(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
2024年4月18日(18.04.2024)

WIPO | PCT

(10) 国際公開番号
WO 2024/079800 A1

(51) 国際特許分類:
G06F 9/455 (2006.01) *G06F 11/36* (2006.01)

(21) 国際出願番号： PCT/JP2022/037936

(22) 国際出願日： 2022年10月11日(11.10.2022)

(25) 国際出願の言語： 日本語

(26) 国際公開の言語： 日本語

(71) 出願人: 日本電信電話株式会社 (**NIPPON TELEGRAPH AND TELEPHONE CORPORATION**) [JP/JP]; 〒1008116 東京都千代田区大手町一丁目５番１号 Tokyo (JP).

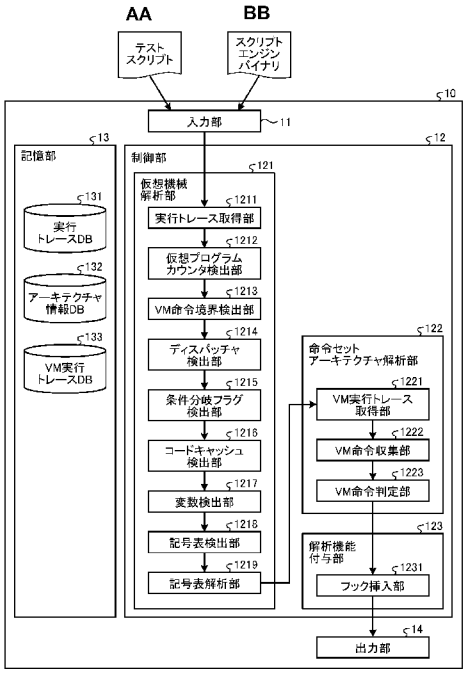
(72) 発明者: 碓井 利宣(**USUI, Toshinori**); 〒1808585 東京都武蔵野市緑町３丁目９－１１ ＮＴＴ知的財産センタ内 Tokyo (JP). 川古谷 裕平(**KAWAKOYA, Yuhei**); 〒1808585 東京都武蔵野市緑町３丁目９－１１ ＮＴＴ知的財産センタ内 Tokyo (JP). 岩村 誠(**IWAMURA, Makoto**); 〒1808585 東京都武蔵野市緑町３丁目９－１１ ＮＴＴ知的財産センタ内 Tokyo (JP).

(74) 代理人: 弁理士法人酒井国際特許事務所 (**SAKAI INTERNATIONAL PATENT OFFICE**); 〒1000013 東京都千代田区霞が関３丁目８番１号 虎ノ門ダイビルイースト 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, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN,

(54) **Title:** ANALYSIS FUNCTION ADDITION DEVICE, ANALYSIS FUNCTION ADDITION METHOD, AND ANALYSIS FUNCTION ADDITION PROGRAM

(54) 発明の名称： 解析機能付与装置、解析機能付与方法および解析機能付与プログラム



11	Input unit
12	Control unit
13	Storage unit
14	Output unit
121	Virtual machine analysis unit
122	Instruction set architecture analysis unit
123	Analysis function addition unit
131	Execution trace database
132	Architecture information database
133	VM execution trace database
1211	Execution trace acquisition unit
1212	Virtual program counter detection unit
1213	VM instruction boundary detection unit
1214	Dispatcher detection unit
1215	Conditional branch flag detection unit
1216	Code cache detection unit
1217	Variable detection unit
1218	Symbol table detection unit
1219	Symbol table analysis unit
1221	VM execution trace acquisition unit
1222	VM instruction collection unit
1223	VM instruction determination unit
1231	Hook insertion unit
AA	Test script
BB	Script engine binary

(57) **Abstract:** A virtual machine analysis unit (121) analyzes a VM of a script engine and acquires information relating to the architecture of the script engine. A symbol table detection unit (1218) detects a symbol table, which holds information relating to variables, on the basis of the acquired information relating to the architecture. A symbol table analysis unit (1219) analyzes the structure of the symbol table. An instruction set architecture analysis unit (122) acquires information pertaining to an instruction set architecture, which is the structure of virtual machine instructions, on the basis of the

WO 2024/079800 A1

［続葉有］

HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, 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, SC, SD, SL, ST, 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, ME, 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)）

acquired information relating to the architecture. A VM instruction determination unit (1223) determines, using the symbol table and the information pertaining to the instruction set architecture, a virtual machine instruction that corresponds to a variable held by the symbol table.

(57) 要約：仮想機械解析部（１２１）が、スクリプトエンジンのＶＭを解析し、スクリプトエンジンのアーキテクチャに関する情報を取得する。記号表検出部（１２１８）が、取得されたアーキテクチャに関する情報を基に、変数に関する情報を保持する記号表を検出する。記号表解析部（１２１９）が、該記号表の構造を解析する。命令セットアーキテクチャ解析部（１２２）が、取得されたアーキテクチャに関する情報を基に、仮想機械の命令の体系である命令セットアーキテクチャの情報を取得する。ＶＭ命令判定部（１２２３）が、記号表と、命令セットアーキテクチャの情報とを用いて、該記号表が保持する変数に対応する仮想機械の命令を判定する。

明　細　書

発明の名称 ：

解析機能付与装置、解析機能付与方法および解析機能付与プログラム

技術分野

[0001] 本発明は、解析機能付与装置、解析機能付与方法および解析機能付与プログラムに関する。

背景技術

[0002] ソフトウェアを解析する際に重要な技術の一つに、プログラムが保持する変数の値の追跡がある。これは、プログラムの実行中のある時刻に、どのような変数がプログラムによって使用されており、それらがどのような値を保持しているか、といった情報を取得するものである。例えば、デバッガは実行中の変数や値の情報を追跡する機能を有しており、プログラムの開発やデバッグの際に広く利用されている。どのような変数があり、それらがどのような値を保持しているかは、プログラムの振る舞いに大きく関わるため、ソフトウェアのテストやリバースエンジニアリングなどにおいて、重要な技術である（非特許文献１参照）。

[0003] また、スクリプトによるプログラムの開発が普及している現在では、スクリプトの保持する変数の追跡も、同じく重要な技術である。スクリプトにおいては、変数に関する情報や保持している値は、スクリプトエンジン（インタプリタとも呼ばれる）の持つ記号表によって管理されており、スクリプトエンジンの提供するデバッガなどの解析支援機能を通してその記号表にアクセスし、情報を取得するのが一般的である。

先行技術文献

非特許文献

[0004] 非特許文献1：JongHyup Lee, Thanassis Avgerinos, David Brumley, “TIE: Principled Reverse Engineering of Types in Binary Programs”

発明の概要

発明が解決しようとする課題

[0005] しかしながら、従来技術によれば、変数の情報を取得することが困難な場合がある。これは、スクリプトの実行を司るスクリプトエンジン内の仮想機械（ＶＭ, Virtual Machine）の内部仕様が公開されていない場合が多く、記号表の格納されている位置の特定や、記号表の構造の解析を、支援機能なしに実現するのが難しいためである。例えば、上記のデバッガなどの支援機能が提供されていない場合、ＶＭをリバースエンジニアリングして内部仕様を明らかにし、独自に記号表の解析を実現し、変数や値の情報を獲得する必要がある。これをスクリプトエンジンに対して手動で個別に解析や設計および実装をするのは、かかる労力の観点から、現実的でない。

[0006] 本発明は、上記に鑑みてなされたものであって、デバッガなどの支援機能がなく、内部仕様が未知のＶＭを持つスクリプトエンジンに対しても、手動での個別の解析や設計および実装を要さず、変数の情報を取得する機能の付与を実現可能とすることを目的とする。

課題を解決するための手段

[0007] 上述した課題を解決し、目的を達成するために、本発明に係る解析機能付与装置は、スクリプトエンジンの仮想機械を解析し、スクリプトエンジンのアーキテクチャに関する情報を取得する第１の取得部と、取得された前記アーキテクチャに関する情報を基に、変数に関する情報を保持する記号表を検出する検出部と、検出された前記記号表の構造を解析する解析部と、取得された前記アーキテクチャに関する情報を基に、仮想機械の命令の体系である命令セットアーキテクチャの情報を取得する第２の取得部と、解析された前記記号表と、前記命令セットアーキテクチャの情報とを用いて、該記号表が保持する変数に対応する前記仮想機械の命令を判定する判定部と、を有することを特徴とする。

発明の効果

[0008] 本発明によれば、デバッガなどの支援機能がなく、内部仕様が未知のＶＭを持つスクリプトエンジンに対しても、手動での個別の解析や設計および実装を要さず、変数の情報を取得する機能の付与を実現することが可能となる。

図面の簡単な説明

[0009] [図1]図１は、本実施形態の解析機能付与装置の概略構成を例示する模式図である。

[図2]図２は、仮想プログラムカウンタ検出に用いるテストスクリプト（第一のテストスクリプト）の一例を示す図である。

[図3]図３は、条件分岐フラグ検出に用いるテストスクリプト（第二のテストスクリプト）の一例を示す図である。

[図4]図４は、実行トレースの一例を示す図である。

[図5]図５は、ＶＭ実行トレースの一例を示す図である。

[図6]図６は、解析機能付与処理の処理手順を示すフローチャートである。

[図7]図７は、実行トレース取得処理の処理手順を示すフローチャートである。

[図8]図８は、仮想プログラムカウンタ検出処理の処理手順を示すフローチャートである。

[図9]図９は、ＶＭ命令境界検出処理の処理手順を示すフローチャートである。

[図10]図１０は、ディスパッチャ検出処理の処理手順を示すフローチャートである。

[図11]図１１は、条件分岐フラグ検出処理の処理手順を示すフローチャートである。

[図12]図１２は、コードキャッシュ検出処理の処理手順を示すフローチャートである。

[図13]図１３は、変数検出処理の処理手順を示すフローチャートである。

[図14]図１４は、記号表検出処理の処理手順を示すフローチャートである。

[図15]図１５は、記号表解析処理の処理手順を示すフローチャートである。

[図16]図１６は、ＶＭ実行トレース取得処理の処理手順を示すフローチャートである。

[図17]図１７は、ＶＭ命令収集処理の処理手順を示すフローチャートである。

[図18]図１８は、ＶＭ命令判定処理の処理手順を示すフローチャートである。

[図19]図１９は、フック挿入処理の処理手順を示すフローチャートである。

[図20]図２０は、解析機能付与プログラムを実行するコンピュータの一例を示す図である。

発明を実施するための形態

[0010] 以下、図面を参照して、本発明の一実施形態を詳細に説明する。なお、この実施形態により本発明が限定されるものではない。また、図面の記載において、同一部分には同一の符号を付して示している。

[0011] ［解析機能付与装置の概要］

本実施形態の解析機能付与装置は、スプリクトエンジンに適用され、スクリプトエンジンのバイナリを監視しながらテストスクリプトを実行して、ブランチトレースとメモリアクセストレースとを実行トレースとして取得する。そして、解析機能付与装置は、実行トレースに基づいてＶＭを解析し、スクリプトエンジンのアーキテクチャに関するアーキテクチャ情報であるＶＰＣ（仮想プログラムカウンタ）、ディスパッチャ、条件分岐フラグ、コードキャッシュを取得する。

[0012] さらに、解析機能付与装置は、ＶＰＣおよびディスパッチャを監視しながらテストスクリプトを実行して、ＶＭ実行トレースを取得する。このＶＭ実行トレースを解析することにより、ＶＭ命令を収集するとともに、ＶＭ命令の判定を行い、命令セットアーキテクチャの情報を取得する。

[0013] 解析機能付与装置は、得られたアーキテクチャ情報を基に、記号表を検出して解析を行い、スクリプトエンジンの変数の情報を取得する機能を付与す

る。このように、解析機能付与装置は、ＶＭの内部仕様が未知であるスクリプトエンジンに対しても、実行トレースおよびＶＭ実行トレースの取得に基づく解析により、各種アーキテクチャ情報を検出し、人手によるリバースエンジニアリングを要することなく、変数を取得する機能の付与を実現する。

[0014] また、解析機能付与装置は、多様なスクリプトエンジンに対して、テストスクリプトを用意すれば自動で変数の取得機能を付与することが可能であり、個別の設計や実装を要しない。したがって、様々なスクリプト言語で作成されたスクリプトに対しても、変数の取得が可能となり、より高度なスクリプト解析を実現可能となる。

[0015] ［解析機能付与装置の構成］

図１は、本実施形態の解析機能付与装置の概略構成を例示する模式図である。図１に例示するように、本実施形態の解析機能付与装置１０は、パソコン等の汎用コンピュータで実現され、入力部１１、制御部１２、記憶部１３、および出力部１４を備える。

[0016] 入力部１１は、キーボードやマウス等の入力デバイスを用いて実現され、操作者や外部からの情報の入力を受け付けて、制御部１２に対して入力する。例えば、入力部１１は、テストスクリプトやスクリプトエンジンバイナリの入力を受け付ける。また、入力部１１は、電気通信回線を介した外部の装置から送信された情報の入力を受け付ける。

[0017] テストスクリプトは、スクリプトエンジンを動的解析して実行トレースおよびＶＭ実行トレースを取得する際に、入力されるスクリプトである。このテストスクリプトは、分岐命令の実行やメモリ読み書きの回数に着目し、異なる回数のテストスクリプトを実行したときに生じるスクリプトエンジンの挙動の差分を捉えるために用いられる。このテストスクリプトは、解析の前に予め準備するものであり、手動で作成するものである。この作成には、対象のスクリプト言語の仕様に関する知識が必要となる。本実施形態の解析機能付与装置では、ＶＰＣ検出に用いるテストスクリプト（第一のテストスクリプト）と、条件分岐フラグ検出に用いるテストスクリプト（第二のテスト

スクリプト）とは異なる。

[0018] ここで、図２は、ＶＰＣ検出に用いるテストスクリプト（第一のテストスクリプト）の一例を示す図である。第一のテストスクリプトでは、繰り返し処理を用いる（２行目）。第一のテストスクリプトでは、テストスクリプト内の繰り返し回数（２行目）や繰り返される文の数（３行目から５行目）を増減させることで、実行時の条件を変更し、差分を発生させる。

[0019] また、図３は、条件分岐フラグ検出に用いるテストスクリプト（第二のテストスクリプト）の一例を示す図である。第二のテストスクリプトでは、複数回の条件分岐を用いる（４行目から８行目）。第二のテストスクリプトにおいて、この複数回の条件分岐では、特定の順序のパターンで分岐がなされたり、なされなかったりするように、分岐条件を制御する（１行目、５行目）。第二のテストスクリプトでは、条件分岐の回数や、分岐の成否の順序パターンを変更し、差分を発生させる。

[0020] スクリプトエンジンバイナリは、スクリプトエンジンを構成する実行可能ファイルである。スクリプトエンジンバイナリは、複数の実行可能ファイルによって構成される場合がある。

[0021] 出力部１４は、液晶ディスプレイなどの表示装置、プリンター等の印刷装置等によって実現される。例えば、出力部１４には、後述する解析機能付与処理の結果が表示される。また、出力部１４は、外部の装置に各種情報を出力してもよい。

[0022] 記憶部１３は、ＲＡＭ（Random Access Memory）、フラッシュメモリ（Flash Memory）等の半導体メモリ素子、または、ハードディスク、光ディスク等の記憶装置によって実現される。記憶部１３には、解析機能付与装置１０を動作させる処理プログラムや、処理プログラムの実行中に使用されるデータなどが予め記憶され、あるいは処理の都度一時的に記憶される。

[0023] 記憶部１３は、実行トレースデータベース（ＤＢ）１３１、ＶＭ実行トレースＤＢ１３３およびアーキテクチャ情報ＤＢ１３２を記憶する。

[0024] 実行トレースＤＢ１３１およびＶＭ実行トレースＤＢ１３３は、それぞれ

実行トレース取得部１２１１およびＶＭ実行トレース取得部１２２１によって取得された実行トレースおよびＶＭ実行トレースを格納する。実行トレースＤＢ１３１およびＶＭ実行トレースＤＢ１３３は、解析機能付与装置１０によって管理される。もちろん、実行トレースＤＢ１３１およびＶＭ実行トレースＤＢ１３３は、他の装置（サーバ等）によって管理されていてもよく、この場合には、実行トレース取得部１２１１およびＶＭ実行トレース取得部１２２１は、出力部１４の通信インタフェースを介して、取得した実行トレースおよびＶＭ実行トレースを、実行トレースＤＢ１３１およびＶＭ実行トレースＤＢ１３３の管理サーバ等に出力して、実行トレースＤＢ１３１およびＶＭ実行トレースＤＢ１３３に記憶させる。

[0025] 制御部１２は、ＣＰＵ（Central Processing Unit）やＭＰＵ（Micro Processing Unit）等を用いて実現され、メモリに記憶された処理プログラムを実行する。これにより、制御部１２は、図１に例示するように、仮想機械解析部１２１（第１の取得部）、命令セットアーキテクチャ解析部１２２（第２の取得部）および解析機能付与部１２３（付与部）として機能する。

