

(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
2022年4月21日(21.04.2022)



(10) 国際公開番号

WO 2022/079779 A1

(51) 国際特許分類:

G06F 11/36 (2006.01)

(21) 国際出願番号: PCT/JP2020/038543

(22) 国際出願日: 2020年10月12日(12.10.2020)

(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).

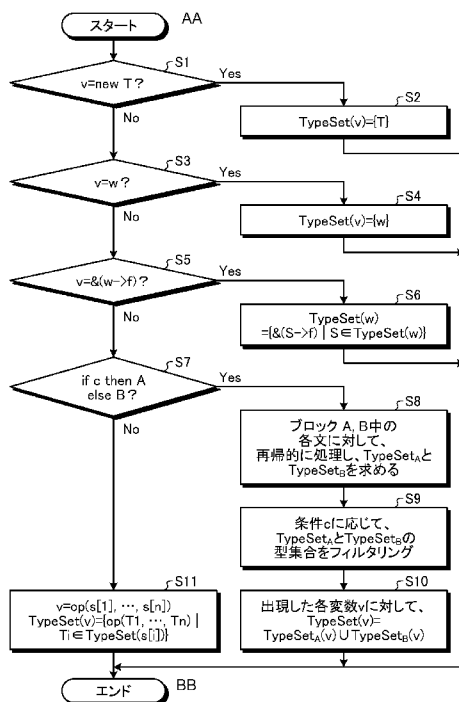
(72) 発明者: 青島達大 (AOSHIMA, Tatsuhiko); 〒1808585 東京都武蔵野市緑町3丁目9-1 NTT 知的財産センタ内 Tokyo (JP). 碓

井利宣 (USUI, Toshinori); 〒1808585 東京都武蔵野市緑町3丁目9-1 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9-1 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9-1 NTT 知的財産センタ内 Tokyo (JP). 三好潤 (MIYOSHI, Jun); 〒1808585 東京都武蔵野市緑町3丁目9-1 NTT 知的財産センタ内 Tokyo (JP).

(74) 代理人: 特許業務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

(54) Title: DETECTION DEVICE AND DETECTION PROGRAM

(54) 発明の名称: 検知装置、および、検知プログラム



S8 Obtain TypeSet_A and TypeSet_B by recursively processing each statement in blocks A, B

S9 Filter type sets TypeSet_A and TypeSet_B according to condition c

S10 TypeSet(v) = TypeSet_A(v) ∪ TypeSet_B(v) for each found variable v

AA Start

BB End

(57) Abstract: This detection device applies abstract interpretation to the source code of a program to collect the types of variables that can be taken at the conditional branches of the source code. The detection device then identifies the dependencies among the variables using a dependent type theory, and filters the set of variable types at the conditional branch destinations using the identified dependencies and the conditions used at the conditional branches. The detection device then performs type checking to determine whether or not each element of the set of types can be cast to a concrete

関 3 丁 目 8 番 1 号 虎 の 門 三 井 ビ ル
ディング 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, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, 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, 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, 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, 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：

一 国際調査報告 (条約第21条(3))

type at the cast destination, and outputs the type checking results. The detection device also identifies the dependencies between function pointers and argument strings when identifying the dependencies among the variables.

(57) 要約：検知装置は、プログラムのソースコードに抽象解釈を適用することにより、ソースコードの条件分岐で取り得る変数の型を収集する。そして、検知装置は、依存型理論により変数間の依存関係を特定し、特定した依存関係と条件分岐で用いられる条件を用いて、条件分岐先における変数の型の集合のフィルタリングを行う。そして、検知装置は、型の集合の各要素がキャスト先の具体型へキャスト可能か否かを判定する型検査を行い、型検査の結果を出力する。また、検知装置は、変数間の依存関係を特定する際、関数ポインターと引数列との間の依存関係も特定する。

明 細 書

発明の名称： 検知装置、および、検知プログラム

技術分野

[0001] 本発明は、プログラムの型混同 (Type Confusion) による脆弱性の検知技術に関する。

背景技術

[0002] 従来、プログラムの型混同による脆弱性の検知技術としてポインター解析がある。このポインター解析は、プログラムに含まれる関数のポインター変数の参照先を列挙することにより、型混同の脆弱性を検知する技術である。

先行技術文献

非特許文献

[0003] 非特許文献1：Zou, C. et al. (2019) “TCD: Statically Detecting Type Confusion Errors in C++ Programs.” ISSRE.

