(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3956131号
(P3956131)
(45)発行日　平成19年8月8日(2007.8.8)　(24)登録日　平成19年5月18日(2007.5.18)

(51)Int.Cl.　F I
G06F 9/45 (2006.01)　G06F 9/44 322F

請求項の数 10 (全 14 頁)

(21)出願番号　特願2002-377992(P2002-377992)
(22)出願日　平成14年12月26日(2002.12.26)
(65)公開番号　特開2004-206625(P2004-206625A)
(43)公開日　平成16年7月22日(2004.7.22)
審査請求日　平成15年8月22日(2003.8.22)

(73)特許権者　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国１０５０４　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
(74)代理人　100086243
弁理士　坂口　博
(74)代理人　100091568
弁理士　市位　嘉宏
(74)代理人　100108501
弁理士　上野　剛史

最終頁に続く

(54)【発明の名称】プログラム変換装置、プログラム変換方法及びプログラム

(57)【特許請求の範囲】
【請求項１】
実行プログラムのソースコードを読み込み、機械語コードに変換するプログラム変換装置において、
コンピュータのプログラム制御されたＣＰＵにて実現され、前記ソースコードの字句解析及び構文解析を行うコード解析部と、
コンピュータのプログラム制御されたＣＰＵにて実現され、前記コード解析部により解析された前記実行プログラムに対し、当該実行プログラム中の手続き呼び出しにおける呼び側の手続きと呼ばれ側の手続きとを検出し、当該呼ばれ側の手続きにおいて引数の参照が実行される条件である参照条件または当該参照条件を包含する所定の条件を評価条件とし、当該評価条件が成立する場合に当該呼び側の手続きに記述されている当該引数の評価を実行するように前記実行プログラムを変形する最適化部と、
コンピュータのプログラム制御されたＣＰＵにて実現され、前記最適化部により変形された前記実行プログラムを前記機械語コードに変換するコード生成部と
を備えることを特徴とするプログラム変換装置。
【請求項２】
実行プログラムのソースコードを読み込み、機械語コードに変換するプログラム変換装置において、
コンピュータのプログラム制御されたＣＰＵにて実現され、前記ソースコードの字句解析及び構文解析を行うコード解析部と、

コンピュータのプログラム制御されたＣＰＵにて実現され、前記コード解析部により解析された前記実行プログラムに対し、当該実行プログラム中で引数の値渡しでの手続き呼び出しが行われる箇所を検出し、当該手続き呼び出しが行われる箇所における引数の評価と当該引数の参照が実行される条件である参照条件の評価の順序を入れ替え、当該参照条件により分岐したパスのうち引数の参照を行わないパスにおける引数評価を削除する最適化部と、

コンピュータのプログラム制御されたＣＰＵにて実現され、前記最適化部により変形された前記実行プログラムを前記機械語コードに変換するコード生成部と

を備えることを特徴とするプログラム変換装置。

【請求項３】

実行プログラムのソースコードを読み込み、機械語コードに変換するプログラム変換装置において、

コンピュータのプログラム制御されたＣＰＵにて実現され、前記ソースコードの字句解析及び構文解析を行うコード解析部と、

コンピュータのプログラム制御されたＣＰＵにて実現され、前記コード解析部により解析された前記実行プログラムに対し、所定の変形を施す最適化部と、

コンピュータのプログラム制御されたＣＰＵにて実現され、前記最適化部により変形された前記実行プログラムを前記機械語コードに変換するコード生成部とを備え、

前記最適化部は、

前記実行プログラム中の手続き呼び出しにおける呼び側の手続きと呼ばれ側の手続きとを検出し、当該呼ばれ側の手続きを前記呼び側の手続きにインライン化する第１の変形手段と、

インライン化された前記呼ばれ側の手続きの先頭から制御フローを遡って引数の評価を行う命令列を得、当該呼ばれ側の手続きにおいて引数の参照が実行される条件である参照条件を当該命令列の前に移動すると共に、当該命令列を複製して当該参照条件から分岐する各ルートに挿入する第２の変形手段と、

前記参照条件から分岐する各ルートに挿入された前記命令列のうち、前記引数が参照されないルートの不用な前記命令列を除去する第３の変形手段と

を備えることを特徴とするプログラム変換装置。

【請求項４】

コンピュータを制御して、処理対象であるプログラムの変形を行うプログラム変換方法であって、

コンピュータのプログラム制御されたＣＰＵにて実現される検出手段が、所定の記憶装置から前記処理対象であるプログラムを読み出し、当該プログラム中の手続き呼び出しにおける呼び側の手続きと呼ばれ側の手続きとを検出する第１のステップと、

コンピュータのプログラム制御されたＣＰＵにて実現される変形手段が、前記呼ばれ側の手続きにおいて引数の参照が実行される条件である参照条件または当該参照条件を包含する所定の条件を評価条件とし、当該評価条件が成立する場合に当該呼び側の手続きに記述されている当該引数の評価を実行するように前記プログラムを変形し、変形したプログラムを所定の記憶装置に格納する第２のステップと

を含むことを特徴とするプログラム変換方法。

【請求項５】

コンピュータを制御して、処理対象であるプログラムの変形を行うプログラム変換方法であって、

前記コンピュータのプログラム制御されたＣＰＵにて実現される検出手段が、所定の記憶装置から前記処理対象であるプログラムを読み出し、当該プログラム中で引数の値渡しでの手続き呼び出しが行われる箇所を検出するステップと、

