(12) 特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2013 年 7 月 25 日(25.07.2013)

WIPO | PCT

(10) 国際公開番号
WO 2013/108730 A1

(51) 国際特許分類:
G06F 9/44 (2006.01) *G06F 11/36* (2006.01)

(21) 国際出願番号: PCT/JP2013/050503

(22) 国際出願日: 2013 年 1 月 15 日(15.01.2013)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(30) 優先権データ:
特願 2012-010333 2012 年 1 月 20 日(20.01.2012) JP

(71) 出願人: 日立オートモティブシステムズ株式会社 **(HITACHI AUTOMOTIVE SYSTEMS, LTD.)** [JP/JP]; 〒3128503 茨城県ひたちなか市高場２５２０番地 Ibaraki (JP).

(72) 発明者: 成沢 文雄**(NARISAWA Fumio)**; 〒3191292 茨城県日立市大みか町七丁目１番１号 株式会社 日立製作所 日立研究所内 Ibaraki (JP). 石郷岡 祐**(ISHIGOOKA Tasuku)**; 〒3191292 茨城県日立市大みか町七丁目１番１号 株式会社 日立製作所 日立研究所内 Ibaraki (JP). 月舘 統宙**(TSUKIDATE Tsunamichi)**; 〒3191292 茨城県日立市大みか町七丁目１番１号 株式会社 日立製作所 日立研究所内 Ibaraki (JP). 永浦 渉**(NAGAURA Wataru)**; 〒3128503 茨城県ひたちなか市高場２５２０番地 日立オートモティブシステムズ株式会社内 Ibaraki (JP). 櫻井 康平**(SAKURAI Kohei)**; 〒3191292 茨城県日立市大みか町七丁目１番１号 株式会社 日立製作所 日立研究所内 Ibaraki (JP). 松原 正裕**(MATSUBARA Masahiro)**; 〒3191292 茨城県日立市大みか町七丁目１番１号 株式会社 日立製作所 日立研究所内 Ibaraki (JP).

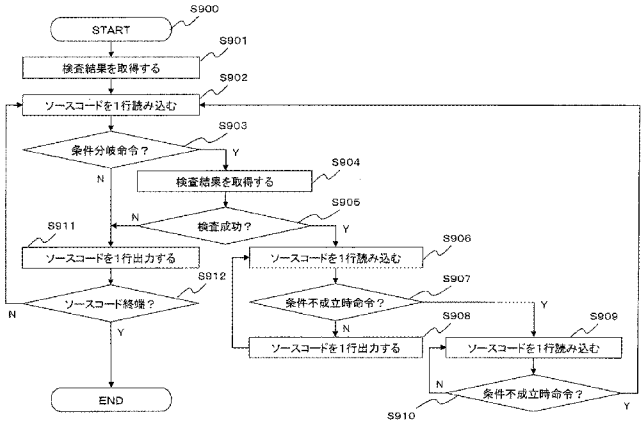
(74) 代理人: 井上 学, 外**(INOUE Manabu** et al.); 〒1008220 東京都千代田区丸の内一丁目６番１号 株式会社 日立製作所内 Tokyo (JP).

(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, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

[続葉有]

(54) **Title**: SOFTWARE VERIFICATION SUPPORT DEVICE, SOFTWARE VERIFICATION SUPPORT METHOD, SOFTWARE VERIFICATION SUPPORT PROGRAM

(54) 発明の名称 : ソフトウェア検証支援装置、ソフトウェア検証支援方法、ソフトウェア検証支援プログラム



S900 Start
S901, S904 Acquire test result
S902, S906, S909 Read one line of source code
S903 Conditional branch instruction?
S905 Test successful?
S907, S910 Instruction when condition has not been satisfied?
S908, S911 Output one line of source code
S912 End of source code?

(57) **Abstract**: The objective of the present invention is to achieve software verification support technology for achieving software which is provided with high safety and reusability and which is capable of suppressing program size. The software verification support device as laid out in the present invention, in a case in which a condition described by a conditional branch instruction is always satisfied, allows instructions executed when the condition is satisfied to remain within source code, thereupon deleting instructions which are executed when the condition is not satisfied from the source code.

(57) 要約: 高い安全性と再利用性を備え、プログラムサイズを抑えることのできるソフトウェアを得るためのソフトウェア検証支援技術を得ることを目的とする。 本発明に係るソフトウェア検証支援装置は、条件分岐命令が記述している条件が必ず成立する場合は、条件が成立したときに実行される命令をソースコード内に残した上で、条件が成立しなかったときに実行される命令をソースコードから削除する。

WO 2013/108730 A1

(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, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, RU, TJ, TM), ヨーロッパ (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, 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（条約第 21 条(3)）

明　細　書

発明の名称 ：

ソフトウェア検証支援装置、ソフトウェア検証支援方法、ソフトウェア検証支援プログラム

技術分野

[0001] 本発明は、ソフトウェアのソースコードを検証する作業を支援する技術に関する。

背景技術

[0002] 自動車などの制御装置は、マイクロコントローラ（以下マイコンと表記）を用いて制御動作を実行する。マイコンに搭載されるソフトウェアは、一般的には制御処理を実施するアプリケーションプログラムとデータ入出力を実施するデバイスドライバやオペレーティングシステム（ＯＳ）などによって構成されている。車両制御装置は、車両乗員の安全性に直接関わる制御を実施するため、高い安全性を要求されている。そのため、制御装置の故障診断を実施し、故障が検出された場合には制御装置を安全な状態へ遷移させる、フェールセーフ処理が用いられている。

[0003] また、近年制御の高度化と規模の増大にともない、車両の安全に関わる入出力装置の数が増大し、その診断ソフトウェアの開発工数が課題となっている。

[0004] 車両制御に用いるソフトウェアは、多くの製品バリエーションに対応するため、複数の製品に亘って再利用することができるように構成されており、これによりソフトウェアの開発効率を高めている。

[0005] 例えば、ソフトウェアが搭載される環境に依拠して処理内容が異なる部分については、モジュールとして切り離しておき、環境に依拠せず共通する部分を再利用することができるように構成する。また、環境に依拠する部分を切り離したモジュールについても、実行時に引き渡すパラメータを変えることによって各環境に対応した処理を実行することができるように構成し、再

利用性を高めている。

[0006] 上記のようにパラメータによって動作を変えるソフトウェアモジュールは、環境に応じて様々なパラメータを受け取る場合があるため、当該モジュールが想定していないパラメータを受け取る可能性がある。この場合、想定外のパラメータによって当該モジュールが想定外の動作を引き起こさないようにするため、受け取ったパラメータに関連する動作を実行する前に、そのパラメータが適正であるか否かを判定する必要がある。

[0007] このようなパラメータの事前検証を実施するためのソフトウェア設計手法として、「防御的プログラミング」と呼ばれる手法がある。防御的プログラミングとは、当該モジュールが想定外の動作条件で呼び出されるかもしれない、との考えに立ち、モジュールが目的とする機能を実施する前に動作条件をチェックし、条件に合致しない場合にはエラー処理などの例外処理を実施する設計手法である。

[0008] 下記特許文献１には、エンジン制御用ソフトウェアの動作検証に関する技術として、ソースコードにアサーション命令を挿入することにより、ソフトウェアの動作を検証する技術が記載されている。

先行技術文献

特許文献

[0009] 特許文献1：特開２００８－２７６５５６号公報

非特許文献

[0010] 非特許文献1：スティーブ・マコネル著，コードコンプリート第２版， 日経ＢＰソフトプレス，２００５年

発明の概要

発明が解決しようとする課題

[0011] 車両制御のようにリアルタイム制御を実施する必要があるソフトウェアは、安全の観点から制御処理の内容が正しいことが要求されるのはもちろんであるが、その制御処理が適正なタイミングで実施されることも同時に求めら

れる。そのため、数マイクロ秒程度の僅かな処理遅延も許されない場合も多い。

[0012] 一方、上述の防御的プログラミングを採用したソフトウェアモジュールは、想定外のパラメータを用いて呼び出された場合にも想定外の動作をしないように、条件分岐命令を用いてパラメータが所定の条件に合致しているか否かを判定し、条件に合致する場合のみ処理を実行するように構成される。そのため、条件分岐命令に係る条件判定、条件に合致しないときの処理等を記述する必要があり、ソースコードの量が増大する傾向がある。その結果、プログラム容量の増大と実行速度の低下を招きやすい。

[0013] 本発明は、上記のような課題を解決するためになされたものであり、高い安全性と再利用性を備え、プログラムサイズを抑えることのできるソフトウェアを得るためのソフトウェア検証支援技術を得ることを目的とする。

課題を解決するための手段

[0014] 本発明に係るソフトウェア検証支援装置は、条件分岐命令が記述している条件が必ず成立する場合は、条件が成立したときに実行される命令をソースコード内に残した上で、条件が成立しなかったときに実行される命令をソースコードから削除する。

発明の効果

[0015] 本発明に係るソフトウェア検証支援装置によれば、不要な条件分岐命令をソースコードから削除することができるので、元のソースコードの再利用性を確保しつつプログラムサイズを抑えることができる。また、必ず成立する条件分岐命令を削除するので、想定外のパラメータが条件分岐命令に引き渡されて不測の結果を生じることはなく、安全性を確保することができる。

