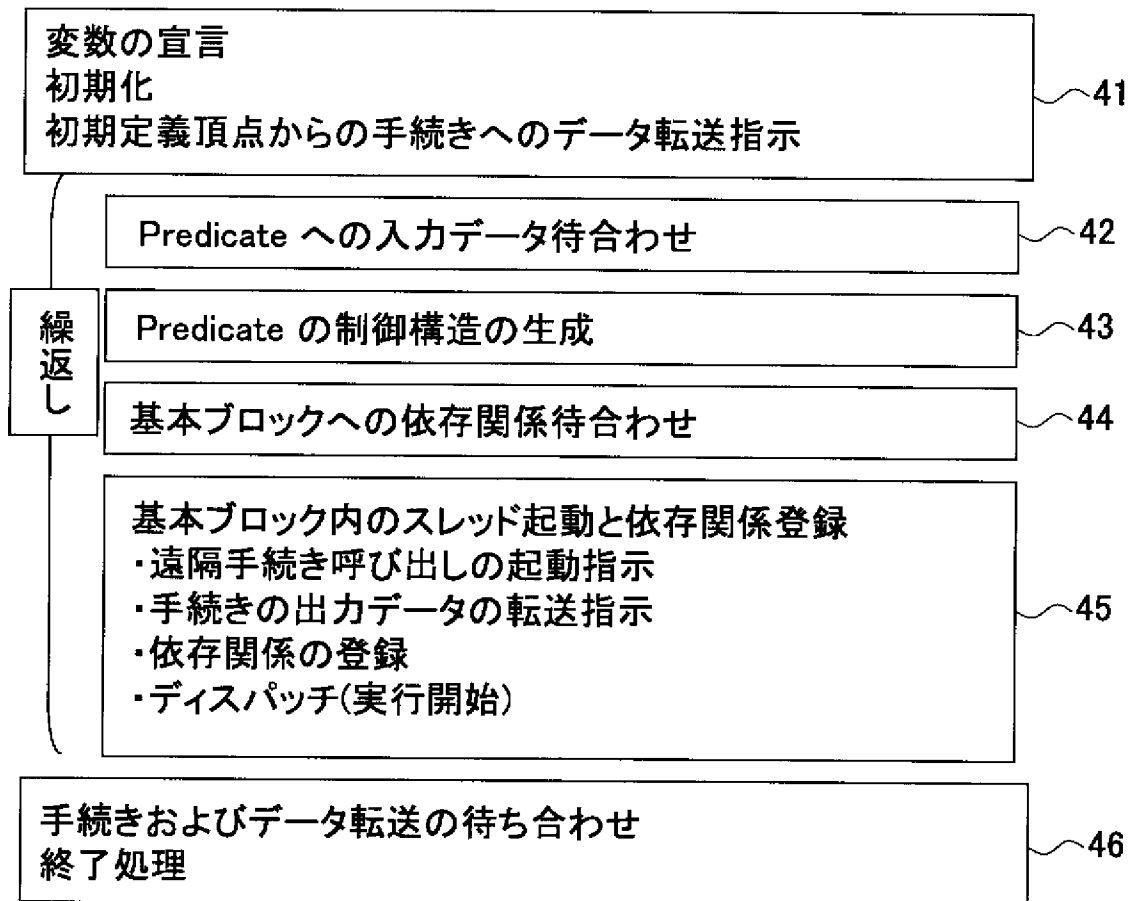
[図22]