[0026] 仮想機械解析部（第１の取得部）１２１は、スクリプトエンジンのＶＭを解析し、スクリプトエンジンのアーキテクチャに関する情報を取得する。具体的には、仮想機械解析部（第１の取得部）１２１は、スクリプトエンジンのバイナリを監視しながらテストスクリプトを実行して、ブランチトレースとメモリアクセストレースとを実行トレースとして取得する。また、仮想機械解析部１２１は、実行トレースに基づいてＶＭを解析し、スクリプトエンジンのアーキテクチャに関する情報を取得する。アーキテクチャ情報は、仮想プログラムカウンタ、ディスパッチャ、条件分岐フラグ、またはコードキャッシュのいずれかを含む。

[0027] 仮想機械解析部１２１は、実行トレース取得部１２１１、仮想プログラムカウンタ検出部１２１２、ＶＭ命令境界検出部１２１３、ディスパッチャ検出部１２１４、条件分岐フラグ検出部１２１５、コードキャッシュ検出部１２１６、変数検出部１２１７、記号表検出部１２１８および記号表解析部１

２１９を有する。

[0028] 実行トレース取得部１２１１は、テストスクリプトおよびスクリプトエンジンバイナリを入力として受け付ける。実行トレース取得部１２１１は、スクリプトエンジンバイナリの実行を監視しながら、テストスクリプトを実行することで、実行トレースを取得する。

[0029] 実行トレースは、ブランチトレースとメモリアクセストレースとによって構成される。ブランチトレースは、実行の際の分岐命令の種類と、分岐元アドレスと分岐先アドレスを記録する。メモリアクセストレースは、メモリ操作の種類と、操作対象のメモリアドレスを記録する。ブランチトレースおよびメモリアクセストレースは、命令フックによって取得可能であることが知られている。実行トレース取得部１２１１が取得した実行トレースは、実行トレースＤＢ１３１に格納される。

[0030] ここで、図４は、実行トレースの一例を示す図である。実行トレースは、traceという要素を有する。traceには、そのログ行がブランチトレースか、メモリアクセストレースかが示される。

[0031] ブランチトレースのログ行は、例えば、図４の１行目から１０行目に記載の書式になっており、type、src、dstの三つの要素からなる。typeは、実行された分岐命令がcall命令によるものか、jmp命令によるものか、ret命令によるものかを示す。また、srcは、分岐元のアドレスを示し、dstは、分岐先のアドレスを示す。

[0032] メモリアクセストレースのログ行は、たとえば、図４の１１行目から１３行目に記載の書式になっており、type、target、valueの三つの要素からなる。typeは、メモリアクセスが読み込みか書き込みかを示す。targetは、メモリアクセスの対象となるメモリアドレスを示す。また、valueには、メモリアクセスの結果の値が格納される。

[0033] 仮想プログラムカウンタ検出部１２１２は、実行トレースＤＢ１３１に格納された第一のテストスクリプトに対する実行トレースを取り出して解析し、ＶＰＣを検出する。仮想プログラムカウンタ検出部１２１２は、メモリの

読み込み回数に着目した差分実行解析とＶＭ命令境界検出部１２１３によって検出された各ＶＭ命令の境界とを用いて複数の実行トレースを解析し、ＶＰＣを検出する。仮想プログラムカウンタ検出部１２１２は、各ＶＭ命令の実行後には、必ずＶＰＣを保持するメモリへの読み込みが発生することを利用し、この読み込み先を発見することで、ＶＰＣを検出する。

[0034] このため、仮想プログラムカウンタ検出部１２１２は、ＶＰＣの検出として、メモリの読み込み回数に着目した差分実行解析を用いる。仮想プログラムカウンタ検出部１２１２は、テストスクリプトを用いて取得された複数のテストスクリプトの実行トレースを比較し、メモリ読み込み回数が、繰り返される回数および繰り返される文の数との双方の増減に比例して変化するメモリを発見する。そして、仮想プログラムカウンタ検出部１２１２は、ＶＭ命令境界検出部１２１３によって検出された各ＶＭ命令の境界を参照して、読み込んだメモリの値が常にＶＭ命令の開始点を指しているものに絞り込む。仮想プログラムカウンタ検出部１２１２は、このメモリをＶＰＣとして検出する。

[0035] ＶＭ命令境界検出部１２１３は、実行トレースをクラスタリングして、各ＶＭ命令の境界を検出する。ＶＭ命令境界検出部１２１３は、実行トレースをクラスタリングして、実行回数が閾値以上のクラスタをＶＭ命令として検出する。クラスタリングでは、複数回実行される連続したコード領域を検出する。これにはたとえば、実行された命令間のコード上の距離が近いものをまとめてもよいし、実行されたコードブロックの共通部分列を探してもよいし、他の方法によってもよい。解析機能付与装置１０は、検出したＶＭ命令を構成する連続した命令列の開始点と終了点とを境界として検出する。ここで検出したＶＭ命令の境界は、ＶＰＣ検出、ディスパッチャ検出において用いられる。

[0036] ディスパッチャ検出部１２１４は、ＶＭ命令境界検出部１２１３が検出したＶＭ命令の境界を基に、スクリプトエンジンバイナリから各ＶＭ命令部分を切り出し、各ＶＭ命令間で類似度が高い部分をディスパッチャとして検出

する。類似度の高い部分の検出には、たとえば系列アライメントアルゴリズムを用いてもよく、その他の方法によってもよい。

[0037] 条件分岐フラグ検出部１２１５は、実行トレースＤＢ１３１に格納された第二のテストスクリプトに対する実行トレースを取り出して解析し、条件分岐フラグを発見する。ここで、スクリプト内で分岐を発生させるＶＭ命令が分岐ＶＭ命令であり、条件分岐フラグは、条件分岐時に分岐がなされるか否かのフラグを保持する領域である。条件分岐フラグ検出部１２１５は、メモリの読み込み回数に着目した差分実行解析を用いて、複数の実行トレースを解析し、条件分岐フラグを検出する。条件分岐フラグ検出部１２１５は、様々なパターンで条件分岐を実行し、その際のメモリの変化のパターンをテストスクリプト上の条件分岐のパターンと照らし合わせることで、条件分岐フラグを格納するメモリを検出する。

[0038] コードキャッシュ検出部１２１６は、実行トレースおよびＶＭ実行トレースを入力として受け付けて、ＶＰＣが指し示すメモリ領域をＶＭ実行トレースから取得する。また、コードキャッシュ検出部１２１６は、当該メモリ領域を確保したメモリ割り当て関数の呼び出し元のコード箇所を実行トレースから取得する。また、コードキャッシュ検出部１２１６は、当該コード箇所で確保された全ての領域をコードキャッシュとして検出する。そして、コードキャッシュ検出部１２１６は、コードキャッシュに書き込みをしているコード箇所を実行トレースから取得する。また、コードキャッシュ検出部１２１６は、当該コード箇所で書き込みされた全ての領域をコードキャッシュの更新として検出し、コードキャッシュおよび更新箇所を返す。

[0039] 変数検出部１２１７は、テストスクリプトを入力として受け付けて、実行トレースＤＢ１３１からテストスクリプトに対応した実行トレースを取り出す。また、変数検出部１２１７は、テストスクリプト中に記述された変数名と値との組を取り出す。そして、変数検出部１２１７は、メモリアクセストレースの書き込みから当該変数名との一致性が所定の閾値より高い値を探索する。また、変数検出部１２１７は、メモリアクセストレースの書込みから

、当該値との一致性が所定の閾値より高い値を探索する。そして変数検出部１２１７は、当該変数に対応する変数名と値の格納先として返す。

[0040] 記号表検出部（検出部）１２１８は、得られたアーキテクチャに関する情報を基に、記号表を検出する。具体的には、記号表検出部１２１８は、スクリプトエンジンバイナリを入力として受け付けて、変数と対応する値の格納先を受け付ける。また、記号表検出部１２１８は、実行トレースＤＢ１３１から実行トレースを取り出して、スクリプトエンジンバイナリの静的解析により値の格納先の周辺に関するメモリ参照の関係を収集する。また、記号表検出部１２１８は、メモリアクセストレースから変数名および値への実行時のメモリ参照を収集し、変数名および値への参照を持つ配列を記号表として検出する。そして、記号表検出部１２１８は、ＡＰＩトレースから、記号表のメモリ領域を確保しているコード箇所を特定し、実行時に記号表の位置を特定可能にして、記号表を返す。

[0041] また、記号表解析部（解析部）１２１９は、検出された記号表の構造を解析する。具体的には、記号表解析部１２１９は、検出された記号表、テストスクリプト、およびスクリプトエンジンバイナリを入力として受け付ける。また、記号表解析部１２１９は、実行トレースＤＢ１３１から実行トレースを取り出して、スクリプトエンジンバイナリの静的解析を実施して、変数の依存関係を収集する。記号表解析部１２１９は、例えば非特許文献１に開示されているような所定の方法で、記号表を構成する構造体の型を推論する。そして、記号表解析部１２１９は、メモリアクセストレースを基に、記号表を構成する構造体の保持する値とテストスクリプトで確保した変数の型の共起から、型情報を保持する変数を特定し、記号表の構造を返す。

[0042] 命令セットアーキテクチャ解析部（第２の取得部）１２２は、取得されたアーキテクチャに関する情報を基に、ＶＭの命令の体系である命令セットアーキテクチャの情報を取得する。具体的には、命令セットアーキテクチャ解析部１２２は、ＶＰＣおよびディスパッチャを監視するとともにＶＭにおいて実行されたＶＭ実行トレースを解析することにより、ＶＭ命令を収集する

とともに、ＶＭ命令の判定を行い、命令セットアーキテクチャの情報を取得する。

[0043] 命令セットアーキテクチャ解析部１２２は、ＶＭ実行トレース取得部１２２１、ＶＭ命令収集部１２２２およびＶＭ命令判定部１２２３（判定部）を有する。

[0044] ＶＭ実行トレース取得部１２２１は、実行トレース取得部１２１１と同じく、テストスクリプトおよびスクリプトエンジンバイナリを入力として受け付ける。ＶＭ実行トレース取得部１２２１は、スクリプトエンジンバイナリの実行を監視しながら、テストスクリプトを実行することで、ＶＭ上で実行された実行トレースであるＶＭ実行トレースを取得する。

[0045] ＶＭ実行トレースは、実行されたＶＭ命令ごとのＶＰＣとＶＭオペコードで構成される。ＶＰＣの記録は、仮想プログラムカウンタ検出部１２１２で検出されたＶＰＣのメモリを監視することで実現できる。ここでのＶＭオペコードは、ＶＭ命令へのポインタとＶＭ命令とを紐づけた各々に仮想的に割り振られた識別子である。ＶＭ実行トレース取得部１２２１が取得したＶＭ実行トレースは、ＶＭ実行トレースＤＢ１３３に格納される。

[0046] ここで、図５は、ＶＭ実行トレースの一例を示す図である。図５は、ＶＭ実行トレースの一部を切り出したものである。ＶＭ実行トレースのログ行は、たとえば、図５に記載の書式になっており、vpc及びpointerの二つの要素からなる。vpcは、ＶＰＣの値を示す。また、pointerは、ポインタキャッシュから取得された、実行されるＶＭ命令ハンドラの先頭を指すポインタの値を示す。

[0047] ＶＭ命令収集部１２２２は、ＶＰＣおよびディスパッチャを入力として受け付ける。また、ＶＭ命令収集部１２２２は、インターネット上から多様なスクリプトを取得する。そして、ＶＭ命令収集部１２２２は、ＶＰＣおよびディスパッチャを監視しながら、スクリプトを実行して、ＶＭ実行トレースを取得する。また、ＶＭ命令収集部１２２２は、ＶＭ実行トレースからＶＭ命令を取得して、ＶＭ命令のリストに追加する。そして、ＶＭ命令収集部１

２２２は、リストにないＶＭ命令が見られなくなった場合に、ＶＭ命令のリストとして返す。

[0048] ＶＭ命令判定部（判定部）１２２３は、記号表と、命令セットアーキテクチャの情報とを用いて、該記号表が保持する変数に対応するＶＭの命令を判定する。具体的には、ＶＭ命令判定部１２２３は、ＶＭ命令のリスト、ＶＭ命令境界、および記号表を入力として受け付ける。また、ＶＭ命令判定部１２２３は、実行トレースＤＢ１３１から実行トレースおよびＶＭ実行トレースを取り出す。

[0049] そして、ＶＭ命令判定部１２２３は、ＶＭ命令のリストおよびＶＭ命令境界、実行トレース、ＶＭ実行トレースから、実行されたＶＭ命令と、関連する実行トレースの部分とを対応付ける。また、ＶＭ命令判定部１２２３は、メモリアクセストレースの読み込みから、記号表の保持する値のメモリ領域を読み込んでいるＶＭ命令を探索し、記号表が保持している変数の値を読み込むＶＭ命令と判定する。また、ＶＭ命令判定部１２２３は、メモリアクセストレースの読み込みから、記号表の保持する値のメモリ領域に書き込んでいるＶＭ命令を探索し、記号表が保持している変数の値を更新するＶＭ命令と判定する。

[0050] 解析機能付与部（付与部）１２３は、記号表と判定されたＶＭの命令とを用いて、スクリプトエンジンに、スクリプトエンジンの変数の情報を出力する機能を付与する。具体的には、解析機能付与部１２３が有するフック挿入部１２３１が、記号表と、記号表の読み書きをするＶＭ命令と、スクリプトエンジンバイナリとを入力として受け付けて、スクリプトエンジンバイナリに記号表の情報を出力する機能をフックで追加する。また、フック挿入部１２３１は、スクリプトエンジンバイナリに記号表を更新するＶＭ命令の実行ごとに、更新された情報を出力する機能をフックで追加する。

[0051] そして、フック挿入部１２３１は、変数の情報を取得する機能を付与されたスクリプトエンジンバイナリを出力する。

[0052] ［解析機能付与処理の処理手順］

次に、本実施形態に係る解析機能付与装置１０による解析機能付与処理の処理手順について説明する。図６は、解析機能付与処理の処理手順を示すフローチャートである。

[0053] まず、入力部１１が、テストスクリプトおよびスクリプトエンジンバイナリを入力として受け取る（ステップＳ１）。

[0054] そして、実行トレース取得部１２１１は、スクリプトエンジンのバイナリを監視しながらテストスクリプトを実行してブランチトレースとメモリアクセストレースを取得する実行トレース取得処理を行う（ステップＳ２）。

[0055] 仮想プログラムカウンタ検出部１２１２は、実行トレースＤＢ１３１に格納された第一のテストスクリプトに対する実行トレースを取り出して解析し、ＶＰＣを発見する仮想プログラムカウンタ検出処理を行う（ステップＳ３）。そして、ＶＭ命令境界検出部１２１３は、ＶＭ命令を検出し、ＶＭ命令の境界を検出するＶＭ命令境界検出処理を行う（ステップＳ４）。ディスパッチャ検出部１２１４は、スクリプトエンジンバイナリから各ＶＭ命令部分を切り出し、各ＶＭ命令間で類似度が高い部分をディスパッチャとして検出するディスパッチャ検出処理を行う（ステップＳ５）。条件分岐フラグ検出部１２１５は、実行トレースＤＢ１３１に格納された第二のテストスクリプトに対する実行トレースを取り出して解析し、条件分岐フラグを発見する条件分岐フラグ検出処理を行う（ステップＳ６）。

[0056] コードキャッシュ検出部１２１６は、実行トレースおよびＶＭ実行トレースを入力として受け付け、コードキャッシュの更新箇所を検出するコードキャッシュ検出処理を行う（ステップＳ７）。また、変数検出部１２１７は、テストスクリプトを入力として受け付け、変数名と値との組を検出する変数検出処理を行う（ステップＳ８）。また、記号表検出部１２１８は、スクリプトエンジンバイナリを入力として受け付け、変数名および値への参照を持つ配列を記号表として検出する記号表検出処理を行う（ステップＳ９）。また、記号表解析部１２１９は、記号表、テストスクリプトおよびスクリプトエンジンバイナリを入力として受け付け、記号表の構造を解析する記号表解

析処理を行う（ステップＳ１０）。

[0057] ＶＭ実行トレース取得部１２２１は、テストスクリプトおよびスクリプトエンジンバイナリを入力として受け付け、スクリプトエンジンバイナリの実行を監視しながら、テストスクリプトを実行することで、ＶＭ実行トレースを取得するＶＭ実行トレース取得処理を行う（ステップＳ１１）。ＶＭ命令収集部１２２２は、ＶＰＣおよびディスパッチャを受け付けて監視しながら、ＶＭ命令のリストを収集するＶＭ命令収集処理を行う（ステップＳ１２）。ＶＭ命令判定部１２２３は、ＶＭ命令のリスト、ＶＭ命令境界および記号表を入力として受け付け、記号表に対応するＶＭ命令を判定するＶＭ命令判定処理を行う（ステップＳ１３）。

[0058] フック挿入部１２３１は、記号表を入力として受付、変数の情報を取得する機能をスクリプトエンジンバイナリに追加するフック挿入処理を行う（ステップＳ１４）。また、フック挿入部１２３１は、変数情報取得機能を付与したスクリプトエンジンを出力部１４に出力する（ステップＳ１５）。これにより、一連の解析機能付与処理が終了する。