図面の簡単な説明

[0016] [図1]本発明に係るソフトウェア検証支援装置が対象とするソフトウェアを搭載する自動車エンジン制御システムの構成例を示す図である。

[図2]本発明に係るソフトウェア検証支援装置１００の構成を示す図である。

[図3]ソフトウェア検証支援装置１００が検証対象とするソースコードの例で

ある。

[図4]図３のソースコードに検査命令を挿入した例を示す図である。

[図5]検査命令生成部１２０が条件分岐命令１７１１から条件式部分を抽出する際に用いる条件式パターン１７２０の例を示す図である。

[図6]検査命令生成部１２０が生成する検査命令のパターンを示す図である。

[図7]ソフトウェア検査支援装置１００が条件分岐命令を削除した後のソースコード例を示す図である。

[図8]ソフトウェア検査支援装置１００が検査命令を生成する処理の動作フローを示す図である。

[図9]ソフトウェア検査支援装置１００がソースコードから条件分岐命令を削除する処理の動作フローを示す図である。

[図10]実施形態２において検査実行部１４０が検査対象のソースコードをコンパイルして実行する際の動作フローである。

発明を実施するための形態

[0017] ＜実施の形態１＞

図１は、本発明に係るソフトウェア検証支援装置が対象とするソフトウェアを搭載する自動車エンジン制御システムの構成例を示す図である。コントロールユニット２００は、エンジンの動作を制御する制御ソフトウェアを実行する装置であり、マイクロコントローラ２１０、駆動回路２２０、通信回路２３０、信号入出力回路２４０を備える。

[0018] 信号入出力回路２４０は、加速度センサ３１１とヨーレートセンサ３１２からそれぞれ検出結果を取得し、マイクロコントローラ２１０に出力する。駆動回路２２０は、マイクロコントローラ２１０からの指示にしたがってアクチュエータ３２０を駆動制御し、これによりエンジンの動作を制御する。通信回路２３０は、通信線３５０を介して診断端末３３０および制御装置３４０と通信する。

[0019] 診断端末３３０は、通信線３５０を介してコントロールユニット２００を診断するための端末である。制御装置３４０は、コントロールユニット２０

０とは別の制御処理を実施する装置である。

[0020] マイクロコントローラ２１０は、入力回路２１１、演算装置２１２、ＲＡＭ（Ｒａｎｄｏｍ　Ａｃｃｅｓｓ　Ｍｅｍｏｒｙ）２１３、ＲＯＭ（Ｒｅａｄ　Ｏｎｌｙ　Ｍｅｍｏｒｙ）２１４、出力回路２１５、通信制御装置２１６を備える。

[0021] 入力回路２１１は、信号入出力回路２４０から各センサの検出結果を受け取り、演算装置２１２に出力する。演算装置２１２は、制御処理を記述した制御ソフトウェアを実行する。ＲＡＭ２１３は、演算装置２１２が制御ソフトウェアを実行する際に一時的に使用するデータなどを格納する。ＲＯＭ２１４は、演算装置２１２が実行する制御ソフトウェアを格納している。出力回路２１５は、駆動回路２２０に対して制御信号を出力する。通信制御装置２１６は、通信回路２３０を介して外部と通信する。

[0022] 上述の加速度センサ３１１、ヨーレートセンサ３１２は、コントロールユニット２００に接続されるセンサの１例でありその他のセンサ、例えば温度、電流、その他の物理量を検知するセンサの検出結果を用いて車両を制御するコントロールユニットについても、本発明に係るソフトウェア検証支援装置が対象とするソフトウェアを実行することができることはいうまでもない。

[0023] 図２は、本発明に係るソフトウェア検証支援装置１００の構成を示す図である。ソフトウェア検証支援装置１００は、制御部１１０、検査命令生成部１２０、検査命令挿入部１３０、検査実行部１４０、検査結果取得部１５０、条件分岐削除部１６０、記憶部１７０を備える。

[0024] 制御部１１０は、例えばＣＰＵ（Ｃｅｎｔｒａｌ　Ｐｒｏｃｅｓｓｉｎｇ　Ｕｎｉｔ）などの演算装置で構成され、各機能部を制御する。

[0025] 検査命令生成部１２０は、ソフトウェアソースコード内の条件分岐命令（典型的には、ｉｆ－ｅｌｓｅ文）が必ず成立するか否かを検査するための検査命令を生成する。例えば後述するａｓｓｅｒｔ命令を、検査命令として用いることができる。検査命令挿入部１３０は、ソースコード内の条件分岐命

令の直前または直後に、検査命令生成部１２０が生成した検査命令を挿入する。

[0026] 検査実行部１４０は、検査命令を挿入したソースコードをコンパイルして実行し、その結果を記憶部１７０に格納する。検査命令は、検査実行部１４０が実行すると、条件分岐命令内の条件式が成立するか否かを示すログを記憶部１７０に格納するように構成されている。したがって、検査実行部１４０が検査命令を含むソースコードをコンパイルして実行すると、条件分岐命令内の条件式が成立するか否かを示すログが記憶部１７０に格納される。

[0027] 検査結果取得部１５０は、検査実行部１４０の実行結果を記憶部１７０から取得し、上述のログにしたがって、条件分岐命令内の条件式が必ず成立するか否かを判定する。

[0028] 条件分岐削除部１６０は、条件分岐命令内の条件式が必ず成立する場合は、条件が成立したときに実行される命令（ｉｆ文内に含まれる命令）をソースコード内に残した上で、条件が成立しなかったときに実行される命令（ｅｌｓｅ文に含まれる命令）をソースコードから削除する。条件分岐命令内の条件式が必ずしも成立しない場合は、上記いずれもソースコード内に残す。

[0029] 記憶部１７０は、ソースコード記憶部１７１、条件式パターン記憶部１７２、検査命令記憶部１７３、検査対象ソースコード記憶部１７４、検査結果記憶部１７５、資源制約条件記憶部１７６を備える。

[0030] ソースコード記憶部１７１は、ソフトウェア検査支援装置１００が検査対象とするソースコードを記憶する。条件式パターン記憶部１７２は、検査命令生成部１２０が検査命令を生成する際に、条件分岐命令から条件式部分を抽出するために用いる抽出パターンを記憶する。詳細は後述する。検査命令記憶部１７３は、検査命令生成部１２０が生成した検査命令を記憶する。検査対象ソースコード記憶部１７４は、検査命令挿入部１３０が検査命令を挿入したソースコードを記憶する。検査結果記憶部１７５は、検査実行部１４０が検査命令を挿入したソースコードをコンパイルして実行した結果を記憶する。資源制約条件記憶部１７６は、検査実行部１４０がソースコードをコ

ンパイルして実行する際に必要となるハードウェアリソースなどの制約条件を記述したデータを記憶する。

[0031] 以上、ソフトウェア検証支援装置１００の構成について説明した。次に、ソースコードに検査命令を挿入する処理について、ソースコードの例とともに説明する。

[0032] 図３は、ソフトウェア検証支援装置１００が検証対象とするソースコードの例である。本ソースコードには、条件分岐命令（ｉｆ｛）１７１１、条件不成立時命令（ｅｌｓｅ｛）１７１２、条件分岐終了命令（｝）１７１３が含まれている。

[0033] 本ソースコードでは、条件分岐命令１７１１に含まれる部分において、サイズＬＩＮＩＴＨの配列を確保し、この配列の最初の要素からパラメータｉＰａｒａｍが指定する番号の要素までを初期化している。これらの処理は、配列を初期化すべきサイズがソフトウェアを実行する環境によって異なるため、これをパラメータｉＰａｒａｍとして汎用化し、任意サイズの配列を初期化することができるように構成して、環境に依拠しない再利用可能なモジュールとすることを図ったものである。

[0034] これらの処理においては、パラメータｉＰａｒａｍが配列サイズＬＩＮＩＴＨ未満であることを前提としているため、条件分岐命令１７１１によって、ＬＩＮＩＴＨ以上の値を有するｉＰａｒａｍを排除している。すなわち、本ソースコードが作成された時点では、いかなる環境を対象として本ソースコードが用いられるか確定していないため、ソフトウェアの動作を不安定にしないように、不確定なパラメータをあらかじめチェックするようにしたものである。これにより、本ソースコードの再利用性を確保しつつ、動作の安全性を確保することができる。このような手法を、前述の防御的プログラミングと呼ぶ。

[0035] 一方、本ソースコードを利用する環境が確定した以降は、パラメータｉＰａｒａｍの値が実行時にとり得る値も確定している場合がある。このような場合には、条件分岐命令１７１１、条件不成立時命令１７１２、条件分岐終

了命令１７１３は冗長な記載となり、却ってソフトウェア容量を不要に増大させる原因となる。

[0036] そこで本発明では、図３に例示するような条件分岐命令のうち、条件式が必ず成立するものについては、条件分岐命令１７１１、条件不成立時命令１７１２、および条件分岐終了命令１７１３をソースコードから削除し、さらには条件不成立時命令１７１２内に含まれるエラー処理などの不要な命令も併せて削除することとした。

[0037] 具体的には、ソフトウェア検証支援装置１００は、本ソースコードを利用する環境に固有のパラメータを組み込んだ上で、本ソースコードをコンパイルして実際に実行し、条件分岐命令１７１１が記述している条件式が実際に成立するか否かを検証する。以下、そのための具体的な手法を説明する。

[0038] なお、本ソースコードを利用する環境に固有のパラメータは、例えばＣ言語のヘッダファイルなどのデータファイルに記述することもできるし、環境固有のパラメータを一括して記述したソースコードを本ソースコードとともにコンパイルして実行モジュール内に組み込むことによって本ソースコード内に取り込むこともできる。

