(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
2014 年 3 月 6 日(06.03.2014)

WIPO | PCT

(10) 国際公開番号
WO 2014/034330 A1

(51) 国際特許分類:
G06F 11/36 (2006.01)

(21) 国際出願番号: PCT/JP2013/069967

(22) 国際出願日: 2013 年 7 月 24 日(24.07.2013)

(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 2012-190826 2012 年 8 月 31 日(31.08.2012) 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: 松原 正裕(MATSUBARA Masahiro); 〒1008280 東京都千代田区丸の内一丁目６番６号 株式会社 日立製作所内 Tokyo (JP). 櫻井 康平 (SAKURAI Kohei); 〒1008280 東京都千代田区丸の内一丁目６番６号 株式会社 日立製作所内 Tokyo (JP). 成沢 文雄 (NARISAWA Fumio); 〒1008280 東京都千代田区丸の内一丁目６番６号 株式会社 日立製作所内 Tokyo (JP). 山中 久光 (YAMANAKA Hisamitsu); 〒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). 根本 守 (NEMOTO Mamoru); 〒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).

(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, 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,

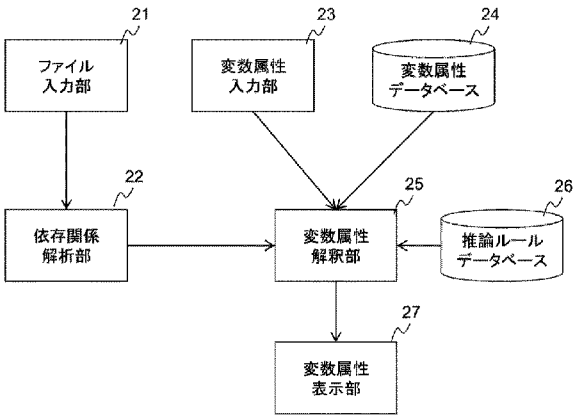
[続葉有]

(54) **Title**: SOFTWARE VERIFICATION PROGRAM AND SOFTWARE VERIFICATION SYSTEM

(54) 発明の名称 : ソフトウェア検証用プログラムおよびソフトウェア検証システム

[図2]

図2



21 File input unit
22 Dependency analysis unit
23 Variable attribute input unit
24 Variable attribute database
25 Variable attribute interpreting unit
26 Inductive rule database
27 Variable attribute display unit

(57) **Abstract**: To reduce time and trouble in matching the inputs and outputs of a software model and a hardware or other external environment model when creating and compositing the external environment model and the software model as a system verification model, a variable attribute interpreting unit (25) carries out induction to the extent possible for an attribute of each variable which is described in source code from dependency information which a dependency analysis unit (22) has derived, and variable attribute information which a variable attribute input unit (23) and/or a variable attribute database (24) hold. A rule which is recorded in an inductive rule database (26) is employed in the induction. A variable attribute display unit (27) displays in a display device (13) a combination of the variable attribute information whereon the variable attribute interpreting unit (25) has carried out the induction, and the variable attribute information which the variable attribute input unit (23) and the variable attribute database (24) hold.

(57) 要約:

[続葉有]

WO 2014/034330 A1

SV, SY, TH, 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, 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, KM, ML, MR, NE, SN, TD, TG).

添付公開書類:

— 国際調査報告（条約第 21 条(3)）

システムの検証モデルとして、ハードウェアなどの外部環境モデルとソフトウェアモデルとを作成して合成する際に、ソフトウェアモデルと外部環境モデルとの入出力を整合させる手間を低減するために、変数属性解釈部２５は、依存関係解析部２２が導出した依存関係情報と、変数属性入力部２３と変数属性データベース２４のいずれか、または両方が持つ変数属性情報とから、ソースコードに記述された各変数の属性について、可能な限り推論を行う。推論には、推論ルールデータベース２６に記録されたルールを用いる。変数属性表示部２７は、変数属性解釈部２５が推論した変数属性情報を、変数属性入力部２３と変数属性データベース２４の持つ変数属性情報と併せて、表示装置１３に表示する。

明　細　書

発明の名称 ：

ソフトウェア検証用プログラムおよびソフトウェア検証システム

技術分野

[0001] 本発明はソフトウェアの構造を解析するソフトウェアツールに係り、特にデータフローを利用した変数の属性解析と、検証用のモデルを構築するソフトウェアツールに関する。

背景技術

[0002] 近年、ソフトウェアの規模が巨大化し、構造や処理の複雑さが増しており、設計者や検証者がソフトウェアの全容を把握することが困難になっている。また、ソフトウェアが取りうる全ての状態をカバーするテストケースを用意して検証することが難しくなっている。特に車載機器等の制御を行う制御装置では、装置の電動化に伴う制御用ソフトウェアの規模増大により、ソフトウェアを含む電子制御装置に不具合が潜在し、システムの安全性を損なう恐れが増していることを背景として、種々の機能安全規格が制定されている。これらの制御システムでは、ハードウェアの不具合がソフトウェアの不具合を誘発することもあるため、ソフトウェアの検証だけでは不十分であり、ハードウェアとソフトウェアとを合わせて検証する必要がある。しかし特定のタイミングのみに発生する不具合は、テストで検出することが難しい。タイミング関連の不具合を引き起こす要因としては、複数のハードウェアの並行動作や、割り込み処理を挙げることができる。

[0003] システムの安全性を確保する上でも、設計者や検証者がソフトウェア、特に過去のソフトウェア資産や他の設計者が記述したソースコードの構造や処理を理解することが好ましい。また、ソフトウェアの取りうる状態に対するテストの網羅性を向上させ、不具合の原因を排除することが要求される。

[0004] 設計者や検証者による理解を容易にすることを目的として、プログラムに対し静的解析や動的解析を行い、その結果を関数コールグラフなどの図形と

して視覚化する手法とソフトウェアツールがある。視覚化対象のうち、ソフトウェアの構成上で重要な情報の１つとして、命令間や変数間の依存関係がある。ここで依存関係は、実行順序により実行結果が変化する可能性のある関係である。

[0005] 一方、システムの状態を網羅的に検証する技術として、モデル検査がある。モデル検査技術では、システムを状態遷移モデルに置き換え、システムに要求される性質をプロパティやアサーションとして定義し、状態遷移モデルと共にツールに入力することで、当該ツールがシステムの取りうる状態を調べ上げ、与えられたプロパティやアサーションが満たされるか否かを判定する。満たされない場合には、反例を出力する。

[0006] ハードウェアとソフトウェアの複合動作や相互作用により発生する不具合をモデル検査で検出するには、例えば特許文献１に開示されているように、システムの検証モデルとして、ハードウェアなどの外部環境モデル（ソフトウェアに対する外部環境を表現する検証用モデル）と、ソフトウェアモデル(ソフトウェアを表現する検証用モデル)とを作成し、合成する必要がある。ソフトウェアモデルの生成には、ソースコードを変換する方法がある。ただしソースコードを全て検証用モデルに変換すると、メモリ不足による状態探索未完了の問題（状態爆発と呼ばれる）が発生する虞があるため、コードの一部を抽出するスライシングと呼ばれる処理がなされることもある。

先行技術文献

特許文献

[0007] 特許文献1：特開２００９-１２３１５９号公報

発明の概要

発明が解決しようとする課題

[0008] 特許文献１記載のように、ソースコードの一部を抽出してソフトウェアモデルに変換する場合、ソフトウェアモデルの入出力に関係するハードウェアや、検証項目に関係するが抽出されなかったソフトウェアは、外部環境モデ

ルとする必要がある。検証者はこの外部環境モデルを、検証目的に応じて詳細なものにしたり、抽象度の高いものにしたりできる。ただしソフトウェアモデル、外部環境モデル、およびシステムの検証モデルを用意するのは一般的に手間が掛かる。

[0009] また、特許文献１記載の技術では、外部環境モデルは、ソフトウェアのうち検査対象とする一部分の状態遷移情報から、検査対象外である他部分の挙動を模擬し、検査対象部分と他部分とのメッセージ通信を生成するものである。このため、検査対象とするソフトウェアの状態遷移情報がないと、外部環境モデルを生成できない。また、外部環境モデルにて故障などソフトウェアで想定していない挙動を取ることができない。

[0010] このような制約を受けない外部環境モデルを検査に適用するには、検査対象とするソフトウェアに応じて外部環境モデルを用意するか、用意された外部環境モデルに応じてソフトウェアから検査対象を抽出する必要がある。このとき、ソフトウェアモデルの入出力と、外部環境の入出力とは整合しなくてはならない。このため、ソースコードから一部を抽出する際に、外部環境モデルの入出力と整合するように、外部環境との境界となる変数を決めて抽出範囲を調整する必要がある。または逆に、ソフトウェアモデルの入出力に合わせて、外部環境モデルを用意する必要がある。入出力が整合するには、変数が表現するデータの種別が同じであり、単位や値域が同じである必要がある。

[0011] ソフトウェアモデルと外部環境モデルとの入出力を整合させるには、入出力となる変数の属性を一致させる必要があり、検証者は入出力変数の属性を調べる必要がある。一方で、コードの抽出範囲を過小に限定してしまうと、不具合の原因をモデル化できないため、不具合の原因をモデルに含めつつ状態爆発を回避して検証しようとするならば、ソフトウェアからコードを抽出する範囲を柔軟に調整する必要がある。しかし、抽出範囲を調整すると、調整の度に入出力変数が替わるため、変数の属性を調べなおさなければならず、規模が増大し複雑さが増しているソフトウェアについて、外部環境モデル

と接続しうる変数を探索して、各変数の属性を調べることには多大な労力が必要となる。

[0012] 本発明はこのような課題に鑑みてなされたものであり、本発明の目的は、システムの検証モデルとして、ハードウェアなどの外部環境モデルとソフトウェアモデルとを作成して合成する際に、ソフトウェアモデルと外部環境モデルとの入出力を整合させる手間を低減することにある。

課題を解決するための手段

[0013] 上記課題を解決するため本発明は、検証対象となるソフトウェアのソースコードを入力とし、前記ソースコードを解析して、モデル検査用のモデルを生成する検査用モデル生成プログラムにおいて、前記プログラムは、前記コンピュータに前記ソースコードの変数等の依存関係を導出する依存関係解析部と、前記ソースコードの一部の変数について属性情報を記録する変数属性記録部と、前記依存関係と前記変数属性記録部の属性情報から、前記ソースコードの各変数について属性を推論する変数属性解釈部と、前記属性情報を表示する変数属性情報表示部と、を有するように構成する。