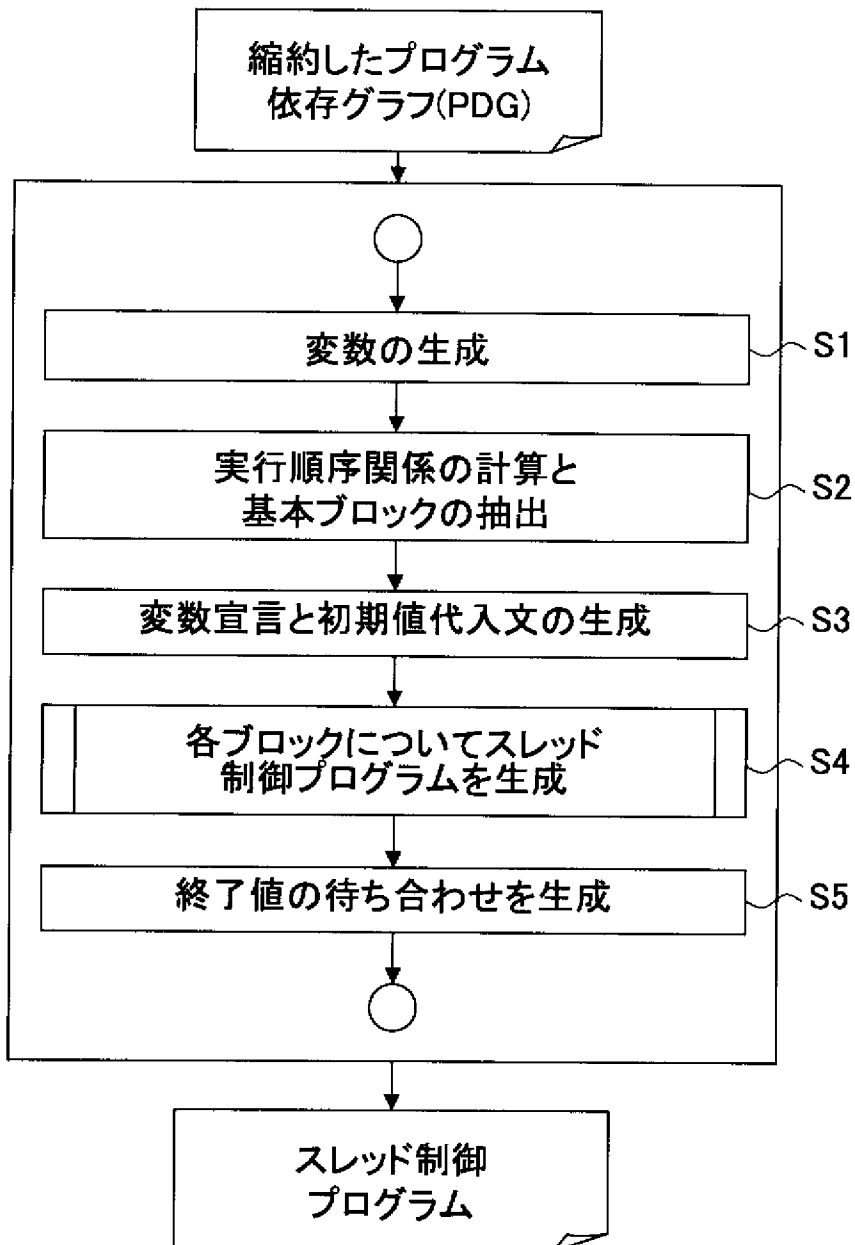
[図23]