前記コンピュータのプログラム制御されたＣＰＵにて実現される変形手段が、前記プログラムの前記手続き呼び出しが行われる箇所における引数の評価と当該引数の参照が実行される条件である参照条件の評価の順序を入れ替えるステップと、

前記変形手段が前記参照条件により分岐したパスのうち引数の参照を行わないパスにおける引数評価を削除し、かかる変形がなされたプログラムを所定の記憶装置に格納するステップと
を含むことを特徴とするプログラム変換方法。
【請求項６】
コンピュータを制御して、処理対象であるプログラムの変形を行うプログラム変換方法であって、
前記コンピュータのプログラム制御されたＣＰＵにて実現される検出手段が、所定の記憶装置から前記処理対象であるプログラムを読み出し、当該プログラム中の手続き呼び出しにおける呼び側の手続きと呼ばれ側の手続きとを検出する第１のステップと、
前記コンピュータのプログラム制御されたＣＰＵにて実現されるインライン化手段が、前記呼ばれ側の手続きを前記呼び側の手続きにインライン化する第２のステップと、
前記コンピュータのプログラム制御されたＣＰＵにて実現される変形手段が、インライン化された前記呼ばれ側の手続きの先頭から制御フローを遡って引数の評価を行う命令列を得、当該呼ばれ側の手続きにおいて引数の参照が実行される条件である参照条件を当該命令列の前に移動すると共に、当該命令列を複製して当該参照条件から分岐する各ルートに挿入する第３のステップと、
前記変形手段が、前記参照条件から分岐する各ルートに挿入された前記命令列のうち、前記引数が参照されないルートの不用な前記命令列を除去する第４のステップと、
前記不用な命令列が除去されたプログラムを所定の記憶装置に格納する第５のステップと
を含むことを特徴とするプログラム変換方法。
【請求項７】
コンピュータを制御して、実行プログラムのソースコードを機械語コードに変換するプログラムであって、
所定の記憶装置から処理対象である前記実行プログラムのソースコードを読み出し、当該実行プログラム中の手続き呼び出しにおける呼び側の手続きと呼ばれ側の手続きとを検出する第１の処理を前記コンピュータのプログラム制御されたＣＰＵにて実現される検出手段に実行させ、
前記呼ばれ側の手続きにおいて引数の参照が実行される条件である参照条件または当該参照条件を包含する所定の条件を評価条件とし、当該評価条件が成立する場合に当該呼び側の手続きに記述されている当該引数の評価を実行するように前記実行プログラムを変形する第２の処理を前記コンピュータのプログラム制御されたＣＰＵにて実現される変形手段に実行させ、
変形された前記実行プログラムを前記機械語コードに変換し、所定の記憶装置に格納する第３の処理を前記変形手段に実行させることを特徴とするプログラム。
【請求項８】
コンピュータを制御して、実行プログラムのソースコードを機械語コードに変換するプログラムであって、
所定の記憶装置から処理対象である前記実行プログラムのソースコードを読み出し、当該実行プログラム中で引数の値渡しでの手続き呼び出しが行われる箇所を検出する処理を前記コンピュータのプログラム制御されたＣＰＵにて実現される検出手段に実行させ、
前記手続き呼び出しが行われる箇所における引数の評価と当該引数の参照が実行される条件である参照条件の評価の順序を入れ替える処理を前記コンピュータのプログラム制御されたＣＰＵにて実現される変形手段に実行させ、
前記参照条件により分岐したパスのうち引数の参照を行わないパスにおける引数評価を削除し、かかる変形がなされた前記実行プログラムを変形する処理を前記変形手段に実行させ、
変形された前記実行プログラムを前記機械語コードに変換し、所定の記憶装置に格納する処理を前記変形手段に実行させることを特徴とするプログラム。

