(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2009-187285
(P2009-187285A)
(43)公開日　平成21年8月20日(2009.8.20)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 9/45	(2006.01)	G06F	9/44	322F	5B081

審査請求 未請求　請求項の数 13　ＯＬ　（全 19 頁）

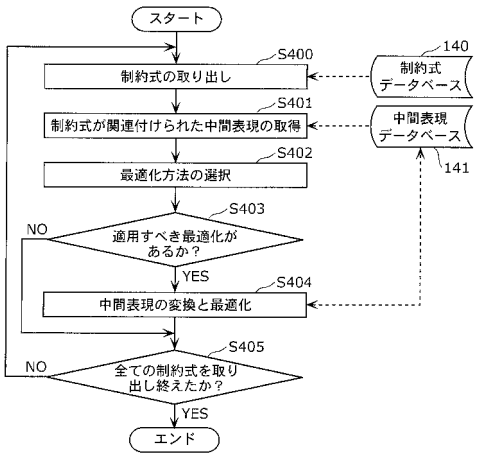
(21)出願番号　特願2008-26556(P2008-26556)
(22)出願日　平成20年2月6日(2008.2.6)

(71)出願人　000005821
パナソニック株式会社
大阪府門真市大字門真１００６番地
(74)代理人　100109210
弁理士　新居　広守
(72)発明者　濱田　智雄
大阪府門真市大字門真１００６番地　松下電器産業株式会社内
(72)発明者　服部　大
大阪府門真市大字門真１００６番地　松下電器産業株式会社内

Ｆターム(参考)　5B081 CC21

(54)【発明の名称】プログラム変換方法、プログラム変換装置およびプログラム

(57)【要約】
【課題】数の特性や２つ以上の変数間の関連を、プログラマあるいはプロファイラが、最適化の際のヒント情報として与えることで、適用可能な最適化の種類やその効果を拡大する。
【解決手段】コンピュータにより入力プログラムを変換するプログラム変換方法であって、入力プログラムに含まれる変数、定数、演算子、擬似変数および組み込み関数のうちの少なくとも１つで構成され、かつ結果が真偽値となる制約式を取得する取得ステップ（Ｓ４００）と、取得された前記制約式に基づいて、前記入力プログラムを最適化することにより、前記入力プログラムを変換する変換ステップ（Ｓ４０１～Ｓ４０４）とを含む。
【選択図】図５



【特許請求の範囲】
【請求項１】
コンピュータにより入力プログラムを変換するプログラム変換方法であって、
入力プログラムに含まれる変数、定数、演算子、擬似変数および組み込み関数のうちの少なくとも１つを含み、かつ結果が真偽値となる制約式を取得する取得ステップと、
取得された前記制約式に基づいて、前記入力プログラムを最適化することにより、前記入力プログラムを変換する変換ステップとを含む
ことを特徴とするプログラム変換方法。
【請求項２】
前記変換ステップでは、前記制約式に関連付けられた前記入力プログラムに含まれる関数または文を、前記制約式が真となる場合の条件に基づいて最適化する
ことを特徴とする請求項１に記載のプログラム変換方法。
【請求項３】
前記制約式における擬似変数として、前記入力プログラム中の関数の仮引数を指定するための変数である擬似仮引数変数を用いる
ことを特徴とする請求項１または２に記載のプログラム変換方法。
【請求項４】
前記制約式における擬似変数として、前記入力プログラム中の関数宣言における引数を指定するための変数である擬似宣言部引数を用いる
ことを特徴とする請求項１または２に記載のプログラム変換方法。
【請求項５】
前記制約式における組み込み関数は、引数が定数の場合は真を、定数でない場合に偽を返す定数判定組み込み関数である
ことを特徴とする請求項１または２に記載のプログラム変換方法。
【請求項６】
前記取得ステップでは、１つの関数定義に関連付けられ、かつ仮引数、定数および演算子からなる制約式である仮引数制約式を取得し、
前記変換ステップは、
前記関連付けられた関数定義で示される関数の複製を作成するステップと、
複製された関数の内部の処理を、前記仮引数制約式が真になるとみなして最適化するステップと、
前記仮引数制約式に含まれる仮引数を実引数で置き換えた制約式が真となり、かつ前記関連付けられた関数定義で示される関数を呼び出す関数呼び出し文について、当該関数呼び出し文が呼び出す関数を前記複製された関数に置き換えるステップとを含む
ことを特徴とする請求項１に記載のプログラム変換方法。
【請求項７】
前記取得ステップでは、１つの関数呼び出し文に関連付けられ、かつ前記擬似仮引数変数、定数および演算子からなる制約式である実引数制約式を取得し、
前記変換ステップは、
前記実引数制約式中の前記擬似仮引数変数を、前記関数呼び出し文が呼び出す関数の実引数で置き換えた式が常に真である場合に、前記関数呼び出し文が呼び出す関数の複製を作成するステップと、
複製された関数の内部の処理を、前記実引数制約式中の前記擬似仮引数変数を、前記関数呼び出し文が呼び出す前記関数の仮引数で置き換えた式が真になるとみなして最適化するステップと、
前記関数呼び出し文が呼び出す関数を前記複製された関数に置き換えるステップとを含む
ことを特徴とする請求項３に記載のプログラム変換方法。
【請求項８】
前記取得ステップでは、１つの関数宣言に関連付けられ、かつ前記擬似宣言部引数、定

数および演算子からなる制約式である擬似宣言部引数制約式を取得し、
前記変換ステップは、
前記入力プログラムに前記関数宣言に対応する関数の定義が存在する場合に、前記擬似宣言部引数制約式中の擬似宣言部引数を、対応する関数の仮引数変数に置き換えた仮引数制約式を用いて、当該関数の複製を、当該仮引数制約式が真になるとみなして最適化するステップと、
前記入力プログラム中に前記関数宣言に対応する関数呼び出し文が存在する場合に、前記擬似宣言部引数制約式中の擬似宣言部引数を、対応する関数呼び出しの実引数変数に置き換えた実引数制約式が常に真である場合に、当該関数呼び出し文を、複製される関数への呼び出し文に置き換えるステップとを含む
ことを特徴とする請求項４に記載のプログラム変換方法。
【請求項９】
前記取得ステップでは、１つの関数内の変数または大域変数、定数、および演算子からなる関数内制約式を取得し、
前記変換ステップは、
前記関数から、前記関数内制約式に含まれる全ての変数について共通する生存区間に含まれる文の並び、または前記生存区間に含まれる文の並びを含む文を最適化対象文として取得するステップと、
前記関数内制約式を条件式とし、
当該条件式の結果が真である場合の処理として、前記最適化対象文を前記関数内制約式が真になるとみなして最適化した文を有し、かつ前記条件式の結果が偽である場合の処理として、前記最適化対象文を有する条件分岐文を生成するステップと、
前記最適化対象文を生成された前記条件分岐文に置き換えるステップとを含む
ことを特徴とする請求項１に記載のプログラム変換方法。
【請求項１０】
前記取得ステップでは、１つの関数内の変数または大域変数、定数、および演算子からなる関数内制約式を取得し、
前記変換ステップでは、
前記関数から、２つ以上の分岐条件と２つ以上の分岐先と、前記分岐先毎に排他的処理を持つ分岐文または分岐文の並びとを取得するステップと、
前記関数内制約式と同じ式が前記２つ以上の分岐条件のいずれかに含まれている場合には、分岐文または分岐文の並びの条件式の評価順序を入れ替えるステップとを含む
ことを特徴とする請求項１に記載のプログラム変換方法。
【請求項１１】
前記変換ステップでは、さらに、変換された前記入力プログラムを目的プログラムに変換する
ことを特徴とする請求項１～１０のいずれか１項に記載のプログラム変換方法。
【請求項１２】
入力プログラムを変換するプログラム変換装置であって、
入力プログラムに含まれる変数、定数、演算子、擬似変数および組み込み関数のうちの少なくとも１つで構成され、かつ結果が真偽値となる制約式を取得する取得手段と、
取得された前記制約式に基づいて、前記入力プログラムを最適化することにより、前記入力プログラムを変換する変換手段とを備える
ことを特徴とするプログラム変換装置。
【請求項１３】
入力プログラムを変換するためのプログラムであって、
入力プログラムに含まれる変数、定数、演算子、擬似変数および組み込み関数のうちの少なくとも１つで構成され、かつ結果が真偽値となる制約式を取得する取得ステップと、
取得された前記制約式に基づいて、前記入力プログラムを最適化することにより、前記入力プログラムを変換する変換ステップとをコンピュータに実行させる