発明の概要

発明が解決しようとする課題

[0004] しかし、上記のポインター解析は、解析対象の分岐条件を無視するので、プログラムの動的な型変換による分岐を考慮できず、型の候補を過剰に列挙してしまう。その結果、プログラムの型混同の脆弱性の検知において誤検知が発生しやすいという問題がある。そこで、本発明は、前記した問題を解決し、プログラムの型混同の脆弱性の検知において、誤検知を低減することを課題とする。

課題を解決するための手段

[0005] 前記した課題を解決するため、本発明は、プログラムのソースコードを読み込むローダー部と、前記読み込んだソースコードに抽象解釈を適用することにより、前記ソースコードの条件分岐で取り得る変数の型を収集し、依存型理論により前記変数間の依存関係を特定し、前記依存関係と前記条件分岐で用いられる条件を用いて、前記収集した型の集合のうち前記条件分岐先に

おける型の集合のフィルタリングを行う命令解釈部と、を備えることを特徴とする。

発明の効果

[0006] 本発明によれば、プログラムの型混同の脆弱性の検知において、誤検知を低減することができる。

図面の簡単な説明

[0007] [図1]図1は、検知装置の処理の概要を説明するための図である。

[図2]図2は、検知装置の構成例を示す図である。

[図3]図3は、依存型の構文の定義の例を示す図である。

[図4]図4は、文脈の定義域（変数名）を説明するための図である。

[図5]図5は、射影の種類を説明するための図である。

[図6]図6は、図2の型検査部の処理手順の例を示すフローチャートである。

[図7]図7は、図2の命令解釈部の処理手順の例を示すフローチャートである。

[図8]図8は、図7のS40における処理手順の例を示すフローチャートである。

[図9]図9は、図7のS44における処理手順の例を示すフローチャートである。

[図10]図10は、検知装置の動作例を説明するための図である。

[図11]図11は、検知プログラムを実行するコンピュータの構成例を示す図である。

発明を実施するための形態

[0008] 以下、図面を参照しながら、本発明を実施するための形態（実施形態）について説明する。本発明は、本実施形態に限定されない。

[0009] 本実施形態の検知装置は、プログラムの型混同の脆弱性の誤検知を低減するため、検知対象のプログラムのあらゆる実行パスから型を集める抽象解釈を採用する。そして、検知装置は、変数間の依存関係を依存型（dependent type）理論により表現し、分岐条件とその依存関係を解決し、収集した型の

集合をフィルタリングする。

- [0010] 図1を用いて、検知装置の処理の概要を説明する。まず、検知装置は、検知対象のプログラムの命令文の入力を受け付けると、オブジェクト確保の度に、命令文の変数(v)が取り得る型の集合(型集合)を初期化する(S1の $v = \text{new } T?$ でYes $\rightarrow$ S2の $\text{TypeSet}(v) = \{T\}$)。
- [0011] また、S1で命令文の $v = \text{new } T$ ではなく(S1でNo)、 $v = w$ の場合(S3でYes)、検知装置は、値の代入について、変数wという型式として表現された型を利用し、そのみを含む集合に置き変える(S4: $\text{TypeSet}(v) = \{w\}$)。
- [0012] 一方、命令文の $v = w$ ではなく(S3でNo)、 $v = \&(w \rightarrow f)$ の場合(S5でYes)、検知装置は、アドレス操作の度に、型集合の各要素を基点にしたポインター操作の結果であることを型式で表現する(S6: $\text{TypeSet}(w) = \{\&(S \rightarrow f) \mid S \in \text{TypeSet}(w)\}$)。
- [0013] 一方、命令文の $v = \&(w \rightarrow f)$ ではなく(S5でNo)、条件分岐である場合(S7でYes)、S8へ進む。そして、検知装置は、例えば、まず、条件cで分岐されるブロックA,B中の各文に対して、再帰的に処理し、 TypeSet_A と TypeSet_B を求める(S8)。次に、検知装置は、条件cに応じて、射影機構により TypeSet_A と TypeSet_B の型集合をフィルタリングする(S9)。そして、検知装置は、S9のフィルタリングにより出現した各変数vに対して、 $\text{TypeSet}(v) = \text{TypeSet}_A(v) \cup \text{TypeSet}_B(v)$ を求める(S10)。
- [0014] 一方、命令文が条件分岐ではない場合(S7でNo)、検知装置は、 $v = \text{op}(s[1], \dots, s[n])$ 、 $\text{TypeSet}(v) = \{\text{op}(T_1, \dots, T_n) \mid T_i \in \text{TypeSet}(s[i])\}$ とする(S11)。
- [0015] 検知装置は、上記のようにして得られた型集合($\text{TypeSet}(v)$)に対し、型検査を行い、プログラムの型混同の脆弱性の検知を行う。このように、検知装置は、条件分岐を考慮して型候補の収集を行うので、例えば、C++の動的な型変換(仮想継承関係にあるクラス間のdynamic_cast)による分岐が含まれる場合でも、型の候補を正確に列挙することができる。その結果、検知装置は、プログラムの型混同の脆弱性の誤検知を低減することができる。