[0059] ［実行トレース取得処理の処理手順］

次に、図７は、図６に示す実行トレース取得処理の処理手順を示すフローチャートである。

[0060] まず、実行トレース取得部１２１１は、テストスクリプトおよびスクリプトエンジンバイナリを入力として受け取る（ステップＳ２１）。そして、実行トレース取得部１２１１は、受け取ったスクリプトエンジンに対して、ブランチトレースを取得するためのフックを施す（ステップＳ２２）。また、実行トレース取得部１２１１は、受け取ったスクリプトエンジンに対して、メモリアクセストレースを取得するためのフックも施す（ステップＳ２３）。

[0061] そして、実行トレース取得部１２１１は、その状態で受け取ったテストスクリプトをスクリプトエンジンに入力して実行させ（ステップＳ２４）、それによって取得される実行トレースを実行トレースＤＢ１３１に格納する（

ステップＳ２５）。

[0062] 実行トレース取得部１２１１は、入力されたテストスクリプトを全て実行し終えているか否かを判定する（ステップＳ２６）。実行トレース取得部１２１１は、入力されたテストスクリプトを全て実行し終えている場合（ステップＳ２６：Ｙｅｓ）、処理を終了する。これに対し、実行トレース取得部１２１１は、入力されたテストスクリプトを全て実行していない場合（ステップＳ２６：Ｎｏ）、ステップＳ２４のテストスクリプトの実行に戻って処理を続ける。

[0063] ［仮想プログラムカウンタ検出処理の処理手順］

次に、図８は、図６に示す仮想プログラムカウンタ検出処理の処理手順を示すフローチャートである。

[0064] まず、仮想プログラムカウンタ検出部１２１２は、実行トレースＤＢ１３１から第一のテストスクリプトによる実行トレースを一つ取り出す（ステップＳ３１）。続いて、仮想プログラムカウンタ検出部１２１２は、実行トレースのうちのメモリアクセストレースに着目し、メモリ読み込み先ごとに読み込み回数を数え上げる（ステップＳ３２）。

[0065] 仮想プログラムカウンタ検出部１２１２は、実行トレースの取得に用いた第一のテストスクリプトを入力として受け取り（ステップＳ３３）、その第一のテストスクリプトを解析して繰り返しの回数と繰り返される文の数とを取得する（ステップＳ３４）。

[0066] 続いて、仮想プログラムカウンタ検出部１２１２は、実行トレースＤＢ１３１から、繰り返し回数や繰り返される文の数の異なる第一のテストスクリプトによる実行トレースを、さらに一つ取り出す（ステップＳ３５）。そして、仮想プログラムカウンタ検出部１２１２は、メモリアクセストレースに着目し、メモリ読み込み先ごとに読み込み回数を数え上げる（ステップＳ３６）。また、仮想プログラムカウンタ検出部１２１２は、実行トレースの取得に用いた第一のテストスクリプトを入力として受け取り（ステップＳ３７）、テストスクリプトを解析して、繰り返しの回数と繰り返される文の数と

を取得する（ステップＳ３８）。

[0067] ここで、仮想プログラムカウンタ検出部１２１２は、繰り返し回数や繰り返される文の増減に比例して読み込み回数が変化するメモリ読み込み先のみに絞り込む（ステップＳ３９）。さらに、仮想プログラムカウンタ検出部１２１２は、ステップＳ３９において絞り込んだメモリ読み込み先を、読み込んだメモリの値が常にＶＭ命令の開始点を指しているものに絞り込む（ステップＳ４０）。

[0068] そして、仮想プログラムカウンタ検出部１２１２は、メモリ読み込み先を一つのみに絞り込めたか否かを判定する（ステップＳ４１）。仮想プログラムカウンタ検出部１２１２は、メモリ読み込み先を一つのみに絞り込めていない場合（ステップＳ４１：Ｎｏ）、ステップＳ３５に戻り、次の実行トレースを一つ取り出して処理を継続する。一方、仮想プログラムカウンタ検出部１２１２は、メモリ読み込み先を一つのみに絞り込めた場合（ステップＳ４１：Ｙｅｓ）、絞り込まれたメモリ読み込み先を仮想プログラムカウンタとしてアーキテクチャ情報ＤＢ１３２に格納して（ステップＳ４２）、処理を終了する。

[0069] ［ＶＭ命令境界検出処理の処理手順］

次に、図９は、図６に示すＶＭ命令境界検出処理の処理手順を示すフローチャートである。

[0070] まず、ＶＭ命令境界検出部１２１３は、実行トレースＤＢ１３１から実行トレースを取り出す（ステップＳ５１）。ＶＭ命令境界検出部１２１３は、実行トレースを所定の方法でクラスタリングする（ステップＳ５２）。クラスタリングは、いずれの手法を用いてもよい。

[0071] ＶＭ命令境界検出部１２１３は、実行回数が閾値以上のクラスタをＶＭ命令として検出する（ステップＳ５３）。そして、ＶＭ命令境界検出部１２１３は、ＶＭ命令を構成する連続した命令列の開始点と終了点とを境界とする（ステップＳ５４）。ＶＭ命令境界検出部１２１３は、ＶＭ命令の境界を返り値として出力して（ステップＳ５５）、ＶＭ命令境界検出処理を終了する

。

[0072] ［ディスパッチャ検出処理の処理手順］

次に、図１０は、図６に示すディスパッチャ検出処理の処理手順を示すフローチャートである。

[0073] まず、ディスパッチャ検出部１２１４は、スクリプトエンジンバイナリを入力として受け取る（ステップＳ６１）。ディスパッチャ検出部１２１４は、ＶＭ命令境界検出部１２１３から、ＶＭ命令の境界を受け取る（ステップＳ６２）。

[0074] ディスパッチャ検出部１２１４は、ＶＭ命令境界検出部１２１３から受け取ったＶＭ命令の境界を基に、スクリプトエンジンバイナリから各ＶＭ命令部分を切り出す（ステップＳ６３）。ディスパッチャ検出部１２１４は、各ＶＭ命令間でコード間の類似度を所定の方法で算出する（ステップＳ６４）。類似度の算出手法は、コード間の類似度を算出できる手法であれば、どの手法でもよい。

[0075] ディスパッチャ検出部１２１４は、ステップＳ６４において算出した類似度を基に、全ＶＭ命令間で類似度が高い部分を取り出す（ステップＳ６５）。そして、ディスパッチャ検出部１２１４は、ＶＭ命令の終端部分であるかを判定する（ステップＳ６６）。

[0076] ＶＭ命令の終端部分でない場合（ステップＳ６６：Ｎｏ）、ディスパッチャ検出部１２１４は、ステップＳ６５に戻り処理を続ける。また、ＶＭ命令の終端部分である場合（ステップＳ６６：Ｙｅｓ）、ディスパッチャ検出部１２１４は、取り出した部分をディスパッチャとして出力して（ステップＳ６７）、処理を終了する。

[0077] ［条件分岐フラグ検出処理の処理手順］

次に、図１１は、図６に示す条件分岐フラグ検出処理の処理手順を示すフローチャートである。

[0078] まず、条件分岐フラグ検出部１２１５は、実行トレースＤＢ１３１から第二のテストスクリプトによる実行トレースを一つ取り出す（ステップＳ７１

）。そして、条件分岐フラグ検出部１２１５は、メモリアクセストレースに着目し、メモリ読み込み先ごとに読み込み回数を数え上げる（ステップＳ７２）。

[0079] また、条件分岐フラグ検出部１２１５は、実行トレースの取得に用いた第二のテストスクリプトを、入力として受け取り（ステップＳ７３）、この第二のテストスクリプトを解析して、条件分岐の回数とTrue／Falseの順序パターンを取得する（ステップＳ７４）。そして、条件分岐フラグ検出部１２１５は、条件分岐の回数に比例して読み込み回数が変化するメモリ読み込み先のみに絞り込む（ステップＳ７５）。さらに、条件分岐フラグ検出部１２１５は、読み込んだメモリの値がTrue／Falseの順序パターンに合わせて二つの値を行き来しているメモリ読み込み先のみに絞り込む（ステップＳ７６）。

[0080] 条件分岐フラグ検出部１２１５は、メモリ読み込み先を一つのみに絞り込めたか否かを判定する（ステップＳ７７）。条件分岐フラグ検出部１２１５は、メモリ読み込み先を一つのみに絞り込めていない場合（ステップＳ７７：Ｎｏ）、ステップＳ７１に戻り、次の実行トレースを一つ取り出して処理を継続する。一方、条件分岐フラグ検出部１２１５は、メモリ読み込み先を一つのみに絞り込めた場合（ステップＳ７７：Ｙｅｓ）、絞り込まれた読み込み先を仮想プログラムカウンタとしてアーキテクチャ情報ＤＢ１３２に格納し（ステップＳ７８）、処理を終了する。

[0081] ［コードキャッシュ検出処理の処理手順］

次に、図１２は、図６に示す条件分岐フラグ検出処理の処理手順を示すフローチャートである。

[0082] コードキャッシュ検出部１２１６は、実行トレースおよびＶＭ実行トレースを入力として受け付けて（ステップＳ８１）、ＶＰＣが指し示すメモリ領域をＶＭ実行トレースから取得する（ステップＳ８２）。また、コードキャッシュ検出部１２１６は、当該メモリ領域を確保したメモリ割り当て関数の呼び出し元のコード箇所を実行トレースから取得する（ステップＳ８３）。また、コードキャッシュ検出部１２１６は、当該コード箇所で確保された全

ての領域をコードキャッシュとして検出する（ステップＳ８４）。そして、コードキャッシュ検出部１２１６は、コードキャッシュに書き込みをしているコード箇所を実行トレースから取得する（ステップＳ８５）。また、コードキャッシュ検出部１２１６は、当該コード箇所で書き込みされた全ての領域をコードキャッシュの更新として検出し（ステップＳ８６）、コードキャッシュおよび更新箇所を返して（ステップＳ８７）、処理を終了する。

[0083] ［変数検出処理の処理手順］

次に、図１３は、図６に示す条件分岐フラグ検出処理の処理手順を示すフローチャートである。

[0084] 変数検出部１２１７は、テストスクリプトを入力として受け付けて（ステップＳ９１）、実行トレースＤＢ１３１からテストスクリプトに対応した実行トレースを取り出す（ステップＳ９２）。また、変数検出部１２１７は、テストスクリプト中に記述された変数名と値との組を取り出す（ステップＳ９３）。そして、変数検出部１２１７は、メモリアクセストレースの書き込みから当該変数名との一致性が所定の閾値より高い値を探索する（ステップＳ９４）。また、変数検出部１２１７は、メモリアクセストレースの書き込みから、当該値との一致性が所定の閾値より高い値を探索する（ステップＳ９５）。そして変数検出部１２１７は、当該変数に対応する変数名と値の格納先として返し（ステップＳ９６）、処理を終了する。

[0085] ［記号表検出処理の処理手順］

次に、図１４は、図６に示す条件分岐フラグ検出処理の処理手順を示すフローチャートである。

[0086] 記号表検出部（抽出部）１２１８は、得られたアーキテクチャに関する情報を基に、記号表を検出する。具体的には、記号表検出部１２１８は、スクリプトエンジンバイナリを入力として受け付けて（ステップＳ１０１）、変数と対応する値の格納先を受け付ける（ステップＳ１０２）。また、記号表検出部１２１８は、実行トレースＤＢ１３１から実行トレースを取り出して（ステップＳ１０３）、スクリプトエンジンバイナリの静的解析により値の

格納先の周辺に関するメモリ参照の関係を収集する（ステップＳ１０４）。また、記号表検出部１２１８は、メモリアクセストレースから変数名および値への実行時のメモリ参照を収集し（ステップＳ１０５）、変数名および値への参照を持つ配列を記号表として検出する（ステップＳ１０６）。そして、記号表検出部１２１８は、ＡＰＩトレースから、記号表のメモリ領域を確保しているコード箇所を特定し、実行時に記号表の位置を特定可能にして（ステップＳ１０７）、記号表を返し（ステップＳ１０８）、処理を終了する。

[0087] ［記号表解析処理の処理手順］

次に、図１５は、図６に示す条件分岐フラグ検出処理の処理手順を示すフローチャートである。

[0088] また、記号表解析部（解析部）１２１９は、検出された記号表の構造を解析する。具体的には、記号表解析部１２１９は、検出された記号表、テストスクリプト、およびスクリプトエンジンバイナリを入力として受け付ける（ステップＳ１１１～Ｓ１１３）。また、記号表解析部１２１９は、実行トレースＤＢ１３１から実行トレースを取り出して、スクリプトエンジンバイナリの静的解析を実施して、変数の依存関係を収集する（ステップＳ１１４～Ｓ１１５）。記号表解析部１２１９は、例えば非特許文献１に開示されているような所定の方法で、記号表を構成する構造体の型を推論する（ステップＳ１１６）。そして、記号表解析部１２１９は、メモリアクセストレースを基に、記号表を構成する構造体の保持する値とテストスクリプトで確保した変数の型の共起から、型情報を保持する変数を特定し、記号表の構造を返して（ステップＳ１１７～Ｓ１１８）、処理を終了する。

[0089] ［ＶＭ実行トレース取得処理の処理手順］

次に、図１６は、図６に示すＶＭ実行トレース取得処理の処理手順を示すフローチャートである。

[0090] まず、ＶＭ実行トレース取得部１２２１は、テストスクリプトおよびスクリプトエンジンバイナリを入力として受け取る（ステップＳ１２１）。そし

て、ＶＭ実行トレース取得部１２２１は、受け取ったスクリプトエンジンに対して、ＶＰＣおよびＶＭオペコードを記録するためのフックを施す（ステップＳ１２２）。

[0091] ＶＭ実行トレース取得部１２２１は、その状態で受け取ったテストスクリプトをスクリプトエンジンに入力して実行させ（ステップＳ１２３）、それによって取得されるＶＭ実行トレースをＶＭ実行トレースＤＢ１３３に格納する（ステップＳ１２４）。

[0092] ＶＭ実行トレース取得部１２２１は、入力されたテストスクリプトを全て実行したか否かを判定する（ステップＳ１２５）。ＶＭ実行トレース取得部１２２１は、入力されたテストスクリプトを全て実行し終えていない場合（ステップＳ１２５：Ｎｏ）、ステップＳ１２３のテストスクリプトの実行に戻って処理を続ける。一方、ＶＭ実行トレース取得部１２２１は、入力されたテストスクリプトを全て実行し終えている場合（ステップＳ１２５：Ｙｅｓ）、処理を終了する。

[0093] ［ＶＭ命令収集処理の処理手順］

次に、図１７は、図６に示す条件分岐フラグ検出処理の処理手順を示すフローチャートである。

[0094] ＶＭ命令収集部１２２２は、ＶＰＣおよびディスパッチャを入力として受け付ける（ステップＳ１３１）。また、ＶＭ命令収集部１２２２は、インターネット上から多様なスクリプトを取得する（ステップＳ１３２）。そして、ＶＭ命令収集部１２２２は、ＶＰＣおよびディスパッチャを監視しながら、スクリプトを実行して、ＶＭ実行トレースを取得する（ステップＳ１３３）。また、ＶＭ命令収集部１２２２は、ＶＭ実行トレースからＶＭ命令を取得して、ＶＭ命令のリストに追加する（ステップＳ１３４～Ｓ１３５）。

[0095] そして、ＶＭ命令収集部１２２２は、リストにないＶＭ命令の有無を確認する（ステップＳ１３６）。ＶＭ命令収集部１２２２は、リストにないＶＭ命令が見られる場合に（ステップＳ１３６：Ｎｏ）、ステップＳ１３２に処理を戻す。一方、ＶＭ命令収集部１２２２は、リストにないＶＭ命令が見ら

れなくなった場合に（ステップＳ１３６：Ｙｅｓ）、ＶＭ命令のリストとして返し（ステップＳ１３７）、処理を終了する。

[0096] ［ＶＭ命令判定処理の処理手順］

次に、図１８は、図６に示す条件分岐フラグ検出処理の処理手順を示すフローチャートである。

[0097] ＶＭ命令判定部（判定部）１２２３は、記号表と、命令セットアーキテクチャの情報とを用いて、該記号表が保持する変数に対応するＶＭの命令を判定する。具体的には、ＶＭ命令判定部１２２３は、ＶＭ命令のリスト、ＶＭ命令境界、および記号表を入力として受け付ける（ステップＳ１４１～Ｓ１４２）。また、ＶＭ命令判定部１２２３は、実行トレースＤＢ１３１から実行トレースおよびＶＭ実行トレースを取り出す（ステップＳ１４３）。

[0098] そして、ＶＭ命令判定部１２２３は、ＶＭ命令のリストおよびＶＭ命令境界、実行トレース、ＶＭ実行トレースから、実行されたＶＭ命令と、関連する実行トレースの部分とを対応付ける（ステップＳ１４４）。また、ＶＭ命令判定部１２２３は、メモリアクセストレースの読み込みから、記号表の保持する値のメモリ領域を読み込んでいるＶＭ命令を探索し、記号表が保持している変数の値を読み込むＶＭ命令と判定する（ステップＳ１４５～Ｓ１４６）。また、ＶＭ命令判定部１２２３は、メモリアクセストレースの読み込みから、記号表の保持する値のメモリ領域に書き込んでいるＶＭ命令を探索し、記号表が保持している変数の値を更新するＶＭ命令と判定し（ステップＳ１４７～Ｓ１４８）、処理を終了する。