[0014] 次に、外部環境モデルデータベースと、前記変数属性情報に応じて前記外部環境モデルデータベースから利用可能なモデルを引き出す外部環境モデル候補抽出部とを有するように構成する。

[0015] 次に、前記依存関係に対して範囲限定の指定を受け付ける依存関係限定部と、前記の限定された依存関係からソースコードのスライシングを行うソースコード抽出部と、前記の抽出されたソースコードから検査用のソフトウェアモデルを生成するモデル変換部と、前記外部環境モデル候補抽出部が引き出したモデルから利用するモデルを選ぶ外部環境モデル選択部と、前記ソフトウェアモデルと、前記外部環境モデル選択部が選んだ外部環境モデルとから、検査用のシステムモデルを合成するシステムモデル合成部と、を有するように構成する。

発明の効果

[0016] 検査用モデル生成プログラムが、変数の一部(マイコン入出力を行うレジス

タ変数や、関数の引数など)に関する属性情報から、変数の依存関係を辿りつつ、他の変数の属性を推論することにより、前記プログラムのユーザは全ての変数について属性情報を自ら調査する必要が無くなり、ソフトウェアモデルと外部環境モデルとの入出力を整合させる手間が軽減される。

[0017] また検査用モデル生成プログラムが、推論されたものを含む変数属性情報から、外部環境モデルとして利用可能なものをデータベースから抽出することにより、前記プログラムのユーザは外部環境モデルの選択に掛かる工数を低減することができる。

[0018] またユーザが変数を１つ以上選択すると、検査用モデル生成プログラムが、選択された変数と依存関係のある変数の属性情報から、スライシング後のソフトウェアモデルに接続可能な外部環境モデルをデータベースから探索し、候補として提示する。逆に、外部環境モデルを選択すると、その外部環境モデルの入出力変数の属性から、接続可能なソフトウェアの変数を探索し、スライシングの条件（スライス基準）として設定することで、選択された外部環境モデルと接続可能なソフトウェアモデルの変換元となるソースコードを抽出する。以上により、ユーザが検証のためにハードウェアとソフトウェアの双方をモデル化する作業に掛かる工数を低減することができる。

図面の簡単な説明

[0019] [図1]本発明の構成を表す図である。

[図2]本発明の機能構成を表す図である。

[図3]変数の属性情報を推論する処理フローを表す図である。

[図4]属性推論処理の詳細な処理フローを表す図である。

[図5]解析対象例のソースコードを表す図である。

[図6]変数依存関係グラフの例を表す図である。

[図7]変数属性データベースに格納されているテーブルを表す図である。

[図8]推論ルールデータベースに格納されているテーブルを表す図である。

[図9]変数属性情報を表す図である。

[図10]変数属性情報を表す図である。

[図11]本発明の機能構成を表す図である。

[図12]システムの検証モデルを生成する処理フローを表す図である。

[図13]外部環境モデルの候補リストを表す図である。

[図14]システム検査用モデルを表す図である。

発明を実施するための形態

[0020] 以下、図面を用いて本発明の実施形態について説明を行う。

実施例 1

[0021] 図１は、本発明をシステムとして実現した場合のシステム構成である。本発明は、例えばソフトウェアツールとして実現される。システム１０において、処理を実行するためのソフトウェアは検証モデル生成プログラム１１３１として、例えばコンピュータ１１上のＲＯＭ１１３に格納されており、ＣＰＵ１１１によって読み出され、ＲＡＭ１１２を作業用記憶領域として用い、実行される。これにより、コンピュータ１１は検証モデル生成プログラム１１３１によって規定された処理を実行する。その処理内容は図２以降で説明する。解析対象となるソースコードファイルは、コンピュータ１１上のハードディスク１１４などの記憶装置に格納されている。解析実行指示などのユーザ操作は、キーボードなどの入力装置１２から行うことができる。

[0022] なお、検証モデル生成プログラム１１３１による出力は、例えばディスプレイなどの表示装置１３になされるが、それ以外に、図示しない外部コンピュータへのネットワークを介した出力、ＣＤ－ＲＯＭなどの外部記憶媒体へのデータファイル形式での書き込みによる出力であってもよい。

[0023] 図２は、検証モデル生成プログラム１１３１の基本的な機能構成を示している。ファイル入力部２１は、解析対象となるソースコードファイルをユーザが１ないし複数選択できる機能を有しており、ユーザによるファイル指定に従い、ソースコードファイルをコンピュータの記憶媒体（ハードディスク１１４など）から読み込む。依存関係解析部２２は、ユーザが解析実行を指示すると、ファイル入力部２１が読み込んだソースコードファイルを解析し、ソースコードから生成されるプログラムの依存関係を導出する。ここで依

存関係とは、プログラムの実行順序により演算結果に影響のありうる関係のことであり、具体的には、ステートメント間の依存関係であるプログラム依存関係や、変数間の依存関係である変数依存関係である。プログラム依存関係を変数単位に分解すると変数依存関係になるので、プログラム依存関係と変数依存関係は情報としては等価とみなせる。プログラム依存関係や変数依存関係の導出方法については、文献に譲り、本明細書では割愛する。

[0024] 変数属性入力部２３は、ソースコードに記述されたある変数に関する属性について、ユーザからの入力を受け付ける。以降では特に明記しない限り、変数の属性といえば、ソフトウェアモデルの入出力の整合に関係するものを指すとする。変数属性データベース２４は、ファイル入力部２１が読み込んでいるソースコードの一部の変数について、属性情報を記録している。変数属性解釈部２５は、依存関係解析部２２が導出した依存関係情報と、変数属性入力部２３と変数属性データベース２４のいずれか、または両方が持つ変数属性情報とから、ソースコードに記述された各変数の属性について、可能な限り推論を行う。推論には、推論ルールデータベース２６に記録されたルールを用いる。変数属性表示部２７は、変数属性解釈部２５が推論した変数属性情報を、変数属性入力部２３と変数属性データベース２４の持つ変数属性情報と併せて、表示装置１３に表示する。

[0025] 図３は、検証モデル生成プログラム１１３１が図２に示された機能を用いて行う処理であり、入力となるソースコードに記述された変数の属性情報を推論する処理のフローを示している。

[0026] 処理を開始すると、ステップ３１にて、依存関係解析部２２は、入力となるソースコードの依存関係を解析し、データとして出力する。例として、変数ａと変数ｂについて、ａ＝ｂという式がソースコードに記述されているとすると、変数依存関係としては「ａ←ｂ」と表現できる。これは変数ｂが変数ａに影響を与えるという意味である。逆に「ａ→ｂ」と記述してもよく、表現のルールを決めればよい。本実施例では前者を採用する。また、ａは依存元、ｂは依存先と呼ぶことにする。

[0027] 次にステップ３２にて、変数属性解釈部２５は、変数属性入力部２３や変数属性データベース２４から、入力となるソースコードに記述された変数の属性情報を取得し、依存関係解析部２２の出力した依存関係データにおける各変数と対応付ける。なおステップ３２は、該当する変数の属性情報が空のときだけ実行し、それ以外ではスキップする。

[0028] 次にステップ３３にて、変数属性解釈部２５は、依存関係データにおいて属性推論処理の始端となる変数を抽出する。本実施例では、入力となるソースコードがプログラムとして実行される際に入力となる変数の属性情報をもとに、前向きに（依存先の変数から、依存元の変数へ）属性情報の推論を進めるものとする。また、本実施例の処理は再帰的になっている。このため始端変数として、影響する変数がない、すなわちどの変数からも依存されていない（依存元になる変数が存在しない）変数を抽出する。ただし依存関係がループしており該当する変数が存在しない場合には、任意の変数を１つ選択する。

[0029] 次にステップ３４にて、変数属性解釈部２５は、対象とする始端変数のそれぞれについて、属性を推論する処理（詳細は図４に記載）を行う。ステップ３４は、ステップ３３にて抽出された全ての推論始端となる変数について、それぞれ実行する。以上をもって処理を終了する。

[0030] なお、図３の処理を実行するタイミングは、ユーザによる実行指示時か、ユーザが変数属性入力部２３に対し属性情報を入力したときとする。処理を終了したら、変数属性表示部２７が、出力された依存関係情報を表示装置１３に表示する。

[0031] 図４は、ある変数の属性情報を推論する処理であり、図３のステップ３４（「属性推論」処理）の詳細フローを示している。

[0032] 処理を開始すると、ステップ４１は、属性情報を推論する対象変数の全ての依存元変数について、それぞれ実行する。ステップ４１にて、変数属性解釈部２５は、対象とする依存元変数について、属性情報を推論する処理（図４の処理）を行う。次にステップ４２にて、変数属性解釈部２５は、対象変

数と依存元変数とがデータ依存関係にあるかを判定する。データ依存関係であれば、対象変数の属性情報を更新する必要が無いので処理を終了し、そうでなければ、ステップ４３に進む。次にステップ４３にて、変数属性解釈部２５は、いずれかの依存元変数の属性情報がステップ４１にて更新されたか否かを判定する。１変数でも更新されていればステップ４４に進み、１つも更新されていなければ、対象変数の属性情報を保持して、処理を終了する。次にステップ４４にて、変数属性解釈部２５は、対象変数が推論ルールデータベース２６に記録されている推論ルールに該当するか否かを判定する。１ルールでも該当すればステップ４５に進み、該当するルールがなければ、対象変数の属性情報を保持して、処理を終了する。次にステップ４５にて、変数属性解釈部２５は、ステップ４４にて該当した推論ルールに基づき推論処理を実行し、その結果を対象変数の属性情報に反映する。以上をもって処理を終了する。なお上記は依存関係がループしていない場合の処理であり、ループが存在する場合には、１度「属性推論」処理の対象になった変数は対象外にすることが必要となる。

[0033] 以下では、図３と図４の処理例を示す。図５のソースコード５０は、解析対象として検証モデル生成プログラム１１３１に入力する、Ｃ言語で記述されたソースコードの例である。左端の数字は行番号である。

[0034] 図６の変数依存関係グラフ６１は、図５のソースコード５０を依存関係解析部２２が解析し出力する変数依存関係データをグラフ化したものである。この変数依存関係グラフ６１は、例えば表示装置１３に表示される。変数依存関係グラフ６１のノードは、変数の名称や実体（記憶領域）だけでなく、記述位置（および実行パス）でも区別した変数である。例えば「ｓｔｒｏｋｅ（Ｌ２０@ｃｏｎｔｒｏｌ）」と表記された変数は、ソースファイルの２０行目、関数ｃｏｎｔｒｏｌ内に記述された変数ｓｔｒｏｋｅであること意味している。変数の区別には、記述された行番号だけでなく列番号や、ソースファイル名を用いてもよい。また、図６のグラフには記載されていないが、各ノードには、変数の型のようにソースコードのみから判明する属性や、