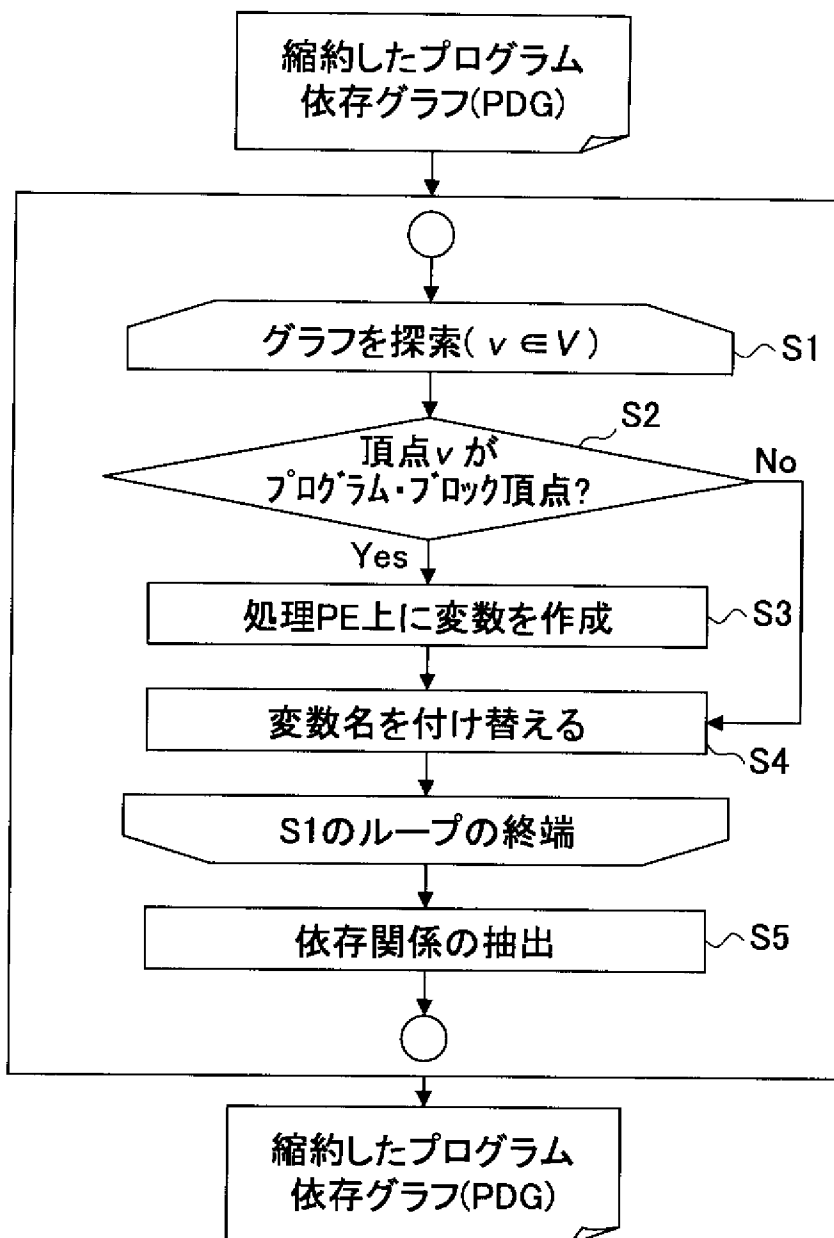
[図24]



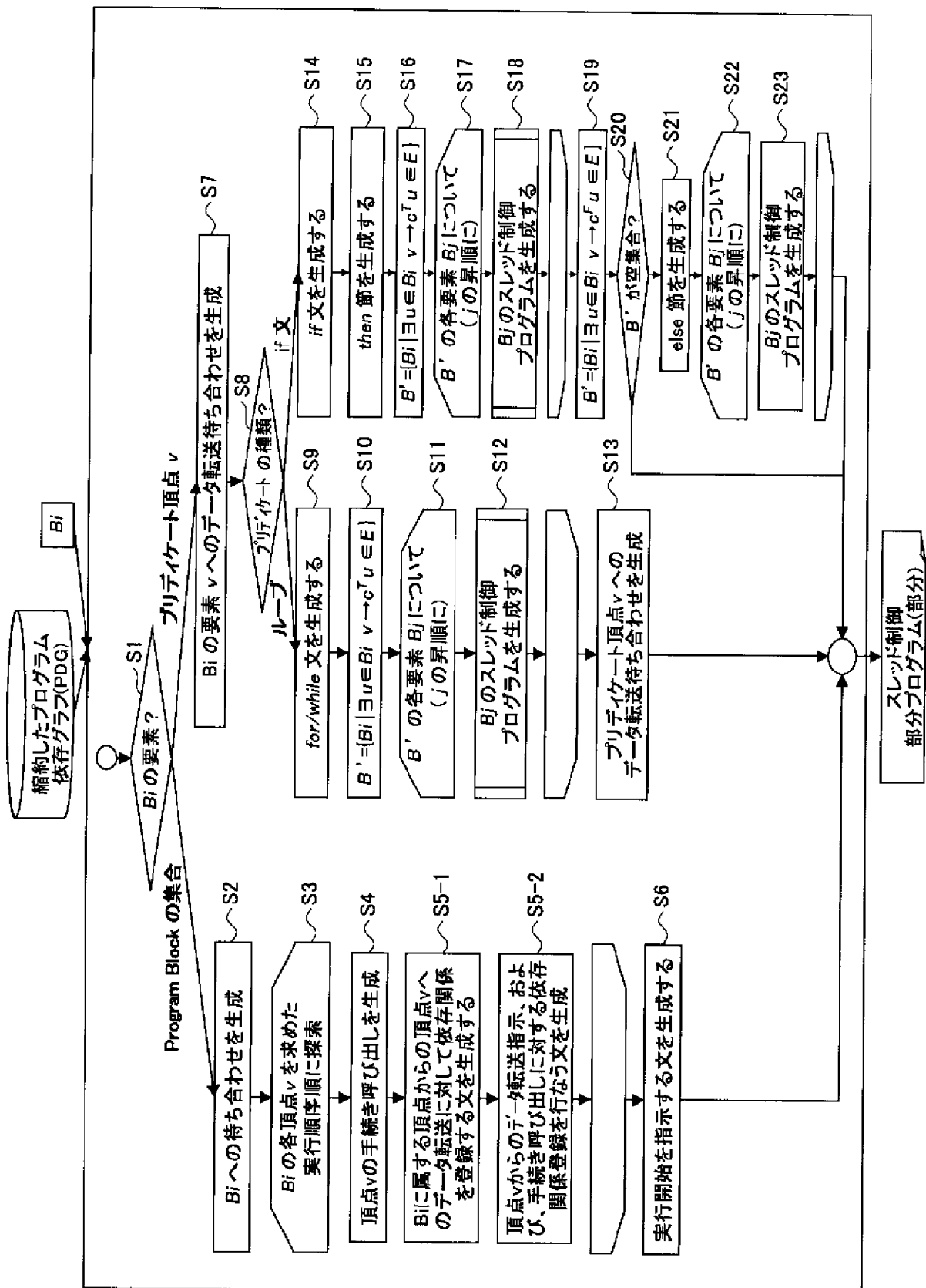
[図25]