[0039] 一般にソースコードには、条件分岐命令が複数含まれている。必ずしも全ての条件分岐命令が、上述のような冗長命令であるわけではないので、ソースコードが利用される環境に依拠して変わる可能性がある条件式を記述した部分のみ、削除するか否かを判定すればよい。そこで図３に示すソースコードには、削除するか否かを判定する対象とする条件分岐命令を識別するため、ソースコードの内容に影響しないコメント文として、条件分岐命令識別コメント１７１４、条件不成立時命令識別コメント１７１５、条件分岐終了命令識別コメント１７１６が付与されている。

[0040] 条件分岐命令識別コメント１７１４、条件不成立時命令識別コメント１７１５、条件分岐終了命令識別コメント１７１６はそれぞれ、条件分岐命令１７１１、条件不成立時命令１７１２、および条件分岐終了命令１７１３が記述されている箇所をソフトウェア検証支援装置１００が識別するためのもの

である。コメント文の形式は任意でよいが、図３では例として、「@ＣＣ__ｉｆ__識別番号」「@ＣＣ__ｅｌｓｅ__識別番号」「@ＣＣ__ｅｎｄｉｆ__識別番号」という形式でコメントを付与した。

[0041] 条件分岐命令１７１１が記述している条件式が実際に成立するか否かを検証するためには、ソースコードをコンパイルして実行することに加えて、条件式が成立した際にその旨を何らかのログとして記録することが望ましい。例えば、条件分岐命令１７１１内に、適当なログを記録する命令を新たに挿入することが考えられる。条件分岐命令１７１１内の命令は、その条件式が成立した場合に限って実行されるので、条件分岐命令１７１１内にログ記録命令を挿入しておけば、その命令が実行されたということは条件式が成立したということと等価だからである。

[0042] 本実施形態１では、これと同様の機能を備えた関数として、ソフトウェアの動作検証においてよく用いられるａｓｓｅｒｔ命令を用いる。ａｓｓｅｒｔ命令は、指定した条件式が成立したときはｔｒｕｅを返し、成立しなかったときはｆａｌｓｅを返す、またはこれと同様の動作をする関数である。本実施形態１では、次の図４に示す__ＣＰＲＯＶＥＲ__ａｓｓｅｒｔという検査命令を用いることとする。これは、指定した条件式が成立したか否かを示すログを、検査結果記憶部１７５に格納するように構成された命令である。

[0043] 図４は、図３のソースコードに検査命令を挿入した例を示す図である。図４において、条件分岐命令１７１１の直前に、検査命令１７１７が新たに挿入されている。

[0044] 検査命令挿入部１３０は、条件分岐命令識別コメント１７１４をソースコード内に見つけると、対応する条件分岐命令１７１１が必ず成立するか否かを検証するための検査命令１７１７を、条件分岐命令１７１１の直前（または直後）に挿入する。

[0045] 検査命令１７１７は、条件分岐命令１７１１が成立するか否かを検証するためのものであるため、検査命令１７１７が指定する条件式は条件分岐命令１７１１内の条件式と同一である必要がある。検査命令生成部１２０は、条

件分岐命令１７１１から条件式部分を抽出し、これを条件式として指定した検査命令１７１７を生成する。

[0046] 図５は、検査命令生成部１２０が条件分岐命令１７１１から条件式部分を抽出する際に用いる条件式パターン１７２０の例を示す図である。条件式パターンとは、プログラムコード内において条件式が開始または終了する旨を指定する文字列と、条件式そのものを記述した文字列とを抽出するための文字列テンプレートであり、条件式パターン記憶部１７２内にデータとして記憶されている。例えば正規表現を用いて条件式パターン１７２０を記述することができるが、これに限られるものではなく、指定した文字列表現に合致する文字列を抽出することができればよい。図５では、正規表現を用いて条件式パターン１７２０を構成した例を示した。

[0047] 図５に示す条件式パターン１７２０は、条件式が開始または終了する旨を指定する文字列を抽出するための定型パターン１７２１と、定型パターン１７２１の間に記述されている任意の文字列を抽出するための任意パターン１７２２とを有する。検査命令生成部１２０は、条件式パターン１７２０を用いて、ｉｆ文内に記述されている条件式（図３～図４の例では、ｉＰａｒａｍ＜ＬＩＮＩＴＨ）を抽出することができる。

[0048] なお、条件式パターン１７２０は、複数の抽出パターンを記述することもできる。検査命令生成部１２０は、いずれかの抽出パターンに合致する文字列を発見したときはその条件式を用いて検査命令１７１７を生成し、全ての抽出パターンに合致しない条件分岐命令については空の条件を指定して検査命令１７１７を生成する。

[0049] 図６は、検査命令生成部１２０が生成する検査命令のパターンを示す図である。検査命令生成部１２０は、図６に示す検査命令パターン１７３０に準じて検査命令を生成し、検査命令記憶部１７３に格納する。検査命令挿入部１３０は、検査命令記憶部１７３が格納している検査命令を読み出し、ソースコードに挿入する。

[0050] 検査命令パターン１７３０は、行番号部１７３１、挿入箇所指示部１７３

２、検査命令固定部１７３３、条件式部１７３４を有する。

[0051] 行番号部１７３１は、検査命令を挿入すべきソースコード内の行番号を指定する。挿入箇所指示部１７３２は、行番号部１７３１が指定する行の直前に検査命令を挿入するか、直後に挿入するかを指定する。検査命令固定部１７３３は、検査命令を記述した文字列のうち、条件式を除いた固定部分である。条件式部１７３４は、検査命令生成部１２０が条件式パターン１７２０を用いて条件分岐命令から抽出する条件式を埋め込む部分である。

[0052] 図７は、ソフトウェア検査支援装置１００が条件分岐命令を削除した後のソースコード例を示す図である。図３～図４で例示したソースコードのうち、条件分岐命令１７１１、条件不成立時命令１７１２、条件分岐終了命令１７１３、条件分岐命令識別コメント１７１４、条件不成立時命令識別コメント１７１５、条件分岐終了命令識別コメント１７１６、検査命令１７１７がソースコードから削除され、さらには条件不成立時命令１７１２内に含まれるエラー処理などの不要な命令も併せて削除されている。この結果、削除した部分に対応するオブジェクトコードは生成されないので、プログラム容量およびプログラム実行負荷を低減することができる。

[0053] 以上、ソフトウェア検査支援装置１００がソースコードから条件分岐命令を削除する際の基本的な考え方について説明した。以下では、ソフトウェア検査支援装置１００の具体的な動作手順を説明する。

[0054] 図８は、ソフトウェア検査支援装置１００が検査命令を生成する処理の動作フローを示す図である。以下、図８の各ステップについて説明する。

（図８：ステップＳ８００）

ソフトウェア検査支援装置１００は、検査対象であるソースコードをコンパイルして実行する前に、本動作フローを実行して検査命令を生成する。

（図８：ステップＳ８０１）

検査命令生成部１２０は、ソースコード記憶部１７１から検査対象であるソースコードを１行分読み出す。また、図６で説明した行番号部１７３１の値を１加算して更新する。

[0055]（図８：ステップＳ８０２）

検査命令生成部１２０は、ステップＳ８０１で読み込んだ１行分のソースコード内に、条件分岐命令識別コメント１７１４（＠ＣＣ＿ｉｆ＿識別番号）が含まれているか否かを判定する。含まれている場合はステップＳ８０３へ進み、含まれていない場合はステップＳ８０５へスキップする。

（図８：ステップＳ８０３）

検査命令生成部１２０は、条件分岐命令に含まれる条件式を、図５で説明した条件式パターンにしたがって抽出する。

[0056]（図８：ステップＳ８０４）

検査命令生成部１２０は、ステップＳ８０３で抽出した条件式と、図６で説明した検査命令パターンにしたがって、検査命令を生成し、検査命令記憶部１７３に格納する。行番号部１７３１には、ステップＳ８０１で更新した行番号を用いる。挿入箇所指示部１７３２の値は、例えばあらかじめ指定しておけばよい。また、条件分岐命令識別コメント１７１４に含まれている識別番号を記憶しておき、検査命令の実行結果を検査結果記憶部１７５に格納する際に、併せて格納する。

（図８：ステップＳ８０５）

ステップＳ８０１で読み込んだソースコードの終端に達している場合はステップＳ８０６へ進み、読み込んでいない行が残っている場合はステップＳ８０１に戻って同様の処理を繰り返す。

[0057]（図８：ステップＳ８０６）

本動作フローが終了した後、検査命令挿入部１３０は、検査命令生成部１２０が生成して検査命令記憶部１７３に格納した検査命令を、行番号部１７３１が指定する行の直前または直後に挿入し、検査対象ソースコード記憶部１７４に格納する。検査実行部１４０は、検査命令が挿入されたソースコードをコンパイルして実行し、検査命令の実行結果を検査結果記憶部１７５に格納する。また、必要に応じて資源制約条件記憶部１７６からハードウェアリソースに関する制約条件を取得し、これを満たす範囲内で検査を実施する

。

[0058] 図９は、ソフトウェア検査支援装置１００がソースコードから条件分岐命令を削除する処理の動作フローを示す図である。以下、図９の各ステップについて説明する。

（図９：ステップＳ９００）

ソフトウェア検査支援装置１００は、検査対象であるソースコードをコンパイルして実行した後に本動作フローを実行し、条件分岐命令をソースコードから削除する。

（図９：ステップＳ９０１）

検査結果取得部１５０は、検査結果記憶部１７５から検査命令の実行結果を取得する。検査命令の実行結果は、ソースコード内に複数存在する検査命令それぞれの条件式が成立したか否か、および検査命令の識別番号（条件分岐命令識別コメント１７１４に含まれている識別番号）を含む。検査結果取得部１５０は、各検査命令を識別番号によって区別した上で、それぞれの検査結果をリストアップする。