変数属性入力部２３や変数属性データベース２４から与えられた属性、変数属性解釈部２５が推論した属性が対応付けられ、データとしてＲＡＭ１１２やハードディスク１１４に記憶されている。

[0035] また図６において、依存関係を示すエッジ（矢印）、および演算式における関係を示すエッジ（破線）には、関係の種類を示すデータが対応付けられて記憶されている。依存関係のエッジにおいて、ｄｅｆは依存元の変数を依存先の変数が定義（代入などにより値を決めていること）を示している。ｕｓｅはデータ依存関係（依存元の変数が、依存先の変数の値を利用していること）を示している。ｃｏｎｄは制御依存関係（依存先の変数の値により実行パスが変わるため、依存元の変数の値に影響があること）を示している。

[0036] 図７の変数属性テーブル７１は、変数属性データベース２４に記録されている変数属性の例を示している。変数属性テーブル７１の項目は、変数名、ｉｎ／ｏｕｔ（変数がプログラムの入力か出力か）、単位、データ種別からなる。これらの値は予め設定されている。変数ｒｅｇ＿ａｄ＿１は、入力用の変数であり、データ種別は（例えば自動車のブレーキペダルの）ストローク量であることがわかる。またその単位は「（Ａ／Ｄ変換値）」とあるが、これはＡ／Ｄ（Ａｎａｌｏｇ／Ｄｉｇｉｔａｌ）変換により得られる値であり、単位は特に決められていないことを示している。

[0037] 図８の推論ルールテーブル８１は、推論ルールデータベース２６に記録されている推論ルールの例を示している。推論ルールテーブル８１の項目は、前件と後件からなる。前件の条件が成立するときに、後件の条件も成立することを意味している。前件と後件は、述語論理で表現されている。これらの値は予め設定されている。

[0038] 推論ルールテーブル８１の推論ルールＮｏ．１にて、ＸとＹはある変数を示しており、Ｄｅｆ（Ｘ，Ｙ）「ＸがＹを定義している」ならば、ＳａｍｅＡｔｔｒ（Ｘ，Ｙ）「ＹはＸと同じ属性情報を有する」ことを意味している。

[0039] 推論ルールテーブル８１の推論ルールＮｏ．２にて、Ｘ，Ｙ，Ａはある変

数を示しており、Ｃｏｎｄ（Ｘ，Ａ）「ＡがＸと制御依存関係にある（Ａが依存元）」かつ、Ｃｏｎｄ（Ｙ，Ａ）「ＡがＹと制御依存関係にある」かつ、Ｃｏｍｐ（Ｘ，Ｙ）「ＸとＹとは演算式内にて大小比較の関係にある（大小比較記号で結ばれている）」ならば、Ｆｌａｇ（Ａ）「Ａは（異常などの）判定フラグであり、単位はなし」、かつＳａｍｅＡｔｔｒ（Ｘ，Ｙ）であることを意味している。なお、フラグＡの判定対象とするデータ種別は、例えば変数Ｘのデータ種別とする。

[0040] 推論ルールテーブル８１の推論ルールＮｏ．３にて、Ｘ，Ｙ，Ｚはある変数を示しており、Ｄｅｆ（Ｘ，Ｙ，Ｚ）「ＸとＹがＺを定義している」かつ、Ｍｕｌ（Ｘ，Ｙ）「ＸとＹとは演算式内にて乗算の関係にある（乗算記号で結ばれている）」ならば、ＭｕｌＡｔｔｒ（Ｘ，Ｙ，Ｚ）「Ｚのデータ種別はＸ（またはＹ）と同じであり、かつ、Ｚの単位はＸとＹの単位を乗じたもの」であることを意味している。同様に、和差算、除算についても推論ルールを記述することができる。

[0041] 推論ルールテーブル８１の推論ルールＮｏ．４にて、Ｘ，Ｙはある変数を示しており、Ｃｏｍｐ（Ｘ，Ｙ）かつ、Ｖａｃａｎｔ（Ｙ）「Ｙの属性が空」ならば、Ｔｈｒｓｈ（Ｙ）「Ｙのデータ種目は『閾値』である」であることを意味している。ただし、Ｔｈｒｓｈ（Ｙ）に「？」が付いているのは、確定でないことを意味している。なお、閾値Ｙのデータ種別は、変数Ｘのデータ種別とする。

[0042] 推論ルールテーブル８１の推論ルールＮｏ．５にて、Ｘ，Ｙ，Ｚはある変数を示しており、Ｄｅｆ（Ｘ，Ｙ，Ｚ）かつ、Ｖａｃａｎｔ（Ｙ）ならば、ＳａｍｅＡｔｔｒ（Ｘ，Ｚ）であることを意味している。ただし、ＳａｍｅＡｔｔｒ（Ｘ，Ｚ）に「？」が付いており、確定でないことを意味している。

[0043] 推論ルールテーブル８１の推論ルールＮｏ．６にて、Ｙ，Ａはある変数を示しており、推論ルールＮｏ．２の前件が成立し、かつＥｒｒＴｈｒｓｈ（Ｙ）「Ｙが異常閾値」ならば、ＥｒｒＦｌａｇ（Ａ）「Ａは異常判定フラグ

（異常を示すフラグ）」であることを意味している。前件に推論ルールＮｏ．２を用いているので、Ｎｏ．６が優先されるとする。つまり、Ｆｌａｇ（Ａ）よりＥｒｒＦｌａｇ（Ａ）の適用が優先される。

[0044] 図９は、変数依存関係グラフ６１のノードと対応付けられた変数属性テーブル９１を示しており、図３の処理を１回実行した後の値が入っている。変数属性テーブル９１の項目は、変数名、単位、データ種別からなる。

[0045] 変数属性テーブルにおいて、変数は実体で区別されており、記述位置では区別されていない。このため、変数依存関係グラフ６１のノードと、変数属性テーブル９１の行とは、ｎ：１対応となっている。例えば変数ｓｔｒｏｋｅについて、変数依存関係グラフ６１には対応するノードが２つあるが、変数属性テーブル９１には対応する行は１つのみである。

[0046] 以下では、変数属性解釈部２５が推論処理により変数属性テーブル９１を生成する過程を説明する。変数属性解釈部２５は、依存関係解析部２２から変数依存関係グラフ６１のデータを受け取り、変数依存関係グラフ６１に含まれる変数から、変数属性テーブル９１のデータ構造を生成する。この時点で、変数属性テーブル９１には変数名のみが入っており、残りのデータは空である。各変数の属性情報の更新を示すフラグ（変数属性テーブル９１に記載なし）はクリアする。

[0047] 次に変数属性解釈部２５は、変数依存関係グラフ６１に含まれる変数の属性情報を、変数属性データベース２４の変数属性テーブル７１から取得する。ただし本実施例では、入力用変数の属性情報のみを取得する。すると、変数ｒｅｇ＿ａｄ＿１に関する属性情報が取得されるので、変数属性テーブル９１に値を入れる。この際、変数ｒｅｇ＿ａｄ＿１の属性情報の更新を示すフラグをセットする。

[0048] 次に変数属性解釈部２５は、変数依存関係グラフ６１の推論始端となる変数を抽出する。全ノードについて、依存元となる変数の有無を判定し、無いと判定されたノードだけ残すと、変数ｓｔｒｏｋｅ（Ｌ２０＠ｃｏｎｔｒｏｌ）が抽出される。この変数に対して、変数属性解釈部２５は図４の「属性

推論」処理を行う。

[0049] 変数ｓｔｒｏｋｅ（Ｌ２０@ｃｏｎｔｒｏｌ）に対して「属性推論」処理が実行されると、「属性推論」は再帰的な処理となっているため、変数依存関係グラフ６１の全ノードに対して「属性推論」処理が実行される。変数ｒｅｇ＿ａｄ＿１（Ｌ７@ｉｎｐｕｔ）は、依存先変数を持たないので、属性情報が保持される。この属性情報は、変数属性テーブル７１から取得した値と同じである。

[0050] 次に、変数ｓｔｒｏｋｅ＿ｒａｗ（Ｌ７@ｉｎｐｕｔ）に対して、ステップ４２～４５の処理が実行される。依存先変数のｒｅｇ＿ａｄ＿１（Ｌ７@ｉｎｐｕｔ）は、変数属性テーブル７１から値を取得したことで、属性情報が更新されているので、推論ルールテーブル８１から合致する推論ルールが検索される。合致する推論ルールとしてＮｏ．１が抽出され（推論ルールのＸがｒｅｇ＿ａｄ＿１，Ｙがｓｔｒｏｋｅ＿ｒａｗ）、ｓｔｒｏｋｅ＿ｒａｗの属性は、ｒｅｇ＿ａｄ＿１と同じ内容に更新される。

[0051] 同様に他の変数の属性情報も更新していく。変数ｓｔｒｏｋｅ＿ｒａｗ（Ｌ１６@ｃｏｎｖｅｒｔ）と変数ｓｔｒｏｋｅ＿ｒａｗ（Ｌ１０@ｃｈｅｃｋ）は、変数ｓｔｒｏｋｅ＿ｒａｗ（Ｌ７@ｉｎｐｕｔ）とデータ依存関係なので、属性情報は更新しない（更新済みである）。

[0052] 変数ｓｔｒｏｋｅ＿ｍａｘ（Ｌ１０@ｃｈｅｃｋ）は、推論ルールテーブル８１の推論ルールＮｏ．４に合致するので、データ種別は「ストローク量に対する閾値」、単位は「（Ａ／Ｄ変換値）」と判定される。ただし確定ではないので、属性値に「？」が付いている。

[0053] 変数ｅｒｒ＿ｓｔｒｏｋｅ（Ｌ１１@ｃｈｅｃｋ）は、推論ルールテーブル８１の推論ルールＮｏ．２に合致する（推論ルールのＸがｓｔｒｏｋｅ＿ｒａｗ，Ｙがｓｔｒｏｋｅ＿ｍａｘ，Ａがｅｒｒ＿ｓｔｒｏｋｅ）ので、後件のＦｌａｇ（Ａ）から、ｅｒｒ＿ｓｔｒｏｋｅの属性情報について、データ種別はストローク量に対する判定フラグであり、判定フラグなので単位はなしに更新される。またＳａｍｅＡｔｔｒ（Ｘ，Ｙ）から、ｓｔｒｏｋｅ＿