ことを特徴とするプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、プログラムの実行速度向上およびプログラムのサイズ削減を実現するプログラム変換方法に関する。
【背景技術】
【０００２】
入力プログラムから目的機械が実行可能なプログラムを生成する言語処理系においては、生成するプログラムの実行速度やプログラムのサイズを削減するために、さまざまな最適化が行われる。
【０００３】
上記最適化の１つに、プログラム中の変数が取る値に着目し、変数が取りうる値とその値の出現頻度を調べ、当該変数が出現頻度が高い値を取った場合の処理を高速化する手法がある（例えば、特許文献１参照。）。
【特許文献１】特開２００２－２２２０８８号公報
【発明の開示】
【発明が解決しようとする課題】
【０００４】
しかしながら、従来の技術で、ユーザがヒント情報として与えたり、プロファイラの解析結果から自動生成したりするのは、プログラム中の個々の変数が取りうる値とその出現頻度である。したがって、当該情報を用いた最適化は、関数内あるいは関数間の定数伝播と、それに伴う定数畳み込みとに制限される。
【０００５】
本発明は、変数の特性や２つ以上の変数間の関連を、プログラマあるいはプロファイラが、最適化の際のヒント情報として与えることで、適用可能な最適化の種類やその効果を拡大することを目的とする。
【課題を解決するための手段】
【０００６】
本発明のある局面に係るプログラム変換方法は、コンピュータにより入力プログラムを変換するプログラム変換方法であって、入力プログラムに含まれる変数、定数、演算子、擬似変数および組み込み関数のうちの少なくとも１つを含み、かつ結果が真偽値となる制約式を取得する取得ステップと、取得された前記制約式に基づいて、前記入力プログラムを最適化することにより、前記入力プログラムを変換する変換ステップとを含むことを特徴とする。
【０００７】
上述のような制約式を用いて入力プログラムを最適化する。このため、変数の特性や２つ以上の変数間の関連を制約式として与えることができ、従来に比べ、適用可能な最適化の種類やその効果を拡大することができる。
【０００８】
好ましくは、前記取得ステップでは、１つの関数定義に関連付けられ、かつ仮引数、定数および演算子からなる制約式である仮引数制約式を取得し、前記変換ステップは、前記関連付けられた関数定義で示される関数の複製を作成するステップと、複製された関数の内部の処理を、前記仮引数制約式が真になるとみなして最適化するステップと、前記仮引数制約式に含まれる仮引数を実引数で置き換えた制約式が真となり、かつ前記関連付けられた関数定義で示される関数を呼び出す関数呼び出し文について、当該関数呼び出し文が呼び出す関数を前記複製された関数に置き換えるステップとを含むことを特徴とする。
【０００９】
複製された関数の内部の処理を、仮引数制約式が真になるとみなして最適化するため、仮引数制約式の演算処理が不要になる。このため、関数呼び出し文から呼び出される関数

の処理量を削減できる。

【００１０】

さらに好ましくは、前記取得ステップでは、１つの関数呼び出し文に関連付けられ、かつ前記擬似仮引数変数、定数および演算子からなる制約式である実引数制約式を取得し、前記変換ステップは、前記実引数制約式中の前記擬似仮引数変数を、前記関数呼び出し文が呼び出す関数の実引数で置き換えた式が常に真である場合に、前記関数呼び出し文が呼び出す関数の複製を作成するステップと、複製された関数の内部の処理を、前記実引数制約式中の前記擬似仮引数変数を、前記関数呼び出し文が呼び出す前記関数の仮引数で置き換えた式が真になるとみなして最適化するステップと、前記関数呼び出し文が呼び出す関数を前記複製された関数に置き換えるステップとを含むことを特徴とする。

【００１１】

複製された関数の内部の処理を、実引数制約式中の擬似仮引数変数を、関数呼び出し文が呼び出す関数の仮引数で置き換えた式が真になるとみなして最適化する。このため、当該式の演算処理が不要になる。これによって、関数呼び出し文から呼び出される関数の処理量を削減できる。

【００１２】

さらに好ましくは、前記制約式における擬似変数として、前記入力プログラム中の関数宣言における引数を指定するための変数である擬似宣言部引数を用いることを特徴とする。

【００１３】

また、前記取得ステップでは、１つの関数宣言に関連付けられ、かつ前記擬似宣言部引数、定数および演算子からなる制約式である擬似宣言部引数制約式を取得し、前記変換ステップは、前記入力プログラムに前記関数宣言に対応する関数の定義が存在する場合に、前記擬似宣言部引数制約式中の擬似宣言部引数を、対応する関数の仮引数変数に置き換えた仮引数制約式を用いて、当該関数の複製を、当該仮引数制約式が真になるとみなして最適化するステップと、前記入力プログラム中に前記関数宣言に対応する関数呼び出し文が存在する場合に、前記擬似宣言部引数制約式中の擬似宣言部引数を、対応する関数呼び出しの実引数変数に置き換えた実引数制約式が常に真である場合に、当該関数呼び出し文を、複製される関数への呼び出し文に置き換えるステップとを含むことを特徴とする。

【００１４】

さらに好ましくは、前記取得ステップでは、１つの関数内の変数または大域変数、定数、および演算子からなる関数内制約式を取得し、前記変換ステップは、前記関数から、前記関数内制約式に含まれる全ての変数について共通する生存区間に含まれる文の並び、または前記生存区間に含まれる文の並びを含む文を最適化対象文として取得するステップと、前記関数内制約式を条件式とし、当該条件式の結果が真である場合の処理として、前記最適化対象文を前記関数内制約式が真になるとみなして最適化した文を有し、かつ前記条件式の結果が偽である場合の処理として、前記最適化対象文を有する条件分岐文を生成するステップと、前記最適化対象文を生成された前記条件分岐文に置き換えるステップとを含むことを特徴とする。

【００１５】

制約式の結果が真となる場合には、最適化された文を実行することができる。このため、与えられた制約式を満足するときの、当該関数の処理量を削減できる。

【００１６】

さらに好ましくは、前記取得ステップでは、１つの関数内の変数または大域変数、定数、および演算子からなる関数内制約式を取得し、前記変換ステップでは、前記関数から、２つ以上の分岐条件と２つ以上の分岐先と、前記分岐先毎に排他的処理を持つ分岐文または分岐文の並びとを取得するステップと、前記関数内制約式と同じ式が前記２つ以上の分岐条件のいずれかに含まれている場合には、分岐文または分岐文の並びの条件式の評価順序を入れ替えるステップとを含むことを特徴とする。

【００１７】

例えば、制約式により指定された条件式の評価を優先させるように条件式の評価順序を入れ替えると、制約式で与えられた式が成立する際の条件式の評価に要する処理量を削減することができる。

【００１８】

なお、本発明は、このような特徴的なステップを含むプログラム変換方法として実現することができるだけでなく、プログラム変換方法に含まれる特徴的なステップを手段とするプログラム変換装置として実現したり、プログラム変換方法に含まれる特徴的なステップをコンピュータに実行させるプログラムとして実現したりすることもできる。そして、そのようなプログラムは、ＣＤ－ＲＯＭ（Ｃｏｍｐａｃｔ　Ｄｉｓｃ－Ｒｅａｄ　Ｏｎｌｙ　Ｍｅｍｏｒｙ）等の記録媒体やインターネット等の通信ネットワークを介して流通させることができるのは言うまでもない。