[0099] ［フック挿入処理の処理手順］

次に、図１９は、図６に示すフック挿入処理の処理手順を示すフローチャートである。

[0100] フック挿入部１２３１は、記号表と、記号表の読み書きをするＶＭ命令と、スクリプトエンジンバイナリとを入力として受け付けて（ステップＳ１５１～Ｓ１５３）、スクリプトエンジンバイナリに記号表の情報を出力する機能をフックで追加する（ステップＳ１５４）。また、フック挿入部１２３１

は、スクリプトエンジンバイナリに記号表を更新するＶＭ命令の実行ごとに、更新された情報を出力する機能をフックで追加する（ステップＳ１５５）。そして、フック挿入部１２３１は、変数の情報を取得する機能を付与されたスクリプトエンジンバイナリを出力し（ステップＳ１５６）、処理を終了する。

[0101] ［効果］

以上、説明したように、本実施形態の解析機能付与装置１０において、仮想機械解析部（第１の取得部）１２１が、スクリプトエンジンのＶＭを解析し、スクリプトエンジンのアーキテクチャに関する情報を取得する。記号表検出部（検出部）１２１８は、取得されたアーキテクチャに関する情報を基に、変数に関する情報を保持する記号表を検出する。記号表解析部（解析部）１２１９は、該記号表の構造を解析する。命令セットアーキテクチャ解析部（第２の取得部）１２２が、取得されたアーキテクチャに関する情報を基に、仮想機械の命令の体系である命令セットアーキテクチャの情報を取得する。ＶＭ命令判定部（判定部）１２２３が、記号表と、命令セットアーキテクチャの情報とを用いて、該記号表が保持する変数に対応する仮想機械の命令を判定する。

[0102] 具体的には、アーキテクチャ情報は、仮想プログラムカウンタ、ディスパッチャ、条件分岐フラグ、またはコードキャッシュのいずれかを含む。また、命令セットアーキテクチャ解析部１２２は、仮想プログラムカウンタおよびディスパッチャを監視するとともに仮想機械において実行された仮想機械実行トレースを解析することにより、命令セットアーキテクチャの情報を取得する。

[0103] これにより、本実施形態の解析機能付与装置１０は、ＶＭの内部仕様が未知のスクリプトエンジンに対しても、実行トレースおよびＶＭ実行トレースの取得に基づく解析により各種アーキテクチャ情報を検出することが可能となる。したがって、スクリプトエンジンの変数の情報を取得することが可能となる。

[0104] また、解析機能付与部１２３が、解析された記号表と判定された仮想機械の命令とを用いて、スクリプトエンジンに、スクリプトエンジンの変数の情報を出力する機能を付与する。これにより、解析機能付与装置１０は、人手でのリバースエンジニアリングを要することなく、変数情報の取得機能の付与を実現できる。

[0105] また、解析機能付与装置１０では、多様なスクリプトエンジンに対して、テストスクリプトさえ用意すれば自動で変数情報の取得機能を付与できるため、個別の設計や実行を要することなく、変数情報の取得機能の付与を実現できる。したがって、様々なスクリプト言語で作成されたスクリプトに対しても、変数情報の取得機能が可能となり、より高度なスクリプト解析を実現できるようになる。

[0106] また、解析機能付与装置１０によれば、スクリプトエンジンを解析し、コードカバレッジを計測する機能を後付けで付与することにより、多種多様なスクリプト言語のスクリプトエンジンに対して、変数の情報を取得する機能の自動的な付与を実現できる。

[0107] このように、本実施形態の解析機能付与装置１０は、多種多様なスクリプト言語で記述されたスクリプトの変数情報の解析に有用であり、デバッガなどの支援機能の不在やVMの内部仕様が未知であるため変数情報の取得が難しいスクリプトに対しても、解析を実現することに適している。したがって、様々なスクリプトエンジンに変数の情報を取得する機能を付与することで、スクリプトの変数を解析して、スクリプトの持つ機能のより深く解釈することが可能となる。

[0108] ［プログラム］

上記実施形態に係る解析機能付与装置１０が実行する処理をコンピュータが実行可能な言語で記述したプログラムを作成することもできる。一実施形態として、解析機能付与装置１０は、パッケージソフトウェアやオンラインソフトウェアとして上記の解析機能付与処理を実行する解析機能付与プログラムを所望のコンピュータにインストールさせることによって実装できる。

例えば、上記の解析機能付与プログラムを情報処理装置に実行させることにより、情報処理装置を解析機能付与装置１０として機能させることができる。ここで言う情報処理装置には、デスクトップ型またはノート型のパーソナルコンピュータが含まれる。また、その他にも、情報処理装置にはスマートフォン、携帯電話機やＰＨＳ（Personal Handyphone System）などの移動体通信端末、さらには、ＰＤＡ（Personal Digital Assistant）などのスレート端末などがその範疇に含まれる。また、解析機能付与装置１０の機能を、クラウドサーバに実装してもよい。

[0109] 図２０は、解析機能付与プログラムを実行するコンピュータの一例を示す図である。コンピュータ１０００は、例えば、メモリ１０１０と、ＣＰＵ１０２０と、ハードディスクドライブインタフェース１０３０と、ディスクドライブインタフェース１０４０と、シリアルポートインタフェース１０５０と、ビデオアダプタ１０６０と、ネットワークインタフェース１０７０とを有する。これらの各部は、バス１０８０によって接続される。

[0110] メモリ１０１０は、ＲＯＭ（Read Only Memory）１０１１およびＲＡＭ１０１２を含む。ＲＯＭ１０１１は、例えば、ＢＩＯＳ（Basic Input Output System）等のブートプログラムを記憶する。ハードディスクドライブインタフェース１０３０は、ハードディスクドライブ１０３１に接続される。ディスクドライブインタフェース１０４０は、ディスクドライブ１０４１に接続される。ディスクドライブ１０４１には、例えば、磁気ディスクや光ディスク等の着脱可能な記憶媒体が挿入される。シリアルポートインタフェース１０５０には、例えば、マウス１０５１およびキーボード１０５２が接続される。ビデオアダプタ１０６０には、例えば、ディスプレイ１０６１が接続される。

[0111] ここで、ハードディスクドライブ１０３１は、例えば、ＯＳ１０９１、アプリケーションプログラム１０９２、プログラムモジュール１０９３およびプログラムデータ１０９４を記憶する。上記実施形態で説明した各情報は、例えばハードディスクドライブ１０３１やメモリ１０１０に記憶される。

[0112] また、解析機能付与プログラムは、例えば、コンピュータ１０００によって実行される指令が記述されたプログラムモジュール１０９３として、ハードディスクドライブ１０３１に記憶される。具体的には、上記実施形態で説明した解析機能付与装置１０が実行する各処理が記述されたプログラムモジュール１０９３が、ハードディスクドライブ１０３１に記憶される。

[0113] また、解析機能付与プログラムによる情報処理に用いられるデータは、プログラムデータ１０９４として、例えば、ハードディスクドライブ１０３１に記憶される。そして、ＣＰＵ１０２０が、ハードディスクドライブ１０３１に記憶されたプログラムモジュール１０９３やプログラムデータ１０９４を必要に応じてＲＡＭ１０１２に読み出して、上述した各手順を実行する。

[0114] なお、解析機能付与プログラムに係るプログラムモジュール１０９３やプログラムデータ１０９４は、ハードディスクドライブ１０３１に記憶される場合に限られず、例えば、着脱可能な記憶媒体に記憶されて、ディスクドライブ１０４１等を介してＣＰＵ１０２０によって読み出されてもよい。あるいは、解析機能付与プログラムに係るプログラムモジュール１０９３やプログラムデータ１０９４は、ＬＡＮやＷＡＮ（Wide Area Network）等のネットワークを介して接続された他のコンピュータに記憶され、ネットワークインタフェース１０７０を介してＣＰＵ１０２０によって読み出されてもよい。

[0115] 以上、本発明者によってなされた発明を適用した実施形態について説明したが、本実施形態による本発明の開示の一部をなす記述および図面により本発明は限定されることはない。すなわち、本実施形態に基づいて当業者等によりなされる他の実施形態、実施例および運用技術等は全て本発明の範疇に含まれる。

符号の説明

[0116] １０ 解析機能付与装置

１１ 入力部

１２ 制御部

１３　記憶部

１４　出力部

１２１　仮想機械解析部

１２２　命令セットアーキテクチャ解析部

１２３　解析機能付与部

１３１　実行トレースＤＢ

１３２　アーキテクチャ情報ＤＢ

１３３　ＶＭ実行トレースＤＢ

１２１１　実行トレース取得部

１２１２　仮想プログラムカウンタ検出部

１２１３　ＶＭ命令境界検出部

１２１４　ディスパッチャ検出部

１２１５　条件分岐フラグ検出部

１２１６　コードキャッシュ検出部

１２１７　変数検出部

１２１８　記号表検出部

１２１９　記号表解析部

１２２１　ＶＭ実行トレース取得部

１２２２　ＶＭ命令収集部

１２２３　ＶＭ命令判定部

１２３１　フック挿入部

請求の範囲

[請求項1] スクリプトエンジンの仮想機械を解析し、スクリプトエンジンのアーキテクチャに関する情報を取得する第１の取得部と、

取得された前記アーキテクチャに関する情報を基に、変数に関する情報を保持する記号表を検出する検出部と、

検出された前記記号表の構造を解析する解析部と、

取得された前記アーキテクチャに関する情報を基に、仮想機械の命令の体系である命令セットアーキテクチャの情報を取得する第２の取得部と、

解析された前記記号表と、前記命令セットアーキテクチャの情報とを用いて、該記号表が保持する変数に対応する前記仮想機械の命令を判定する判定部と、

を有することを特徴とする解析機能付与装置。

[請求項2] 前記記号表と判定された前記仮想機械の命令とを用いて、スクリプトエンジンに、該スクリプトエンジンの変数の情報を出力する機能を付与する付与部をさらに有することを特徴とする請求項１に記載の解析機能付与装置。

[請求項3] 前記アーキテクチャに関する情報は、仮想プログラムカウンタ、ディスパッチャ、条件分岐フラグ、またはコードキャッシュのいずれかを含むことを特徴とする請求項１に記載の解析機能付与装置。

[請求項4] 前記第２の取得部は、前記仮想プログラムカウンタおよび前記ディスパッチャを監視するとともに前記仮想機械において実行された仮想機械実行トレースを解析することにより、前記命令セットアーキテクチャの情報を取得することを特徴とする請求項３に記載の解析機能付与装置。

[請求項5] 解析機能付与装置が実行する解析機能付与方法であって、

スクリプトエンジンの仮想機械を解析し、スクリプトエンジンのアーキテクチャに関する情報を取得する第１の取得工程と、

取得された前記アーキテクチャに関する情報を基に、変数に関する情報を保持する記号表を検出する検出工程と、

検出された前記記号表の構造を解析する解析工程と、

取得された前記アーキテクチャに関する情報を基に、仮想機械の命令の体系である命令セットアーキテクチャの情報を取得する第２の取得工程と、

解析された前記記号表と、前記命令セットアーキテクチャの情報とを用いて、該記号表が保持する変数に対応する前記仮想機械の命令を判定する判定工程と、

を含んだことを特徴とする解析機能付与方法。

[請求項6] スクリプトエンジンの仮想機械を解析し、スクリプトエンジンのアーキテクチャに関する情報を取得する第１の取得ステップと、

取得された前記アーキテクチャに関する情報を基に、変数に関する情報を保持する記号表を検出する検出ステップと、

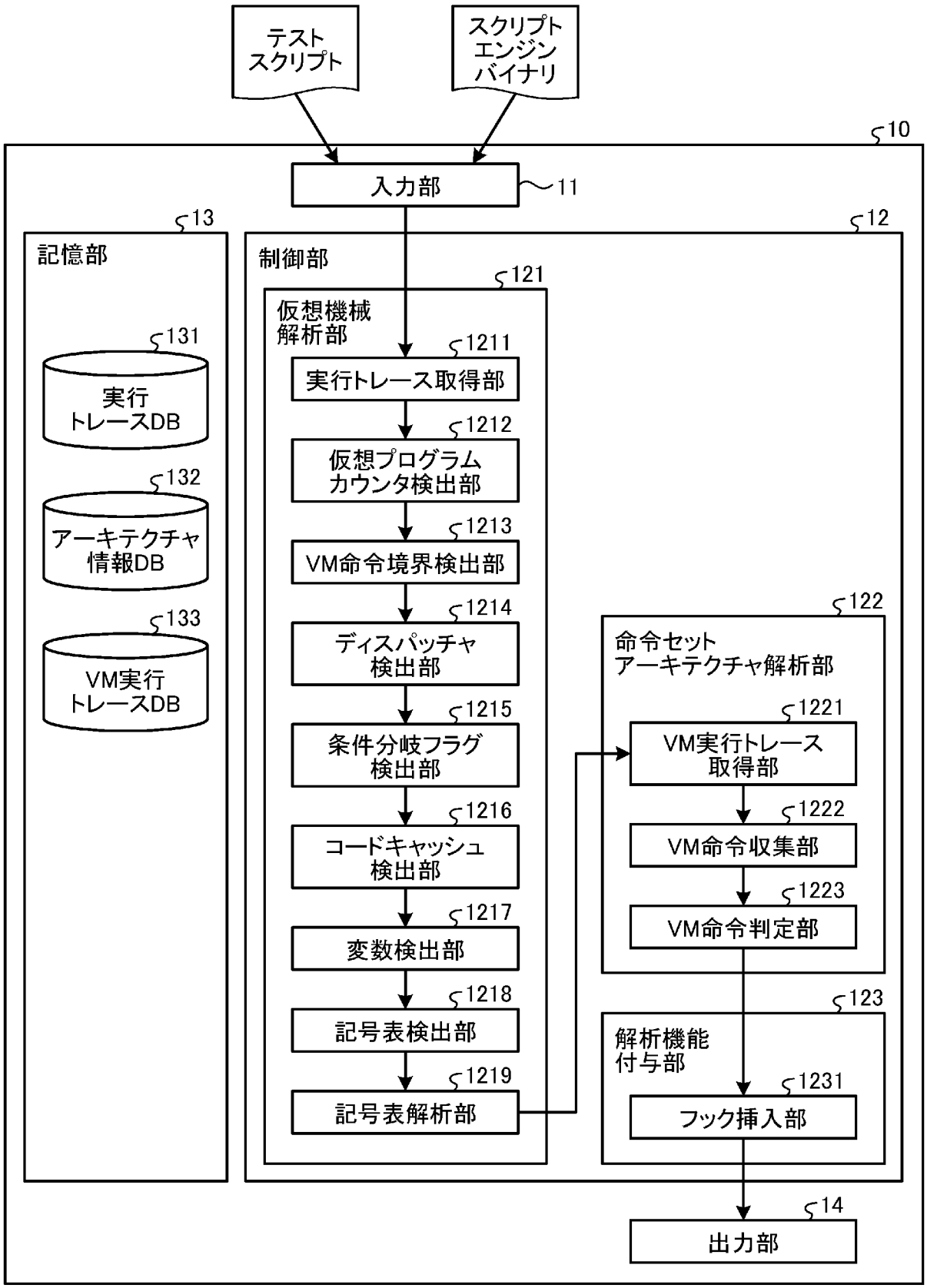
検出された前記記号表の構造を解析する解析ステップと、

取得された前記アーキテクチャに関する情報を基に、仮想機械の命令の体系である命令セットアーキテクチャの情報を取得する第２の取得ステップと、

解析された前記記号表と、前記命令セットアーキテクチャの情報とを用いて、該記号表が保持する変数に対応する前記仮想機械の命令を判定する判定ステップと、

をコンピュータに実行させるための解析機能付与プログラム。

[図1]


テスト
スクリプト
スクリプト
エンジン
バイナリ
10
入力部
11
13
記憶部
12
制御部
121
仮想機械
解析部
131
実行
トレースDB
132
アーキテクチャ
情報DB
133
VM実行
トレースDB
1211
実行トレース取得部
1212
仮想プログラム
カウンタ検出部
1213
VM命令境界検出部
1214
ディスパッチャ
検出部
1215
条件分岐フラグ
検出部
1216
コードキャッシュ
検出部
1217
変数検出部
1218
記号表検出部
1219
記号表解析部
122
命令セット
アーキテクチャ解析部
1221
VM実行トレース
取得部
1222
VM命令収集部
1223
VM命令判定部
123
解析機能
付与部
1231
フック挿入部
14
出力部

[図2]