ｍａｘの属性は、ｓｔｒｏｋｅ＿ｒａｗと同じ内容に更新される。

[0054] 変数ｅｒｒ＿ｓｔｒｏｋｅ（Ｌ１５＠ｃｏｎｖｅｒｔ）はｅｒｒ＿ｓｔｒｏｋｅ（Ｌ１１＠ｃｈｅｃｋ）とデータ依存関係なので、属性情報は更新しない。

[0055] 変数ｓｔｒｏｋｅ（Ｌ１６＠ｃｏｎｖｅｒｔ）は、推論ルールテーブル８１の推論ルールＮｏ．３に合致する（推論ルールのＸがｓｔｒｏｋｅ＿ｒａｗ，Ｙがｃ＿ｉｎ＿ｓｔｒｏｋｅ，Ｚがｓｔｒｏｋｅ）。しかし、ｃ＿ｉｎ＿ｓｔｒｏｋｅの属性情報が空であり不明なので、推論を進めることはできない。その一方で、推論ルールＮｏ．５にも合致するので、ｓｔｒｏｋｅデータ種別は「ストローク量」と判定される。ただし確定ではないので、属性値に「？」が付いている。

[0056] 以上で、図３の処理を終了する。同様にして、出力用変数の属性情報をもとに、後ろ向きに推論を進めてもよい。

[0057] 以上のようにして、一部の変数の属性情報から、他の変数の属性情報を自動的に推論することができ、ユーザが手作業で変数の依存関係を追跡し、属性情報を解釈していくより、短い時間で変数の属性情報を得ることができる。

[0058] 図１０は、ユーザからの変数属性情報の入力を受けて図３の推論処理を再度実行し、変数属性テーブル９１を更新した変数属性テーブル１０１を示している。

[0059] 変数属性テーブル９１の値が推論処理により導出された後で、ユーザが変数属性入力部２３に対し、変数ｓｔｒｏｋｅ＿ｍａｘのデータ種別が「異常閾値」、単位が「（Ａ／Ｄ変換値）」であると入力する。これらの属性値は、データ種別と単位と、それぞれに取りうる値が用意されており、候補としてリスト表示される。ユーザは表示装置１３上でリストを目視し、入力装置１２から選択の上、確定させる。検証モデル生成プログラム１１３１内では、データ種別の候補として、「閾値」や「異常閾値」などが予め定義されている。また、「異常閾値」は「閾値」の下位概念であると定義されている。

このため、変数ｓｔｒｏｋｅ＿ｍａｘのデータ種別が「閾値」のときは、リスト上で現在の値として「閾値」が選択状態にあり、その次の候補として下位概念である「異常閾値」がリスト上で順に表示される。

[0060] ユーザが属性情報の入力を確定させると、検証モデル生成プログラム１１３１は図３の推論処理を実行する。すると変数ｅｒｒ＿ｓｔｒｏｋｅの属性情報が、推論ルールＮｏ．６により更新され、データ種別は「ストローク量の異常判定フラグ」となる。

[0061] さらにユーザが変数ｃ＿ｉｎ＿ｓｔｒｏｋｅの属性情報として、データ種別を「係数」、単位を「ｃｍ／（Ａ／Ｄ変換値）」と入力して確定させると、検証モデル生成プログラム１１３１は図３の推論処理を実行し、変数ｓｔｒｏｋｅの属性情報が推論ルールＮｏ．３により更新され、データ種別は「ストローク量」、単位は「ｃｍ」となる。単位を示すデータは、分子と分母とそれぞれに要素を用意して、分子と分母で重複するものを除去すればよい。なお、単位系の演算規則は、推論ルールテーブル８１と同様に、検証モデル生成プログラム１１３１内に定義しておくものとする。

[0062] 以上のようにして、ユーザが変数の属性情報を追加入力することで、属性情報を推論できる変数の個数を増やし、全変数に対する割合を向上させることができるので、ユーザが手作業で変数の属性情報を解釈するより、さらに短い時間で変数の属性情報を得ることができる。

図１１は、検証モデル生成プログラム１１３１の機能構成を示しており、これらの機能は図２の基本機能構成と併せて動作する。

[0063] 依存関係限定部１１１は、依存関係解析部２２から解析対象のソースコードに関する依存関係情報を受け取り、依存関係情報に含まれる各変数について個別に、ソースコード抽出時に残すか否かのデータ（残存設定データ）と対応付けて管理する。残存設定データは２値（Ｔｒｕｅ；残す、Ｆａｌｓｅ；残さない）で示され、依存関係情報のうちソースコード抽出に利用する範囲を限定する意味がある。残存設定データの初期値は、例えば全変数についてＴｒｕｅとする。また依存関係解析部２２は、ユーザ入力や外部環境モデ

ル選択部１１６からの指定を受けて、残存設定データを変更する。

[0064] 依存関係情報において、ソースコード抽出時に残る設定となっている変数と、残らない設定となっている変数とに依存関係があるとき、残る変数がソフトウェアモデルと外部環境モデルとの境界になる。以下、このような変数を境界変数と呼ぶ。例えば変数依存関係グラフ６１において、ｓｔｒｏｋｅ＿ｒａｗ（Ｌ７＠ｉｎｐｕｔ）を残すと設定し、ｒｅｇ＿ａｄ＿１（Ｌ７＠ｉｎｐｕｔ）を残さないと設定したとき、ｓｔｒｏｋｅ＿ｒａｗ（Ｌ７＠ｉｎｐｕｔ）は境界変数である。

[0065] 依存関係解析部２２の残存設定データに対する、ユーザ入力や外部環境モデル選択部１１６からの指定内容は、各変数の残存設定でもよいし、境界に指定する変数でも良い。例えばｓｔｒｏｋｅ＿ｒａｗ（Ｌ７＠ｉｎｐｕｔ）を入力側の境界変数に指定するとき、ｓｔｒｏｋｅ＿ｒａｗ（Ｌ７＠ｉｎｐｕｔ）の残存設定はＴｒｕｅとなり、ｒｅｇ＿ａｄ＿１（Ｌ７＠ｉｎｐｕｔ）の残存設定はＦａｌｓｅとなる。

[0066] ソースコード抽出部１１２は、ファイル入力部２１から解析対象のソースコードを受け取り、依存関係限定部１１１による依存関係情報とその残存設定データとに基づいて、ソースコードの抽出（スライシング）を行う。抽出したソースコードはファイルに出力する。スライシングの方法として例えば、ソースコードを構文解析した後、依存関係情報に含まれる変数のうち、残存設定がＴｒｕｅの変数が記述されているステートメント、およびそのステートメントを包含する構文（関数など）、関連する構文（変数宣言など）を残し、それ以外のコードを除去する。

[0067] モデル変換部１１３は、ソースコード抽出部１１２が出力するスライシング後のソースコードを、モデル検査器が扱えるモデル記述言語に変換する。例えばＳＰＩＮ（Ｓｉｍｐｌｅ Ｐｒｏｍｅｌａ ＩＮｔｅｒｐｒｉｔｅｒ）というモデル検査器では、モデル記述言語はＰｒｏｍｅｌａである。モデル変換の方法として、ソースコードを構文解析した後、ソースコードの構文から、モデル記述言語の構文へ変換する規則に基づいて各構文を変換する。

この変換規則は予め用意しておく。変換により得られるモデルは、検査対象となるソフトウェアに関する検証用モデルであり、検査対象となるシステムに関する検証用モデルの部品である。

[0068] 外部環境モデルデータベース１１４は、検証対象となるソフトウェアの外部環境について予めコーディングまたは自動コード生成用にテンプレート化されたモデル部品について、属性と紐付けられて記憶されている。外部環境モデル候補抽出部１１５は、依存関係限定部１１１から残存設定データと対応付けられた依存関係情報を、変数属性解釈部２５から検証対象となるソフトウェアの各変数の属性情報を受け取り、残存設定がＴｒｕｅとなっている変数の属性情報の一部（データ項目や単位など）を検索キーとして、インタフェースが合致する外部環境モデルを、外部環境モデルデータベース１１４より抽出する。抽出された外部環境モデルは、検証対象となるソフトウェアの外部環境となりうるモデルである。

[0069] 外部環境モデル選択部１１６は、外部環境モデル候補抽出部１１５により抽出された外部環境モデルの属性（名称等）を、ユーザに対して表示装置１３を用いてリスト表示し、ユーザからの各外部環境モデルの利用有無について、入力装置１２から指定を受け付ける。利用有無は、デフォルトでは「なし」である。ユーザが外部環境モデルの１つについて、利用有無を「あり」に指定すると、選択された外部環境モデルのインタフェースとなっている変数の属性情報が、依存関係限定部１１１に渡され、依存関係限定部１１１は検証対象となるソフトウェアの変数のうち、外部環境モデルのインタフェース変数と属性が合致する変数を依存関係情報から検索し、境界変数として残存設定を変更する。具体的には、境界変数は残す設定とする。インタフェース変数が入力のとき、境界変数から依存元方向（より出力に近い方）の変数は、残らない設定の変数を残る設定に変更する（例えば残存設定がＴｒｕｅの変数が出現するまで）。また、境界変数から依存先方向（より入力に近い方）の変数は、残らない設定とする。インタフェース変数が出力のときには、入力のときと残存設定の変更が逆になる。なお外部環境モデル選択部１１

６は、利用有無が「あり」に指定された外部環境モデルをファイルに出力してもよい。

[0070] モデル合成部１１７は、モデル変換部１１３が出力するソフトウェアモデルと、外部環境モデル選択部１１６にて利用ありに指定された外部環境モデルとから、検証対象となるシステムの検証モデルを合成する。この際、外部環境モデルのインタフェース変数と、対応するソフトウェアの境界変数とで、一致しない属性情報がある場合、外部環境モデルのインタフェース変数の属性情報を変更する。例えば、インタフェース変数の名称を、境界変数の名称と同一に変更する。合成された検証モデルは、ハードディスク１４にファイルとして出力される。

[0071] 図１２は、検証モデル生成プログラム１１３１が図２や図１１に示された機能を用いて行う処理であり、システムの検証モデルを生成する処理のフローを示している。

[0072] 処理を開始すると、ステップ１２１にて、検証モデル生成プログラム１１３１は、図３の処理を行い、検証対象となるソフトウェアの依存関係情報を生成する。依存関係解析部２２から、依存関係限定部１１１に、依存関係情報が渡されて管理される。

[0073] ステップ１２２にて、外部環境モデル候補抽出部１１５は、ステップ１２１にて出力された依存関係情報をもとに、外部環境モデルデータベース１１４から外部環境モデルの候補を抽出する。