【発明の効果】

【００１９】

先に述べた制約式を用いて、入力プログラムを変換することにより、適用可能な最適化の種類やその効果を拡大することができる。

【発明を実施するための最良の形態】

【００２０】

以下、本発明の実施の形態に係るプログラム変換装置について図面を参照しながら説明する。

【００２１】

図１は、プログラム変換装置の外観図である。プログラム変換装置２００は、通常のコンピュータからなり、当該コンピュータ上でプログラムを実行することにより実現される。

【００２２】

図２は、プログラム変換装置の機能的な構成を示すブロック図である。

【００２３】

プログラム変換装置２００は、高級言語で記述された入力プログラム１１０と、制約式集合１１１とを入力として受け、入力プログラム１１０を変換し、変換した結果である目的プログラム１２０を出力する装置である。ここで、制約式集合１１１には、入力プログラム１１０内の変数、定数および演算子、並びにプログラム変換装置２００が提供する擬似変数および組み込み関数の少なくとも１つで構成され、結果が真偽値となる式（以下、「制約式」と呼ぶ。）の集合が含まれる。

【００２４】

プログラム変換装置２００は、中間表現生成部２３０と、制約式集合解析部２３１と、中間表現変換部２３２と、目的プログラム生成部２３３と、制約式データベース１４０と、中間表現データベース１４１とを備える。

【００２５】

中間表現生成部２３０は、入力プログラム１１０を、プログラム変換装置２００の内部データ表現である中間表現に変換し、記憶装置である中間表現データベース１４１に格納する処理部である。

【００２６】

制約式集合解析部２３１は、制約式集合１１１に含まれる制約式をプログラム変換装置２００の内部データ表現である式表現に変換し、記憶装置である制約式データベース１４０に格納する処理部である。

【００２７】

中間表現変換部２３２は、中間表現データベース１４１に含まれている入力プログラム１１０の内部表現を、制約式データベース１４０の情報に基づいて、変換し、中間表現データベース１４１に格納する処理部である。

【００２８】

目的プログラム生成部２３３は、中間表現データベース１４１の内容から、目的機械に

適した書式の目的プログラム１２０を生成し、出力する処理部である。

【００２９】

図３は、プログラム変換装置２００が実行するプログラム変換処理を示すフローチャートである。

【００３０】

中間表現生成部２３０は、入力プログラム１１０を中間表現に変換し、中間表現データベース１４１に格納する（ステップＳ１３０）。中間表現として一般的な形式には、３番地コードや抽象構文木が使用され、本発明においてもそれらを用いる事ができる。

【００３１】

制約式集合解析部２３１は、制約式集合１１１をプログラム変換装置２００の内部データ表現である式表現に変換し、制約式データベース１４０に格納する（ステップＳ１３１）。式表現として一般的な形式には、式木構造が挙げられる。

【００３２】

中間表現変換部２３２は、中間表現データベース１４１の内容を、制約式データベース１４０の情報に基づいて、変換し、中間表現データベース１４１に格納する（ステップＳ１３２）。変換の目的は、演算の高速化あるいはプログラムサイズの削減である。

【００３３】

目的プログラム生成部２３３は、中間表現データベース１４１の内容から、目的機械に適した書式の目的プログラム１２０を生成し、出力する（ステップＳ１３３）。

【００３４】

図４は、制約式の記述方法の一例について説明するための図である。

【００３５】

同図に示される入力プログラム１１０において、指示子３００は、関数ｓａｍｐｌｅ＿１の仮引数からなる仮引数制約式である。指示子３００の１行目は、仮引数ｐ０が条件式ｐ０＞０の成立を前提とした関数ｓａｍｐｌｅ＿１の処理の最適化を行なうことを指示し、指示子３００の２行目は、ｐ０＞２５５の成立を前提とした場合の処理の最適化を行うことを指示する。

【００３６】

指示子３０１は、呼び出される関数（ここでは、関数ｓａｍｐｌｅ２）の仮引数に相当する擬似変数を用いた、実引数制約式である。例では“＄０”，“＄１”，．．．をそれぞれ、呼ばれる関数の０番目の仮引数，１番目の仮引数，．．．としている。指示子３０１の１行目は、関数ｓａｍｐｌｅ２の０番目の引数を２で割った時の余りが０の場合のｓａｍｐｌｅ２内の処理の最適化を行なうことを指示し、指示子３００の２行目は、関数ｓａｍｐｌｅ２の０番目の引数を２で割ったときの余りが１の関数の場合のｓａｍｐｌｅ２内の処理の最適化を行うことを指示する。

【００３７】

指示子３０２は、関数内の変数からなる関数内制約式である。指示子３０２は、ｐ０＝＝５かつｐ１＝＝３が成立する場合の次行の処理の最適化を指示する。

【００３８】

上記の例では、制約式を指示子にて与えたが、変数の名前と各変数の生存区間に関する情報が含まれていれば、例えば入力ファイルとは別のファイルでも制約式を定義可能である。

【００３９】

次に、中間表現変換部２３２が実行する中間表現の最適化処理（ステップＳ１３２）について、具体的な変換例を示しながら説明する。

【００４０】

図５は、中間表現変換部２３２が実行する中間表現の最適化処理（ステップＳ１３２）の概要を示すフローチャートである。なお、以下の各処理の詳細については、図６以降の図を用いて説明する。

【００４１】

中間表現変換部２３２は、制約式データベース１４０内の全ての制約式を順次取り出し、取り出された制約式をｅとする（ステップＳ４００）。

【００４２】

中間表現変換部２３２は、制約式ｅに関連付けられた、中間表現データベース１４１の文、関数または宣言を取り出し、取り出された文または関数または宣言を中間表現ｓとする（ステップＳ４０１）。

【００４３】

中間表現変換部２３２は、制約式ｅとそれが関連する中間表現ｓとを解析し、入力プログラム１１０の内部表現に適用可能でかつ有効な最適化方法を選択する（ステップＳ４０２）。ここで、得られた最適化方法をｍとする。

【００４４】

適用可能でかつ有効な最適化方法が得られた場合には（ステップＳ４０３でＹＥＳ）、中間表現変換部２３２は、制約式ｅと最適化方法ｍに基づいて、中間表現ｓの変換と最適化を実行する。また、中間表現変換部２３２は、変換及び最適化が施された中間表現ｓを中間表現データベース１４１に格納する（ステップＳ４０４）。

【００４５】

適用可能でかつ有効な最適化方法が得られなかった場合（ステップＳ４０３でＮＯ）または中間表現最適化処理（ステップＳ４０４）の後、中間表現変換部２３２は、制約式データベース１４０内の全ての制約式を取り出し終えたか否かを判断する（ステップＳ４０５）。全ての制約式を取り出し終えていれば（ステップＳ４０５でＹＥＳ）、中間表現変換部２３２は、処理を終了する。取り出し終えていない制約式がある場合には（ステップＳ４０５でＮＯ）、中間表現変換部２３２は、ステップＳ４００以降の処理を繰り返す。

【００４６】

以下では、制約式の種類ごとに中間表現変換部２３２が実行する中間表現の最適化処理（ステップＳ１３２、図５）について説明する。

【００４７】

図６は、関数の仮引数、定数および演算子からなる仮引数制約式を用いた入力プログラム１１０の変換処理を示すフローチャートである。

【００４８】

中間表現変換部２３２は、制約式データベース１４０内の全ての制約式を順次取り出し、取り出された制約式をｅとする（ステップＳ５００）。

【００４９】

制約式ｅが、仮引数制約式である場合には（ステップＳ５０１でＹＥＳ）、中間表現変換部２３２は、制約式ｅに含まれる変数を仮引数として持つ関数ｐを、中間表現データベース１４１より取得する（ステップＳ５０２）。