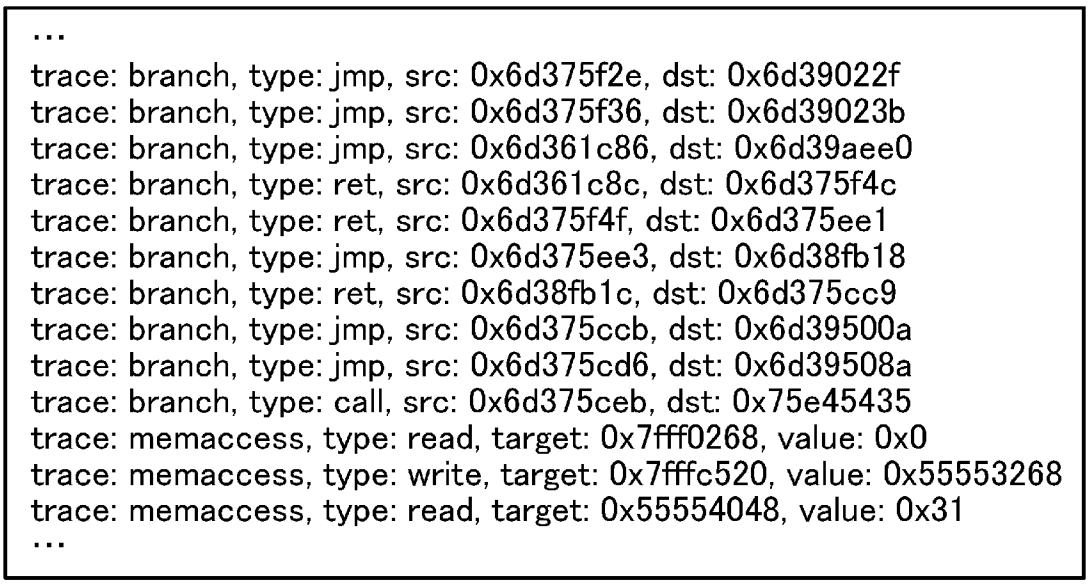
```
a = 0
For N = 0 To 10
    a = a + N
    a = a * N
    a = a - N
Next
```

[図3]

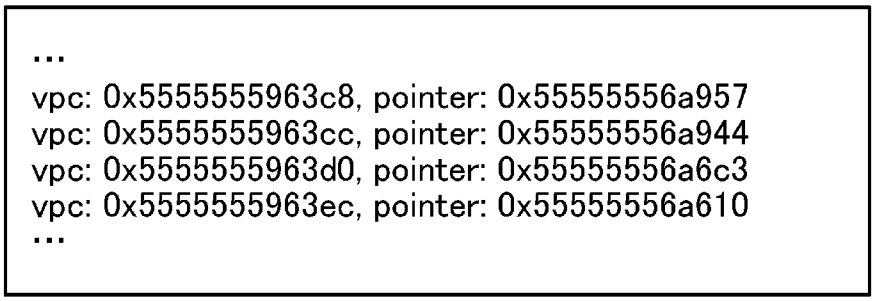
```
bool_array = Array(True, False, True, True, False)

a = 0
For Each bool in bool_array
    If bool Then
        a = a + 1
    End If
Next
```

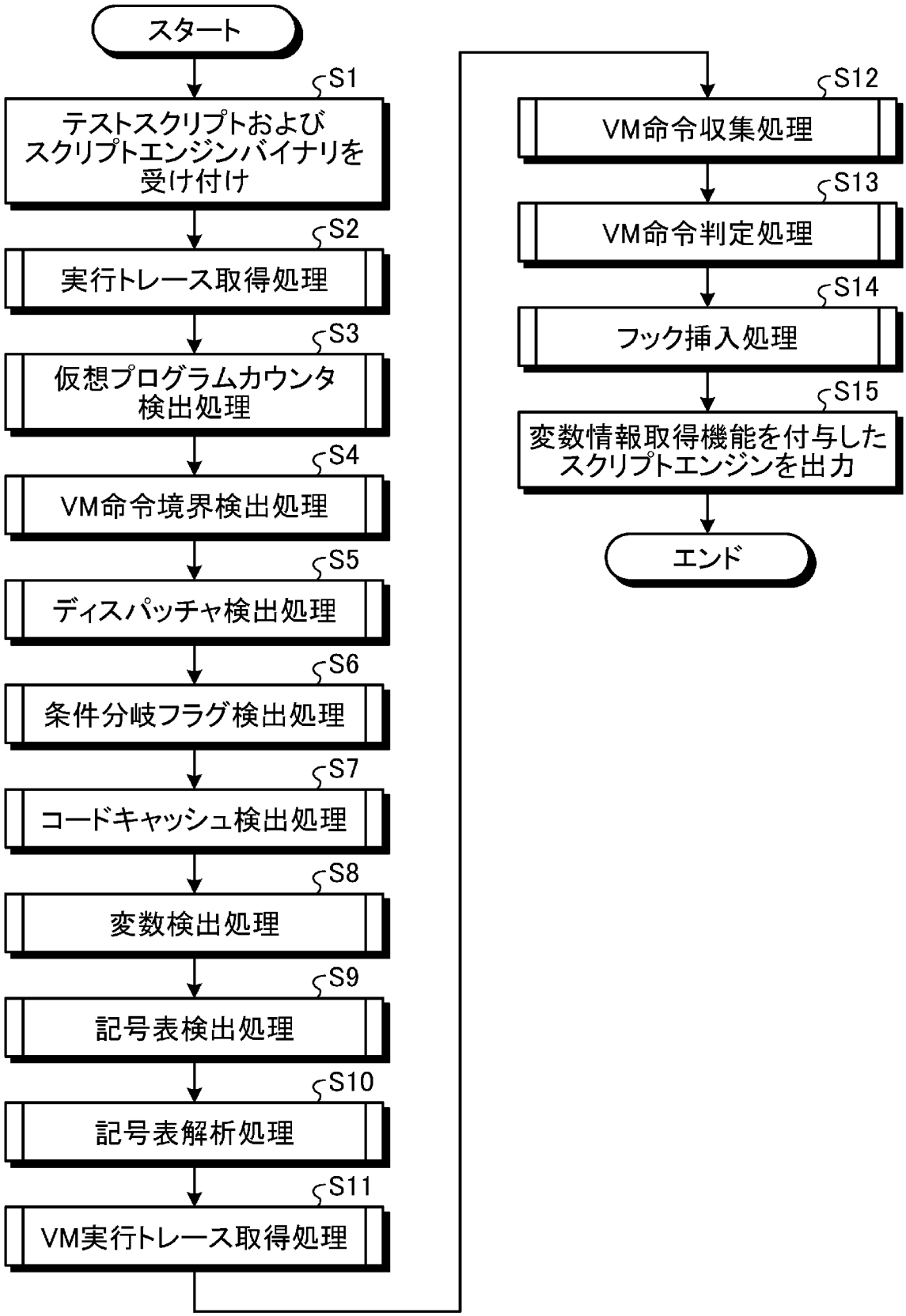
[図4]


…
trace: branch, type: jmp, src: 0x6d375f2e, dst: 0x6d39022f
trace: branch, type: jmp, src: 0x6d375f36, dst: 0x6d39023b
trace: branch, type: jmp, src: 0x6d361c86, dst: 0x6d39aee0
trace: branch, type: ret, src: 0x6d361c8c, dst: 0x6d375f4c
trace: branch, type: ret, src: 0x6d375f4f, dst: 0x6d375ee1
trace: branch, type: jmp, src: 0x6d375ee3, dst: 0x6d38fb18
trace: branch, type: ret, src: 0x6d38fb1c, dst: 0x6d375cc9
trace: branch, type: jmp, src: 0x6d375ccb, dst: 0x6d39500a
trace: branch, type: jmp, src: 0x6d375cd6, dst: 0x6d39508a
trace: branch, type: call, src: 0x6d375ceb, dst: 0x75e45435
trace: memaccess, type: read, target: 0x7fff0268, value: 0x0
trace: memaccess, type: write, target: 0x7fffc520, value: 0x55553268
trace: memaccess, type: read, target: 0x55554048, value: 0x31
…

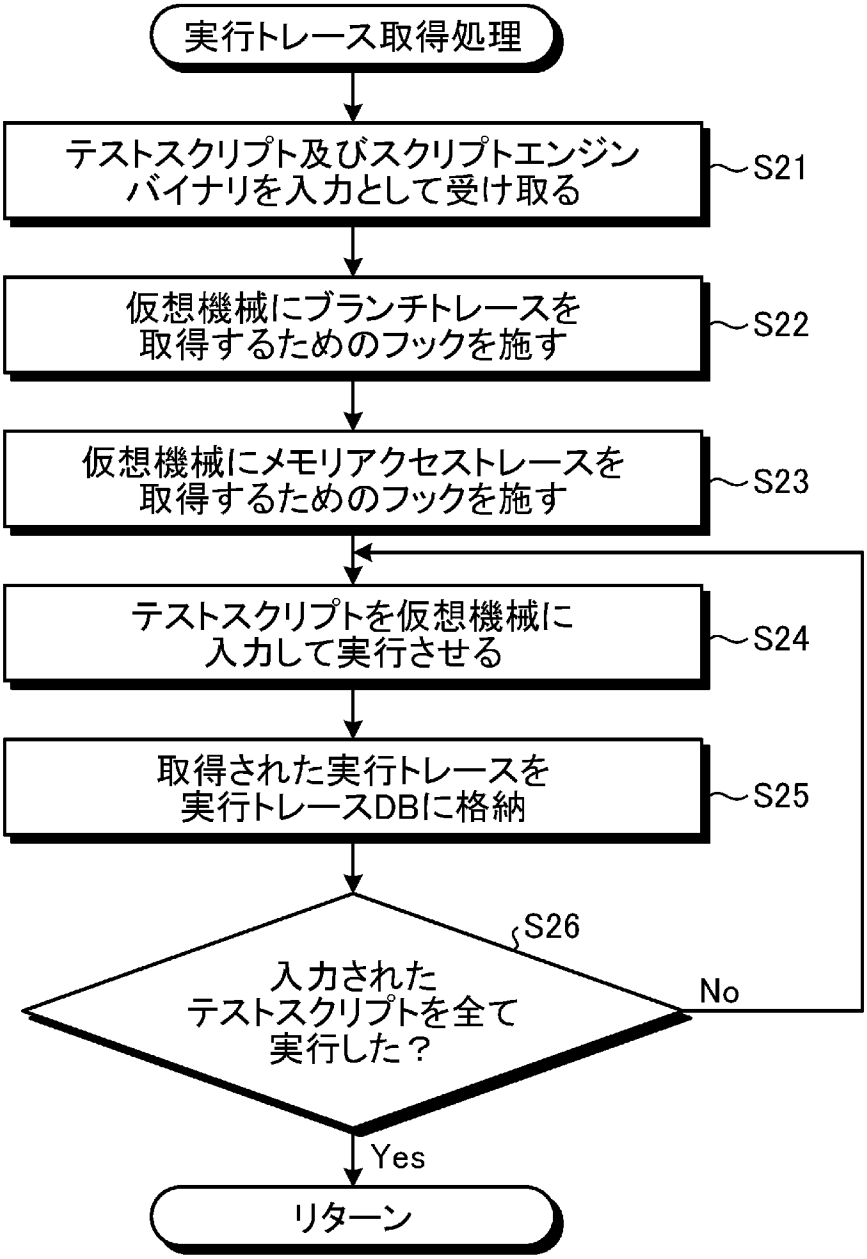
[図5]


…
vpc: 0x5555555963c8, pointer: 0x55555556a957
vpc: 0x5555555963cc, pointer: 0x55555556a944
vpc: 0x5555555963d0, pointer: 0x55555556a6c3
vpc: 0x5555555963ec, pointer: 0x55555556a610
…

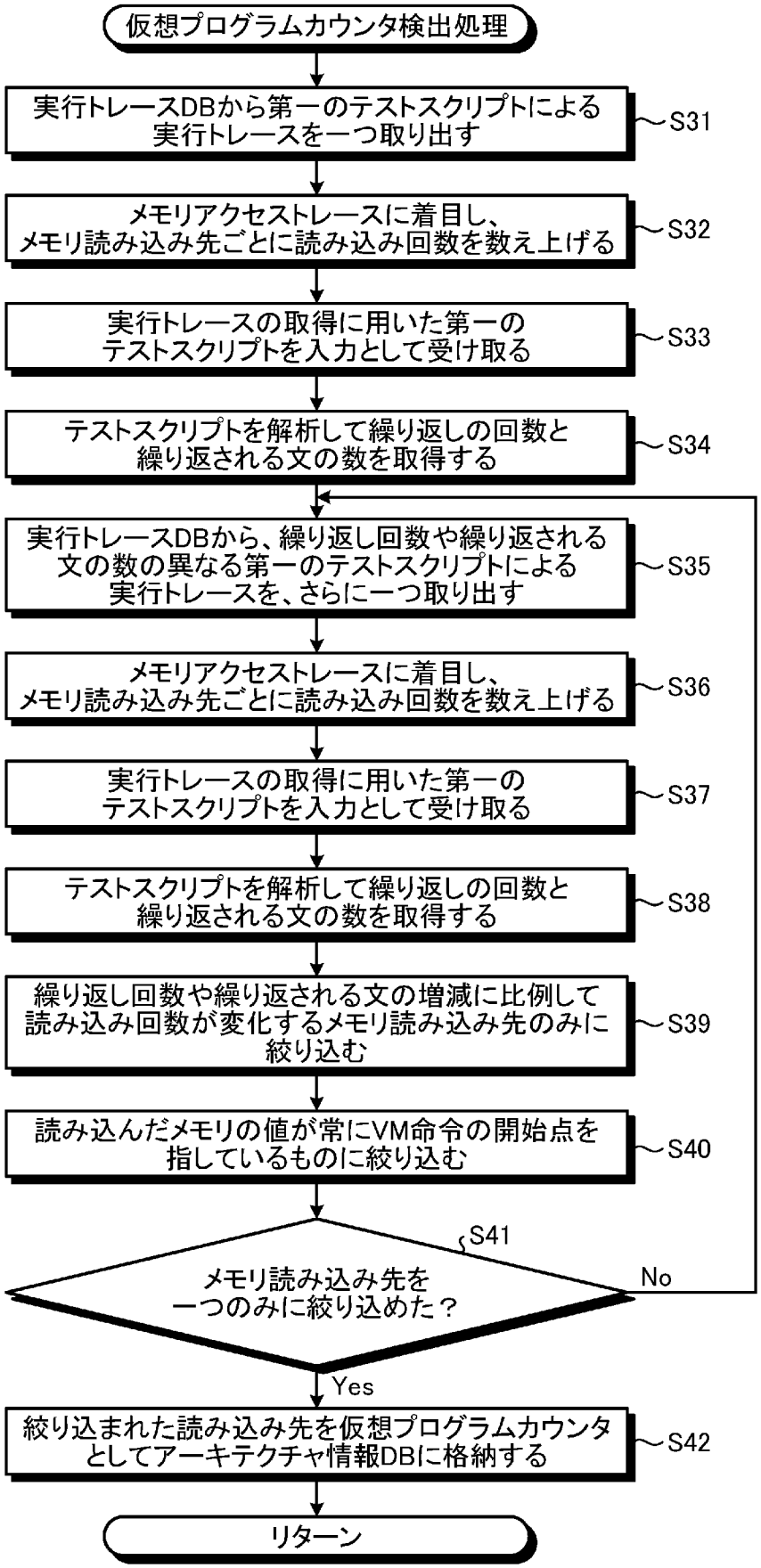
[図6]


スタート
S1
テストスクリプトおよび
スクリプトエンジンバイナリを
受け付け
S2
実行トレース取得処理
S3
仮想プログラムカウンタ
検出処理
S4
VM命令境界検出処理
S5
ディスパッチャ検出処理
S6
条件分岐フラグ検出処理
S7
コードキャッシュ検出処理
S8
変数検出処理
S9
記号表検出処理
S10
記号表解析処理
S11
VM実行トレース取得処理
S12
VM命令収集処理
S13
VM命令判定処理
S14
フック挿入処理
S15
変数情報取得機能を付与した
スクリプトエンジンを出力
エンド

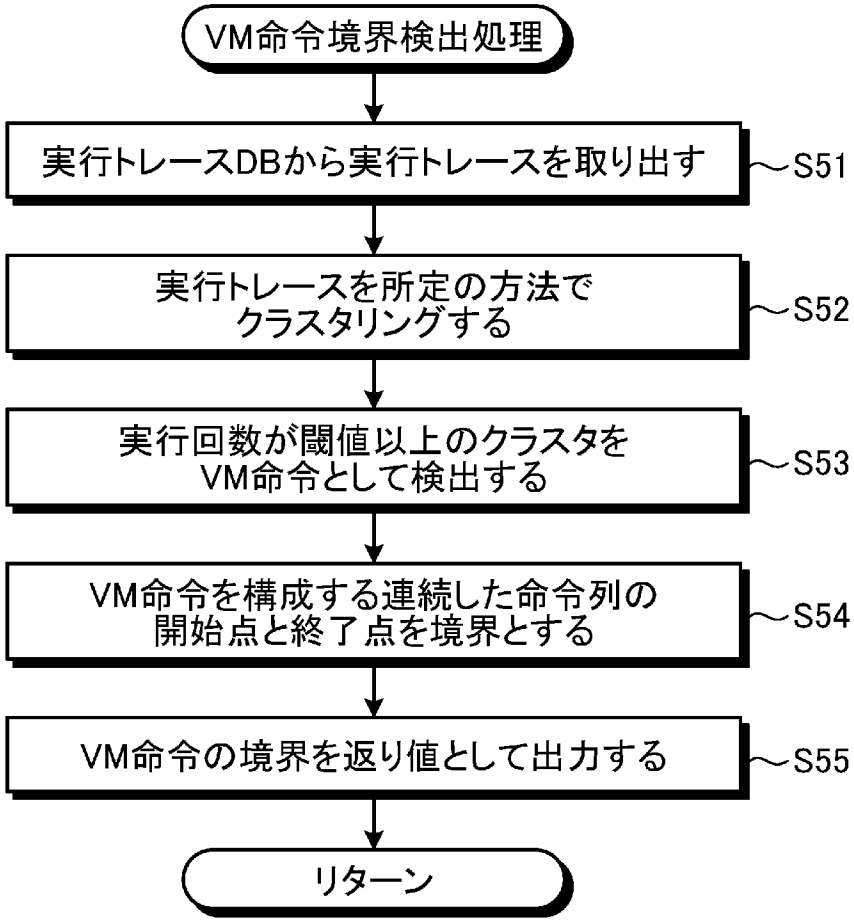
[図7]