[0016] 次に、図2を用いて検知装置の構成例を説明する。検知装置10は、ローダー部11と、トレーサー部12と、命令解釈部13と、型検査部14と、文脈15と、キュー16とを備える。

[0017] ローダー部11は、プログラムのソースコードの読み込みと解釈とを行う。トレーサー部12は、命令解釈部13で処理すべきソースコードの関数、ブロック等を管理する。また、トレーサー部12は、命令解釈部13による処理の収束判定を行う。命令解釈部13は、文脈15と型検査部14の情報を参照しながら、各命令を解釈し、各命令により操作される変数の型集合を計算する。例えば、命令解釈部13は、トレーサー部12により指示された関数またはブロックの各命令を順に解釈し、操作される変数の型集合を計算する。

[0018] 型検査部14は、型情報の管理と型検査を行う。例えば、型検査部14は、型の宣言リスト（型情報）に基づき、型キャスト（型変換）が妥当かどうかを判定し、キャスト先の型を計算する。文脈15は、各関数のブロックごとに、変数名から当該変数名に対応する型集合への写像を管理する。キュー16は、命令解釈部13が次に処理する関数、ブロックのリストを保持する。

[0019] 以下、図2を参照しながら、検知装置10の動作例を説明する。まず、ローダー部11が検知対象のプログラムのソースコードを読み込むと、型検査部14で保持される環境情報、文脈15で保持する型集合、キュー16で保持するリストそれぞれを初期化する（（1）～（3））。例えば、（2）において、ローダー部11は、文脈15のグローバル変数の値を評価し、型集合を初期化する。

[0020] 次に、トレーサー部12は、ローダー部11で読み込んだソースコードの解析を開始する（（4））。なお、解析を始める関数は、例えば、mainでもよいし、検知装置10のユーザが指定したものであってもよい。トレーサー部12は、ソースコードの解析の過程で、キュー16から次に処理する関数／ブロックを取り出し（（5））、先行ブロックが分岐元であれば、その分

岐条件を解決し、文脈 15 を更新する（（6））。

[0021] 命令解釈部 13 は、トレーサ部 12 により取り出されたブロック中の各命令を順に処理する（（7））。例えば、命令解釈部 13 は、型検査部 14 および文脈 15 で保持される情報の参照しながら、各命令を解釈し、当該命令により操作される変数の型集合を計算する。また、命令解釈部 13 は、上記の計算結果に基づき、型検査部 14 および文脈 15 で保持される情報を更新し、処理の過程で条件分岐による分岐先があれば、それら（分岐先の情報）をキュー 16 に追加する。

[0022] その後、トレーサ部 12 が、命令解釈部 13 の処理により、文脈 15 中のすべての型集合が収束したと判定すると（（8））、型検査部 14 による型検査のエラー情報（型キャストに違反した箇所とその理由）を出力する。このようにすることで検知装置 10 は、プログラムの型混同の脆弱性の検知を行うことができる。