【００５０】

中間表現変換部２３２は、関数ｐを複製した関数ｐ´を生成する（ステップＳ５０３）。

【００５１】

中間表現変換部２３２は、関数ｐ´を、制約式ｅが真であるとして最適化する（ステップＳ５０４）。

【００５２】

中間表現変換部２３２は、関数ｐを呼び出す関数呼び出し文の集合Ｓを、中間表現データベース１４１より求める（ステップＳ５０５）。

【００５３】

中間表現変換部２３２は、関数呼び出し文の集合Ｓの要素を順次取り出し、取り出された関数呼び出し文をｓとする（ステップＳ５０６）。

【００５４】

中間表現変換部２３２は、制約式ｅに含まれる変数、即ち関数ｐの仮引数を、関数呼び出し文ｓの対応する実引数で置き換えた、制約式ｅ´を生成する（ステップＳ５０７）。

【００５５】
制約式ｅ´が常に真であることが保障される場合には（ステップＳ５０８でＹＥＳ）、中間表現変換部２３２は、関数呼び出し文ｓの呼び出す関数をｐからｐ´に置き換えて（ステップＳ５０９）、ｐ´と更新した関数呼び出し文ｓとを中間表現データベース１４１に登録する（ステップＳ５１０）。
【００５６】
制約式ｅ´が常に真であることが保障されない場合（ステップＳ５０８でＮＯ）、またはステップＳ５１０の処理の後、中間表現変換部２３２は、集合Ｓに含まれる全ての関数呼び出し文ｓを取り出し終えたか否かを判定する（ステップＳ５１１）。取り出していない関数呼び出し文ｓが存在する場合には（ステップＳ５１１でＮＯ）、中間表現変換部２３２は、Ｓ５０６以降の処理を繰り返し実行する。
【００５７】
全ての関数呼び出し文ｓを取り出し終えた場合（ステップＳ５１１でＹＥＳ）、または取り出した制約式ｅが仮引数制約式ではない場合（ステップＳ５０１でＮＯ）、中間表現変換部２３２は、制約式データベース１４０内の全ての制約式を取り出し終えたか否かを判断する（ステップＳ５１２）。全ての制約式を取り出し終えていれば（ステップＳ５１２でＹＥＳ）、中間表現変換部２３２は、処理を終了する。取り出し終えていない制約式がある場合には（ステップＳ５１２でＮＯ）、中間表現変換部２３２は、ステップＳ５００以降の処理を繰り返す。
【００５８】
次に、図７を用いて仮引数制約式を用いたプログラムの変換の実行例について説明する。図７（ａ）は、入力プログラム１１０の一例であるプログラム６００を示す図である。図７（ｂ）は、プログラム６００の変換結果であるプログラム６１０を示す図である。
【００５９】
中間表現変換部２３２は、指示子６０１“ｂ＞０”を、制約式ｅとして、制約式データベース１４０から取り出したものとする（ステップＳ５００）。
【００６０】
取り出した制約式“ｂ＞０”が仮引数制約式であるため（ステップＳ５０１でＹＥＳ）、中間表現変換部２３２は、制約式“ｂ＞０”が付与された関数ｃｌｉｐを、中間表現データベース１４１より取得する（ステップＳ５０２）。
【００６１】
中間表現変換部２３２は、関数ｃｌｉｐの複製の関数ｃｌｉｐ´を作成する（ステップＳ５０３）。
【００６２】
中間表現変換部２３２は、関数ｃｌｉｐ´を制約式“ｂ＞０”が真であるとして最適化する（ステップＳ５０４）。この実行例では、“ｂ＞０”を評価するｉｆ文の条件式が定数“ｔｒｕｅ”に置き換えられるため、条件判定と、ｅｌｓｅ側の処理とを削減でき、関数ｃｌｉｐ´の処理内容は、文６１２となる。
【００６３】
中間表現変換部２３２は、関数ｃｌｉｐを呼び出す関数呼び出し文６０３の集合Ｓを、中間表現データベース１４１より求める（ステップＳ５０５）。
【００６４】
中間表現変換部２３２は、集合Ｓから関数呼び出し文６０３を取り出す（ステップＳ５０６）。
【００６５】
中間表現変換部２３２は、制約式“ｂ＞０”の仮引数ｂを、関数呼び出し文６０３の実引数ａで置き換えた、制約式“ａ＞０”を得る（ステップＳ５０７）。
【００６６】
関数呼び出し文６０３の直前の条件式“ｉｆ（ａ＞５）”から分かるように、関数呼び出し文６０３の実行時には制約式“ａ＞０”は常に真であることが保障される。制約式“

ａ＞０”は常に真であることが保障されるため（ステップＳ５０８でＹＥＳ）、中間表現変換部２３２は、関数呼び出し文６０３が呼び出す関数を、ｃｌｉｐ関数から、ｃｌｉｐ´関数に置き換え、更新した関数呼び出し文６１３を作成する（ステップＳ５０９）。

【００６７】

以上のように、プログラム６００をプログラム６１０に変換することにより、ｃｌｉｐ関数に含まれる分岐処理を削減することができる。

【００６８】

図８は、関数の実引数、定数および演算子からなる実引数制約式を用いた入力プログラム１１０の変換処理を示すフローチャートである。

【００６９】

中間表現変換部２３２は、制約式データベース１４０内の全ての制約式を順次取り出し、取り出された制約式をｅとする（ステップＳ７００）。

【００７０】

制約式ｅが実引数制約式の場合には（ステップＳ７０１でＹＥＳ）、中間表現変換部２３２は、中間表現データベース１４１から、制約式ｅが付与された関数呼び出し文ｓを取得する（ステップＳ７０２）。

【００７１】

中間表現変換部２３２は、制約式ｅの擬似仮引数変数を、関数呼び出し文ｓに含まれる関数の実引数に置き換えた、制約式ｅ´を生成する（ステップＳ７０３）。

【００７２】

中間表現変換部２３２は、制約式ｅ´が常に真であるか否かを判定する（ステップＳ７０４）。常に真であることが保障される場合には（ステップＳ７０４でＹＥＳ）、中間表現変換部２３２は、中間表現データベース１４１から、関数呼び出し文ｓが呼び出す関数ｐを取得する（ステップＳ７０５）。

【００７３】

中間表現変換部２３２は、制約式ｅに含まれる擬似仮引数を、関数ｐの仮引数で置き換えた、制約式ｅ´´を生成する（ステップＳ７０６）。

【００７４】

中間表現変換部２３２は、関数ｐの複製ｐ´を生成する（ステップＳ７０７）。

【００７５】

中間表現変換部２３２は、関数ｐ´を制約式ｅ´´が真であるという条件を用いて最適化する（ステップＳ７０８）。

【００７６】

中間表現変換部２３２は、中間表現データベース１４１に、関数ｐ´と同じ処理を行う関数ｐ´´が格納されているか否かを判断する（ステップＳ７０９）。関数ｐ´´が中間表現データベース１４１に格納されている場合には（ステップＳ７０９でＹＥＳ）、中間表現変換部２３２は、関数呼び出し文ｓが呼び出す関数をｐからｐ´´に置き換えて、更新した関数呼び出し文ｓを中間表現データベース１４１に登録する（ステップＳ７１０）。

【００７７】

関数ｐ´´が中間表現データベース１４１に格納されていない場合には（ステップＳ７０９でＹＥＳ）、中間表現変換部２３２は、関数ｐ´を中間表現データベース１４１に登録する（ステップＳ７１１）。また、中間表現変換部２３２は、関数呼び出し文ｓが呼び出す関数をｐからｐ´に置き換えて、更新した関数呼び出し文ｓを中間表現データベース１４１に登録する（ステップＳ７１２）。

【００７８】

中間表現変換部２３２は、制約式データベース１４０内の全ての制約式を取り出し終えたか否かを判断する（ステップＳ７１３）。全ての制約式を取り出し終えていれば（ステップＳ７１３でＹＥＳ）、中間表現変換部２３２は、処理を終了する。取り出し終えていない制約式がある場合には（ステップＳ７１３でＮＯ）、中間表現変換部２３２は、ステ

ップＳ７００以降の処理を繰り返す。

【００７９】

次に、図９を用いて実引数制約式を用いたプログラムの変換の実行例について説明する。図９（ａ）は、入力プログラム１１０の一例であるプログラム８００を示す図である。図９（ｂ）は、プログラム８００の変換結果であるプログラム８１０を示す図である。

【００８０】

中間表現変換部２３２は、制約式ｅとして指示子８０１“＄０＞０”を、制約式データベース１４０から取り出したものとする（ステップＳ７００）。

【００８１】

取り出した制約式ｅが実引数制約式であるため（ステップＳ７０１でＹＥＳ）、中間表現変換部２３２は、制約式ｅが付与された関数呼び出し文８０２を取得する（ステップＳ７０２）。