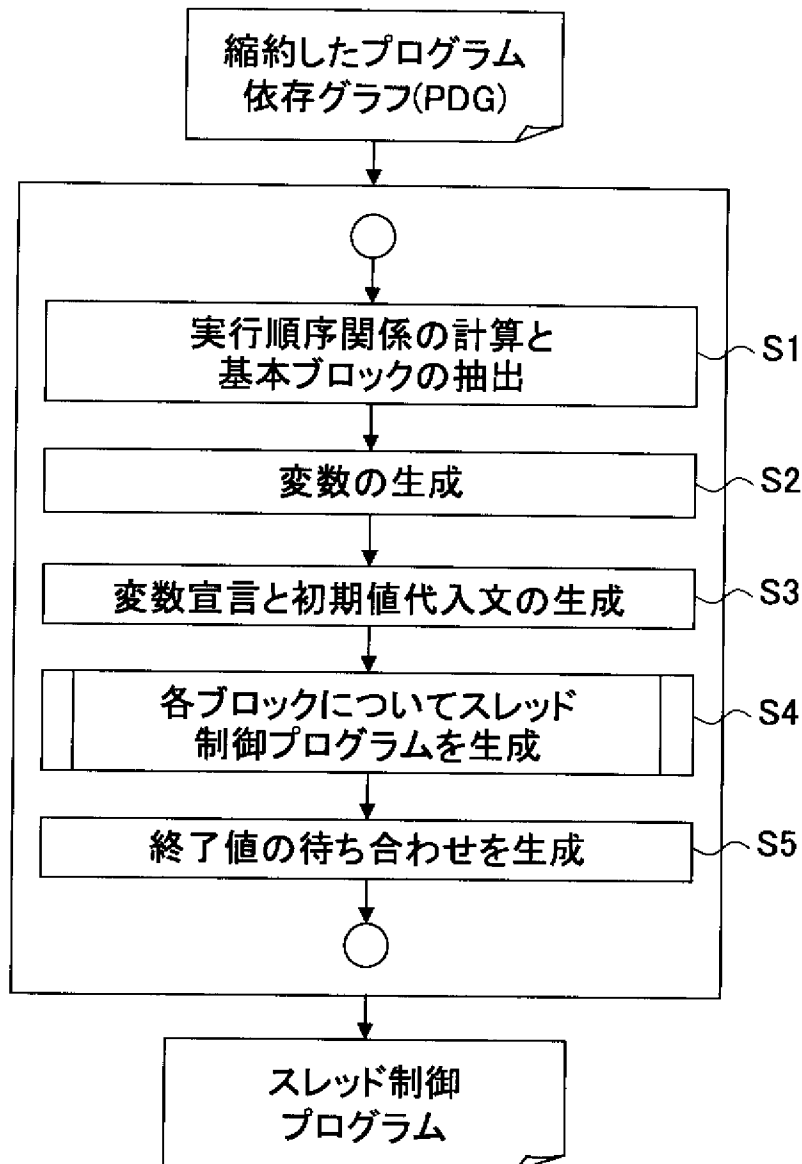
[図26]