【請求項９】
コンピュータを制御して、実行プログラムのソースコードを機械語コードに変換するプログラムであって、
所定の記憶装置から処理対象である前記実行プログラムのソースコードを読み出し、当該実行プログラム中の手続き呼び出しにおける呼び側の手続きと呼ばれ側の手続きとを検出する第１の処理を前記コンピュータのプログラム制御されたＣＰＵにて実現される検出手段に実行させ、
前記呼ばれ側の手続きを前記呼び側の手続きにインライン化する第２処理を前記コンピュータのプログラム制御されたＣＰＵにて実現されるインライン化手段に実行させ、
インライン化された前記呼ばれ側の手続きの先頭から制御フローを遡って引数の評価を行う命令列を得、当該呼ばれ側の手続きにおいて引数の参照が実行される条件である参照条件を当該命令列の前に移動すると共に、当該命令列を複製して当該参照条件から分岐する各ルートに挿入する第３の処理を前記コンピュータのプログラム制御されたＣＰＵにて実現される変形手段に実行させ、
前記参照条件から分岐する各ルートに挿入された前記命令列のうち、前記引数が参照されないルートの不用な前記命令列を除去する第４の処理を前記コンピュータのプログラム制御されたＣＰＵにて実現される変形手段に実行させ、
前記不用な命令列が除去された前記実行プログラムを前記機械語コードに変換し、所定の記憶装置に格納する第５の処理を前記コンピュータのプログラム制御されたＣＰＵにて実現される検出手段に実行させることを特徴とするプログラム。
【請求項１０】
前記第３の処理では、前記参照条件を、当該参照条件を包含する他の条件に置き換える処理を前記変形手段にさらに実行させることを特徴とする請求項９に記載のプログラム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、コンピュータプログラムのコンパイルにおいて実行される最適化処理に関する。
【０００２】
【従来の技術】
コンピュータによるデータ処理を制御するプログラムを記述するプログラミング言語のうち、引数を値渡しする言語、すなわち殆どの手続き型言語では、手続き呼び出しにおける呼び側の手続き（以下、単に呼び側と称す）が、手続き呼び出しに先立ち全ての引数を評価する（これを先行評価（eager evaluation）という）。
【０００３】
引数の先行評価には、引数評価を呼び側本体の計算と並列に実行することで、当該引数評価を行うことによる遅延時間を、手続き呼び出しの遅延時間と重複させて隠せる（実質的に遅延を低減させられる）という利点がある。その反面、手続き呼び出しの呼ばれ側の手続き（以下、単に呼ばれ側と称す）で実際に参照されない引数まで評価されてしまう無駄がある。
【０００４】
そこで、プログラムの開発においては、引数の先行評価を行いながら、その無駄を省くための方策が求められる。
Ｓｃｈｅｍｅという言語では、引数の評価は通常、先行評価で行うが、約束（promise）と呼ばれる閉包（closure）として呼ばれ側に渡し、遅延評価（lazy evaluation）に切り替えることができる仕様となっている（例えば、非特許文献１参照）。
【０００５】
また、プログラマがプログラムの作成時に、引数が呼ばれ側で実際に参照されるための条件（以下、参照条件と称す）から導かれる適切な条件（以下、評価条件と称す）を呼び側に挿入することにより、呼ばれ側で引数を参照するか否かに応じて適切に先行評価を行う

ことができる。そして自ら呼び側のソースコードに評価条件を挿入し、手続き呼び出しそのものを保護することができる。

【０００６】

さらに、従来から用いられているプログラムの最適化の手法として、部分不用コード除去（partial dead code elimination）がある。この最適化は、後続制御フローの一部分でのみ値が参照される命令を、値を参照する命令の直前に移動し、その部分不用性を除く（例えば、非特許文献２参照）。

【０００７】

【非特許文献１】

R. Kelsey, W. Clinger, J. Rees (eds.), "Revised^5 Report on the Algorithmic Language Scheme", Higher-Order and Symbolic Computation, Vol. 11, No. 1, September, 1998, and ACM SIGPLAN Notices, Vol. 33, No. 9, October, 1998.

【非特許文献２】

J. Knoop, O. Ruthing and B. Steffen, "Partial Dead Code Elimination", In PLDI '94, pp. 147-158, June 1994.

【非特許文献３】

Steven S. Muchnick, "Advanced Compiler Design and Implementation", Morgan Kaufmann Publishers, Inc., 1997, pp. 465 - 470.

【非特許文献４】

Steven S. Muchnick, "Advanced Compiler Design and Implementation", Morgan Kaufmann Publishers, Inc., 1997, pp. 592 - 597.

【０００８】

【発明が解決しようとする課題】

しかしながら、上述した従来の各手法では、それぞれ次のような問題がある。Ｓｃｈｅｍｅにおける約束（promise）は、呼び側にdelay、呼ばれ側にforceをつけることによって、引数の評価を先行評価から遅延評価に切り替える方式である。このため、呼び側、呼ばれ側双方のソースコードの変更を伴い煩雑である。また当然ながら、遅延評価に切り替えると、並列実行により引数評価の遅延時間を隠し実質的に遅延を低減できるという先行評価の利点は失われる。

【０００９】

また、プログラミングにおいて評価条件を呼び側のソースコードへ挿入する手法は、本来呼ばれ側に集約されていた参照条件を多数の呼び側に分散させることである。すなわち、参照条件に基づく評価条件を、その呼ばれ側の手続きを呼び出す全ての呼び側に挿入しなくてはならない。そのため、プログラムの変更や保守を行う場合の容易さを損なう。また、呼ばれ側の詳細は必ずしも公開されているとは限らず、仮に公開されていたとしても、参照条件が呼び側から呼び出せるパブリック関数やパブリックメンバであるとは限らない。そのため、呼び側のソースコードに評価条件を挿入することが、原理上不可能な場合もあり得る。

【００１０】

さらに、部分不用コード除去は、任意の不用コードを除去できる強力な最適化であるが、計算量が命令数の３から５乗と非常に大きい。したがって、Ｊａｖａ（米国サン・マイクロシステムズ社の商標）におけるＪＩＴ（Just In Time）コンパイラのようにプログラムの実行時に動的にコンパイルを行うコンパイラに用いるには適当ではない。

【００１１】

そこで、本発明は、引数の先行評価を、呼ばれ側で必要とするか否かに応じて実行する効率的なプログラムを生成することを可能とすることを目的とする。

また本発明は、プログラムのコンパイルの際に、そのような効率的なプログラムに最適化するコンパイラを提供することを目的とする。

【００１２】

【課題を解決するための手段】

上記の目的を達成する本発明は、実行プログラムのソースコードを読み込み、機械語コードに変換する、次のように構成されたプログラム変換装置として実現される。すなわち、このプログラム変換装置は、ソースコードの字句解析及び構文解析を行うコード解析部と、この実行プログラムを変形する最適化部と、この最適化部により変形された実行プログラムを機械語コードに変換するコード生成部とを備える。そして、この最適化部は、コード解析部により解析された実行プログラムに対し、この実行プログラム中の手続き呼び出しにおける呼び側の手続きと呼ばれ側の手続きとを検出し、呼び側の手続きに記述されている引数の評価を、所定の評価条件で保護し、この引数を参照する場合に評価を行うように実行プログラムを変形することを特徴とする。