【００８２】

中間表現変換部２３２は、制約式“＄０＞０”の擬似仮引数＄０を、関数呼び出し文８０２の実引数に置き換えた、式ｅ´として“ａ＞０”を得る（ステップＳ７０３）。

【００８３】

関数呼び出し文８０２の直前の条件式“ｉｆ（ａ＞５）”から分かるように、関数呼び出し文８０２の実行時には、式ｅ´が真であることが保障される（ステップＳ７０４でＹＥＳ）。このため、中間表現変換部２３２は、関数呼び出し文ｓが呼び出す関数ｃｌｉｐを取り出す（ステップＳ７０５）。

【００８４】

中間表現変換部２３２は、制約式“＄０＞０”の擬似仮引数を、関数ｃｌｉｐの仮引数に置き換えた、式ｅ´´として、“ｂ＞０”を得る（ステップＳ７０６）。

【００８５】

中間表現変換部２３２は、関数ｃｌｉｐの複製関数ｃｌｉｐ´を生成する（ステップＳ７０７）。

【００８６】

中間表現変換部２３２は、関数ｃｌｉｐ´を、制約式ｅ´´すなわち“ｂ＞０”を用いて最適化する（ステップＳ７０８）。この実行例では、“ｂ＞０”を評価するｉｆ文の条件式が定数“ｔｒｕｅ”に置き換えられるため、条件判定とｅｌｓｅ側の処理とを削減でき、関数ｃｌｉｐ´の処理内容は、文８１３となる。

【００８７】

ここでは、中間表現データベース１４１中に、関数ｃｌｉｐ´と同じ処理を行う関数がないものとする（ステップＳ７０９でＮＯ）。このため、中間表現変換部２３２は、関数ｃｌｉｐ´を中間表現データベース１４１に登録する（ステップＳ７１１）。

【００８８】

中間表現変換部２３２は、関数呼び出し文８０２が呼び出す関数をｃｌｉｐ関数からｃｌｉｐ´関数に置き換えて、更新した関数呼び出し文８１２を作成する（ステップＳ７１２）。

【００８９】

以上のように、プログラム８００をプログラム８１０に変換することにより、ｃｌｉｐ関数に含まれる分岐処理を削減することができる。

【００９０】

図１０は、関数内の変数、定数および演算子からなる関数内制約式を用いた入力プログラム１１０の変換処理を示すフローチャートである。

【００９１】

中間表現変換部２３２は、制約式データベース１４０内の全ての制約式を順次取り出し、取り出された制約式をｅとする（ステップＳ９００）。

【００９２】

中間表現変換部２３２は、制約式ｅが関数内制約式であるか否かを判定する（ステップ

Ｓ９０１）。制約式ｅが関数内制約式であれば（ステップＳ９０１でＹＥＳ）、中間表現変換部２３２は、中間表現データベース１４１から、制約式ｅが付与された関数ｐを取り出す（ステップＳ９０２）。

【００９３】

中間表現変換部２３２は、関数ｐにおける、制約式ｅの中の全ての変数に共通の生存区間を求め、同区間内に存在する文のリストＳを求める（ステップＳ９０３）。

【００９４】

中間表現変換部２３２は、リストＳの文を各分岐先に持ち、条件分岐文を制約式ｅとした、条件分岐文ｉを生成し、中間表現データベース１４１に登録する（ステップＳ９０４）。

【００９５】

中間表現変換部２３２は、条件分岐文ｉの条件式が真となる時に実行される文を、制約式ｅが真である条件を用いて、最適化し、最適化した結果を中間表現データベース１４１に格納する（ステップＳ９０５）。

【００９６】

次に、図１１を用いて関数内制約式を用いたプログラムの変換の実行例について説明する。図１１（ａ）は、入力プログラム１１０の一例であるプログラム１０００を示す図である。図１１（ｂ）は、プログラム１０００の変換結果であるプログラム１０１０を示す図である。

【００９７】

中間表現変換部２３２は、制約式ｅとして、指示子１００１“ａ＋ｂ＝＝１０”を、制約式データベース１４０から取り出したものとする（ステップＳ９００）。

【００９８】

取り出した制約式ｅが関数内制約式であるため（ステップＳ９０１でＹＥＳ）、中間表現変換部２３２は、制約式ｅが付与された関数ｆｕｎｃを取り出す（ステップＳ９０２）。

【００９９】

中間表現変換部２３２は、関数ｆｕｎｃにおける、制約式ｅの中の全ての変数に共通の生存区間を求め、同区間内に存在する文のリストＳを求める（ステップＳ９０３）。この実行例では、制約式ｅ中の全ての変数に共通の生存区間は、文１００２の開始から終了までの区間である。このため、文１００２のみからなる文のリストがリストＳとなる。

【０１００】

中間表現変換部２３２は、リストＳ中の文、即ち文１００２を、分岐先に持ち、条件分岐文を制約式ｅとした条件分岐文ｉを生成する（ステップＳ９０４）。実行例では文１０１１が条件分岐文ｉと合致する。

【０１０１】

中間表現変換部２３２は、条件分岐文ｉの条件文、即ち“ａ＋ｂ＝＝１０”が真となるものとみなして、条件式が真となる時に実行される文を最適化する（ステップＳ９０５）。実行例では、文１００２の右辺式“（ａ＋ｂ）＊（ａ＋ｂ）”を、制約式“ａ＋ｂ＝＝１０”を用いて、“１０＊１０”に変形でき、かつ“１０＊１０”は“１００”である事から、“１００”に置換される。置換した結果が文１０１３となる。

【０１０２】

以上のようにプログラム１０００をプログラム１０１０に変換する事で、２つの加算と１つの乗算処理を、条件判定処理と分岐処理とに置き換えることができる。

【０１０３】

図１２は、関数内の変数、定数および演算子からなる関数内制約式を用いた入力プログラム１１０の変換処理を示すフローチャートである。

【０１０４】

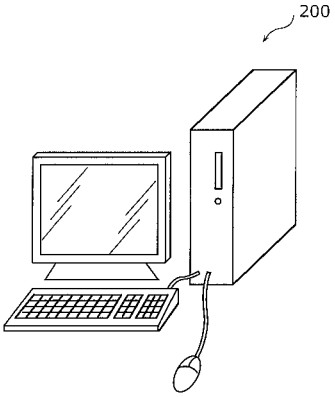
中間表現変換部２３２は、制約式データベース１４０内の全ての制約式を順次取り出し、取り出された制約式をｅとする（ステップＳ１１００）。