[0023] [依存型の構文の定義例]

図 3 を用いて、本実施形態における依存型の構文の定義の例を説明する。

図 3 は、BNF 記法による依存型の構文の定義を示す。

[0024] 本実施形態では、具体型（ConcreteType）は、C/C++ 標準で規定された基本型や複合型とするが、他のプログラミング言語で扱える型や、整数／実数区間([a, b]や(a, b)等)、多次元実ベクトル空間中の凸集合等でもよい。

[0025] また、値（Value）は、C/C++ 標準で規定された型に対応する値としているが、具体型の値であれば、これに限定されない。さらに、型式（TypeExpr）の修正には注意を要するが、フィールドアドレスの参照操作における添え字の数を 2 つ以上にする等、表現力が変わらない範囲での修正に関しては、限定されない。

[0026] [文脈の定義と収束判定の例]

次に、図 4 を用いて、本実施形態における文脈の定義と収束判定について説明する。検知装置 10 は、ポインターを介した破壊的な代入が可能なプログラムを解析するため、変数の値を代入箇所ごとに区別する必要がある。そ

ここで、検知装置 10 のトレーサ部 12 は、代入箇所を記録するため、Loc を用いる。Loc は、関数名 (funcName)、ブロック名 (BlockName)、命令番号 (InsnIdx) を持つ。命令番号は 1 から始まるものとする。また、命令番号 0 は、先行ブロックからの回収結果を格納する用途等で擬似的に使う。なお、トレーサ部 12 は、トレース対象のオブジェクトが同じ型でも、初期化された箇所が異なれば、別として扱う。

[0027] 文脈 15 は、変数の取り得る値を保持するため、変数名 RegPath を型集合へ対応させる写像を管理する。トレーサ部 12 は、収束判定のために、変更を伴う代入の度に、文脈 15 のバージョンをインクリメントする。また、トレーサ部 12 は、現在のノードのバージョンも文脈 15 のバージョンに更新する。そして、トレーサ部 12 は、現在のノードのバージョンが、先行するすべてのノードのバージョン以降であれば、収束したと判定する。

[0028] なお、トレーサ部 12 は、以下の (1) ~ (3) の条件をすべて満たすときに、変更を伴う代入と判定する。

[0029] (1) 変数が射影機構により制約されていない。

(2) 変数にはすでに値が代入されていて、変数の代入箇所が現在のノードで、以下の 2 つの条件のいずれかを満たす。

(2-1) 前回の変数の代入箇所より命令番号が新しい。

(2-2) 前回の代入における型集合と異なる値を代入する。

(3) 変数の代入箇所の命令番号が 0 でない。

[0030] [射影機構の定義と文脈の動作]

図 5 を用いて、本実施形態における射影の種類と文脈 15 の動作について説明する。検知装置 10 の命令解釈部 13 は、射影機構により、特定の変数の型集合の要素を制約する。射影の種類は、例えば、以下のようなものがある。

[0031] ・関数ポインターと引数列間の依存関係を解決するため、一命令でのみ有効な at。

・分岐条件により型集合をフィルタリングするため、分岐先でのみ有効な、t

rue-branch、false-branch。

・エラーにより追跡対象外とするときにフィルタリングするため、それ以降の命令でのみ有効なsince。

[0032] 射影の種類は、プログラム中のどの箇所でも有効かが明確に定義可能なものであれば上記のものに限定されない。

[0033] 射影機構は、文脈 15 と同じく、制約する変数名と対応する型集合への写像を持つ。

[0034] なお、命令解釈部 13 が変数の型集合を取得するときは、射影機構により制約されているかどうかを確認し、射影機構により制限されている場合は、射影機構により制限された型集合を返す。一方、射影機構により制限されていない場合、文脈 15 が保持する型集合を返す。

[0035] また、命令解釈部 13 が変数へ型集合を代入するときは、射影機構により制約されているかどうかを確認し、射影機構により制約されている場合は、射影機構へ代入し、射影機構により制約されていなければ、文脈 15 へ代入する。

[0036] [型検査部]