[0059] （図９：ステップＳ９０２）

条件分岐削除部１６０は、検査命令を挿入したソースコードを、検査対象ソースコード記憶部１７４から１行ずつ読み出す。

（図９：ステップＳ９０３）

条件分岐削除部１６０は、ステップＳ９０２で読み込んだ１行分のソースコード内に、条件分岐命令識別コメント１７１４が含まれているか否かを判定する。含まれている場合はステップＳ９０４に進み、含まれていない場合はステップＳ９１１へ進む。

[0060] （図９：ステップＳ９０４）

条件分岐削除部１６０は、条件分岐命令識別コメント１７１４に含まれている識別番号をキーにして、検査結果記憶部１７５から当該条件分岐命令に対応する検査命令の検査結果を取得する。

（図９：ステップＳ９０５）

ステップＳ９０４で取得した検査結果が検査成功である場合はステップＳ９０６へ進み、検査失敗である場合はステップＳ９１１へ進む。

（図９：ステップＳ９０５：補足）

本ステップにおける検査結果が検査成功である場合は、対応する条件分岐命令内の条件文が成立していることを示しているので、ステップＳ９０６以降のステップに進んで冗長な命令文を削除する。本ステップにおける検査結果が失敗である場合は、ステップＳ９０２で読み込んだ条件分岐命令内の条件文が必ずしも成立するとは限らないことを示しているので、ステップＳ９１１に進んでその条件分岐命令はソースコード内に残す。

[0061] （図９：ステップＳ９０６）

条件分岐削除部１６０は、直前に読み込んだソースコード行の次行を、検査対象ソースコード記憶部１７４から読み出す。

（図９：ステップＳ９０７）

条件分岐削除部１６０は、ステップＳ９０６で読み込んだ１行分のソースコード内に、条件不成立時命令識別コメント１７１５が含まれているか否かを判定する。含まれている場合はステップＳ９０９に進み、含まれていない場合はステップＳ９０８へ進む。

[0062] （図９：ステップＳ９０８）

条件分岐削除部１６０は、ステップＳ９０６で読み込んだ１行分のソースコードを、そのまま検査対象ソースコード記憶部１７４に書き戻す。本ステップの後はステップＳ９０６に戻り、同様の処理を繰り返す。

（図９：ステップＳ９０９）

条件分岐削除部１６０は、直前に読み込んだソースコード行の次行を、検査対象ソースコード記憶部１７４から読み出す。

[0063] （図９：ステップＳ９１０）

条件分岐削除部１６０は、ステップＳ９０９で読み込んだ１行分のソースコード内に、条件分岐終了命令識別コメント１７１６が含まれているか否かを判定する。含まれていない場合はステップＳ９０９へ戻って同様の処理を

繰り返す。含まれている場合はステップＳ９０２に戻って同様の処理を繰り返す。

（図９：ステップＳ９１１）

条件分岐削除部１６０は、ステップＳ９０２で読み込んだ１行分のソースコードを、そのまま検査対象ソースコード記憶部１７４に書き戻す。

（図８：ステップＳ９１２）

ソースコードの終端に達している場合は本動作フローを終了し、読み込んでいない行が残っている場合はステップＳ９０２に戻って同様の処理を繰り返す。

[0064] ＜実施の形態１：まとめ＞

以上のように、本実施形態１に係るソフトウェア検証支援装置１００は、検査命令を挿入したソースコードをコンパイルして実行し、条件分岐命令内の条件式が必ず成立することが判明した場合は、冗長な条件分岐命令などをソースコード内から削除する。これにより、不要な条件分岐命令、条件不成立時命令、条件分岐終了命令、および条件不成立時命令内に含まれる命令を除去し、コンパイル後のプログラム容量を抑えることができる。また、冗長な命令を削除することにより、処理負荷を軽減して制御処理のリアルタイム性を高めることができる。

[0065] また、本実施形態１に係るソフトウェア検証支援装置１００は、検査命令を挿入したソースコードをコンパイルして実行する際に、ソースコードが利用される環境に固有のパラメータを記述したデータファイルを取り込み、検証対象であるソースコードと組み合わせて実行する。これにより、検査対象のソースコードが利用される環境を模擬し、条件分岐命令内の条件式が当該環境下で確実に成立するか否かを、シミュレーション環境下で検証することができる。

[0066] ＜実施の形態２＞

実施の形態１では、検査実行部１４０が検査命令を埋め込んだソースコードをコンパイルして実行することを説明した。一般に、ソースコードをコン

パイルして実際に実行することによりソースコードの挙動を検査する処理は、ＣＰＵ負荷やメモリ使用量などのハードウェアリソースを多大に消費するため、ハードウェアリソースに関する何らかの制約条件が課される場合がある。本発明の実施形態２では、その具体例を説明する。検査実行部１４０の動作以外については実施形態１と概ね同様であるため、以下では差異点を中心に説明する。

[0067] 図１０は、本実施形態２において検査実行部１４０が検査対象のソースコードをコンパイルして実行する際の動作フローである。以下、図１０の各ステップについて説明する。

（図１０：ステップＳ１０００）

検査実行部１４０は、図８のステップＳ８０６において、本動作フローを開始する。

（図１０：ステップＳ１００１～Ｓ１００２）

検査実行部１４０は、検査対象ソースコード記憶部１７４から検査対象のソースコードを読み出す（Ｓ１００１）。また、資源制約条件記憶部１７６からハードウェアリソースに関する制約条件を取得する（Ｓ１００２）。

（図１０：ステップＳ１００２：補足）

本実施形態２において、資源制約条件記憶部１７６が記憶しているハードウェアリソースに関する制約条件は、使用メモリ量の最大値と、検査開始からの最大経過時間である。

[0068] （図１０：ステップＳ１００３）

検査実行部１４０は、ステップＳ１００１で読み出した検査対象ソースコードをコンパイルし、実行開始する。本ステップの処理は、一般的なソフトウェア検証ツールを用いて実行することができる。例えば、ＣＢＭＣ（Ｅｄｍｕｎｄ Ｃｌａｒｋｅ著，ＡＮＳＩ－Ｃ Ｂｏｕｎｄｅｄ Ｍｏｄｅｌ Ｃｈｅｃｋｅｒ Ｕｓｅｒ Ｍａｎｕａｌ，２００６年を参照）などのツールを用いることができる。同ツールは、ソフトウェアソースコード内に記載された条件が満たされているか否かを網羅的に検証するソフトウェアである

。

（図１０：ステップＳ１００４）

検査実行部１４０は、検査を実行しながら所定時間待機する。

[0069] （図１０：ステップＳ１００５）

検査実行部１４０は、現在のメモリ使用量を取得する。メモリ使用量を取得する具体的な方法としては、検査プログラムが動作している計算機のオペレーティングシステムのシステムコールなどを用いればよい。取得したメモリ使用量が資源制約条件記憶部１７６によって規定されている最大使用量を超えている場合はステップＳ１００７へ進み、超えていない場合はステップＳ１００６へ進む。

（図１０：ステップＳ１００６）

検査実行部１４０は、現在時刻を取得し、検査を開始してからの経過時間を算出する。

[0070] （図１０：ステップＳ１００７）

検査実行部１４０は、ステップＳ１００３で開始した検査を中断し、使用メモリ量超過エラー処理を実行する。例えば、検査を中断した時点で実行していたソースコードの行番号を、検査結果記憶部１７５に格納する。

（図１０：ステップＳ１００８）

検査実行部１４０は、検査を開始してからの経過時間が資源制約条件記憶部１７６によって規定されている最大経過時間を超えている場合はステップＳ１０１０へ進み、超えていない場合はステップＳ１００９へ進む。

[0071] （図１０：ステップＳ１００９）

検査実行部１４０は、ステップＳ１００３で開始した検査が終了している場合は本動作フローを終了し、検査途中である場合はステップＳ１００４に戻って同様の処理を繰り返す。

（図１０：ステップＳ１０１０）

検査実行部１４０は、ステップＳ１００３で開始した検査を中断し、経過時間超過エラー処理を実行する。例えば、検査を中断した時点で実行してい

たソースコードの行番号を、検査結果記憶部１７５に格納する。

[0072] （図１０：ステップＳ１０１１）

本動作フローが終了した後、ソフトウェア検査支援装置１００は、図９で説明した動作フローを実行する。ステップＳ１００７またはＳ１０１０においてそれぞれの超過処理を実行した旨の検査結果が記録されている場合は、検査を中断した以降のソースコード部分については無条件に残すこととする。条件式が成立するか否かを検証できていないからである。

[0073] ＜実施の形態２：まとめ＞

以上のように、本実施形態２に係るソフトウェア検査支援装置１００は、検査実行部１４０がソースコードをコンパイルして実行するために必要なハードウェアリソース量が所定の制約条件を超過したときは、検査を中断してその旨を記録する。これにより、ハードウェア能力不足のため検査がいつまでも終了しないような事態を避けることができる。

[0074] また、本実施形態２に係るソフトウェア検査支援装置１００は、検査実行部１４０が検査を途中で中断したときは、それ以降のソースコードを無条件に残す。これにより、冗長であることが確実に判明している条件分岐命令等のみをソースコードから削除することができる。

[0075] 以上、本発明者によってなされた発明を実施形態に基づき具体的に説明したが、本発明は前記実施の形態に限定されるものではなく、その要旨を逸脱しない範囲で種々変更可能であることは言うまでもない。