実行トレース取得処理
テストスクリプト及びスクリプトエンジン
バイナリを入力として受け取る
S21
仮想機械にブランチトレースを
取得するためのフックを施す
S22
仮想機械にメモリアクセストレースを
取得するためのフックを施す
S23
テストスクリプトを仮想機械に
入力して実行させる
S24
取得された実行トレースを
実行トレースDBに格納
S25
S26
入力された
テストスクリプトを全て
実行した？
No
Yes
リターン

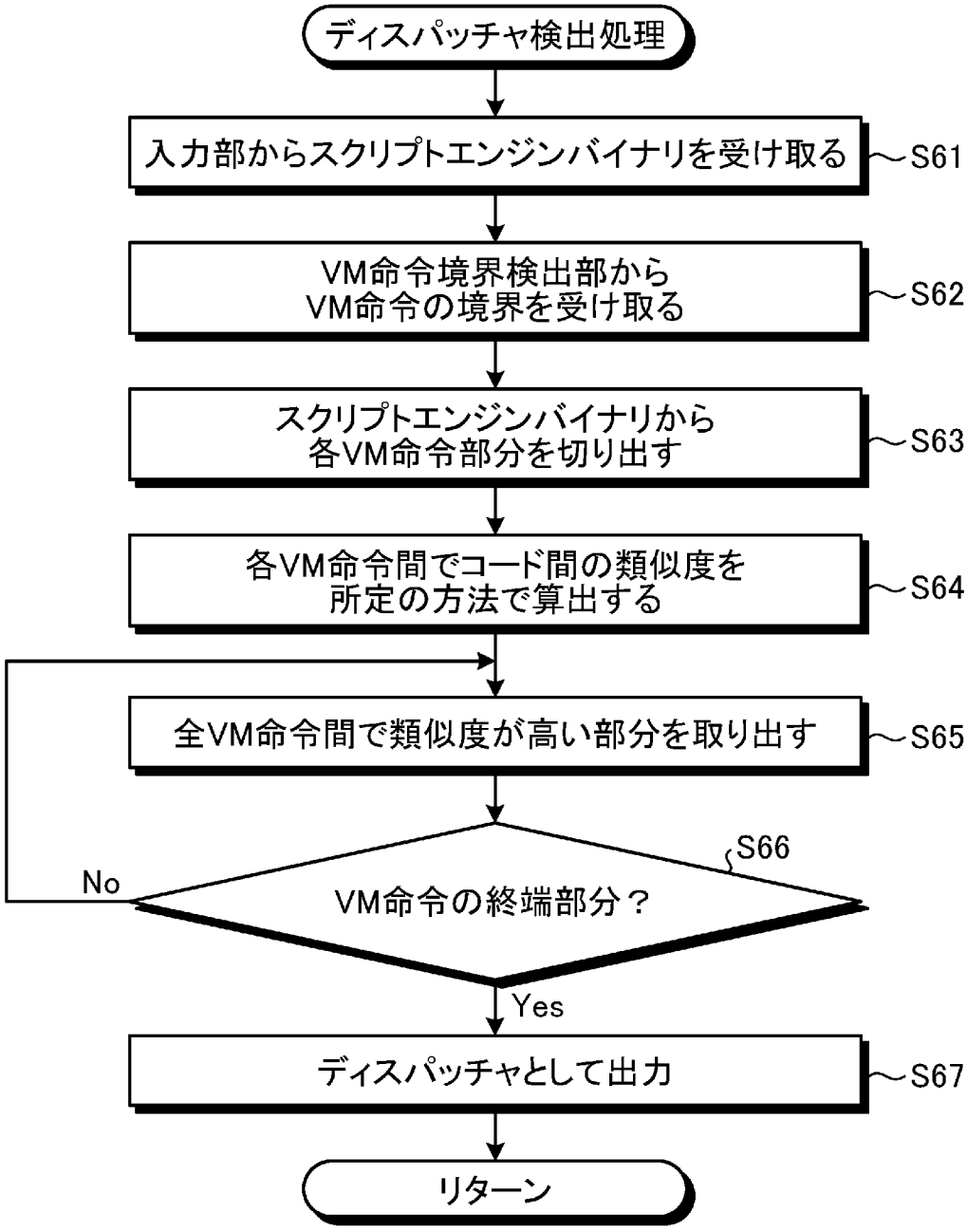
[図8]


仮想プログラムカウンタ検出処理
実行トレースDBから第一のテストスクリプトによる実行トレースを一つ取り出す
S31
メモリアクセストレースに着目し、メモリ読み込み先ごとに読み込み回数を数え上げる
S32
実行トレースの取得に用いた第一のテストスクリプトを入力として受け取る
S33
テストスクリプトを解析して繰り返しの回数と繰り返される文の数を取得する
S34
実行トレースDBから、繰り返し回数や繰り返される文の数の異なる第一のテストスクリプトによる実行トレースを、さらに一つ取り出す
S35
メモリアクセストレースに着目し、メモリ読み込み先ごとに読み込み回数を数え上げる
S36
実行トレースの取得に用いた第一のテストスクリプトを入力として受け取る
S37
テストスクリプトを解析して繰り返しの回数と繰り返される文の数を取得する
S38
繰り返し回数や繰り返される文の増減に比例して読み込み回数が変化するメモリ読み込み先のみに絞り込む
S39
読み込んだメモリの値が常にVM命令の開始点を指しているものに絞り込む
S40
S41
メモリ読み込み先を一つのみに絞り込めた？
No
Yes
絞り込まれた読み込み先を仮想プログラムカウンタとしてアーキテクチャ情報DBに格納する
S42
リターン

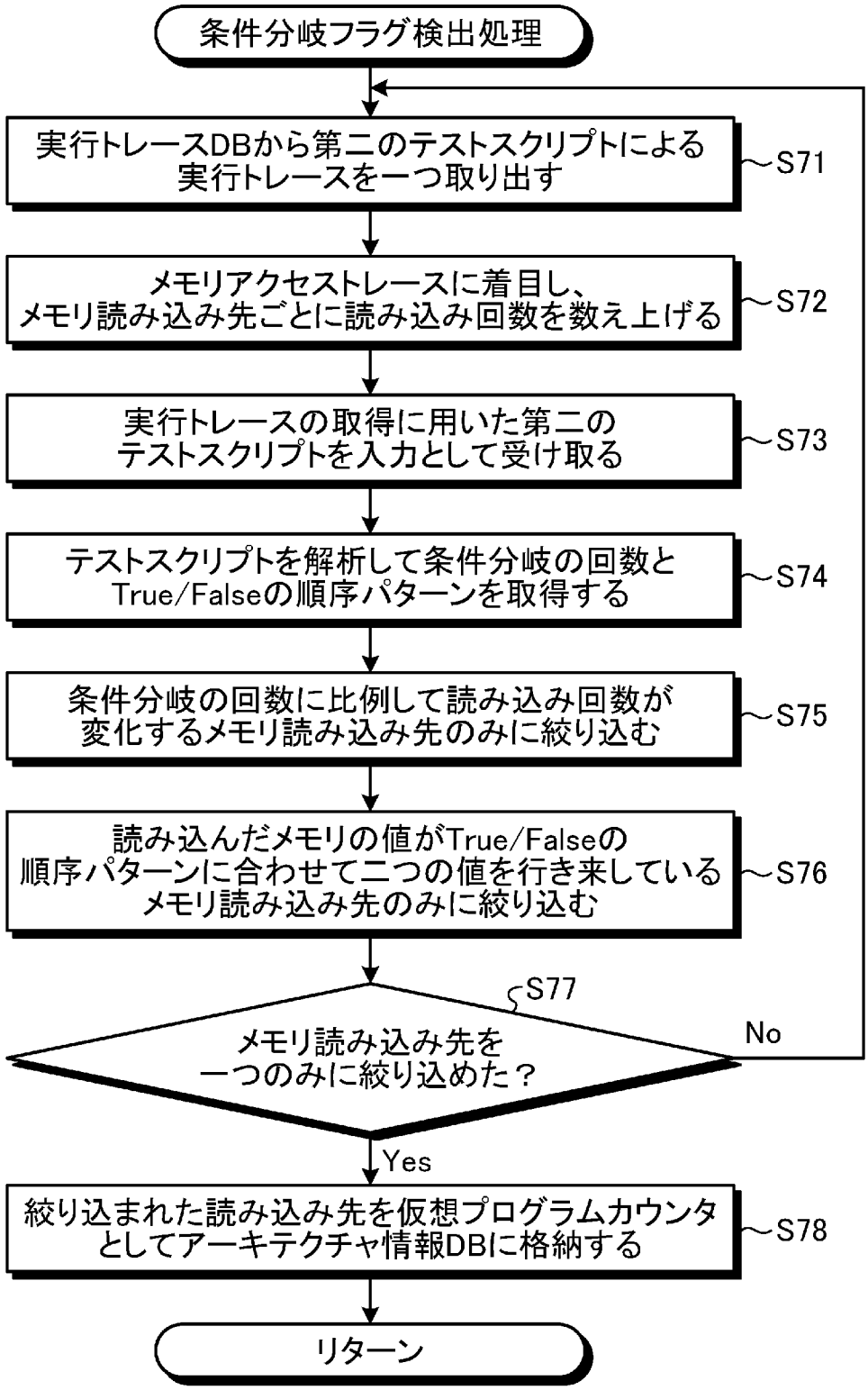
[図9]


VM命令境界検出処理
実行トレースDBから実行トレースを取り出す
S51
実行トレースを所定の方法で
クラスタリングする
S52
実行回数が閾値以上のクラスタを
VM命令として検出する
S53
VM命令を構成する連続した命令列の
開始点と終了点を境界とする
S54
VM命令の境界を返り値として出力する
S55
リターン

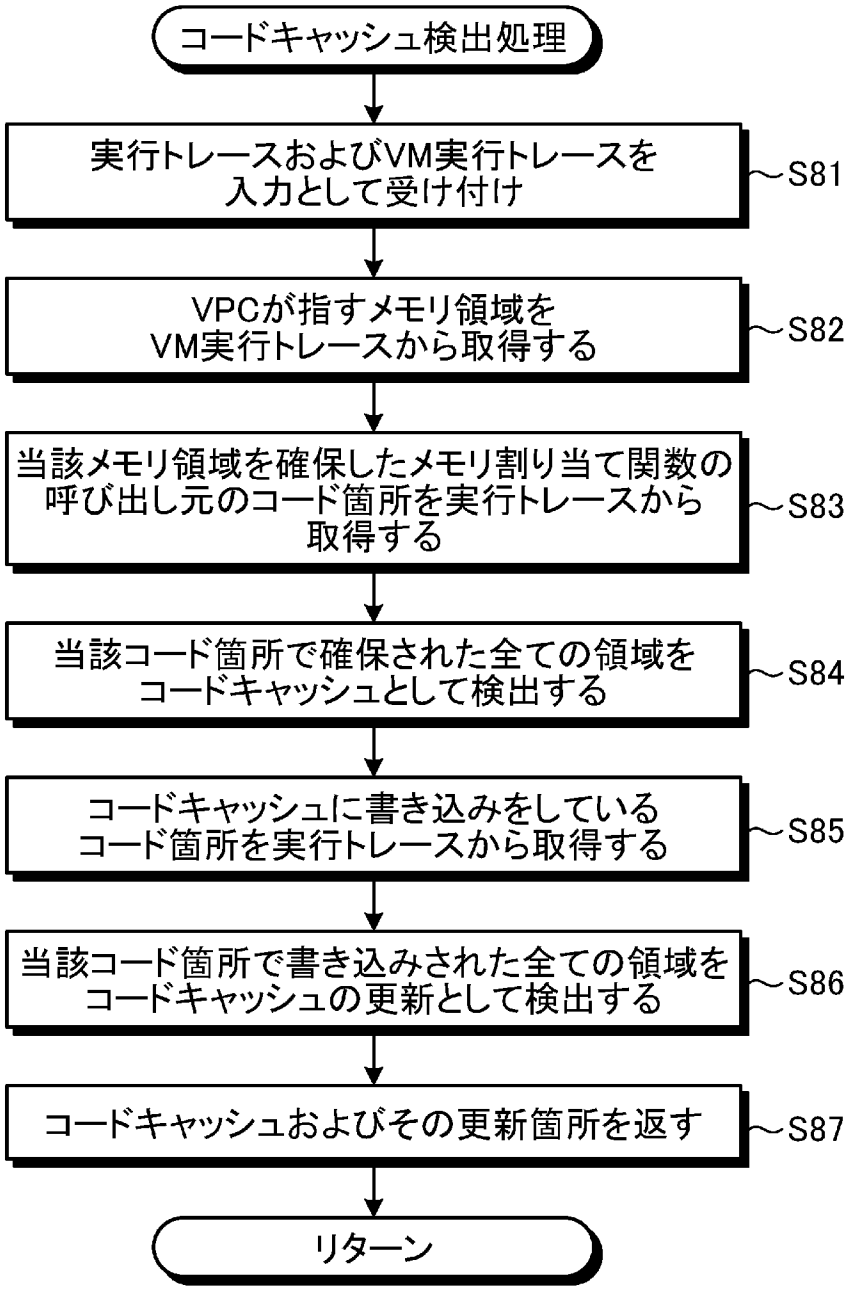
[図10]


ディスパッチャ検出処理
入力部からスクリプトエンジンバイナリを受け取る
S61
VM命令境界検出部から
VM命令の境界を受け取る
S62
スクリプトエンジンバイナリから
各VM命令部分を切り出す
S63
各VM命令間でコード間の類似度を
所定の方法で算出する
S64
全VM命令間で類似度が高い部分を取り出す
S65
S66
VM命令の終端部分？
No
Yes
ディスパッチャとして出力
S67
リターン

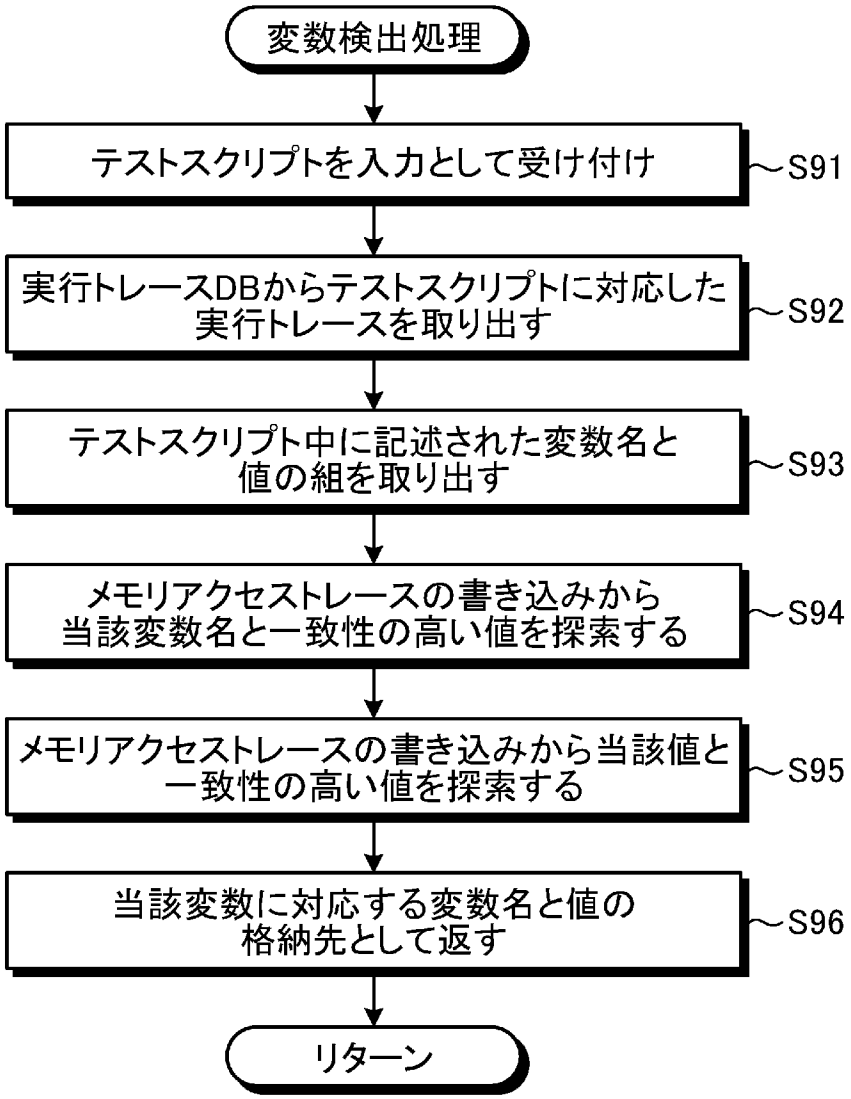
[図11]