この評価条件としては、呼ばれ側の手続きが単純な条件で保護されている場合における、かかる条件または当該条件を包含する条件とすることができる。あるいはこの条件は、呼ばれ側の手続きにて前記引数が参照されるための参照条件または当該条件を包含する条件とすることができる。

【００１３】

また、本発明の他のプログラム変換装置は、コード解析部と、最適化部と、コード生成部とを備え、最適化部は、コード解析部により解析された実行プログラムに対し、この実行プログラム中で引数の値渡しでの手続き呼び出しが行われる箇所を検出し、この手続き呼び出しが行われる箇所における引数評価と参照条件の評価との順序を入れ替えるように実行プログラムを変形することを特徴とする。

【００１４】

本発明のさらに他のプログラム変換装置は、コード解析部と、最適化部と、コード生成部とを備え、最適化部は、実行プログラム中の手続き呼び出しにおける呼び側の手続きと呼ばれ側の手続きとを検出し、呼ばれ側の手続きを呼び側の手続きにインライン化する第１の変形手段と、インライン化された呼ばれ側の手続きの先頭から制御フローを遡って所定の命令列を得、呼ばれ側の手続きにて引数が参照されるための参照条件を、この命令列の前に移動すると共に、この命令列を複製して参照条件から分岐する各ルートに挿入する第２の変形手段と、参照条件から分岐する各ルートに挿入された命令列のうち、引数が参照されないルートの不用な命令列を除去する第３の変形手段とを備えることを特徴とする。

【００１５】

また、上記の目的を達成する他の本発明は、コンピュータを制御して、処理対象であるプログラムの変形を行うプログラム変換方法であって、所定の記憶装置から処理対象であるプログラムを読み出し、このプログラム中の手続き呼び出しにおける呼び側の手続きと呼ばれ側の手続きとを検出する第１のステップと、呼び側の手続きに記述されている引数の評価を、所定の評価条件で保護し、この引数を参照する場合に評価を行うようにプログラムを変形し、変形したプログラムを所定の記憶装置に格納する第２のステップとを含むことを特徴とする。

【００１６】

さらに、本発明の他のプログラム変換方法は、所定の記憶装置から処理対象であるプログラムを読み出し、このプログラム中で引数の値渡しでの手続き呼び出しが行われる箇所を検出する第１のステップと、このプログラムの手続き呼び出しが行われる箇所における引数評価と参照条件の評価との順序を入れ替えるように、プログラムを変形し、変形したプログラムを所定の記憶装置に格納するステップとを含むことを特徴とする。

【００１７】

また、本発明のさらに他のプログラム変換方法は、所定の記憶装置から処理対象であるプログラムを読み出し、このプログラム中の手続き呼び出しにおける呼び側の手続きと呼ばれ側の手続きとを検出する第１のステップと、呼ばれ側の手続きを呼び側の手続きにインライン化する第２のステップと、インライン化された呼ばれ側の手続きの先頭から制御フローを遡って所定の命令列を得、呼ばれ側の手続きにて引数が参照されるための参照条件を、この命令列の前に移動すると共に、この命令列を複製して参照条件から分岐する各ルートに挿入する第３のステップと、参照条件から分岐する各ルートに挿入された命令列の

うち、この引数が参照されないルートの不用な命令列を除去する第４のステップと、不用な命令列が除去されたプログラムを所定の記憶装置に格納する第５のステップを含むことを特徴とする。

【００１８】

さらにまた、本発明は、コンピュータを制御して上述したプログラム変換装置としての各機能を実現するプログラム、あるいはコンピュータに上記のプログラム変換方法における各ステップに対応する処理を実行させるプログラムとしても実現される。このプログラムは、磁気ディスクや光ディスク、半導体メモリ、その他の記録媒体に格納して配布したり、ネットワークを介して配信したりすることにより提供することができる。

【００１９】

【発明の実施の形態】

以下、添付図面に示す実施の形態に基づいて、この発明を詳細に説明する。

図１は、本実施の形態によるデータ処理方法を実現するコンピュータシステムのシステム構成を示す図である。

図１を参照すると、本実施の形態におけるコンピュータシステムは、ソースプログラム（入力コード）をコンパイルするコンパイラ１００と、コンパイラ１００にてコンパイルされたオブジェクトプログラム（出力コード）を実行して種々の処理を行うプログラム実行部２００と、メモリ３００とを備える。コンパイラ１００及びプログラム実行部２００は、パーソナルコンピュータやワークステーションなどのコンピュータシステムにおけるプログラム制御されたＣＰＵにて実現される。メモリ３００は、コンピュータ装置のメインメモリであり、ＲＡＭ等で実現される。メモリ３００には、ＣＰＵを制御してコンパイラ１００として動作させるためのプログラムやコンパイルの対象のプログラムが格納されると共に、コンパイラ１００のコンパイル処理において、レジスタから退避される変数が一時的に格納される。なお、メモリ３００に格納されるプログラムは、必要に応じて、適宜磁気ディスクその他の外部記録装置に保存されることは言うまでもない。