[0076] また、上記各構成、機能、処理部などは、それらの全部または一部を、例えば集積回路で設計することによりハードウェアとして実現することもできるし、プロセッサがそれぞれの機能を実現するプログラムを実行することによりソフトウェアとして実現することもできる。各機能を実現するプログラム、テーブルなどの情報は、メモリやハードディスクなどの記憶装置、ＩＣカード、ＤＶＤなどの記憶媒体に格納することができる。

符号の説明

[0077] １００：ソフトウェア検証支援装置、１１０：制御部、１２０：検査命令

生成部、１３０：検査命令挿入部、１４０：検査実行部、１５０：検査結果取得部、１６０：条件分岐削除部、１７０：記憶部、１７１：ソースコード記憶部、１７２：条件式パターン記憶部、１７３：検査命令記憶部、１７４：検査対象ソースコード記憶部、１７５：検査結果記憶部、１７６：資源制約条件記憶部、２００：コントロールユニット、２１０：マイクロコントローラ、２１１：入力回路、２１２：演算装置、２１３：ＲＡＭ、２１４：ＲＯＭ、２１５：出力回路、２１６：通信制御装置、２２０：駆動回路、２３０：通信回路、２４０：信号入出力回路、３１１：加速度センサ、３１２：ヨーレートセンサ、３２０：アクチュエータ、３３０：診断端末、３４０：制御装置、３５０：通信線。

請求の範囲

［請求項1］ ソフトウェアのソースコードを検証する作業を支援する装置であって、

前記ソースコードをコンパイルして実行した結果を取得する検査部、

前記ソースコード内の条件分岐命令が記述している条件が必ず成立するか否かを前記結果にしたがって判定する検査結果取得部、

前記条件が必ず成立する場合は、前記条件が成立したときに実行される命令を前記ソースコード内に残した上で、前記条件が成立しなかったときに実行される命令を前記ソースコードから削除する条件分岐削除部、

を備えることを特徴とするソフトウェア検証支援装置。

［請求項2］ 前記検査部は、

前記条件分岐命令が記述している条件を指定するパラメータのうち、前記ソフトウェアが搭載される環境に固有のものを記述したパラメータデータを、前記ソースコードに組み込んだ上でコンパイルして実行した結果を取得し、

前記検査結果取得部は、

前記パラメータデータを組み込んだ前記ソースコードの実行結果を取得することにより、前記環境において前記条件が必ず成立するか否かを判定する

ことを特徴とする請求項１記載のソフトウェア検証支援装置。

［請求項3］ 前記条件分岐命令が必ず成立するか否かを検査するための検査命令を前記ソースコード内の前記条件分岐命令の直前または直後に挿入する検査命令挿入部と、

前記ソースコードをコンパイルして実行した結果を記述したログを記憶する記憶部と、 を備え、

前記検査命令は、

前記検査部が前記検査命令を実行したとき、前記条件分岐命令が成立したか否かを示すログを前記記憶部に格納するように構成されており、

前記検査命令挿入部は、

前記検査部が前記ソースコードをコンパイルして実行前に、前記検査命令を前記ソースコード内に挿入し、

前記検査結果取得部は、

前記記憶部に格納されている前記検査命令のログに基づき、前記条件が必ず成立するか否かを判定する

ことを特徴とする請求項１記載のソフトウェア検証支援装置。

［請求項4］ 前記検査命令挿入部は、

前記検査命令を挿入すべき条件分岐命令が記述されている箇所を示す前記ソースコード内のコメント文にしたがって、前記検査命令を挿入する

ことを特徴とする請求項３記載のソフトウェア検証支援装置。

［請求項5］ 前記検査部は、

前記ソースコードをコンパイルして実行するために必要なハードウェアリソース量が所定の上限値を超えたときは前記実行を中断し、

前記検査結果取得部は、

前記条件分岐命令のうち前記中断時点で既に実行されたもの以外については、前記条件が必ず成立するか否か不確定であるものと判定し、

前記条件分岐削除部は、

前記条件が必ず成立するか否か不確定である前記条件分岐命令、その条件が成立したときに実行される命令、およびその条件が成立しなかったときに実行される命令については、前記ソースコードから削除せずに残したままにする

ことを特徴とする請求項１記載のソフトウェア検証支援装置。

[請求項6] 前記検査部は、

前記ソースコードをコンパイルして実行する際に消費するメモリ容量が所定の上限値を超えたときは前記実行を中断する

ことを特徴とする請求項５記載のソフトウェア検証支援装置。

[請求項7] 前記検査部は、

前記ソースコードをコンパイルして実行することを開始してからの経過時間が所定の上限値を超えたときは前記実行を中断する

ことを特徴とする請求項５記載のソフトウェア検証支援装置。

[請求項8] 前記検査部は、

前記実行を中断したときはその時点で実行しようとしていた前記ソースコードの行番号を前記記憶部に格納する

ことを特徴とする請求項５記載のソフトウェア検証支援装置。

[請求項9] ソフトウェアのソースコードを検証する作業を支援する方法であって、

前記ソースコードをコンパイルして実行した結果を取得するステップ、

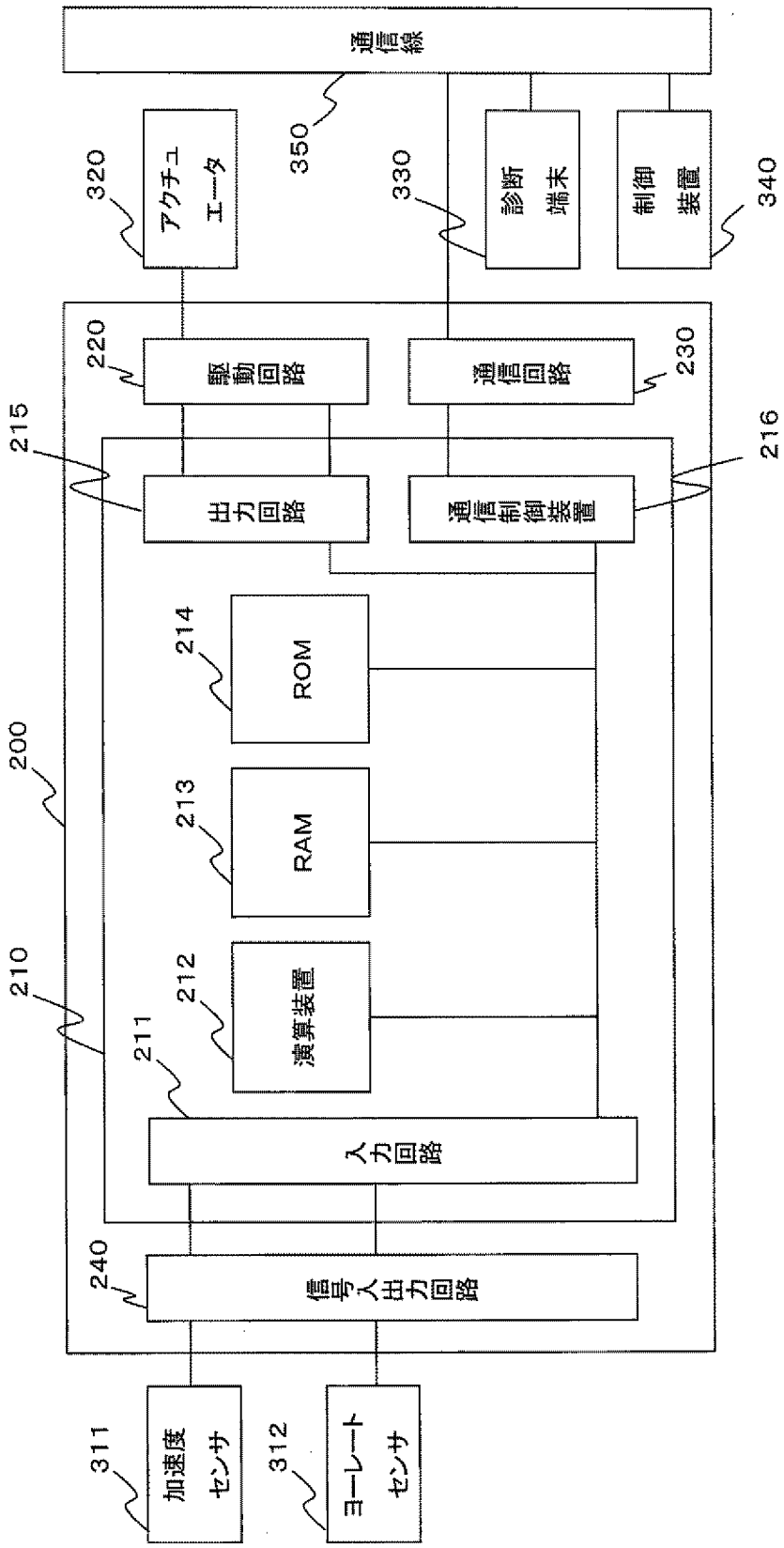
前記ソースコード内の条件分岐命令が記述している条件が必ず成立するか否かを前記結果にしたがって判定するステップ、

前記条件が必ず成立する場合は、前記条件が成立したときに実行される命令を前記ソースコード内に残した上で、前記条件が成立しなかったときに実行される命令を前記ソースコードから削除するステップ、