【０１０５】
中間表現変換部２３２は、制約式ｅが関数内制約式であるか否かを判定する（ステップＳ１１０１）。制約式ｅが関数内制約式であれば（ステップＳ１１０１でＹＥＳ）、中間表現変換部２３２は、制約式ｅが付与された関数ｐを取り出す（ステップＳ１１０２）。
【０１０６】
中間表現変換部２３２は、関数ｐから、２つ以上の分岐条件と２つ以上の分岐先と、前記分岐先ごとに排他的処理を持つ分岐文または分岐文の並びｓを取得する（ステップＳ１１０３）。
【０１０７】
分岐文または分岐文の並びｓの条件式に、制約式ｅと同じ式が含まれている場合には（ステップＳ１１０４でＹＥＳ）、中間表現変換部２３２は、分岐文または分岐文の並びｓの条件式の評価順序を、条件式ｅが先に変換されるよう修正し、中間表現データベース１４１を更新する（ステップＳ１１０５）。
【０１０８】
分岐文または分岐文の並びｓの条件式に、制約式ｅと同じ式が含まれていない場合（ステップＳ１１０４でＮＯ）、またはＳ１１０５の処理の後、中間表現変換部２３２は、関数ｐから全ての分岐文または分岐文の並びｓを取り出し終えたか否かを判定する（ステップＳ１１０６）。取り出し終えていない分岐文または分岐文の並びｓが存在する場合には（ステップＳ１１０６でＮＯ）、中間表現変換部２３２は、Ｓ１１０３以降の処理を繰り返し実行する。
【０１０９】
全ての分岐文または分岐文の並びｓを取り出し終えた場合には（ステップＳ１１０６でＹＥＳ）、中間表現変換部２３２は、制約式データベース１４０より全ての制約式ｅを取り出し終えたか否かを判断する（ステップＳ１１０７）。全ての制約式ｅを取り出し終えていれば（ステップＳ１１０７でＹＥＳ）、中間表現変換部２３２は、処理を終了する。取り出し終えていない制約式ｅが存在する場合には（ステップＳ１１０７でＮＯ）、中間表現変換部２３２は、ステップＳ１１００以降の処理を繰り返し実行する。
【０１１０】
次に、図１３を用いて関数内制約式を用いたプログラムの変換の実行例について説明する。
【０１１１】
図１３（ａ）は、入力プログラム１１０の一例であるプログラム１２００を示す図である。図１３（ｂ）は、プログラム１２００の変換結果であるプログラム１２１０を示す図である。
【０１１２】
中間表現変換部２３２は、制約式ｅとして、指示子１２０１“２０＜＝ａ＆＆ａ＜３０”を、制約式データベース１４０から取り出したものとする（ステップＳ１１００）。
【０１１３】
取り出した制約式ｅが関数内制約式であるため（ステップＳ１１０１でＹＥＳ）、中間表現変換部２３２は、制約式ｅが付与された関数ｆｕｎｃを取り出す（ステップＳ１１０２）。
【０１１４】
中間表現変換部２３２は、関数ｆｕｎｃにおける、２つ以上の分岐条件と２つ以上の分岐先を持った分岐文の並び（以下、「条件分岐文」という。）１２０２を取得する（ステップＳ１１０３）。
【０１１５】
中間表現変換部２３２は、条件分岐文１２０２の条件式１２０３が制約式ｅと同じであると判定する（ステップＳ１１０４でＹＥＳ）。
【０１１６】
このため、中間表現変換部２３２は、条件式１２０３を最初に評価するよう、条件分岐

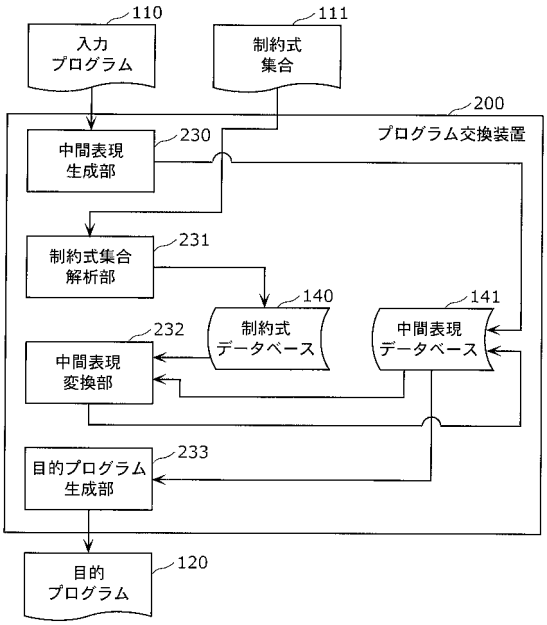
文１２０２を変換する。変換結果は条件分岐文１２１２となる（ステップＳ１１０５）。
【０１１７】
以上のようにプログラム１２００をプログラム１２１０に変換することで、制約式ｅにより指定された条件式の評価を優先させ、プログラムの処理量を削減できる。
【０１１８】
以上説明したように、本発明の実施の形態によれば、変数の特徴あるいは変数間の関係を制約式や出現頻度付き制約式として処理系に与えることが可能になり、その結果、多数の最適化が実施可能となる。
【０１１９】
なお、上述した制約式における組み込み関数は、引数が定数の場合は真を、定数でない場合に偽を返す定数判定組み込み関数であってもよい。
【０１２０】
また、図６および図７を用いて仮引数制約式を用いたプログラムの変換について説明したが、仮引数制約式は、関数の定義ではなく、関数のプロトタイプ宣言（関数宣言）の直前に置かれていてもよい。この場合は、関数宣言に対応する関数の定義について、図６および図７に示したのと同様の方法により最適化を行なう。
【０１２１】
つまり、制約式における擬似変数として、入力プログラム中の関数宣言における引数を指定するための変数である擬似宣言部引数を用いる。関数宣言における引数の名前は、関数の仮引数と一致しないので、関数宣言部の引数名と関数定義の仮引数、関数宣言部の引数名と関数呼び出しの実引数とを対応付けるために擬似宣言部引数を用いている。
【０１２２】
また、この擬似宣言部引数から仮引数制約式を擬似宣言部引数制約式とする。入力プログラムに関数宣言に対応する関数の定義が存在する場合に、擬似宣言部引数制約式中の擬似宣言部引数を、対応する関数の仮引数変数に置き換えた仮引数制約式を用いて、当該関数の複製を、当該仮引数制約式が真になるとみなして最適化する。また、入力プログラム中に関数宣言に対応する関数呼び出し文が存在する場合に、擬似宣言部引数制約式中の擬似宣言部引数を、対応する関数呼び出しの実引数変数に置き換えた実引数制約式が常に真である場合に、当該関数呼び出し文を、複製される関数への呼び出し文に置き換える。
【０１２３】
これにより、関数宣言に関連付けられた制約式を媒介として、関数の複製の生成と、関数呼び出しの置き換えとが、別ファイル上で行われていても、関数を最適化することができる。これにより、分割コンパイルへ対応することが可能となる。
【０１２４】
今回開示された実施の形態はすべての点で例示であって制限的なものではないと考えられるべきである。本発明の範囲は上記した説明ではなくて特許請求の範囲によって示され、特許請求の範囲と均等の意味及び範囲内でのすべての変更が含まれることが意図される。
【産業上の利用可能性】
【０１２５】
本発明にかかるプログラム変換装置は、高級言語で記述された入力プログラムから、目的機械に適したプログラムを生成するコンパイラ装置等に適用できる。
【図面の簡単な説明】
【０１２６】
【図１】プログラム変換装置の外観図である。
【図２】プログラム変換装置の機能的な構成を示すブロック図である。
【図３】プログラム変換装置が実行するプログラム変換処理を示すフローチャートである。
【図４】制約式の記述方法の一例について説明するための図である。
【図５】中間表現変換部が実行する中間表現の最適化処理の概要を示すフローチャートで

ある。
【図６】関数の仮引数、定数および演算子からなる仮引数制約式を用いた入力プログラムの変換処理を示すフローチャートである。
【図７】仮引数制約式を用いたプログラムの変換の実行例について説明するための図である。
【図８】関数の実引数、定数および演算子からなる実引数制約式を用いた入力プログラム１１０の変換処理を示すフローチャートである。
【図９】実引数制約式を用いたプログラムの変換の実行例について説明するための図である。
【図１０】関数内の変数、定数および演算子からなる関数内制約式を用いた入力プログラムの変換処理を示すフローチャートである。
【図１１】関数内制約式を用いたプログラムの変換の実行例について説明するための図である。
【図１２】関数内の変数、定数および演算子からなる関数内制約式を用いた入力プログラム１１０の変換処理を示すフローチャートである。
【図１３】関数内制約式を用いたプログラムの変換の実行例について説明するための図である。
【符号の説明】
【０１２７】
１１０　入力プログラム
１１１　制約式集合
１２０　目的プログラム
１４０　制約式データベース
１４１　中間表現データベース
２００　プログラム変換装置
２３０　中間表現生成部
２３１　制約式集合解析部
２３２　中間表現変換部
２３３　目的プログラム生成部