型検査部 14 は、型情報の管理と型検査を行う。型検査は、キャスト元の型集合の各要素がキャスト先の具体型へキャスト可能かどうかの判定を行うことである。型検査部 14 が型検査に失敗した場合は、そのエラー情報を記録し、型検査に失敗した型を型集合から取り除く。型検査部 14 は、型検査において、C/C++標準の規則（ただし、ポインターのキャストにおいて、アップキャストと関数ポインター間のキャストのみ認めるように修正したもの）を用いるが、これに限らず、具体型の定義に応じ、他のプログラミング言語の規則を用いてもよい。

[0037] また、キャスト先が具体型に限定されるのは、ソースコードで期待する型へキャストされるかどうかを判定するには、ソースコード中に現れる具体型に限定してよいからである。

[0038] なお、型検査部 14 は、追跡対象外の型に関する型検査は行わない。なお

、型検査部 14 が所有型に対する型検査を行う際のキャスト規則は、図 6 を用いて後記する。

[0039] また、型検査部 14 は、型式に対する型検査では、型式の意味論に従い、文脈 15 中の変数を参照しながら、型式を含まない形まで再帰的に展開し、具体型と所有型に対する型検査を適用する。

[0040] [型検査を行う際の処理手順の例]

次に、図 6 を用いて型検査部 14 が、所有型に対する型検査を行う際の処理手順の例を説明する。なお、図 6 に示す処理における、型検査における型検査部 14 への入力は、キャスト元の所有型 S およびキャスト先の具体型 T であり、型検査部 14 からの出力は、キャスト可能か否かの判定結果である。

[0041] まず、型検査部 14 は、入力された S（所有型 S）が具体型であれば（S11で Yes）、具体型 S が具体型 T へキャスト可能かどうかを判定し、その結果を返す（S12）。

[0042] 一方、入力された S（所有型 S）が具体型ではなく（S11で No）、定数型であれば（S13で Yes）、型検査部 14 は、定数の型が具体型 T へキャスト可能かどうかを判定し、キャスト可能なら、その結果を返す（S14）。

[0043] S14の後、型検査部 14 は、定数の値が NULL で、T（具体型 T）がポインター型であれば（S15で Yes）、キャスト可と判定する（S24）。一方、型検査部 14 は、定数の値が NULL で、T（具体型 T）がポインター型でなければ（S15で No）、キャスト不可と判定する（S25）。

[0044] また、S13において、S が定数型ではなく（S13で No）、関数ポインター型である場合（S16で Yes）、型検査部 14 は、T が関数ポインター型か否かを判定する（S17）。そして、T が関数ポインター型である場合（S17で Yes）、型検査部 14 は、キャスト可と判定する（S24）。一方、T が関数ポインター型ではない場合（S17で No）、型検査部 14 は、キャスト不可と判定する（S25）。

[0045] また、S が関数ポインター型ではなく（S16で No）、&S' である場合（S18で Yes）、型検査部 14 は、S' が具体型か否かを判定する（S19）。ここで、型

検査部 1 4 が、 S' は具体型と判定した場合 (S19でYes)、具体型 $S' *$ が具体型 T へキャスト可能かを判定し、その結果を返す (S20)。また、 S が関数ポインター型ではなく (S16でNo)、 $\&S'$ でもない場合 (S18でNo)、型検査部 1 4 は、キャスト不可と判定する (S25)。

[0046] 一方、型検査部 1 4 が、 S' は具体型ではないと判定した場合 (S19でNo)、具体型 T が $T' *$ か否かを判定する (S21)。ここで、型検査部 1 4 が具体型 T は $T' *$ であり (S21でYes)、 $T' = \text{char}$ であると判定した場合 (S22でYes)、キャスト可と判定する (S24)。一方、型検査部 1 4 が、 $T' = \text{char}$ ではないと判定した場合 (S22でNo)、キャスト不可と判定する (S25)。また、型検査部 1 4 が、具体型 T は $T' *$ ではないと判定した場合 (S21でNo)、具体型 S' が具体型 T' へキャスト可能かを判定し、その結果を返す (S23)。

[0047] [命令解釈部]

次に、図 7 を用いて命令解釈部 1 3 の処理手順の例を説明する。なお、図 7 に示す処理における、命令解釈部 1 3 への入力、命令文および命令文における命令の箇所 L であり、命令解釈部 1 3 からの出力は、型集合 (TypeSet) であるものとして説明する。