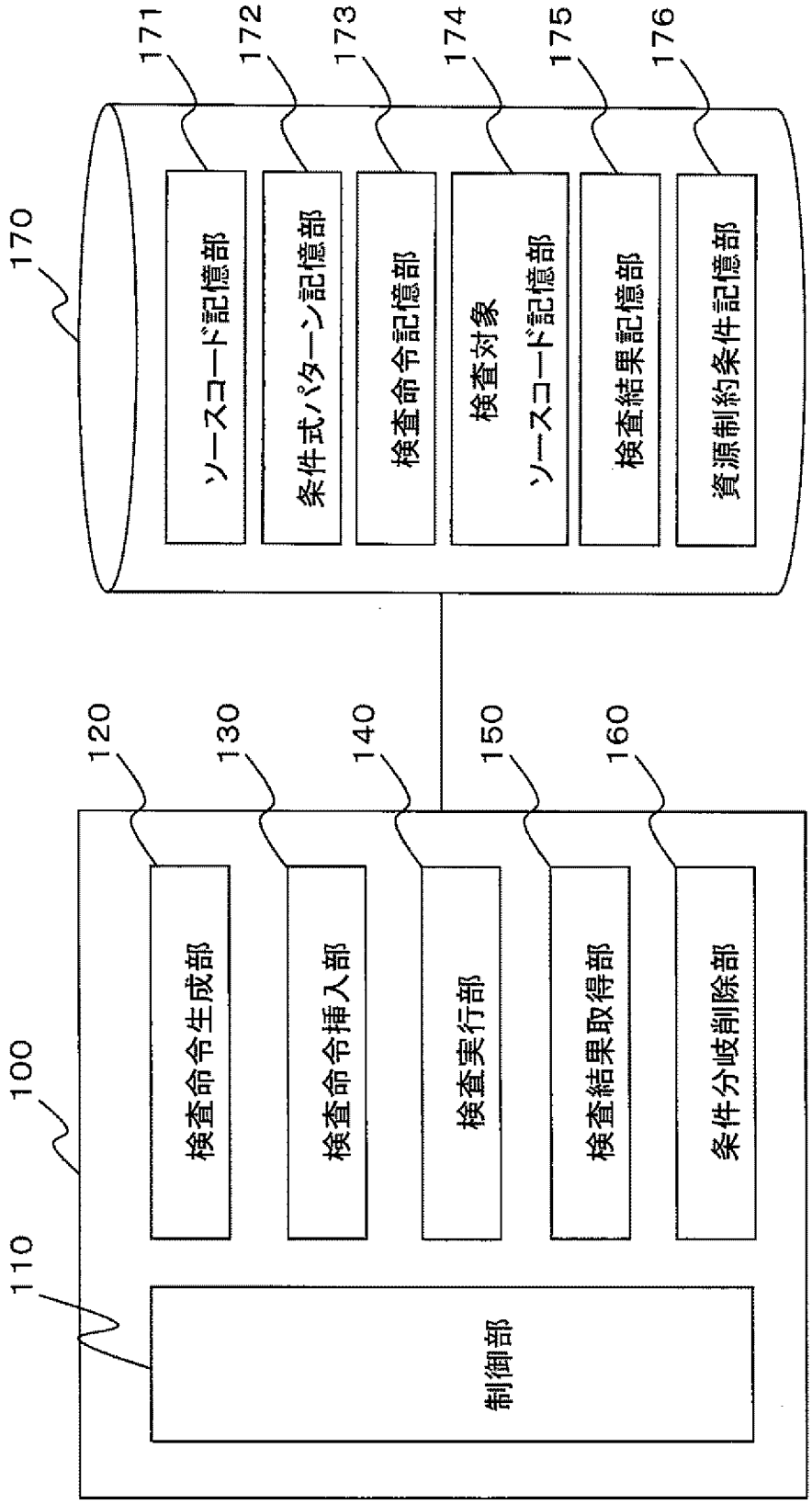
を有することを特徴とするソフトウェア検証支援方法。

[請求項10] 請求項９記載のソフトウェア検証支援方法をコンピュータに実行させることを特徴とするソフトウェア検証支援プログラム。

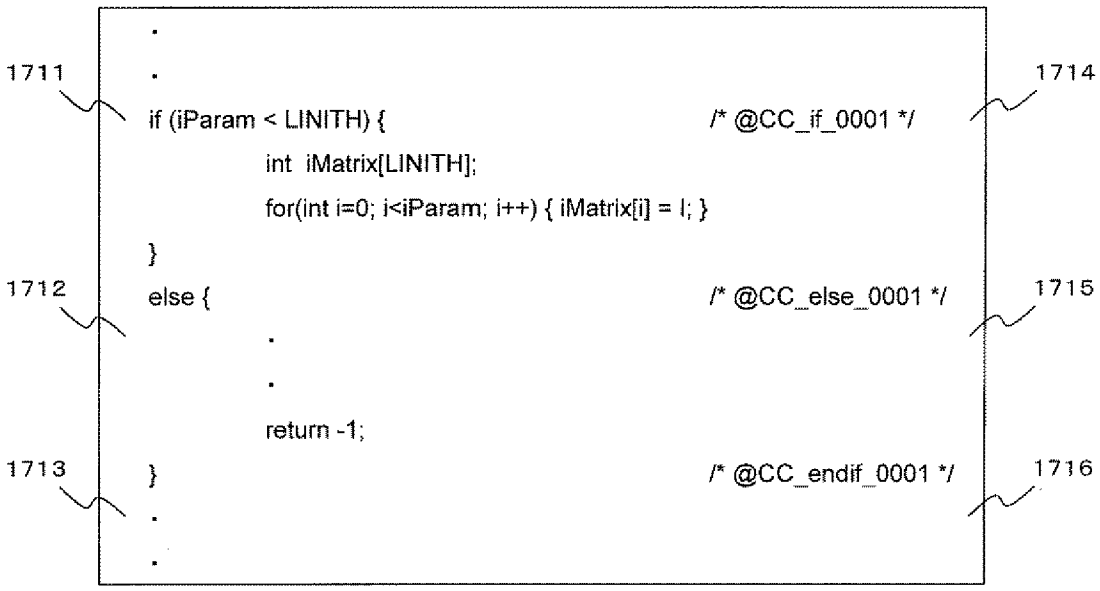
[図1]


通信線
350
320
アクチュエータ
330
診断端末
340
制御装置
220
駆動回路
通信回路
230
215
216
出力回路
通信制御装置
214
ROM
200
213
RAM
210
212
演算装置
211
入力回路
240
信号入出力回路
311
加速度センサ
312
ヨーレートセンサ

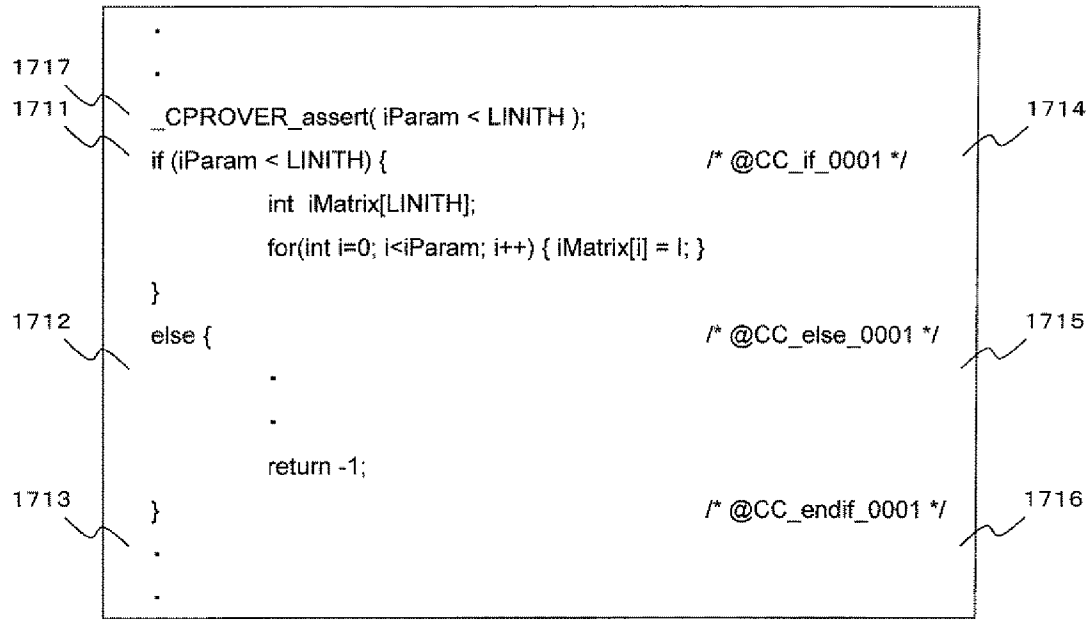
[図2]


100
110
制御部
120
検査命令生成部
130
検査命令挿入部
140
検査実行部
150
検査結果取得部
160
条件分岐削除部
170
171
ソースコード記憶部
172
条件式パターン記憶部
173
検査命令記憶部
174
検査対象
ソースコード記憶部
175
検査結果記憶部
176
資源制約条件記憶部

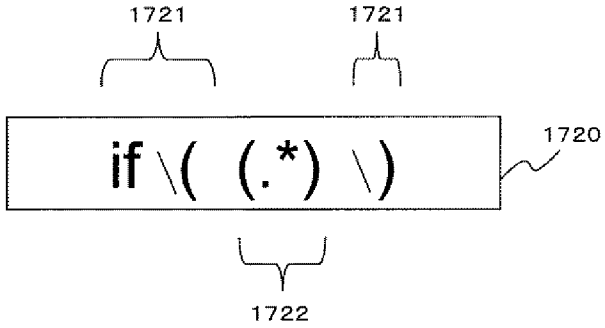
[図3]