【００２０】

図１において、コンパイラ１００は、所定のプログラミング言語で記述された入力コードを入力して処理し、機械語で記述された出力コードを生成して出力する。この入力コードの入力は、コード生成装置４００にて生成された入力コードを直接入力したり、コード生成装置４００にて生成された入力コードを記憶した記憶装置５００から入力したり、ネットワーク６００上に存在するコード生成装置４００や記憶装置５００からネットワーク６００を介して入力したりすることにより行われる。コンパイラ１００により生成された出力コードは、プログラム実行部２００により実行される。

【００２１】

図２は、本実施の形態におけるコンパイラ１００の構成を説明する図である。図２を参照すると、コンパイラ１００は、入力コードの字句解析や構文解析を行うコード解析部１１０と、各種の最適化によるプログラムの変形を行う最適化部１２０と、最適化されたプログラムを出力コードに変換（生成）して出力する出力コード生成部１３０とを備える。

また、最適化部１２０の機能として、所定の手続き呼び出し部分をインライン化するインライン化実行部１０と、プログラム中の引数の評価を行う部分を条件分岐（参照条件）の後ろへ移動させる引数評価移動部２０と、引数評価移動部２０の処理で生じた不用なコードを除去するための不用コード除去部３０とを備える。

図２に示したコンパイラ１００の各構成要素は、コンピュータプログラムにより制御されたＣＰＵにて実現される仮想的なソフトウェアブロックである。ＣＰＵを制御する当該コンピュータプログラムは、磁気ディスクや光ディスク、半導体メモリ等の記録媒体に格納して配布したり、ネットワークを介して伝送したりすることにより提供される。

なお、図２に示した最適化部１２０の各機能は、本実施の形態における特徴的な機能に関するものである。特に図示しないが、プログラムの最適化の手段としては、実際には、本実施の形態で説明する以外の種々の手法があり、本実施の形態による手法と併用することも可能であることは言うまでもない。

【００２２】
本実施の形態は、値渡しの行われる箇所である手続き呼び出しにおいて、呼び側での引数評価を適切な条件（評価条件）で保護することで、呼び側での無駄な引数評価を抑制する。ここで、評価条件は通常、呼ばれ側で引数が実際に参照されるための条件（参照条件）と同一である。すなわち、本実施の形態による最適化の本質は、引数と参照条件とに対する評価順序の入れ替えである。
一般に、呼び側での引数評価と評価後の引数を用いた手続き呼び出しとの間には任意の計算を挿入することができる。しかし、本実施の形態における最適化の主要な対象であるデバッグやログにメッセージを出力するための手続き呼び出しでは、評価後の引数を一時変数に受けずに、手続き呼び出しの実引数部に直接記述する（すなわち引数評価は手続き呼び出しの直前にある）傾向が顕著である。また、その際の呼ばれ側は（デバッグやチューニング時に機能を有効化し、実際の稼動時は無駄なオーバヘッドを避けるため無効化するなどの要求から）手続き全体が唯一の単純な条件（参照条件）で保護されている傾向がある。そのため、条件分岐（参照条件）の直前の命令列を、当該条件分岐の後ろで引数を参照する部分の前へ移動することにより、無駄な引数評価を削減する最適化が可能となる。
【００２３】
引数と参照条件の評価順が逆になったことにより、起きるはずの副作用が起きなかったり、起きないはずの副作用が起きたりすることを避けるため、（参照条件から合成される）評価条件が副作用を起こす場合、引数評価は副作用を起こしてはならない。ここで、副作用とは、例外や値の逃避（escape）などである。副作用の除去、計算量の削減などの理由で評価条件を変更（単純化）することもできるが、その場合は変更後の評価条件が本来の評価条件を包含するように決める。
さらに、呼ばれ側のコードである参照条件から評価条件を合成する都合上、その呼出し点における呼ばれ側の集合は、呼び側のコンパイル時に決まっている必要がある。呼ばれ側が一意に決まる静的呼び出しやテストコードで静的束縛された動的呼び出し、Ｃ++の仮想関数のようにメソッドの実行時オーバロードを伴わない動的呼び出しは、この条件を満たす。Ｊａｖａの仮想関数のようにメソッドの実行時オーバロードを伴う動的呼び出しは、実行時に呼ばれ側の集合が変わったことを検出し、コードを安全側にパッチするランタイムと併用し、呼び側のコンパイル時に既知の呼ばれ側に限り、本実施の形態による最適化を実行する。また、呼び側のコンパイル時に既知の呼ばれ側の集合が複数の要素からなるときは、テストコードを用いて一意に静的束縛した後、本実施の形態による最適化を実行する。
【００２４】
図２に示した構成において、インライン化実行部１０は、プログラム中の手続き呼び出しにおいて、上記の副作用や呼ばれ側の一意性等の条件を満足するかどうかを調べた上で、呼ばれ側の手続きを呼び側に埋め込み、インライン化する。インライン化自体は、従来から最適化の手法の１つとして周知の技術である（例えば、非特許文献３参照）。本実施の形態でも、従来のインライン化の手法を適用することができる。
【００２５】
引数評価移動部２０は、呼び側にある引数評価の部分を参照条件（評価条件）の後ろへ移動する。具体的には、まず、インライン化実行部１０によりインライン化された呼ばれ側（インライン化により呼び側、呼ばれ側という区別はなくなっているが、便宜上、元の呼び側、呼ばれ側に則ってこの称呼を用いる）の手続きの先頭から逆向きに、制御フロー解析を行って制御フローを、制御が合流する基本ブロックの最初の命令まで（評価条件が副作用を起こす場合は、制御が合流する基本ブロックの最初の命令を上限として最下の副作用を起こす命令の直後の命令まで）遡る。そして、得られた命令列の前に参照条件を移動して評価条件とすると共に、当該命令列を複製して評価条件の直後の各ルートへ挿入する。これにより、プログラムの命令列において、引数の評価順と参照条件（評価条件）の評価順とが入れ替わったことになる。なお、参照条件を移動して評価条件とした際、必要があれば上述したように評価条件を単純化することができる。