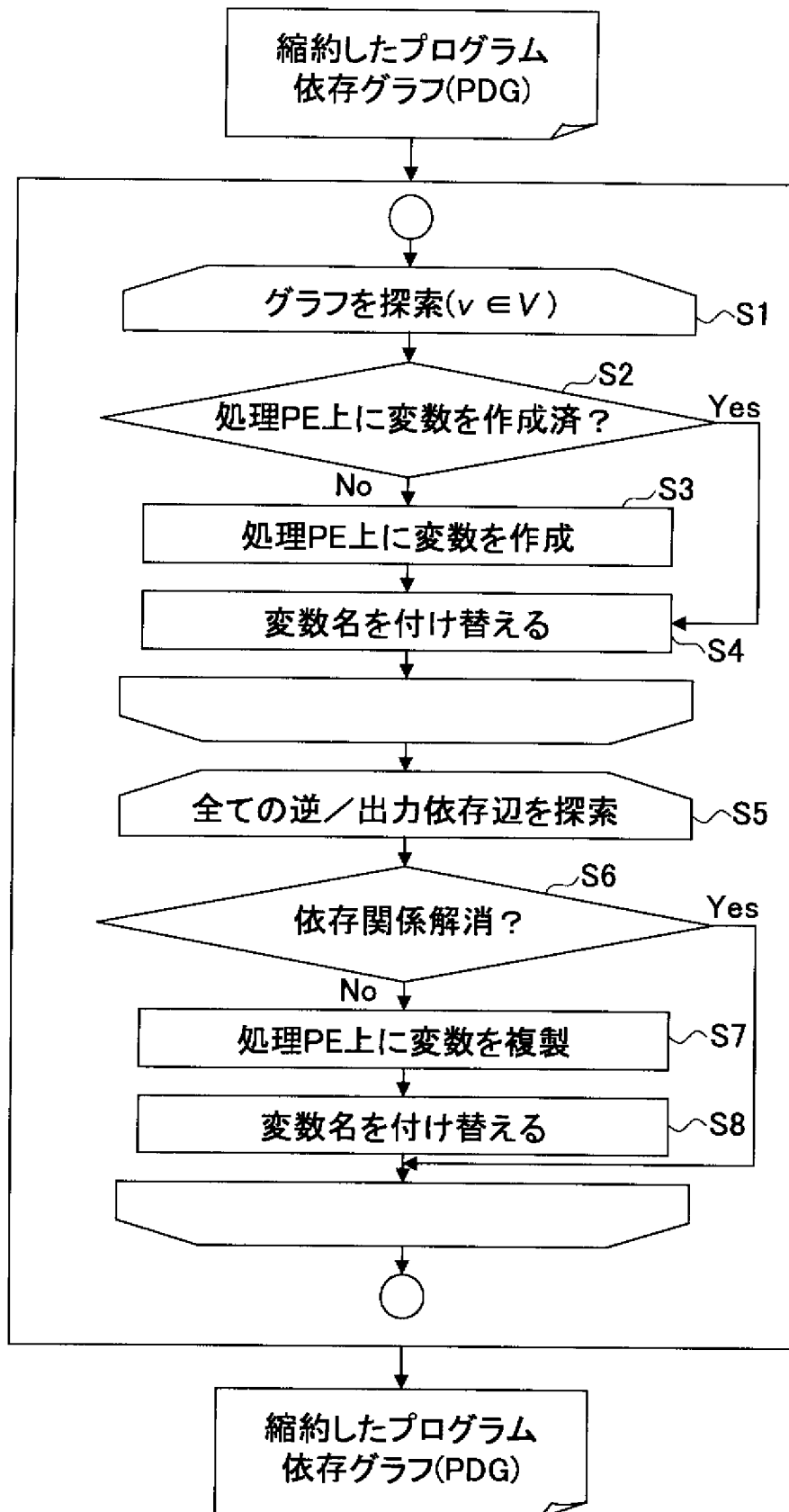
[図27]