[0048] まず、命令解釈部 1 3 は、命令における $v = \text{new } T$ である場合 (S31でYes)、オブジェクトの初期化を行い、 $\text{TypeSet}(v@L) = \{T@L\}$ とする (S32)。例えば、命令解釈部 1 3 は、オブジェクト中のフィールド $T@L[i]$ を、型定義に従い再帰的に初期化し、型式 $T@L$ で定義する。

[0049] 一方、命令における $v = \text{new } T$ ではなく (S31でNo)、 $v = w$ である場合 (S33でYes)、命令解釈部 1 3 は、変数の代入を行い、 $\text{TypeSet}(v@L) = \{w\}$ とする (S34)。例えば、命令解釈部 1 3 は、変数 w の型集合を文脈 1 5 から取得し、変数 v の型へキャスト可能かを型検査を行い、型式 w で定義する。

[0050] また、命令における $v = w$ ではなく (S33でNo)、 $v = \&(w \rightarrow f)$ の場合 (S35でYes)、命令解釈部 1 3 はアドレス操作を行い、 $\text{TypeSet}(v) = \{\&(S \rightarrow f) \mid S \in \text{TypeSet}(v)\}$ とする (S36)。例えば、命令解釈部 1 3 は、変数 w の型集合を文脈 1 5 から取得し、各要素に対するC/C++フィールド参照規則の適用結果で定

義する。

[0051] また、命令における $v = \&(w \rightarrow f)$ ではなく (S35でNo)、 $v = *w$ の場合 (S37でYes)、命令解釈部 13 はポインタの参照を行い、 $\text{TypeSet}(v@L) = \{ *w \}$ とする (S38)。例えば、命令解釈部 13 は、変数 w の型集合を文脈 15 から取得し、各要素にC/C++のポインタ参照規則を適用し、変数 v の型へキャスト可能か型検査を行い、型式 $*w$ で定義する。

[0052] また、命令における $v = *w$ ではなく (S37でNo)、 $*v = w$ の場合 (S39でYes)、命令解釈部 13 はポインタの代入を行う (S40)。なお、このポインタの代入処理については、図 8 を用いて後記する。

[0053] また、命令における $*v = w$ ではなく (S39でNo)、 c という条件に応じて、命令の箇所 L へ移動する場合 (S41でYes)、命令解釈部 13 は、分岐先をキュー 16 へ追加する (S42)。一方、 c という条件に応じ、命令の箇所 L へ移動しない場合 (S41でNo)、命令における $v = f(a[1], \dots, a[n])$ であれば (S43でYes)、命令解釈部 13 は関数を呼び出す (S44)。なお、この関数の呼び出し処理については、図 9 を用いて後記する。

[0054] 一方、命令における $v = f(a[1], \dots, a[n])$ でなければ (S43でNo)、命令解釈部 13 は、 $v = \text{op}(s[1], \dots, s[n])$ 、 $\text{TypeSet}(v) = \{ \text{op}(T_1, \dots, T_n) \mid T_i \in \text{TypeSet}(s[i]) \}$ とする (S45)。また、命令解釈部 13 は、その他の演算では、集合として回収してもよいし、その関係性を型式で表現してもよい。例えば、C++の`dynamic_cast`操作は、型検査を行い、失敗したらエラーを報告し、その型をnullポインタ型へ変換する等の処理を行えばよい。

[0055] [命令解釈部のポインタ代入]

次に、図 8 を用いて、図 7 のS40の処理の例を説明する。図 7 のS40において命令解釈部 13 は、代入先の変数の型集合の要素それぞれに対して、代入処理を行う。ここでの代入処理とは、代入元の型集合が代入先の型へキャスト可能か型検査を行い、間接的なフィールド先も含め、代入先それぞれを、代入元の型集合で定義する処理のことである。