【００２６】
ここで、本実施の形態では、元の呼び側にあった命令列を、呼ばれ側の直前の基本ブロックほぼ１個分にわたって、呼ばれ側へ移動することとした。
図３は、制御フローを遡る範囲を説明する図である。
図３に示すように、制御の合流点を上限として複製され、参照条件（評価条件）の後ろへ移動する。
しかしながら、これは理論上複製し移動可能な上限という意味ではない。原理的には制御フロー解析によって際限なく遡り、参照条件（評価条件）の後ろへ移動することが可能である。したがって、制御の合流点を越えて制御フローを遡り、参照条件（評価条件）の後ろへ移動しても良い。
図４は、制御の合流点を越える範囲の命令列を参照条件（評価条件）の後ろへ移動した様子を示す図である。
【００２７】
ただし、本実施の形態における最適化の主要な対象であるデバッグやログにメッセージを出力するための手続き呼び出しでは、評価後の引数を一時変数に受けずに、手続き呼び出しの実引数部に直接記述する（すなわち引数評価は手続き呼び出しの直前にある）傾向が顕著である。このため、合流点を越えて命令列を移動しても最適化の効果は概ね変わらないと言える。したがって、実用的には、上記のように、制御が合流する基本ブロックの最初の命令まで（もしくは当該基本ブロックの最初の命令を上限として最下の副作用を起こす命令の直後の命令まで）制御フローを遡り、参照条件（評価条件）の後ろへ移動するとすれば十分である。
なお、上記の命令列の複製は、引数を評価する命令を選択して行うのではなく、制御フローで得られた範囲の命令列全体に対して行う。これにより、個々の命令を解析するといった煩雑な作業を避けることができる。
【００２８】
不用コード除去部３０は、引数評価移動部２０により評価条件の直後の各ルートへ移った引数評価を含む命令列のうち、不用なものを除去する。すなわち、引数評価移動部２０は、単純に命令列を条件分岐の後ろへ移動し、かつ複製するのみであるので、評価条件の真偽に関わらず、各ルートに引数評価を含む命令列が挿入されている。そこで、不用コード除去部３０が、評価条件の真偽を調べ、 評価条件が偽（false）となる側のルートに複製された引数評価を、不用コード除去（dead code elimination）により除去する。なお、この不用コード除去については、最適化の手法の１つとして周知の従来手法（例えば、非特許文献４参照）を適用することができる。
【００２９】
次に、本実施の形態による最適化の作用について具体的に説明する。
図５は、本実施の形態による最適化の処理の全体的な流れを説明するフローチャートである。図６乃至図９は、本実施の形態の最適化によるプログラムの制御フローの変化を説明する図である。
初期動作として、コンパイラ１００は、処理対象であるプログラムを入力し、これに対して字句解析や構文解析等の処理を既に行っているものとする。これらの解析の済んだプログラム（中間コード）は、メモリ３００に一時的に格納されている。
図５に示すように、コンパイラ１００において、制御フロー解析等の手段により、処理対象であるプログラム中の手続き呼び出しにおける呼び側と呼ばれ側とが検出される（ステップ５０１）。図６は、検出された呼び側及び呼ばれ側の制御フローの例を示す図である。図６において、呼び側の「手続呼出」によって呼ばれ側の手続きが呼ばれ、実行される。
【００３０】
次に、コンパイラ１００のインライン化実行部１０が、ステップ５０１で検出された手続き呼び出しの呼ばれ側全体が、唯一の単純な条件（参照条件）で保護されているかを調べる（ステップ５０２）。図６に示した手続き呼び出しにおける呼ばれ側はこの条件を満た

す。そこで、次にインライン化実行部１０は、この呼ばれ側を呼び側へインライン化する（ステップ５０３）。インライン化の施された処理対象のプログラムは、メモリ３００に保持される。
図７は、図６に示した制御フローをインライン化した様子を示す図である。この状態では、「引数評価」が「参照条件」の前に存在している。

【００３１】
次に、引数評価移動部２０が、メモリ３００からプログラムを読み出し、インライン化された手続き（元の呼ばれ側）の先頭から逆向きに制御フローを遡り、制御が合流する基本ブロックの最初の命令まで（もしくは当該基本ブロックの最初の命令を上限として、最下の副作用命令の直後まで）の命令列を得る。そして、参照条件を当該命令列の前に移動して評価条件とすると共に、当該命令列を複製して評価条件の直後の各ルートに挿入する（ステップ５０４）。引数評価移動部２０による変形の済んだプログラムは、メモリ３００に保持される。
図８は、図７の状態から「引数評価」を条件分岐の後ろに移動した状態を示す図である。図８において、「参照条件」に代わって「評価条件」という条件分岐が置かれ、その真偽いずれのルートにも「引数評価」が挿入されている。