[0074] ステップ１２３にて、検証モデル生成プログラム１１３１は、入力装置１２からのユーザ入力を確認する。ユーザ入力がある変数の残存設定を変更するものであれば、ステップ１２４に進む。ユーザ入力が外部環境モデル候補抽出部１１５の抽出したある外部環境モデルの利用有無設定を変更するものであれば、ステップ１２５に進む。ユーザ入力がシステムの検証モデル合成を指示するものであれば、ステップ１２７に進む。

[0075] ステップ１２４にて、依存関係限定部１１１は、ユーザ入力に従って依存関係情報における各変数の残存設定を変更し、ステップ１２２に戻る。ステ

ップ１２２では、各変数の残存設定の変更に対応して、外部環境モデル候補の抽出を再度実行する。

[0076] ステップ１２５にて、外部環境モデル選択部１１６は、ユーザ入力に従って各外部環境モデルの利用有無設定を変更し、ステップ１２６に進む。ステップ１２６にて、依存関係限定部１１１は、ステップ１２５にて利用ありと設定された外部環境モデルの、インタフェース変数の属性から、検証対象となるソフトウェアにて対応する境界変数を検索し、依存関係情報における各変数の残存設定を変更する。変更が完了したら、ステップ１２３に戻る。

[0077] ステップ１２７にて、ソースコード抽出部１１２は、変数の残存設定を付与された依存関係情報を利用してソースコードの抽出を行う。ステップ１２８にて、モデル変換部１１３は、ステップ１２７にて抽出されたソースコードを、ソフトウェアの検証用モデルに変換する。ステップ１２９にて、モデル合成部１１７は、ステップ１２８にて出力されたソフトウェアの検証モデルと、ステップ１２５にて利用ありと設定された外部環境モデルとから、システムの検証モデルを合成して出力する。以上により処理を終了する。

[0078] 図１３の外部環境モデル属性テーブル１３１は、外部環境モデルデータベース１１４に記録されている外部環境モデルに関する属性情報の例であり、このテーブルも外部環境モデルデータベース１１４に記録されている。変数属性テーブル１３１の１行は１つの外部環境モデルに紐付いており、項目は外部環境モデル名、インタフェース変数名、ｉｎ／ｏｕｔ（インタフェース変数が入力か出力か）、（インタフェース変数の）単位、（インタフェース変数の）データ種別、（外部環境モデルの）抽象度からなる。これらの値は予め設定されている。

[0079] 外部環境モデル属性テーブル１３１には、Ｓｔｒｏｋｅ１、Ｓｔｒｏｋｅ２、Ｓｔｒｏｋｅ３の３つの外部環境モデルが登録されており、どれもインタフェース変数のデータ種別は「ストローク量」、入出力は「入力」である。一方で、インタフェース変数の単位は、Ｓｔｒｏｋｅ１とＳｔｒｏｋｅ２が「（Ａ／Ｄ変換値）」、Ｓｔｒｏｋｅ３は「ｃｍ」となっている。

[0080] 外部環境モデル候補抽出部１１５は、図１０の変数属性テーブル１０１を変数属性解釈部２５から受け取り、外部環境モデル属性テーブル１３１とデータ種別および単位が合致する外部環境モデルを抽出する。すると、変数ｒｅｇ＿ａｄ＿１とｓｔｒｏｋｅ＿ｒａｗに対してはＳｔｒｏｋｅ１とＳｔｒｏｋｅ２が、変数ｓｔｒｏｋｅに対してはＳｔｒｏｋｅ３が抽出され、外部環境モデル選択部１１６にその属性情報と共に渡される。

[0081] また依存関係限定部１１１は、外部環境モデル選択部１１６から、Ｓｔｒｏｋｅ１～Ｓｔｒｏｋｅ３のうち、ユーザに利用ありと設定された外部環境モデルのインタフェース変数に関する属性情報を受け取り、依存関係情報における各変数の残存設定を変更する。例えばユーザがＳｔｒｏｋｅ３を利用ありに設定したとき、そのインタフェース変数のデータ種別「ストローク量」と単位「ｃｍ」が依存関係限定部１１１に渡され、依存関係限定部１１１は対応する変数ｓｔｒｏｋｅを境界変数として扱う。

[0082] 図１４の検証用モデル１４０は、検証対象システムについてモデル合成部１１７が出力する検証用モデルの例である。なお検証用モデル１４０を出力する前提として、外部環境モデル選択部１１６では外部環境モデルとしてＳｔｒｏｋｅ３がユーザ入力により利用ありに設定されており、依存関係限定部１１１ではユーザ入力により変数ｅｒｒ＿ｓｔｒｏｋｅの残存設定がＦａｌｓｅにされているとする。またモデル合成部１１７が利用する変換ルールは、Ｃ言語の関数をＰｒｏｍｅｌａのインラインマクロに置換し、関数コールグラフにてどの関数からも呼び出しされていない関数に対応するインラインマクロを実行するプロセスを追加するとしている。

[0083] 検証用モデル１４０は、４つの部分から成っている。検証用モデル部品１４１は、ソフトウェアモデルと外部環境用モデルにて使用されている変数や固定値の宣言を、モデル合成部１１７がまとめたものである。検証用モデル部品１４２は、外部環境モデルデータベース１１４に記録されている外部環境モデルＳｔｒｏｋｅ３のコードについて、インタフェース変数名を境界変数ｓｔｒｏｋｅの名称に変更したものである。検証用モデル部品１４３は、

モデル変換部１１３が変換して出力したソフトウェアの検証用モデルについて、さらにモデル合成部１１７が、境界変数ｓｔｒｏｋｅ（１４行目）に値を代入する構文を、外部環境モデルＳｔｒｏｋｅ３のインラインマクロ呼び出しに置換したものである。このようなインラインマクロへの置換は、予め変換ルールとして決めておく。検証用モデル部品１４４は、モデル合成部１１７にて追加されたプロセスである。

[0084] 以上のようにしてシステムの検証用モデルをプログラムが合成することにより、ユーザは外部環境モデルの選択に掛かる工数や、ソフトウェアモデルと外部環境モデルとの入出力を整合させる手間を低減させることができる。またさらに、ユーザがソフトウェアとその外部環境の双方をモデル化する作業に掛かる工数を低減することができる。

符号の説明

[0085] １１　コンピュータ

１１１　ＣＰＵ

１１２　ＲＡＭ

１１３　ＲＯＭ

１１３１　検証モデル生成プログラム

１２　入力装置

１３　表示装置

１１４　ハードディスク

２１　ファイル入力部

２２　依存関係解析部

２３　変数属性入力部

２４　変数属性データベース

２５　変数属性解釈部

２６　推論ルールデータベース

２７　変数属性表示部

請求の範囲

[請求項1] コンピュータにソフトウェアのソースコードを入力させ、前記ソフトウェアの検証を行うためのソフトウェア検証用プログラムにおいて、

前記ソースコードに対して命令または変数の依存関係を導出する依存関係解析部と、

前記ソースコードの一部の変数について属性情報を記録する変数属性記録部と、

前記依存関係と前記変数属性情報から、前記ソースコード中の各変数について、予め設定された推論ルールに従って属性情報を推論する変数属性解釈部と、

前記属性情報を表示する変数属性情報表示部と、を備えることを特徴とするソフトウェア検証用プログラム。

[請求項2] ユーザによる変数属性情報の入力を受け付ける変数属性入力部を備え、

前記変数属性解釈部は、前記ソースコードの各変数に関する属性情報の推論に、前記変数属性部に入力された変数属性情報を利用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用プログラム。

[請求項3] 外部環境モデルをその属性情報と共に記録する外部環境モデルデータベースと、

前記外部環境モデルデータベースから、前記ソースコードを変換したソフトウェアモデルの外部環境となりうるモデルを抽出する外部環境モデル抽出部と、を備え、

前記外部環境モデル抽出部は、前記依存関係に含まれる変数属性情報から、インタフェースの属性が合致する外部環境モデルを検索して、前記ソフトウェアモデルに接続可能な外部環境モデルの候補として出力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用プログラム。

[請求項4] 前記依存関係についてソースコード抽出のための命令または変数の残

存設定の変更をユーザから受け付ける依存関係限定部を備え、

前記外部環境モデル抽出部は、前記依存関係限定部により残存設定が変更された前記依存関係にもとづき、前記インタフェースと属性が合致する変数としては前記残存設定が残る様に設定されている変数のみを用い、前記外部環境モデルデータベースから抽出する外部環境モデルの候補を絞り込むことを特徴とする請求項３記載のソフトウェア検証用プログラム。

［請求項5］ 前記依存関係についてソースコード抽出のための命令または変数の残存設定を変更する依存関係限定部を備え、

前記外部環境モデル抽出部は前記外部環境モデルデータベースから抽出された外部環境モデルに対するユーザによる選択を受け付け、ユーザが前記外部環境モデルを選択すると、前記依存関係限定部は、選択された前記前記外部環境モデルの属性情報からインタフェースが合致する変数を前記依存関係から検索し、前記ソースコードを変換したソフトウェアモデルと、前記外部環境モデルとの境界となる変数として、前記依存関係の残存設定を変更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用プログラム。

［請求項6］ 前記依存関係から前記ソースコードの一部を抽出するソースコード抽出部と、前記ソースコード抽出部によるソースコードを変換してソフトウェアモデルとするモデル変換部と、前記モデル変換部によるソフトウェアモデルと前記外部環境モデル抽出部による外部環境モデルとを合成して、前記ソフトウェアを含むシステムの検証用モデルを出力するモデル合成部と、を備え、

前記モデル合成部は、前記ソフトウェアモデルと、前記外部環境用モデルとのインタフェースについて両者が整合するように修正を行うことを特徴とする請求項３から５いずれか一項に記載のソフトウェア検証用プログラム。