1711
1712
1713
1714
1715
1716
if (iParam < LINITH) { /* @CC_if_0001 */
int iMatrix[LINITH];
for(int i=0; i<iParam; i++) { iMatrix[i] = I; }
}
else { /* @CC_else_0001 */
return -1;
} /* @CC_endif_0001 */

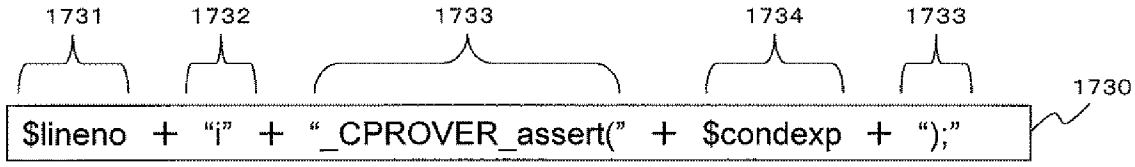
[図4]


1717
1711
1712
1713
1714
1715
1716
_CPROVER_assert(iParam < LINITH);
if (iParam < LINITH) { /* @CC_if_0001 */
int iMatrix[LINITH];
for(int i=0; i<iParam; i++) { iMatrix[i] = I; }
}
else { /* @CC_else_0001 */
return -1;
} /* @CC_endif_0001 */

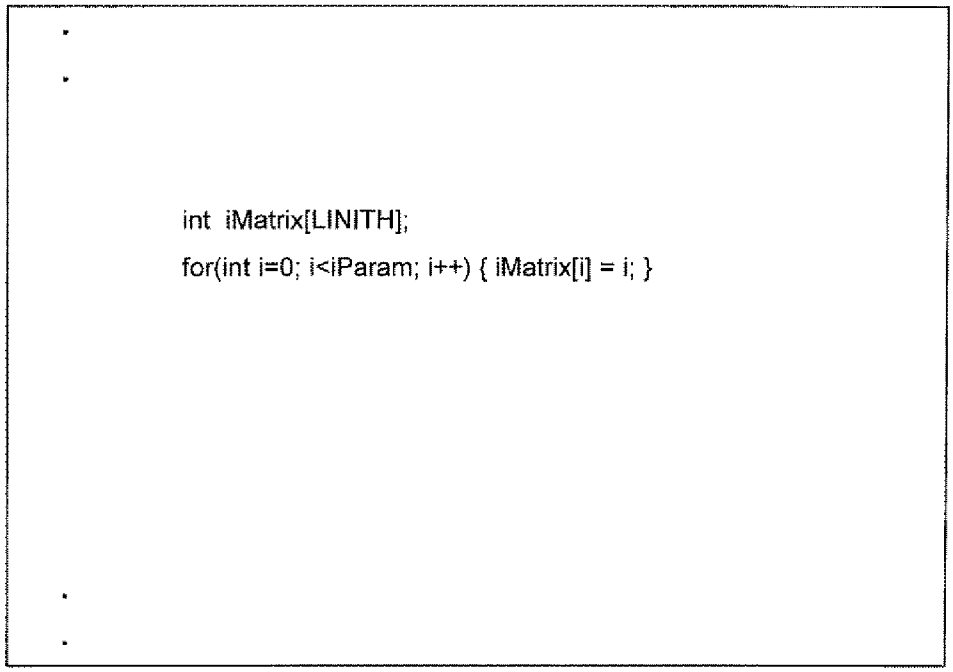
[図5]


1721
1721
if \((.*) \)
1720
1722

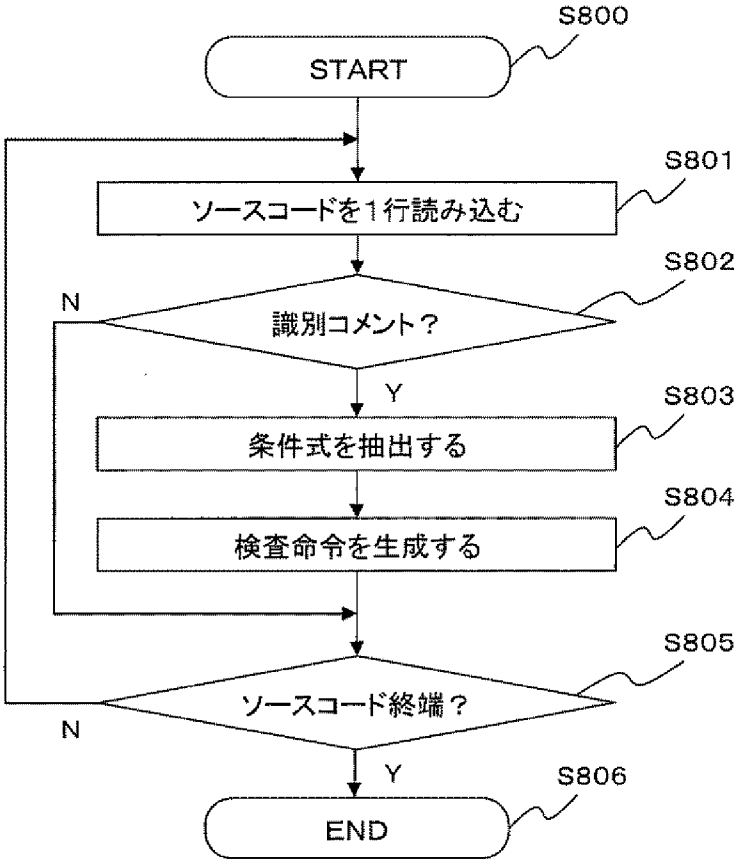
[図6]


1731
1732
1733
1734
1733
1730
$lineno + "i" + "_CPROVER_assert(" + $condexp + ");"

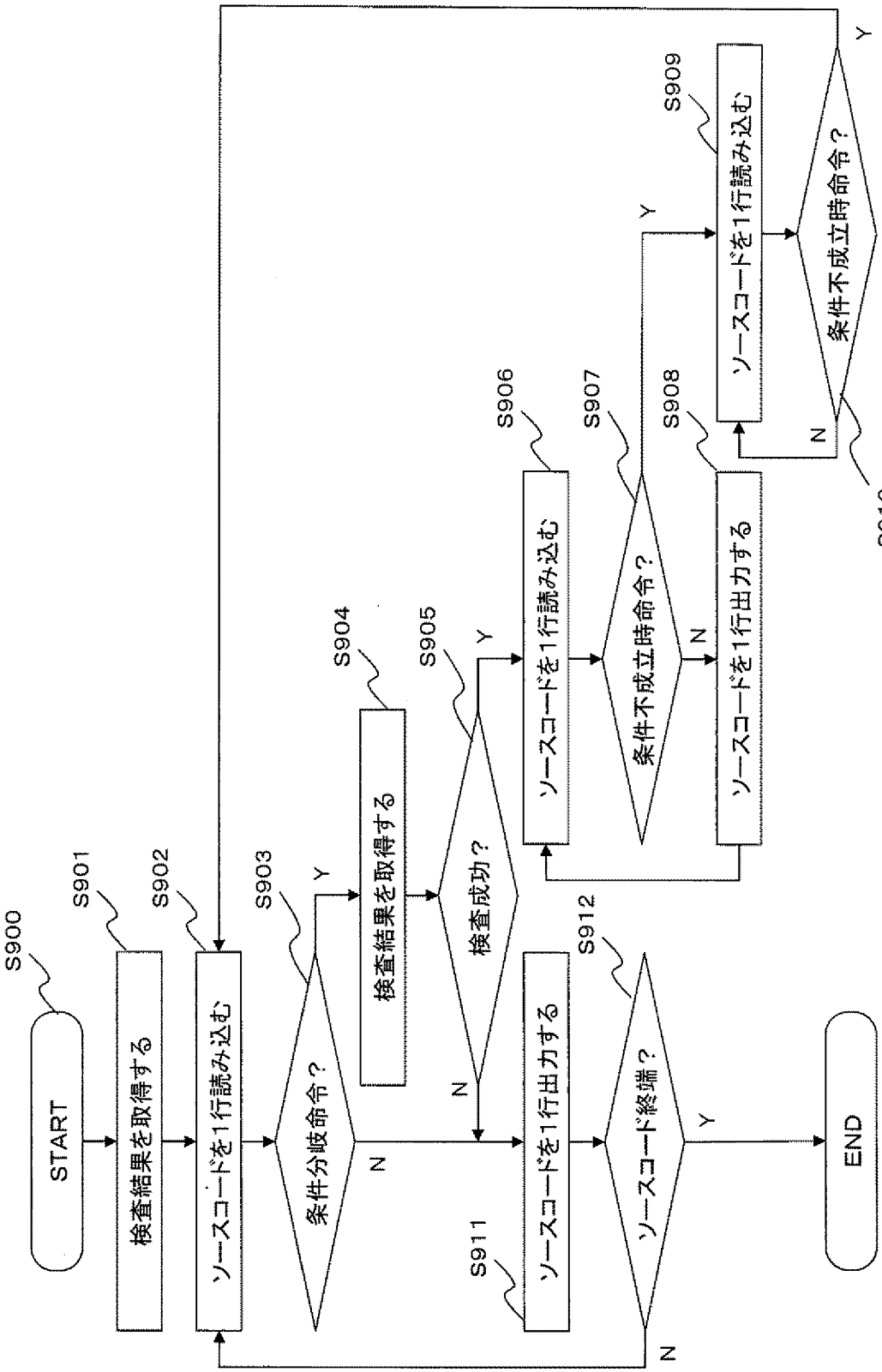
[図7]