【００３２】
最後に、不用コード除去部３０が、メモリ３００からプログラムを読み出し、引数評価移動部２０によって評価条件の後ろの各ルートに挿入された引数評価のうち、不必要な引数評価の命令列を取り除く（ステップ５０５）。不必要な命令列が除去されたプログラムは、メモリ３００に保持される。
図９は、図８の状態から不用な「引数評価」を除去した状態を示す図である。図９において、「引数参照」が行われないルート（「評価条件」が偽のルート）にあった「引数評価」が削除されている。
この後、最適化の施されたプログラムがオブジェクトコードに変換され、当該オブジェクトコードが図１に示したプログラム実行部２００により実行される。

【００３３】
以上のようにして、本実施の形態では、プログラムをコンパイルする際に、引数が呼ばれ側で参照される条件（参照条件）から導かれる条件（評価条件）を、コンパイラ１００が呼び側に自動的に挿入することが実現される。これは、Ｓｃｈｅｍｅで設定可能な遅延評価（lazy evaluation）への切り替えとは異なり、ソースコードの書き換えという繁雑な作業を必要としない。また、評価条件を満たす場合における引数の評価は先行評価されるため、実質的な遅延時間の短縮という利点は失われない。

【００３４】
また、本実施の形態では、プログラムをコンパイルする際にコンパイラ１００が評価条件を呼び側に自動挿入する事が実現される。これは、プログラムの開発者が評価条件を呼び側のソースコードへ手作業で挿入する場合とは異なり、ソースコード上で参照条件は呼ばれ側に集約されたままであるため、プログラムの変更や保守を行う場合の容易さを損なうことはない。また、評価条件の呼び側への挿入の際に実装の詳細を知る必要もなければ、アクセス権の制限を受けることもない。

【００３５】
さらに、本実施の形態では、引数の評価と参照条件（評価条件）の評価の順序を入れ替えた後、不用コードの除去を行うが、その計算量は基本ブロック数（評価条件が副作用を起こす場合は命令数）の１乗に過ぎない。したがって、命令数の３から５乗と非常に大きい計算量を要する従来の部分不用コード除去と異なり、プログラムの実行時に動的にコンパイルを行うコンパイラに用いるのに好適である。

【００３６】
【発明の効果】
以上説明したように、本発明によれば、引数の先行評価を、呼ばれ側で必要とするか否かに応じて行う効率的なプログラムを生成することができる。

また、本発明によれば、プログラムのコンパイルの際に、そのような効率的なプログラムに最適化するコンパイラを提供することができる。

【図面の簡単な説明】

【図１】　本実施の形態によるデータ処理方法を実現するコンピュータシステムのシステム構成を示す図である。

【図２】　本実施の形態におけるコンパイラの構成を説明する図である。

【図３】　本実施の形態において、制御フローを遡る範囲を説明する図である。

【図４】　本実施の形態において制御の合流点を越える範囲の命令列を参照条件（評価条件）の後ろへ移動した様子を示す図である。

【図５】　本実施の形態による最適化の処理の全体的な流れを説明するフローチャートである。

【図６】　本実施の形態の最適化によるプログラムの制御フローの変化を説明する図であり、検出された呼び側及び呼ばれ側の制御フローの例を示す図である。

【図７】　本実施の形態の最適化によるプログラムの制御フローの変化を説明する図であり、図６に示した制御フローをインライン化した様子を示す図である。

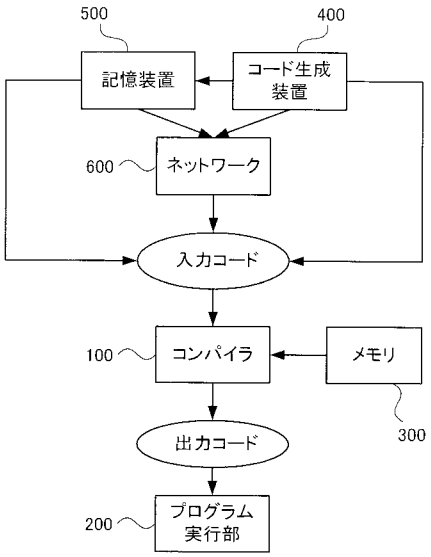
【図８】　本実施の形態の最適化によるプログラムの制御フローの変化を説明する図であり、図７の状態から「引数評価」を条件分岐の後ろに移動した状態を示す図である。

【図９】　本実施の形態の最適化によるプログラムの制御フローの変化を説明する図であり、図８の状態から不用な「引数評価」を除去した状態を示す図である。

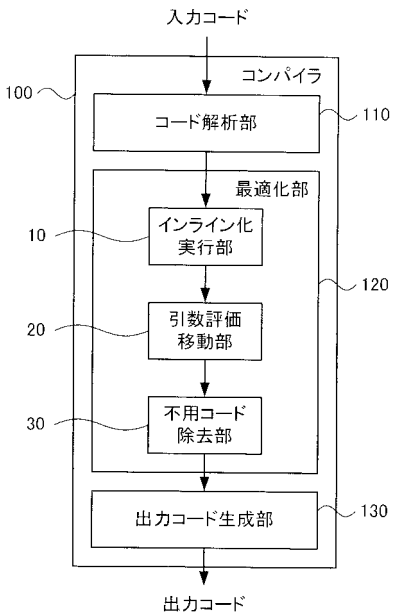
【符号の説明】

１０…インライン化実行部、２０…引数評価移動部、３０…不用コード除去部、１００…コンパイラ、１１０…コード解析部、１２０…最適化部、１３０…出力コード生成部、２００…プログラム実行部、３００…メモリ