［請求項7］ 前記変数属性解釈部は、前記依存関係中で、どの変数からも依存され

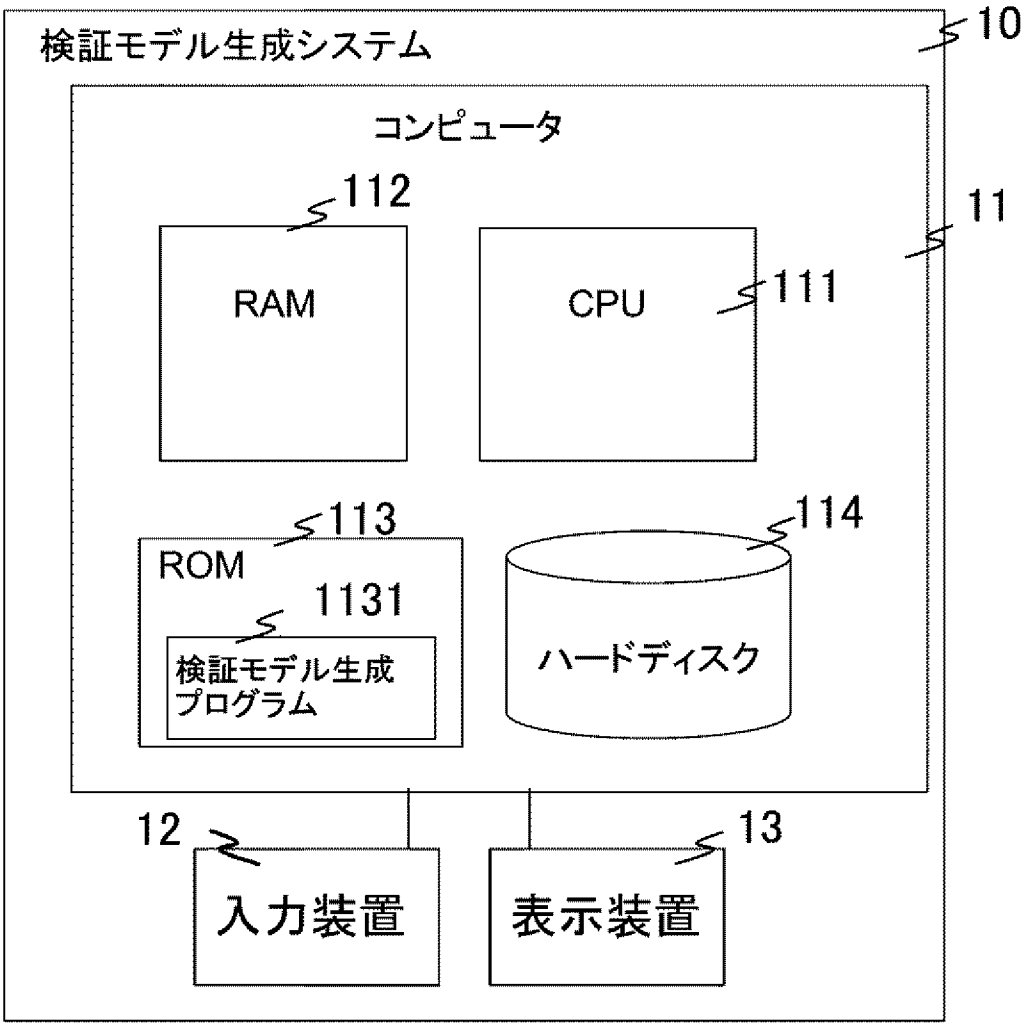
ていない変数を始端変数として選択し、依存先の変数から依存元の変数へと属性情報の推論を行うことを特徴とする請求項１記載のソフトウェア検証用プログラム。

［請求項8］ 前記変数属性解釈部は、前記依存関係中にループがあるときに、一度属性情報の推論を行った前記ループに属する変数については、推論の対象外と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用プログラム。

［請求項9］ ユーザからの入力を受け付ける入力装置と、ユーザへの出力を表示する表示装置と、請求項１記載のソフトウェア検証用プログラムを実行するコンピュータと、を備えるソフトウェア検証システム。

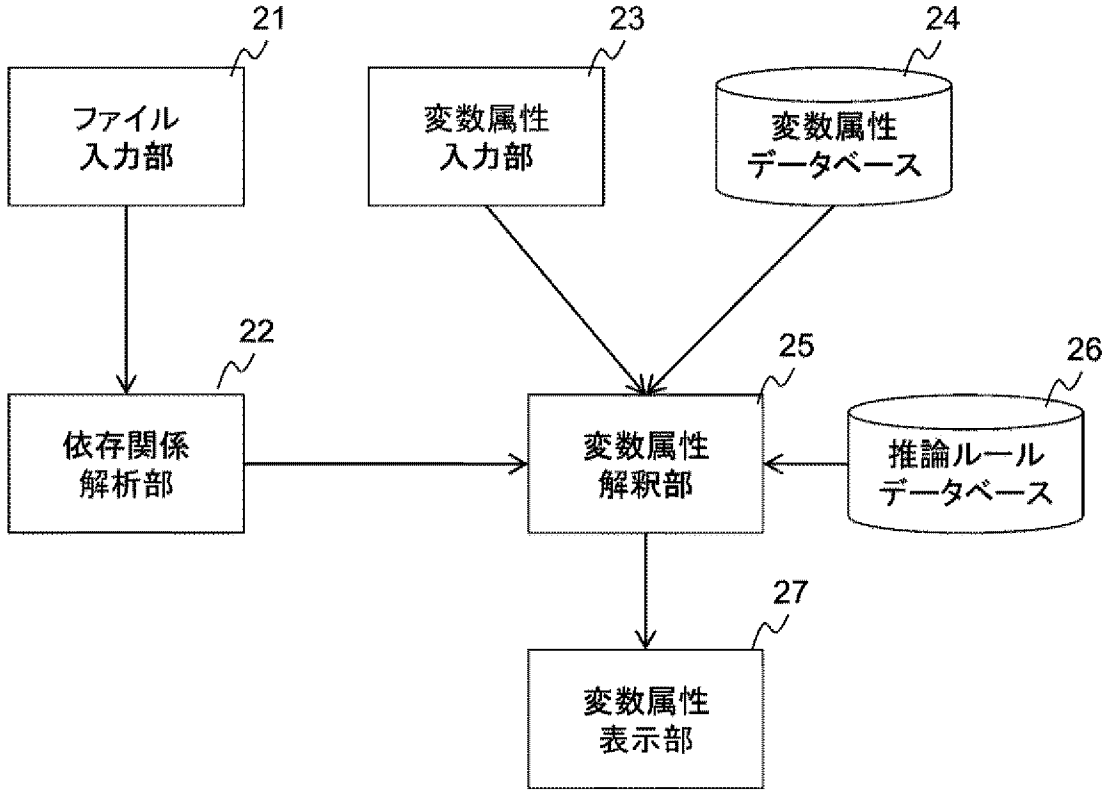
[図1]

図1


検証モデル生成システム
10
コンピュータ
11
112
RAM
CPU
111
113
ROM
1131
検証モデル生成
プログラム
114
ハードディスク
12
入力装置
13
表示装置

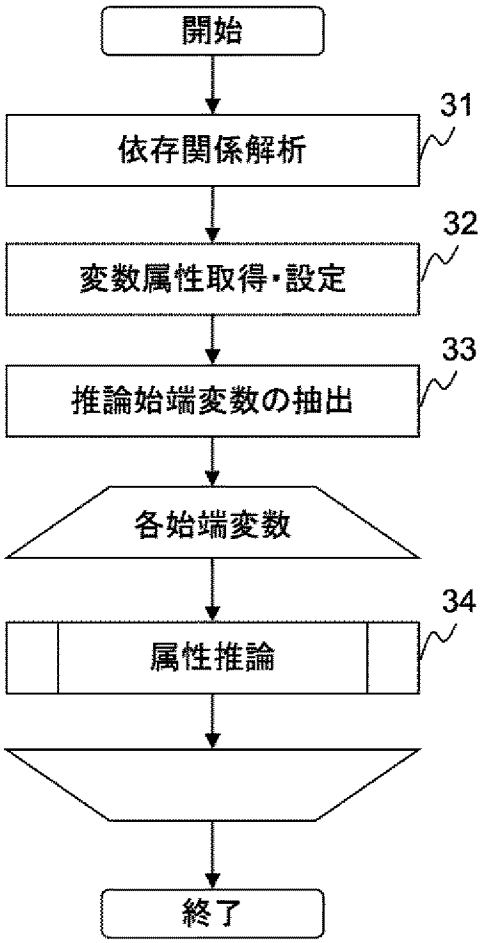
[図2]

図2


21
23
24
ファイル
入力部
変数属性
入力部
変数属性
データベース
22
25
26
依存関係
解析部
変数属性
解釈部
推論ルール
データベース
27
変数属性
表示部

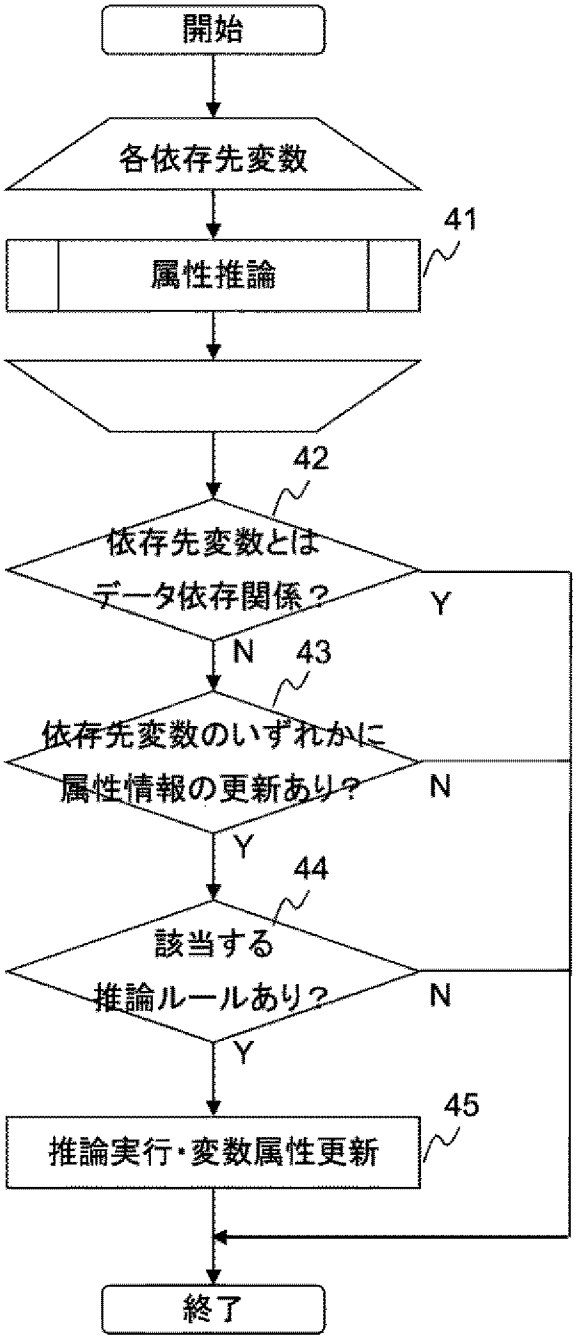
[図3]

図3



[図4]