int iMatrix[LINITH];
for(int i=0; i<iParam; i++) { iMatrix[i] = i; }

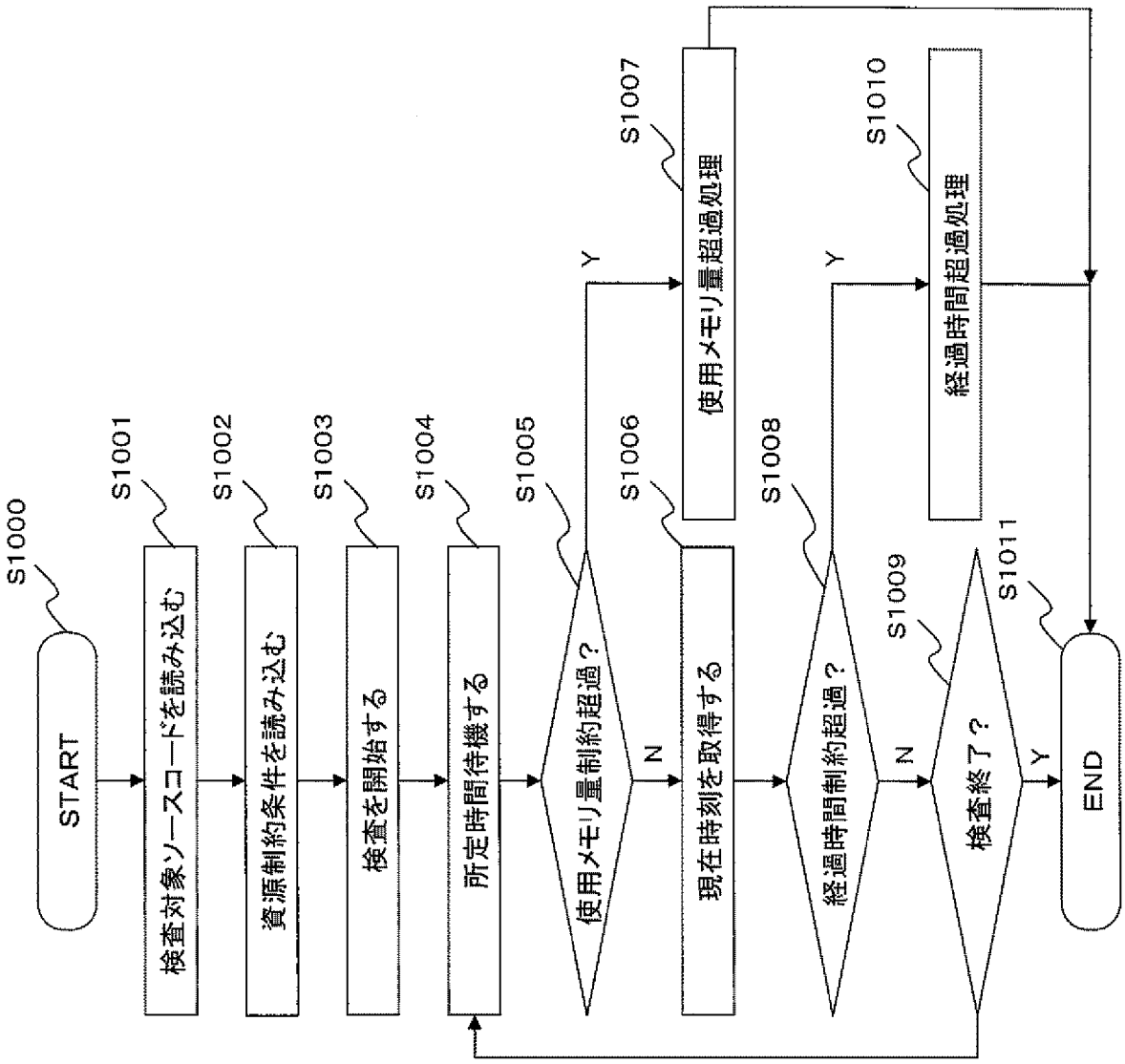
[図8]


S800
START
S801
ソースコードを1行読み込む
S802
識別コメント？
N
Y
S803
条件式を抽出する
S804
検査命令を生成する
S805
ソースコード終端？
N
Y
S806
END

[図9]


START
S900
検査結果を取得する
S901
ソースコードを1行読み込む
S902
条件分岐命令？
S903
Y
N
検査結果を取得する
S904
検査成功？
S905
Y
N
ソースコードを1行読み込む
S906
条件不成立時命令？
S907
Y
N
ソースコードを1行出力する
S908
ソースコードを1行読み込む
S909
条件不成立時命令？
S910
Y
N
ソースコードを1行出力する
S911
ソースコード終端？
S912
Y
N
END

[図10]


S1000
START
S1001
検査対象ソースコードを読み込む
S1002
資源制約条件を読み込む
S1003
検査を開始する
S1004
所定時間待機する
S1005
使用メモリ量制約超過？
Y
S1007
使用メモリ量超過処理
N
S1006
現在時刻を取得する
S1008
経過時間制約超過？
Y
S1010
経過時間超過処理
N
S1009
検査終了？
Y
S1011
END

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2013/050503

A. CLASSIFICATION OF SUBJECT MATTER
G06F9/44(2006.01)i, *G06F11/36*(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)
G06F9/44, G06F11/36

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	1996-2013
Kokai Jitsuyo Shinan Koho	1971-2013	Toroku Jitsuyo Shinan Koho	1994-2013

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y A	JP 2008-015665 A (Hitachi, Ltd.), 24 January 2008 (24.01.2008), paragraphs [0001], [0023], [0035]; fig. 6, 7 (Family: none)	1,9,10 2-8
Y	JP 2005-535055 A (PACT XPP Technologies AG.), 17 November 2005 (17.11.2005), paragraphs [0161] to [0163] & US 2003/0056202 A1 & EP 1466264 A2 & WO 2004/015568 A2 & DE 10240000 A1	1,9,10
A	JP 05-181685 A (Oki Electric Industry Co., Ltd.), 23 July 1993 (23.07.1993), paragraph [0013] (Family: none)	2

☒ 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 13 February, 2013 (13.02.13)	Date of mailing of the international search report 26 February, 2013 (26.02.13)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

Form PCT/ISA/210 (second sheet) (July 2009)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2013/050503

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 2011-113317 A (Fujitsu Ltd.), 09 June 2011 (09.06.2011), paragraph [0012]; fig. 21 (Family: none)	3
A	JP 2007-241426 A (Hitachi Software Engineering Co., Ltd.), 20 September 2007 (20.09.2007), paragraph [0009] (Family: none)	3
A	JP 2011-100420 A (Toshiba Corp.), 19 May 2011 (19.05.2011), claim 4 & CN 102053909 A	4
A	JP 2006-164294 A (Fujitsu Ltd.), 22 June 2006 (22.06.2006), paragraphs [0096], [0097] (Family: none)	5-8

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ２０１３／０５０５０３

Ａ．　発明の属する分野の分類（国際特許分類（ＩＰＣ））

Int.Cl.　G06F9/44(2006.01)i, G06F11/36(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（ＩＰＣ））

Int.Cl.　G06F9/44, G06F11/36

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	１９７１－２０１３年
日本国実用新案登録公報	１９９６－２０１３年
日本国登録実用新案公報	１９９４－２０１３年

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

Ｃ．　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求項の番号
Ｙ Ａ	JP 2008-015665 A（株式会社日立製作所）2008.01.24, 段落【０００１】,【００２３】,【００３５】, 図６，７等（ファミリーなし）	１，９，１０ ２－８
Ｙ	JP 2005-535055 A（ペーアーツェーテー　イクスペーペー　テクノロジーズ　アクチエンゲゼルシャフト）2005.11.17, 段落【０１６１】－【０１６３】等 & US 2003/0056202 A1 & EP 1466264 A2 & WO 2004/015568 A2 & DE 10240000 A1	１，９，１０

☑　Ｃ欄の続きにも文献が列挙されている。

☐　パテントファミリーに関する別紙を参照。

＊　引用文献のカテゴリー
「Ａ」特に関連のある文献ではなく、一般的技術水準を示すもの
「Ｅ」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「Ｌ」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「Ｏ」口頭による開示、使用、展示等に言及する文献
「Ｐ」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
「Ｔ」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「Ｘ」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Ｙ」特に関連のある文献であって、当該文献と他の１以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「＆」同一パテントファミリー文献

国際調査を完了した日	国際調査報告の発送日
１３．０２．２０１３	２６．０２．２０１３

国際調査機関の名称及びあて先
日本国特許庁（ＩＳＡ／ＪＰ）
郵便番号１００－８９１５
東京都千代田区霞が関三丁目４番３号

特許庁審査官（権限のある職員）　５Ｂ　３６６１
金子　秀彦
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	関連する請求項の番号
Ａ	JP 05-181685 A（沖電気工業株式会社）1993.07.23，段落【００１３】等（ファミリーなし）	２
Ａ	JP 2011-113317 A（富士通株式会社）2011.06.09，段落【００１２】，図２１等（ファミリーなし）	３
Ａ	JP 2007-241426 A（日立ソフトウエアエンジニアリング株式会社）2007.09.20，段落【０００９】等（ファミリーなし）	３
Ａ	JP 2011-100420 A（株式会社東芝）2011.05.19，【請求項４】等 & CN 102053909 A	４
Ａ	JP 2006-164294 A（富士通株式会社）2006.06.22，段落【００９６】，【００９７】等（ファミリーなし）	５－８