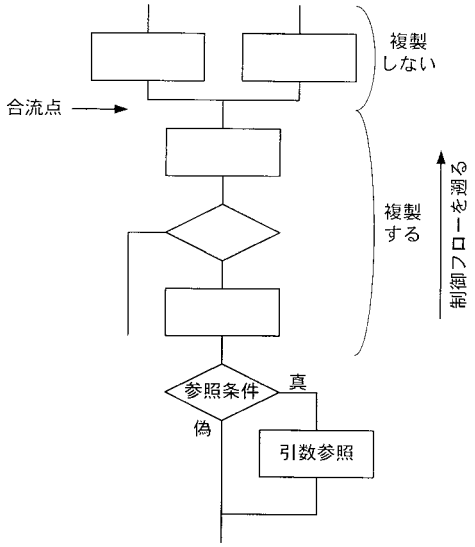
【図１】



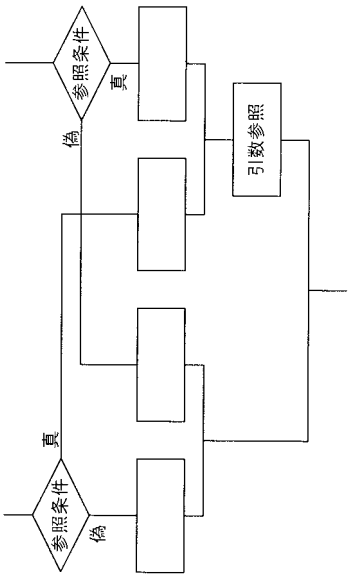
【図２】



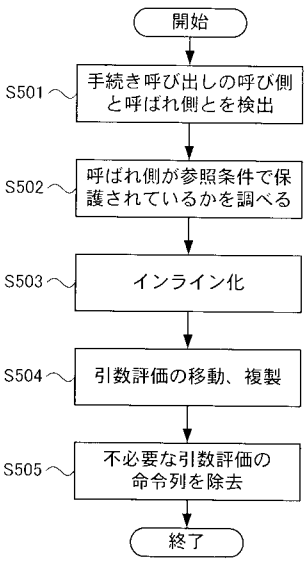
【図３】


複製
しない
合流点
複製
する
制御フローを遡る
参照条件
真
偽
引数参照

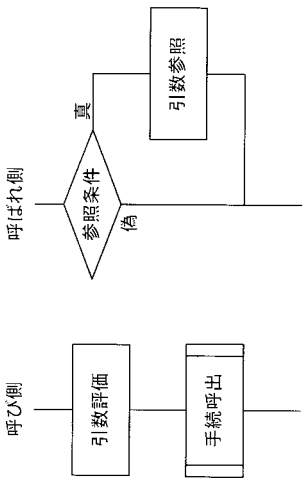
【図４】


参照条件
真
偽
引数参照
真
参照条件
偽

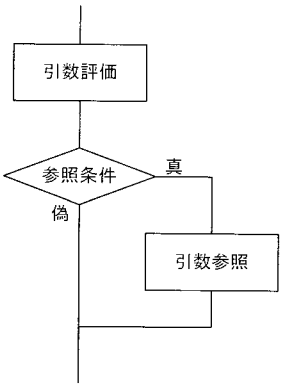
【図５】


開始
S501
手続き呼び出しの呼び側
と呼ばれ側とを検出
S502
呼ばれ側が参照条件で保
護されているかを調べる
S503
インライン化
S504
引数評価の移動、複製
S505
不必要な引数評価の
命令列を除去
終了

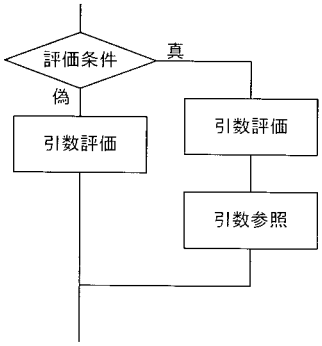
【図６】


引数参照
真
呼ばれ側
参照条件
偽
呼び側
引数評価
手続呼出

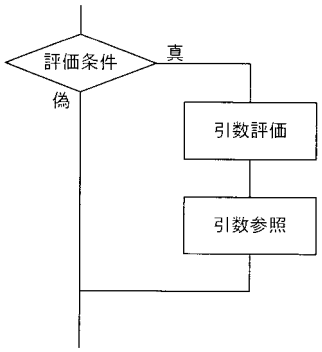
【図７】



【図８】



【図９】



フロントページの続き

(72)発明者　竹内　幹雄
神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　東京基礎研究所内

審査官　久保　光宏

(56)参考文献　特開昭６３－１６３６３６（ＪＰ，Ａ）
特開平１０－４０１０６（ＪＰ，Ａ）
特開平９－２８８５８２（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F9/45,
JSTPlus(JDream2),
CSDB(日本国特許庁)