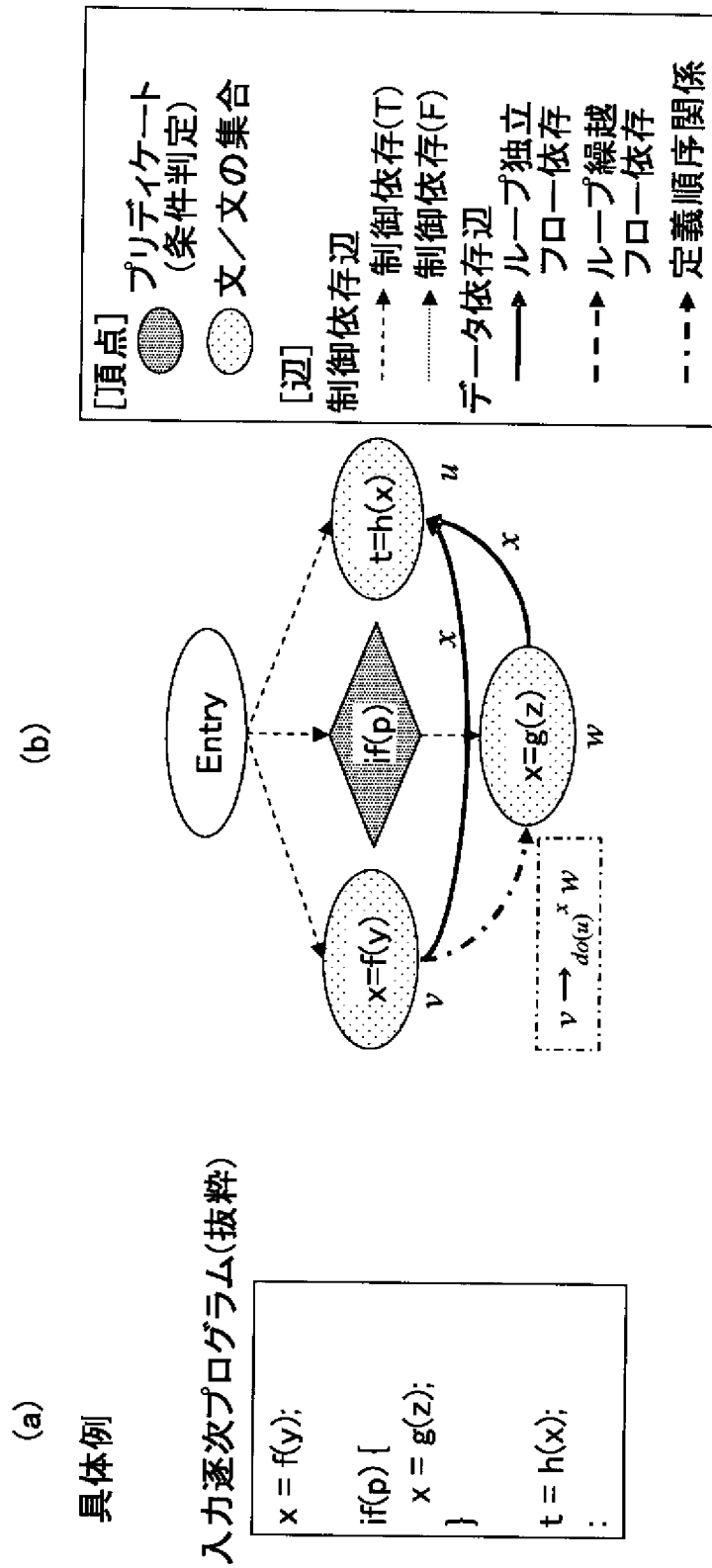
[図28]