図4


開始
各依存先変数
41
属性推論
42
依存先変数とは
データ依存関係？
Y
N
43
依存先変数のいずれかに
属性情報の更新あり？
N
Y
44
該当する
推論ルールあり？
N
Y
45
推論実行・変数属性更新
終了

[図5]

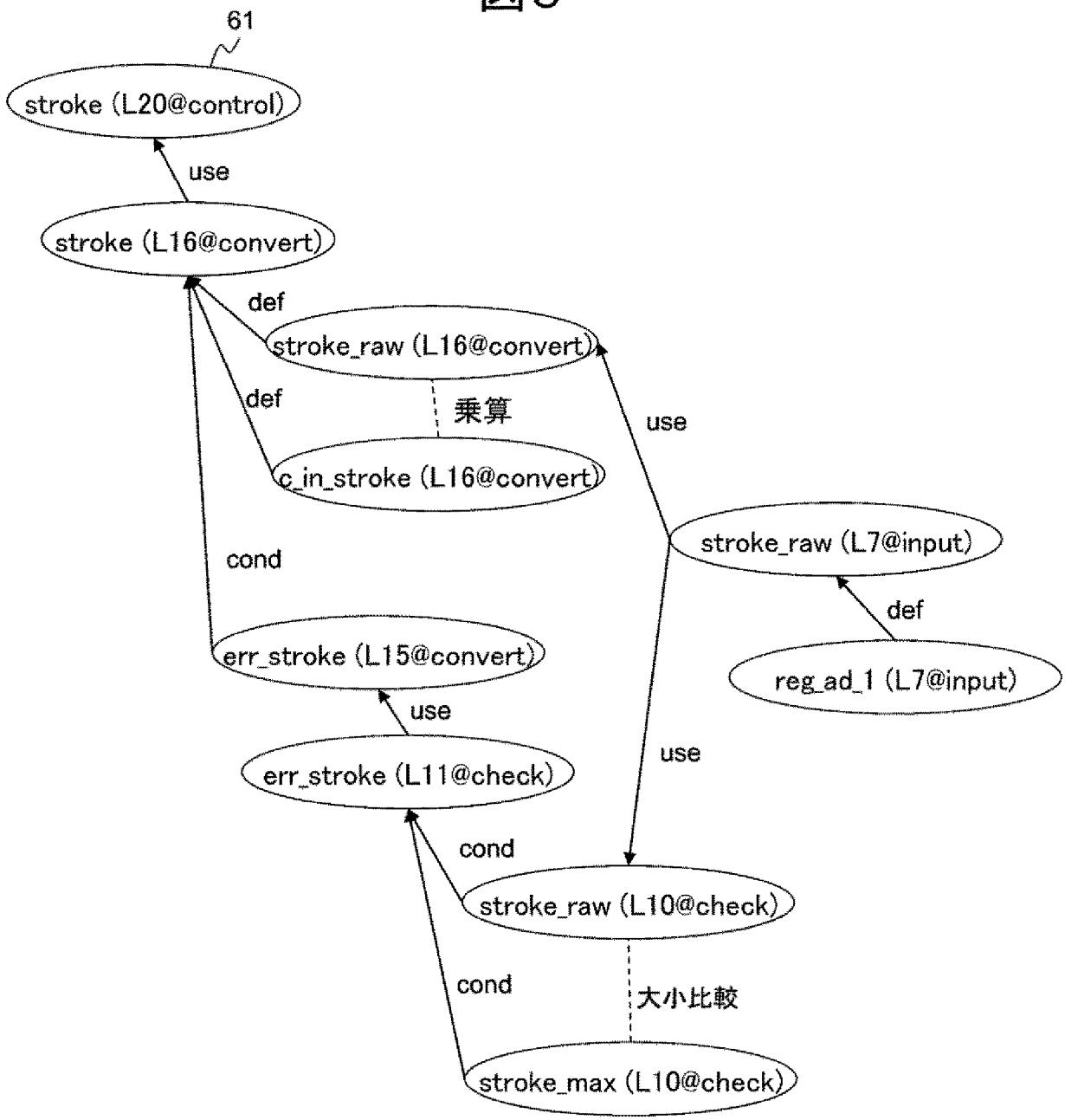
図5

50

```
00: extern unsigned short  stroke, stroke_raw;
01: extern const unsigned short  c_in_stroke = 2, c_out_stroke = 3;
02: extern const unsigned short  stroke_max;
03: extern unsigned char  err_stroke;
04: extern unsigned short  reg_ad_1, reg_da_1;
05:
06: void  input ( ) {
07:   stroke_raw = reg_ad_1;
08: }
09: void  check ( ) {
10:   if ( stroke_raw > stroke_max ) {
11:     err_stroke = 1;
12:   }
13: }
14: void  convert ( ) {
15:   if ( !err_stroke ) {
16:     stroke = c_in_stroke * stroke_raw;
17:   }
18: }
19: void  control ( ) {
20:   reg_da_1 = c_out_stroke * stroke;
21: }
22: void  task ( ) {
23:   input ( );
24:   check ( );
25:   convert ( );
26:   control ( );
27: }
```

[図6]

図6


61
stroke (L20@control)
use
stroke (L16@convert)
def
stroke_raw (L16@convert)
def
乗算
c_in_stroke (L16@convert)
use
stroke_raw (L7@input)
def
reg_ad_1 (L7@input)
cond
err_stroke (L15@convert)
use
err_stroke (L11@check)
use
cond
stroke_raw (L10@check)
cond
大小比較
stroke_max (L10@check)

[図7]

図7

71

変数名	in / out	単位	データ種別
reg_ad_1	入力	（A/D変換値）	ストローク量

[図8]

図8

81

No.	前件	後件
1	Def (X, Y)	SameAttr (X, Y)
2	Cond (X, A) and Cond (Y, A) and Comp (X, Y)	Flag (A) and SameAttr (X, Y)
3	Def (X, Y, Z) and Mul (X, Y)	MulAttr (X, Y, Z)
4	Comp (X, Y) and Vacant (Y)	Thrsh (Y) ?
5	Def (X, Y, Z) and Vacant (Y)	SameAttr (X, Z) ?
6	[2] and ErrThrsh (Y)	ErrFlag (A)

[図9]

図9

91

変数名	単位	データ種別
reg_ad_1	(A/D変換値)	ストローク量
stroke_raw	(A/D変換値)	ストローク量
stroke_max	(A/D変換値)?	閾値? (ストローク量)
c_in_stroke		
stroke		ストローク量?
err_stroke	-	判定フラグ (ストローク量)

[図10]

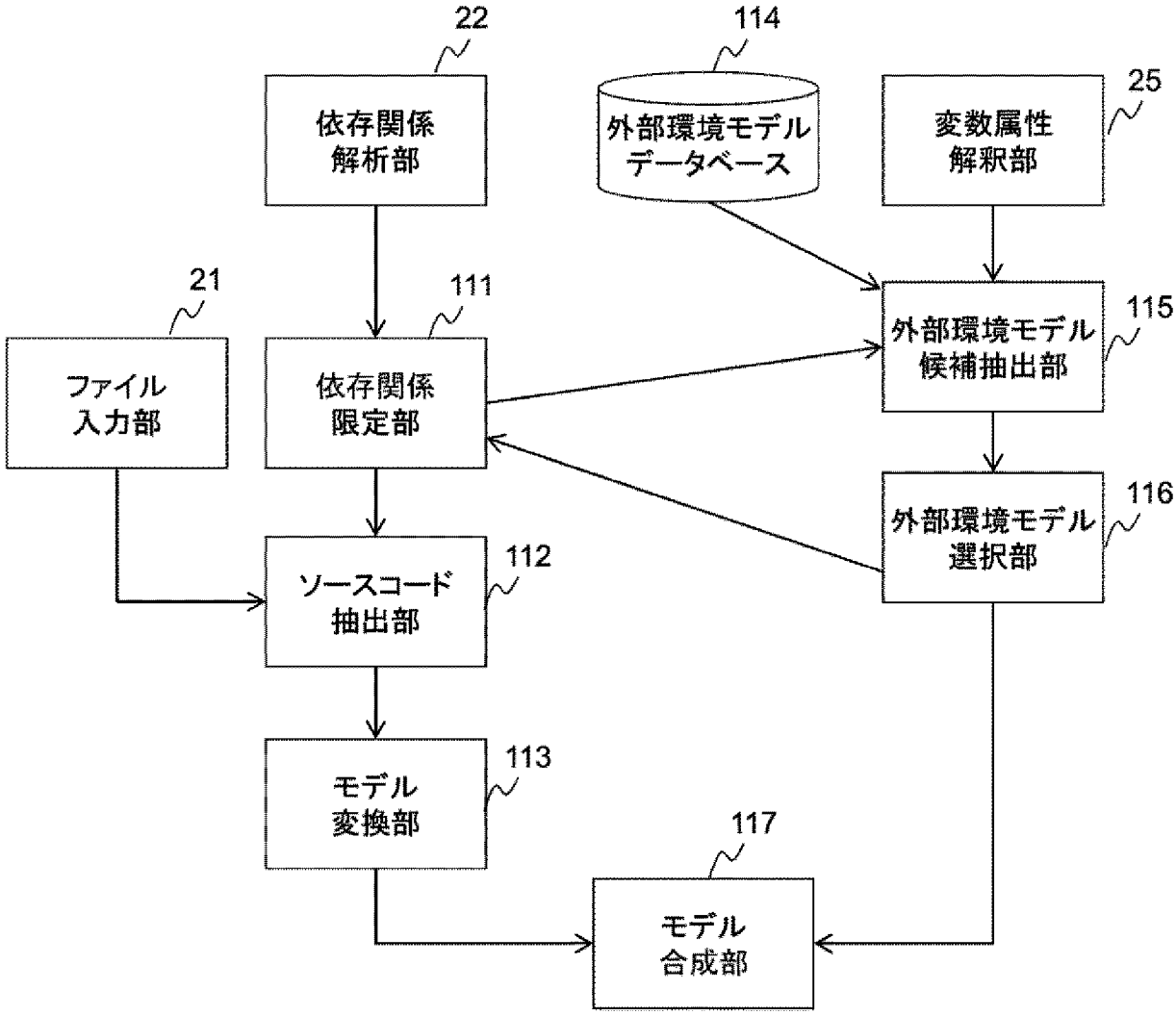
図10

101

変数名	単位	データ種別
reg_ad_1	(A/D変換値)	ストローク量
stroke_raw	(A/D変換値)	ストローク量
stroke_max	(A/D変換値)	異常閾値 (ストローク量)
c_in_stroke	cm / (A/D変換値)	係数
stroke	cm	ストローク量
err_stroke	-	異常判定フラグ (ストローク量)