条件分岐フラグ検出処理
実行トレースDBから第二のテストスクリプトによる実行トレースを一つ取り出す
S71
メモリアクセストレースに着目し、メモリ読み込み先ごとに読み込み回数を数え上げる
S72
実行トレースの取得に用いた第二のテストスクリプトを入力として受け取る
S73
テストスクリプトを解析して条件分岐の回数とTrue/Falseの順序パターンを取得する
S74
条件分岐の回数に比例して読み込み回数が変化するメモリ読み込み先のみに絞り込む
S75
読み込んだメモリの値がTrue/Falseの順序パターンに合わせて二つの値を行き来しているメモリ読み込み先のみに絞り込む
S76
S77
メモリ読み込み先を一つのみに絞り込めた？
No
Yes
絞り込まれた読み込み先を仮想プログラムカウンタとしてアーキテクチャ情報DBに格納する
S78
リターン

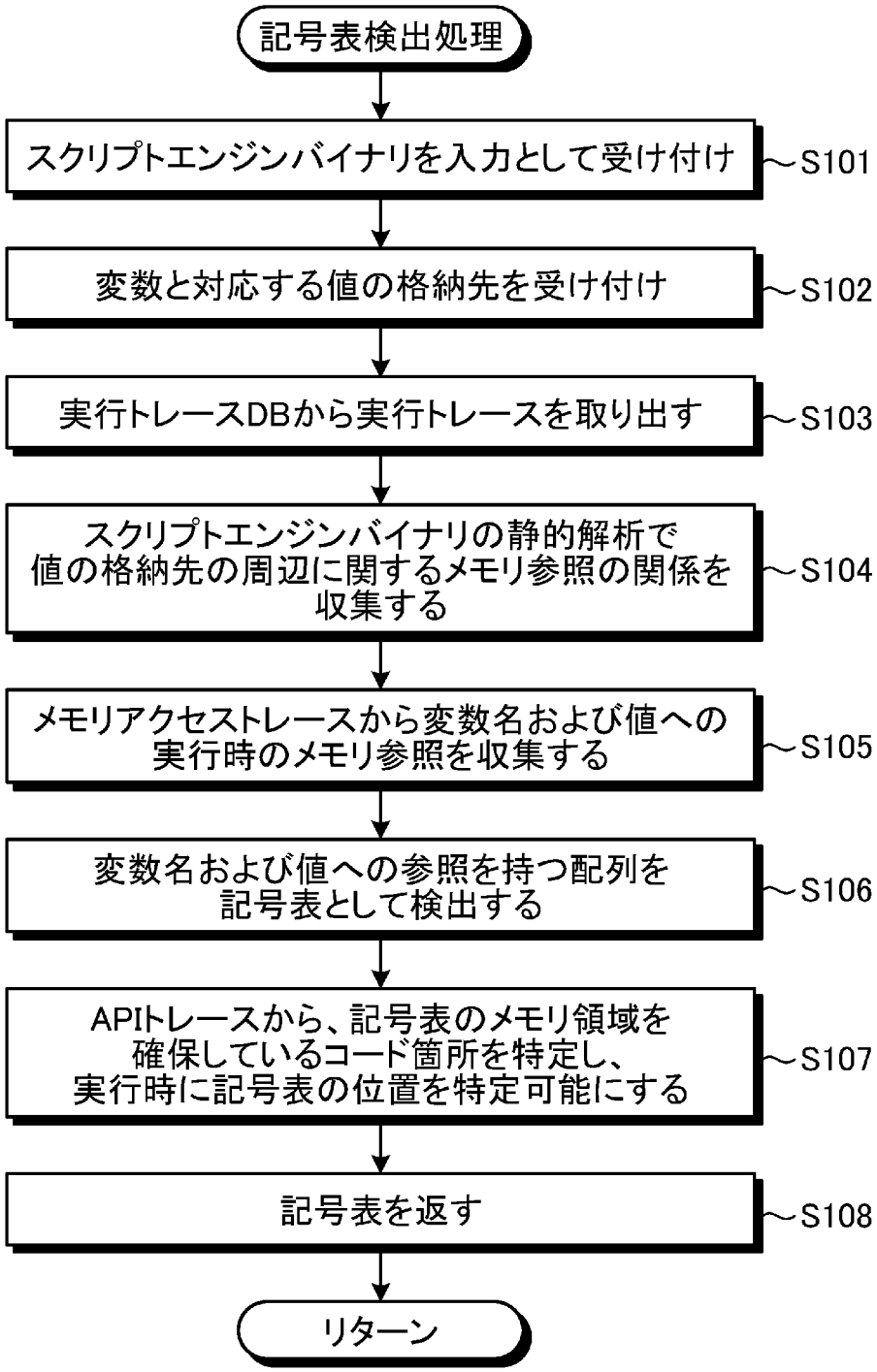
[図12]


コードキャッシュ検出処理
実行トレースおよびVM実行トレースを
入力として受け付け
S81
VPCが指すメモリ領域を
VM実行トレースから取得する
S82
当該メモリ領域を確保したメモリ割り当て関数の
呼び出し元のコード箇所を実行トレースから
取得する
S83
当該コード箇所で確保された全ての領域を
コードキャッシュとして検出する
S84
コードキャッシュに書き込みをしている
コード箇所を実行トレースから取得する
S85
当該コード箇所で書き込みされた全ての領域を
コードキャッシュの更新として検出する
S86
コードキャッシュおよびその更新箇所を返す
S87
リターン

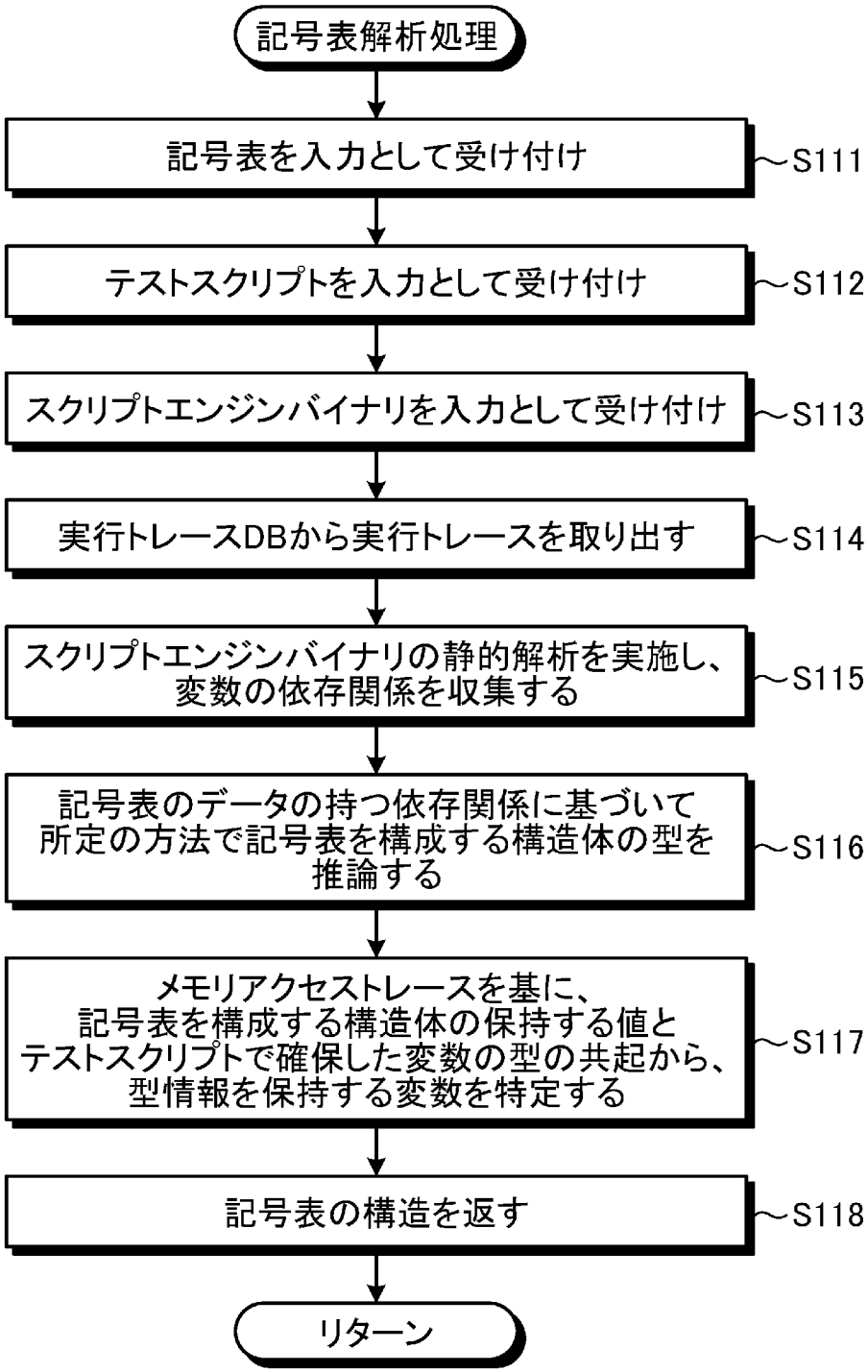
[図13]


変数検出処理
テストスクリプトを入力として受け付け
S91
実行トレースDBからテストスクリプトに対応した実行トレースを取り出す
S92
テストスクリプト中に記述された変数名と値の組を取り出す
S93
メモリアクセストレースの書き込みから当該変数名と一致性の高い値を探索する
S94
メモリアクセストレースの書き込みから当該値と一致性の高い値を探索する
S95
当該変数に対応する変数名と値の格納先として返す
S96
リターン

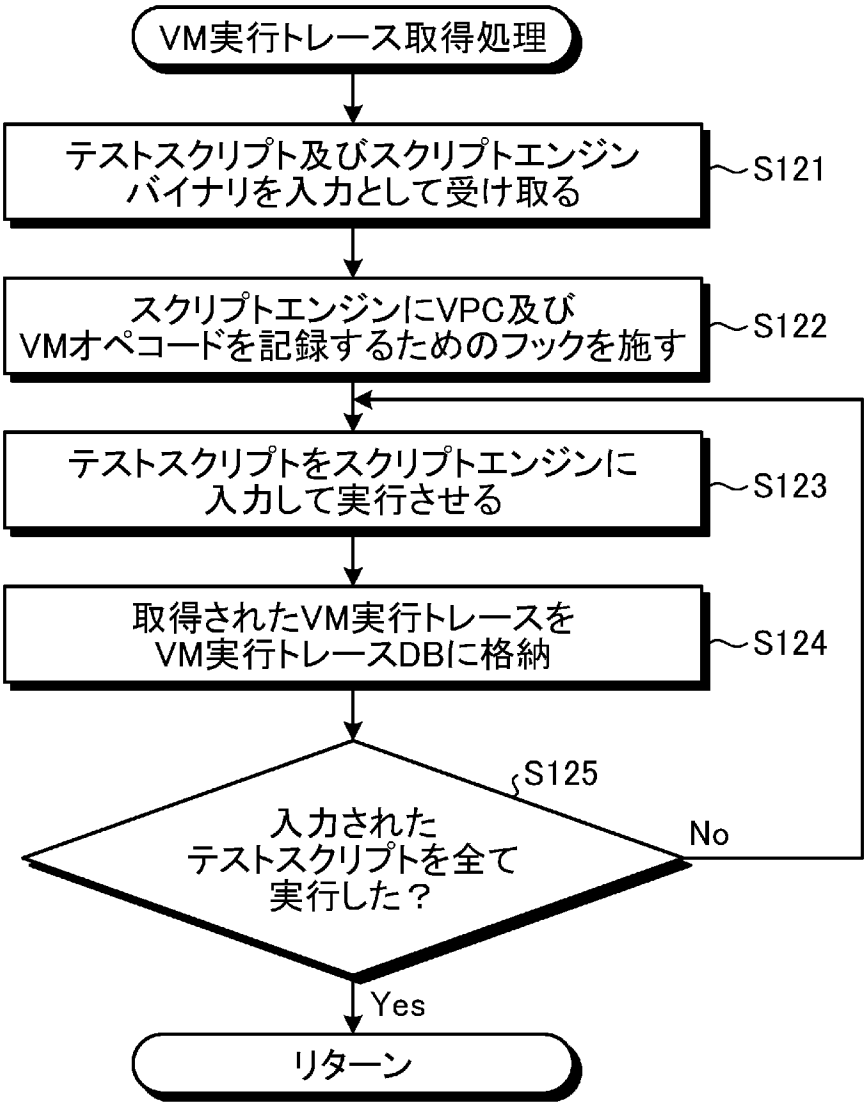
[図14]


記号表検出処理
スクリプトエンジンバイナリを入力として受け付け
S101
変数と対応する値の格納先を受け付け
S102
実行トレースDBから実行トレースを取り出す
S103
スクリプトエンジンバイナリの静的解析で
値の格納先の周辺に関するメモリ参照の関係を
収集する
S104
メモリアクセストレースから変数名および値への
実行時のメモリ参照を収集する
S105
変数名および値への参照を持つ配列を
記号表として検出する
S106
APIトレースから、記号表のメモリ領域を
確保しているコード箇所を特定し、
実行時に記号表の位置を特定可能にする
S107
記号表を返す
S108
リターン

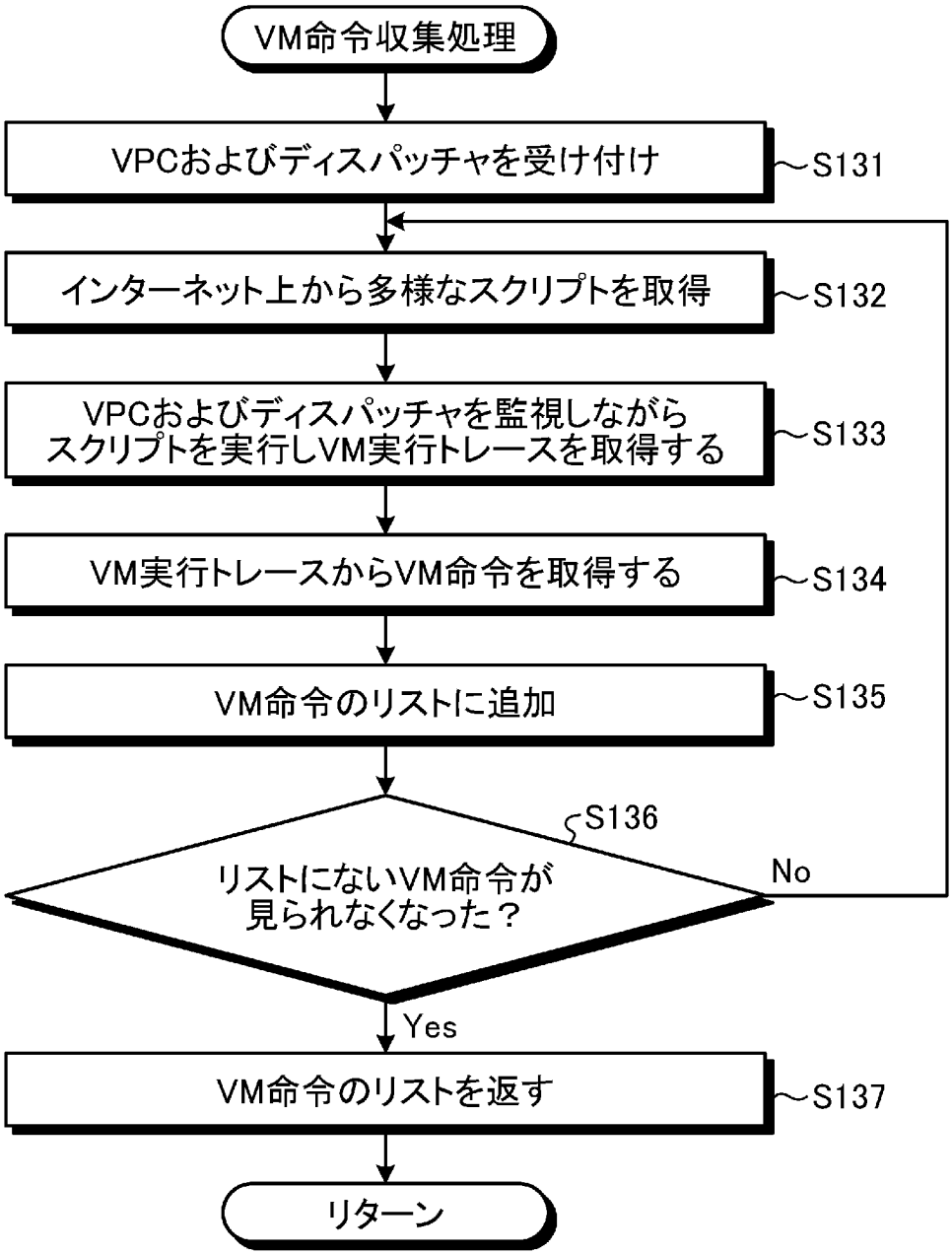
[図15]