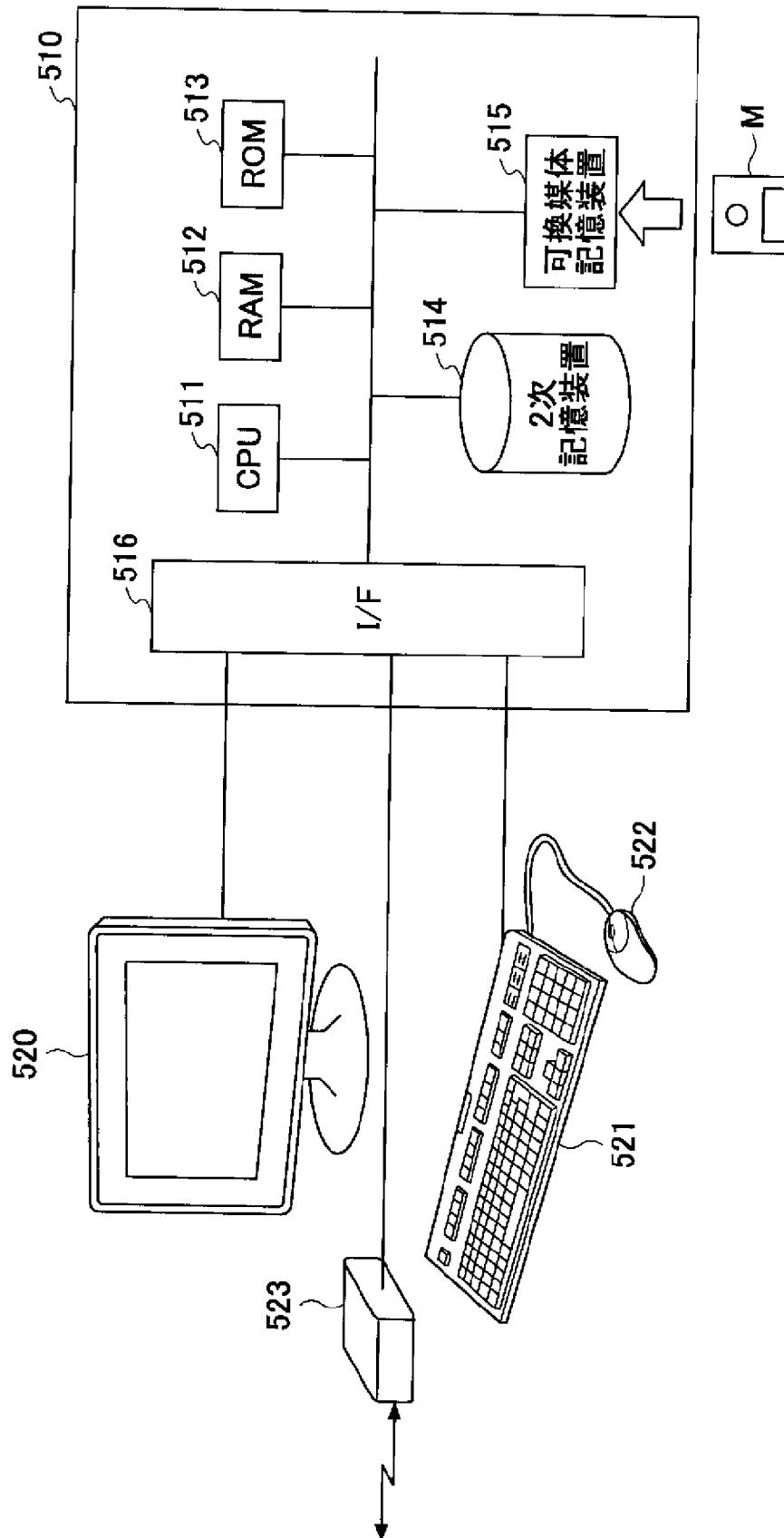
[図29]



[図30]



[図31]



INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2007/056916

A. CLASSIFICATION OF SUBJECT MATTER

G06F9/45 (2006.01) i

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, G06F9/48-9/54, G06F15/16-15/177