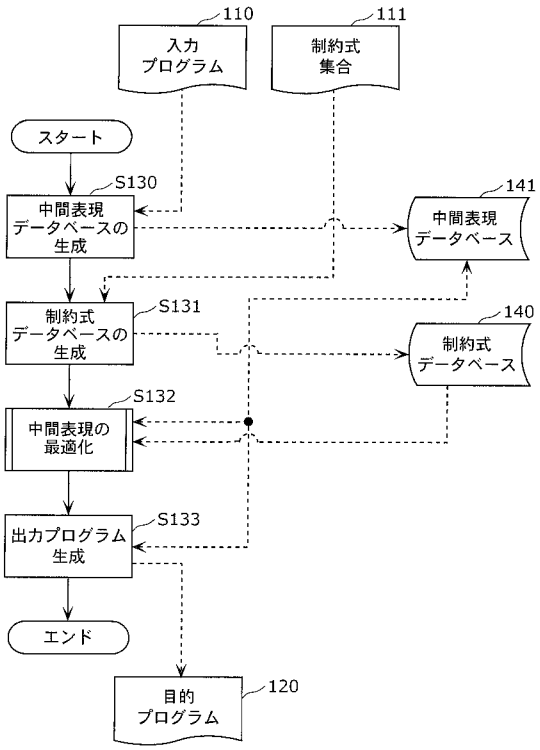
【図１】


200

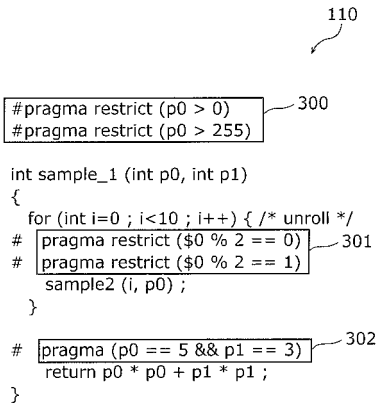
【図２】


110
入力
プログラム
111
制約式
集合
200
プログラム交換装置
中間表現
生成部
230
制約式集合
解析部
231
140
制約式
データベース
141
中間表現
データベース
232
中間表現
変換部
目的プログラム
生成部
233
目的
プログラム
120

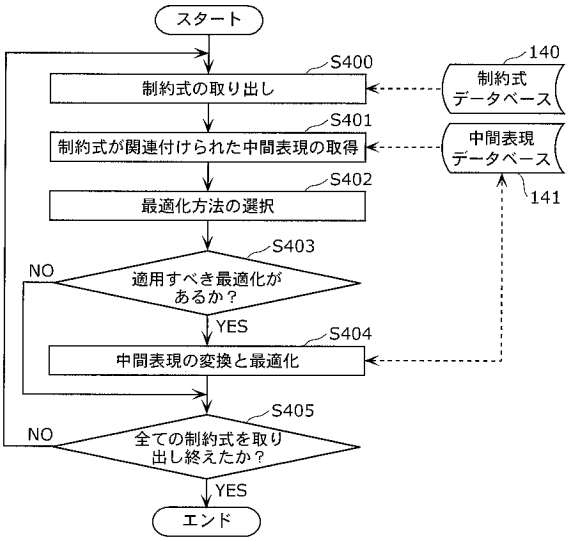
【図３】


110
入力
プログラム
111
制約式
集合
スタート
S130
中間表現
データベースの
生成
141
中間表現
データベース
制約式
データベースの
生成
S131
140
制約式
データベース
S132
中間表現の
最適化
出力プログラム
生成
S133
エンド
目的
プログラム
120

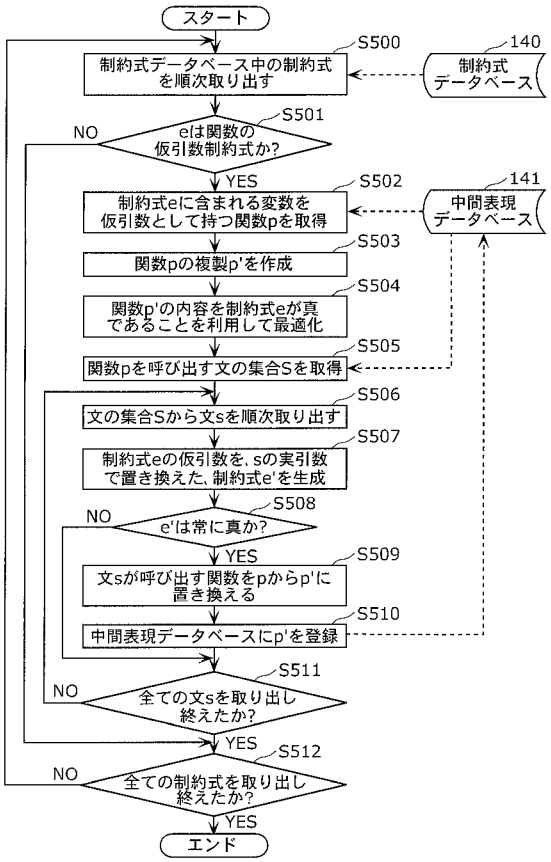
【図４】


110
#pragma restrict (p0 > 0)
#pragma restrict (p0 > 255)
300
int sample_1 (int p0, int p1)
{
for (int i=0 ; i<10 ; i++) { /* unroll */
# pragma restrict ($0 % 2 == 0)
# pragma restrict ($0 % 2 == 1)
301
sample2 (i, p0) ;
}
# pragma (p0 == 5 && p1 == 3)
302
return p0 * p0 + p1 * p1 ;
}

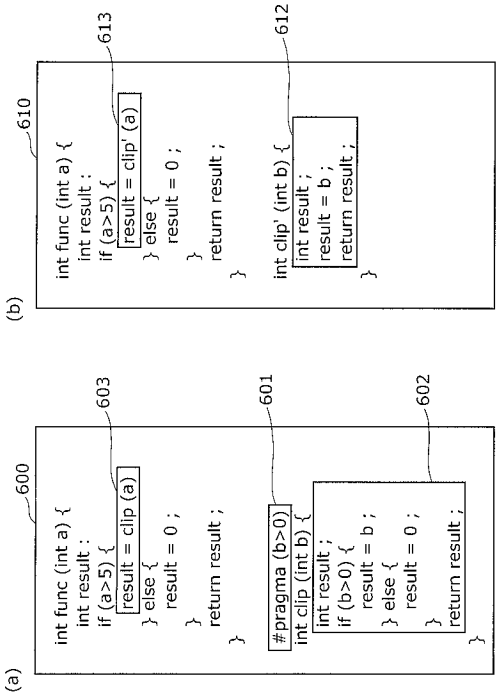
【図５】


スタート
S400
制約式の取り出し
140
制約式データベース
S401
制約式が関連付けられた中間表現の取得
中間表現データベース
141
S402
最適化方法の選択
S403
NO
適用すべき最適化があるか？
YES
S404
中間表現の変換と最適化
S405
NO
全ての制約式を取り出し終えたか？
YES
エンド

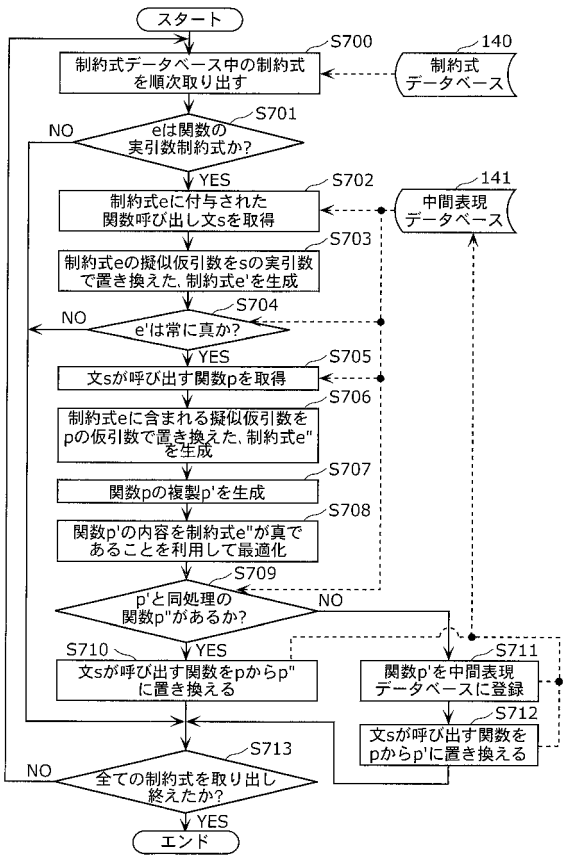
【図６】


スタート
S500
制約式データベース中の制約式を順次取り出す
140
制約式データベース
S501
NO
eは関数の仮引数制約式か?
YES
S502
制約式eに含まれる変数を仮引数として持つ関数pを取得
141
中間表現データベース
S503
関数pの複製p'を作成
S504
関数p'の内容を制約式eが真であることを利用して最適化
S505
関数pを呼び出す文の集合Sを取得
S506
文の集合Sから文sを順次取り出す
S507
制約式eの仮引数を、sの実引数で置き換えた、制約式e'を生成
S508
NO
e'は常に真か?
YES
S509
文sが呼び出す関数をpからp'に置き換える
S510
中間表現データベースにp'を登録
S511
NO
全ての文sを取り出し終えたか?
YES
S512
NO
全ての制約式を取り出し終えたか?
YES
エンド