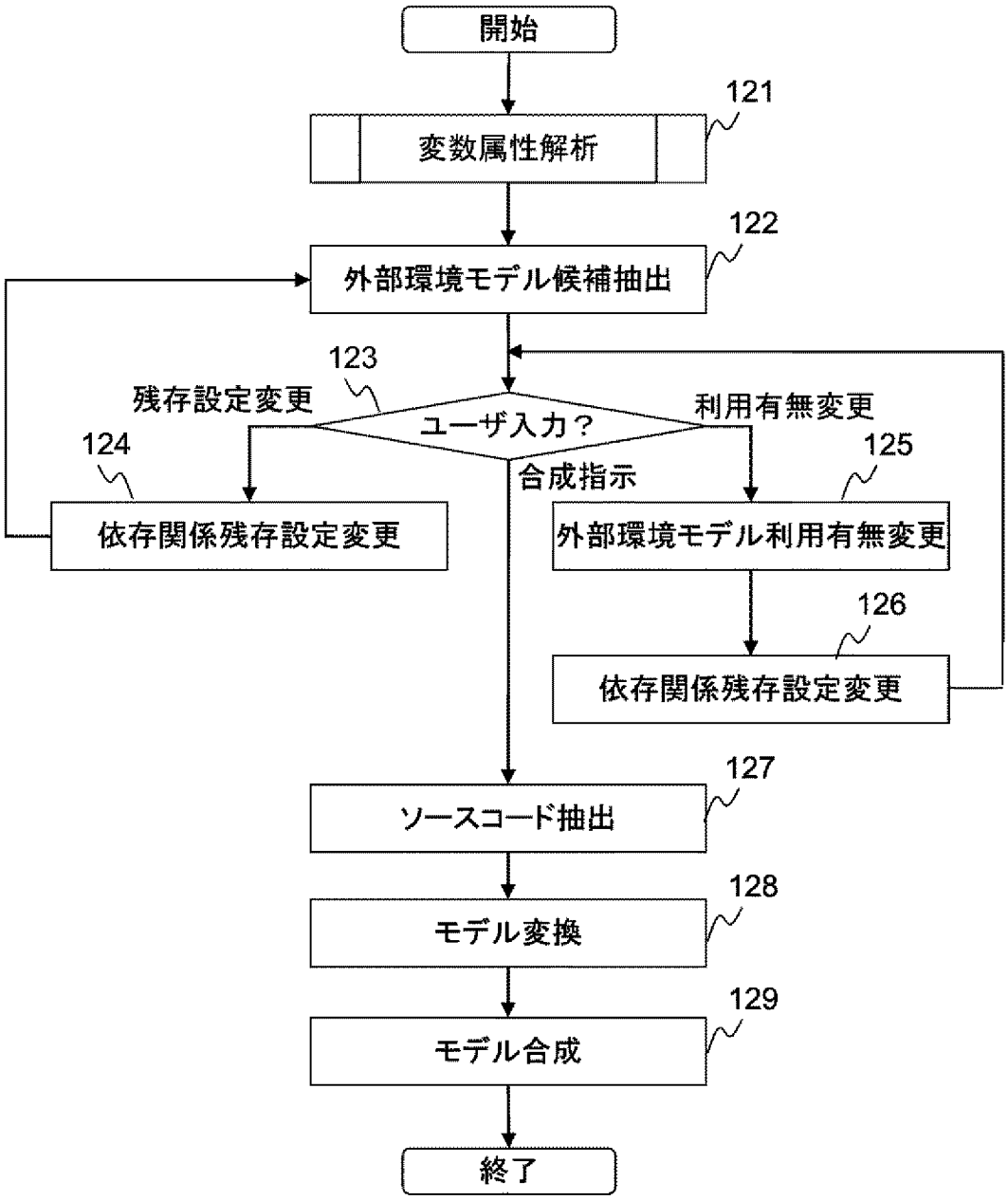
[図11]

図11


22
依存関係
解析部
114
外部環境モデル
データベース
25
変数属性
解釈部
21
ファイル
入力部
111
依存関係
限定部
115
外部環境モデル
候補抽出部
116
外部環境モデル
選択部
112
ソースコード
抽出部
113
モデル
変換部
117
モデル
合成部

[図12]

図12


開始
121
変数属性解析
122
外部環境モデル候補抽出
123
残存設定変更
ユーザ入力？
利用有無変更
124
合成指示
125
依存関係残存設定変更
外部環境モデル利用有無変更
126
依存関係残存設定変更
127
ソースコード抽出
128
モデル変換
129
モデル合成
終了

[図13]

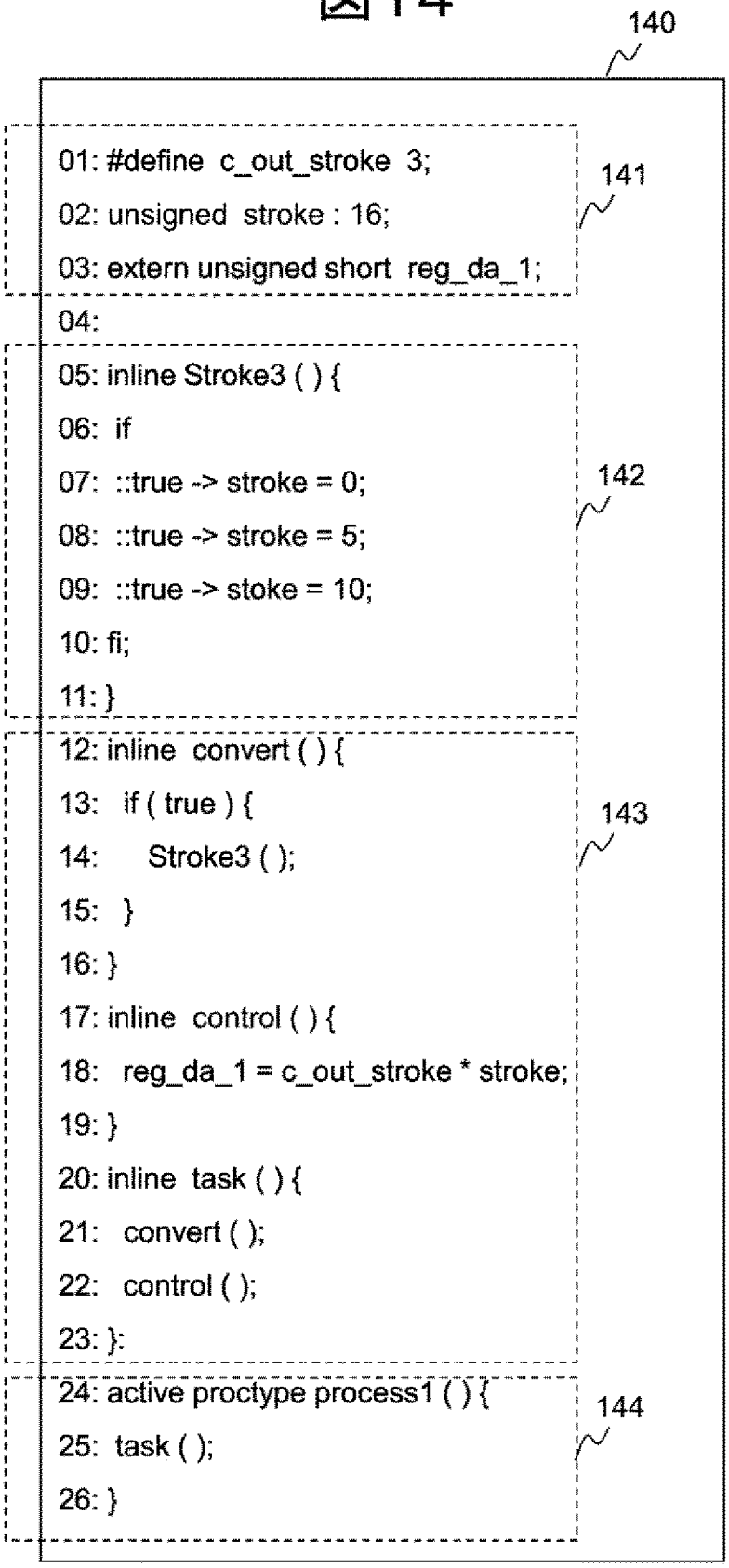
図13

131

外部環境モデル名	インタフェース変数名	in / out	単位	データ種別	抽象度
Stroke1	in_stroke	入力	(A/D変換値)	ストローク量	小
Stroke2	in_stroke	入力	(A/D変換値)	ストローク量	中
Stroke3	in_stroke	入力	cm	ストローク量	大

[図14]

図14


140
01: #define c_out_stroke 3;
02: unsigned stroke : 16;
03: extern unsigned short reg_da_1;
141
04:
05: inline Stroke3 ( ) {
06: if
07: ::true -> stroke = 0;
08: ::true -> stroke = 5;
09: ::true -> stoke = 10;
10: fi;
11: }
142
12: inline convert ( ) {
13: if (true) {
14: Stroke3 ( );
15: }
16: }
17: inline control ( ) {
18: reg_da_1 = c_out_stroke * stroke;
19: }
20: inline task ( ) {
21: convert ( );
22: control ( );
23: };
143
24: active proctype process1 ( ) {
25: task ( );
26: }
144

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2013/069967

A. CLASSIFICATION OF SUBJECT MATTER
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)
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.
X A	JP 11-203116 A (Fujitsu Ltd.), 30 July 1999 (30.07.1999), paragraphs [0066] to [0068]; fig. 10 (Family: none)	1-2,9 3-8
A	JP 2005-18114 A (International Business Machines Corp.), 20 January 2005 (20.01.2005), entire text; all drawings & US 2008/0295080 A1	3-8

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

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

Date of the actual completion of the international search	Date of mailing of the international search report
22 October, 2013 (22.10.13)	05 November, 2013 (05.11.13)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ２０１３／０６９９６７

Ａ．　発明の属する分野の分類（国際特許分類（ＩＰＣ））

Int.Cl.　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.　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
Ｘ	JP 11-203116 A（富士通株式会社）1999.07.30，段落【００６６】－【００６８】，第１０図（ファミリーなし）	１－２，９
Ａ		３－８
Ａ	JP 2005-18114 A（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）2005.01.20，全文、全図 & US 2008/0295080 A1	３－８

☐　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 ２２．１０．２０１３	国際調査報告の発送日 ０５．１１．２０１３
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　３６５２ 塚田　肇 電話番号　０３－３５８１－１１０１　内線　３５４５