記号表解析処理
記号表を入力として受け付け
S111
テストスクリプトを入力として受け付け
S112
スクリプトエンジンバイナリを入力として受け付け
S113
実行トレースDBから実行トレースを取り出す
S114
スクリプトエンジンバイナリの静的解析を実施し、
変数の依存関係を収集する
S115
記号表のデータの持つ依存関係に基づいて
所定の方法で記号表を構成する構造体の型を
推論する
S116
メモリアクセストレースを基に、
記号表を構成する構造体の保持する値と
テストスクリプトで確保した変数の型の共起から、
型情報を保持する変数を特定する
S117
記号表の構造を返す
S118
リターン

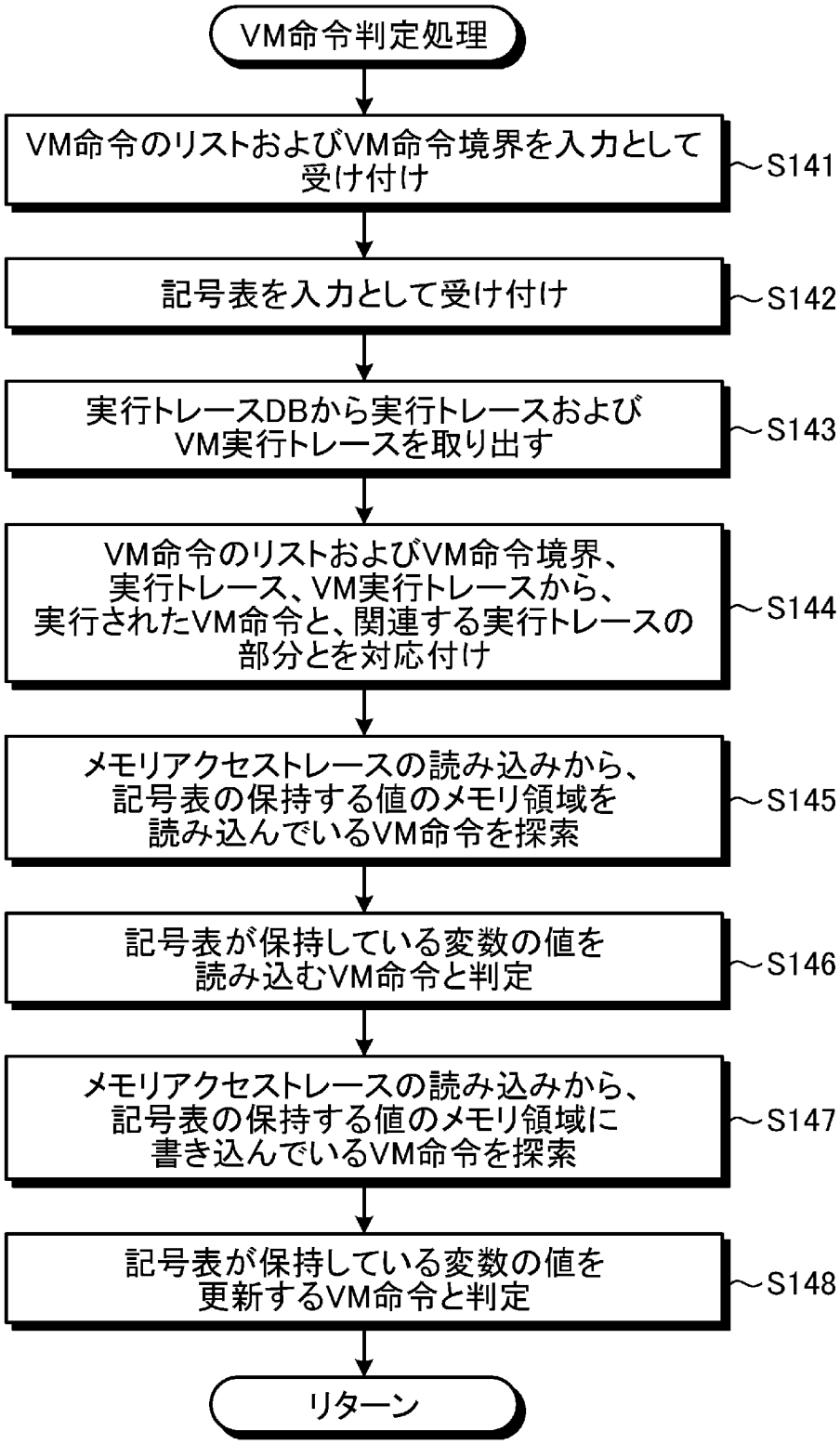
[図16]


VM実行トレース取得処理
テストスクリプト及びスクリプトエンジン
バイナリを入力として受け取る
S121
スクリプトエンジンにVPC及び
VMオペコードを記録するためのフックを施す
S122
テストスクリプトをスクリプトエンジンに
入力して実行させる
S123
取得されたVM実行トレースを
VM実行トレースDBに格納
S124
S125
入力された
テストスクリプトを全て
実行した？
No
Yes
リターン

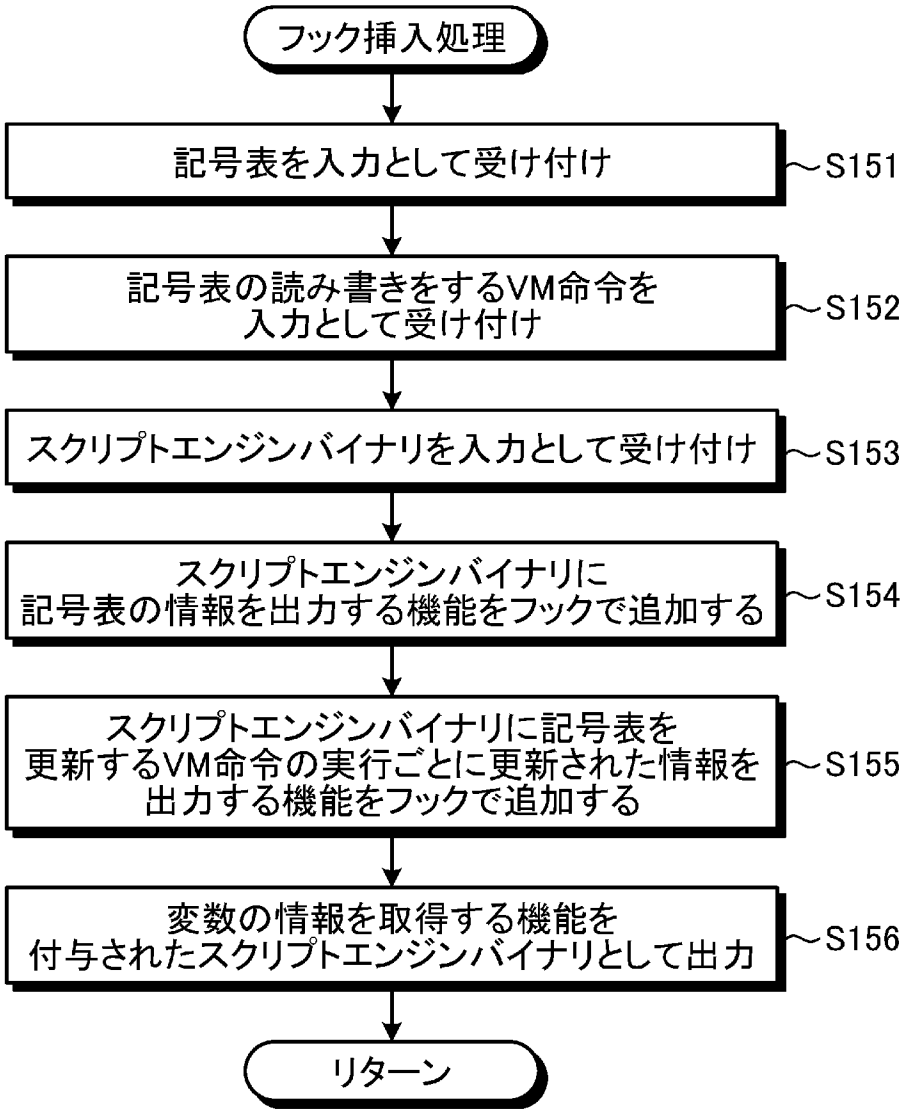
[図17]


VM命令収集処理
VPCおよびディスパッチャを受け付け
S131
インターネット上から多様なスクリプトを取得
S132
VPCおよびディスパッチャを監視しながら
スクリプトを実行しVM実行トレースを取得する
S133
VM実行トレースからVM命令を取得する
S134
VM命令のリストに追加
S135
S136
リストにないVM命令が
見られなくなった？
No
Yes
VM命令のリストを返す
S137
リターン

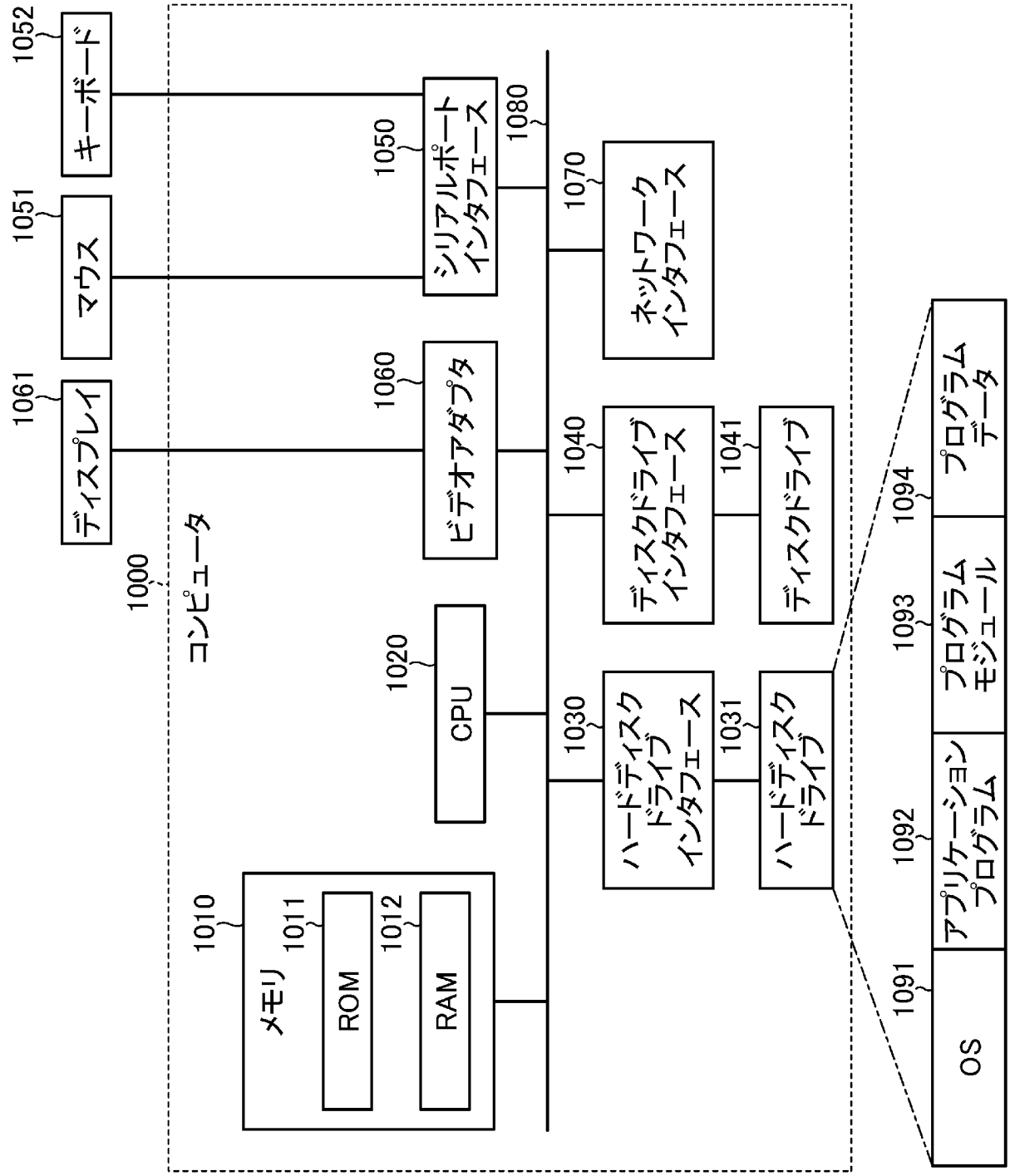
[図18]


VM命令判定処理
VM命令のリストおよびVM命令境界を入力として受け付け
S141
記号表を入力として受け付け
S142
実行トレースDBから実行トレースおよびVM実行トレースを取り出す
S143
VM命令のリストおよびVM命令境界、実行トレース、VM実行トレースから、実行されたVM命令と、関連する実行トレースの部分とを対応付け
S144
メモリアクセストレースの読み込みから、記号表の保持する値のメモリ領域を読み込んでいるVM命令を探索
S145
記号表が保持している変数の値を読み込むVM命令と判定
S146
メモリアクセストレースの読み込みから、記号表の保持する値のメモリ領域に書き込んでいるVM命令を探索
S147
記号表が保持している変数の値を更新するVM命令と判定
S148
リターン

[図19]


フック挿入処理
記号表を入力として受け付け
S151
記号表の読み書きをするVM命令を
入力として受け付け
S152
スクリプトエンジンバイナリを入力として受け付け
S153
スクリプトエンジンバイナリに
記号表の情報を出力する機能をフックで追加する
S154
スクリプトエンジンバイナリに記号表を
更新するVM命令の実行ごとに更新された情報を
出力する機能をフックで追加する
S155
変数の情報を取得する機能を
付与されたスクリプトエンジンバイナリとして出力
S156
リターン

[図20]


1000
コンピュータ
1010
メモリ
1011
ROM
1012
RAM
1020
CPU
1030
ハードディスクドライブインタフェース
1031
ハードディスクドライブ
1040
ディスクドライブインタフェース
1041
ディスクドライブ
1050
シリアルポートインタフェース
1052
キーボード
1051
マウス
1060
ビデオアダプタ
1061
ディスプレイ
1070
ネットワークインタフェース
1080
1091
OS
1092
アプリケーションプログラム
1093
プログラムモジュール
1094
プログラムデータ

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2022/037936

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/455(2006.01)i; ***G06F 11/36***(2006.01)i
FI: G06F9/455 100; G06F11/36 164

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)

G06F8/00-8/77; G06F9/44-9/455; G06F11/36

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Published examined utility model applications of Japan 1922-1996
Published unexamined utility model applications of Japan 1971-2022
Registered utility model specifications of Japan 1996-2022
Published registered utility model applications of Japan 1994-2022

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.
A	WO 2022/180702 A1 (NIPPON TELEGRAPH AND TELEPHONE CORPORATION) 01 September 2022 (2022-09-01) entire text, all drawings	1-6
A	星孝一郎、外２名, 実行時エラー時に変数の推移情報を表示するJava仮想マシンの提案, レクチャーノート／ソフトウェア学37 ソフトウェア工学の基礎XVIII, 30 November 2011, pp. 31-40 entire text, non-official translation (HOSHI, Koichiro et al. Proposal for a Java virtual machine that displays variable transition information during runtime error. Lecture Notes/ Software Science 37 Foundation of Software Engineering XVIII.)	1-6

☐ 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
14 December 2022	**27 December 2022**
Name and mailing address of the ISA/JP **Japan Patent Office (ISA/JP)** **3-4-3 Kasumigaseki, Chiyoda-ku, Tokyo 100-8915** **Japan**	Authorized officer Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/JP2022/037936

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
WO 2022/180702 A1	01 September 2022	(Family: none)	

国際調査報告

国際出願番号
PCT/JP2022/037936

A. 発明の属する分野の分類（国際特許分類（ＩＰＣ））
G06F 9/455(2006.01)i; G06F 11/36(2006.01)i
FI: G06F9/455 100; G06F11/36 164

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））
G06F8/00-8/77; G06F9/44-9/455; 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　１９７１－２０２２年
日本国実用新案登録公報　１９９６－２０２２年
日本国登録実用新案公報　１９９４－２０２２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
A	WO 2022/180702 A1 (日本電信電話株式会社) 01.09.2022 (2022-09-01) 全文，全図	1-6
A	星孝一郎、外２名，実行時エラー時に変数の推移情報を表示するJava仮想マシンの提案，レクチャーノート／ソフトウェア学37 ソフトウェア工学の基礎XVIII，2011.11.30，pp.31-40 全文	1-6

☐ 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抵触するものではなく、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
14.12.2022	27.12.2022

名称及びあて先
日本国特許庁(ISA/JP)
〒100-8915
日本国
東京都千代田区霞が関三丁目４番３号

権限のある職員（特許庁審査官）
杉浦　孝光 5B 5287
電話番号 03-3581-1101 内線 3545

国際調査報告
パテントファミリーに関する情報

国際出願番号
PCT/JP2022/037936

引用文献	公表日	パテントファミリー文献	公表日
WO 2022/180702 A1	01.09.2022	（ファミリーなし）	