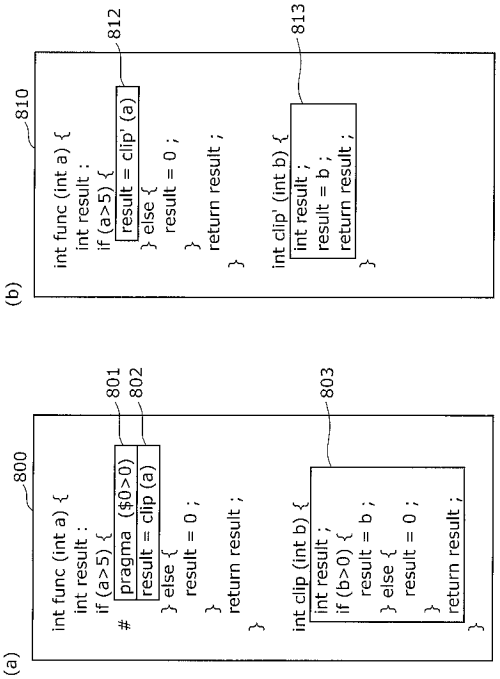
【図７】


(b)
610
613
612
int func (int a) {
int result ;
if (a>5) {
result = clip' (a)
} else {
result = 0 ;
}
return result ;
}
int clip' (int b) {
int result ;
result = b ;
return result ;
}
(a)
600
603
601
602
int func (int a) {
int result ;
if (a>5) {
result = clip (a)
} else {
result = 0 ;
}
return result ;
}
#pragma (b>0)
int clip (int b) {
int result ;
if (b>0) {
result = b ;
} else {
result = 0 ;
}
return result ;
}

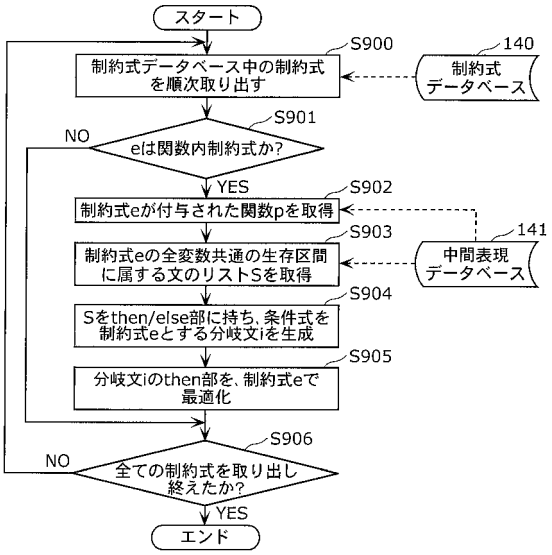
【図８】


スタート
S700
制約式データベース中の制約式を順次取り出す
140
制約式データベース
S701
NO
eは関数の実引数制約式か?
YES
S702
制約式eに付与された関数呼び出し文sを取得
141
中間表現データベース
S703
制約式eの擬似仮引数をsの実引数で置き換えた、制約式e'を生成
S704
NO
e'は常に真か?
YES
S705
文sが呼び出す関数pを取得
S706
制約式eに含まれる擬似仮引数をpの仮引数で置き換えた、制約式e"を生成
S707
関数pの複製p'を生成
S708
関数p'の内容を制約式e"が真であることを利用して最適化
S709
p'と同処理の関数p"があるか?
NO
YES
S710
文sが呼び出す関数をpからp"に置き換える
S711
関数p'を中間表現データベースに登録
S712
文sが呼び出す関数をpからp'に置き換える
S713
NO
全ての制約式を取り出し終えたか?
YES
エンド

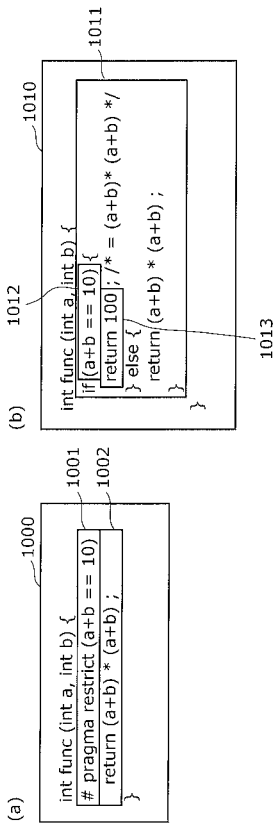
【図９】



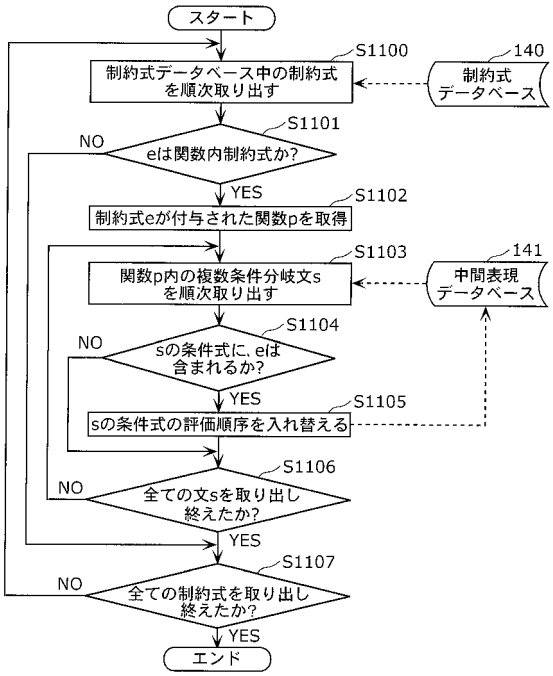
【図１０】



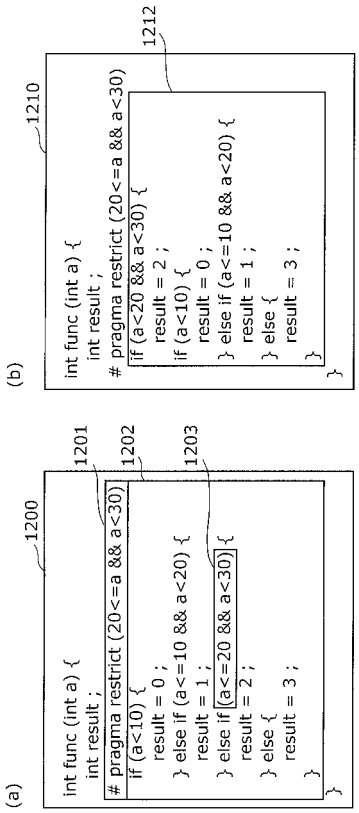
【図１１】



【図１２】



【図１３】


(a)
1200
1201
1202
1203
int func (int a) {
int result ;
# pragma restrict (20<=a && a<30)
if (a<10) {
result = 0 ;
} else if (a<=10 && a<20) {
result = 1 ;
} else if (a<=20 && a<30) {
result = 2 ;
} else {
result = 3 ;
}
}
(b)
1210
1212
int func (int a) {
int result ;
# pragma restrict (20<=a && a<30)
if (a<20 && a<30) {
result = 2 ;
if (a<10) {
result = 0 ;
} else if (a<=10 && a<20) {
result = 1 ;
} else {
result = 3 ;
}
}