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 1996-2007

Kokai Jitsuyo Shinan Koho 1971-2007 Toroku Jitsuyo Shinan Koho 1994-2007

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

CSDB (Nipponkoku Tokkyocho), JSTPlus (JDream2)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Hiroki HONDA et al., 'OSCAR-jo deno Fortran Program Kihon Block no Heiretsu Shori Shuho', The Transactions of the Institute of Electronics, Information and Communication Engineers, 25 September, 1990 (25.09.90), Vol.J73-D-I, No.9, pages 756 to 766, ISSN:0915-1915, Page 758, left column, lines 8 to 26; page 759, left column, line 13 to page 759, right column, line 17; page 760, left column, lines 12 to 16; page 760, right column, line 40 to page 761, right column, line 10,	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
18 April, 2007 (18.04.07)Date of mailing of the international search report
01 May, 2007 (01.05.07)Name and mailing address of the ISA/
Japanese Patent Office

Authorized officer

Facsimile No.

Telephone No.

INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2007/056916

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Hiroki HONDA, 'OSCAR/Fortran Compiler', Information Processing Society of Japan Kenkyu Hokoku, 22 November, 1991 (22.11.91), Vol.91, No.100, pages 1 to 12, Page 6, left column, lines 32 to 37; page 7, left column, lines 20 to 27; page 8, left column, lines 1 to 13	1-10
A	Keiji KIMURA·et al., 'Single Chip Multi Processor-jo deno Kinsairyudo Heiretsu Shori', Transactions of Information Processing Society of Japan, 15 May, 1999 (15.05.99), Vol.40, No.5, pages 1924 to 1934, ISSN:0387-5806, Page 1925, left column, lines 34 to 44	1-10
A	Kiyoshi HAYAKAWA·et al., 'On-chip Multi Processor-jo no Sairyudo Task-kan Doki no tameno PC-C Doki Kiko', The Institute of Electronics, Information and Communication Engineers 1998 Nen Joho·System Society Taikai Koen Ronbunshu, 07 September, 1998 (07.09.98), page 34, Page 34, left column, liens 24 to 33	1-10
A	JP 2005-258920 A (Fujitsu Ltd.), 22 September, 2005 (22.09.05), Par. Nos. [0020], [0021], [0025] (Family: none)	1-10

A. 発明の属する分野の分類（国際特許分類（I P C））

Int.Cl. G06F9/45(2006.01)i

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（I P C））

Int.Cl. G06F9/45, G06F9/48-9/54, G06F15/16-15/177

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	1971-2007年
日本国実用新案登録公報	1996-2007年
日本国登録実用新案公報	1994-2007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 S D B（日本国特許庁）、J S T P l u s（J D r e a m 2）

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
A	本多弘樹・他、「O S C A R 上での F o r t r a n プログラム 基本ブロックの並列処理手法」、電子情報通信学会論文誌, 1990.09.25, 第J73-D-I巻, 第9号, p. 756-766, I S S N : 0915-1915, 第758頁左コラム第8-26行, 第759頁左コラム 第13行-第759頁右コラム第17行, 第760頁 左コラム第12-16行, 第760頁右コラム第40行- 第761頁右コラム第10行	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

18.04.2007

国際調査報告の発送日

01.05.2007

国際調査機関の名称及びあて先

日本国特許庁（I S A / J P）
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）

久保 光宏

5B

9189

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
A	本多弘樹, 「O S C A R / F o r t r a n コンパイラ」, 情報処理学会研究報告, 1991. 11. 22, 第91巻, 第100号, p. 1-12, 第6頁左コラム第32-37行, 第7頁左コラム第20-27行, 第8頁左コラム第1-13行	1-10
A	木村啓二・他, 「シングルチップマルチプロセッサ上での近細粒度 並列処理」, 情報処理学会論文誌, 1999. 05. 15, 第40巻, 第5号, p. 1924-1934, ISSN: 0387-5806, 第1925頁左コラム第34-44行	1-10
A	早川潔・他, 「O n - c h i p M u l t i P r o c e s s o r 上の細粒度タスク間同期のためのP C - C 同期機構」, 電子情報通信 学会1998年情報・システムソサイエティ大会講演論文集, 1998. 09. 07, p. 34, 第34頁左コラム第24-33行	1-10
A	J P 2005-258920 A (富士通株式会社) 2005. 09. 22, 段落 [0020], [0021], [0025] (ファミリーなし)	1-10