[0056] なお、図 8 に示す処理において、命令解釈部 13 への入力、代入先 $*v$ と代

入元wであり、命令解釈部13からの出力は、型集合 (TypeSet) であるものとして説明する。

[0057] 例えば、命令解釈部13は、代入先*vの型集合TypeSet (v@L) を文脈15から取得し、非ローカル変数以外をすべて展開する (S401)。また、命令解釈部13は、代入元wの型集合TypeSet (w) を文脈15から取得し、非ローカル変数以外をすべて展開する (S402)。

[0058] S402の後、命令解釈部13は、TypeSet(*v)、TypeSet(&w)が代入先のソースコード上の型へキャスト可能かどうか型検査を行う (S403)。その後、命令解釈部13は、TypeSet(*v)に含まれる各変数v'に対して、TypeSet (v'@L)=TypeSet (w) とする処理を繰り返す (S404)。その後、*vがローカル変数でなければ (S405でNo)、命令解釈部13は、処理を終了する。

[0059] 一方、*vがローカル変数であるが (S405でYes)、wはローカル変数でなければ (S406でNo)、命令解釈部13は、TypeSet (*v@L)=TypeSet (w@L) とする (S407)。そして、処理を終了する。一方、wがローカル変数であれば (S406でYes)、命令解釈部13は、TypeSet (*v@L)= {w@L} とする (S408)。なお、S408において、命令解釈部13は、代入先*vと代入元wがローカル変数であれば、命令解釈部13は、ローカル変数間の関係を記録する。

[0060] [命令解釈部の関数呼び出し]

次に、図9を用いて、図7のS44の処理の例を説明する。図7のS44において、命令解釈部13は、関数ポインターの型集合の要素それぞれに対して、依存関係の解決を行いつつ、呼び出し処理を行い、呼び出し先としてキュー16に追加する。

[0061] ここで、命令解釈部13は、関数ポインターと引数列間の依存関係を、例えば、Least Common Ancestorアルゴリズムを用いて解析する。そして、命令解釈部13は、上記の解析により特定された先祖変数の型集合の中身を、射影機構atを用いて順に1つずつ射影していくことで、依存関係に応じた候補を絞り込む。これにより、命令解釈部13は、仮想メソッドの呼び出しにおけるthisポインターの参照先を正確に列挙することができる。

[0062] なお、ここでの呼び出し処理とは、引数列の型集合を、呼び出し先の仮引数へ代入する処理と、戻り値を受け取る場合は、呼び出し先の戻り値を代入する処理のことである。

[0063] なお、図9に示す処理において、命令解釈部13への入力は、代入先 v （オプション）、関数ポインター f 、引数列 $a[1], \dots, a[n]$ であり、命令解釈部13からの出力は、型集合（TypeSet）であるものとして説明する。

[0064] まず、命令解釈部13は、関数ポインター f の型集合TypeSet(f)を文脈15から取得し、関数ポインターかどうか型検査し、型式をすべて展開する（S441）。そして、命令解釈部13は、TypeSet(f)中のグローバル関数へのポインター型 F に対して、以下に示すS443～S446の処理を繰り返す（S442）。

[0065] まず、命令解釈部13は、引数列 $a[1], \dots, a[n]$ の型集合を文脈16から取得し、それぞれを呼び出し先の仮引数の型集合へ統合する（S443）。ここで、代入先 v が存在する場合（S444でYes）、命令解釈部13は、TypeSet(v)を呼び出し先の戻り値の型集合と統合する（S445）。そして、命令解釈部13は、呼び出し先 F をキュー16へ追加する（S446）。一方、代入先 v が存在しない場合（S444でNo）、S445をスキップして、呼び出し先 F をキュー16へ追加する（S446）。

[0066] このようにすることで命令解釈部13は、関数ポインターの型集合の要素それぞれに対して、依存関係の解決を行いつつ、呼び出し処理を行い、呼び出し先としてキュー16に追加する。

[0067] [動作例]

次に、図10を用いて検知装置10の動作例を説明する。一般にC++のプログラムにおいては、多様なデータを扱う場合は、クラスObjectを継承する形でArrayやNumberなどのデータを表現する。検知装置10は、ユーザからデータを受け取った後、期待する型かどうかをC++のdynamic_castを用いて、確認する処理を挿入し、型の安全性を確認する

[0068] 例えば、検知装置10は、図10に示すソースコードを解釈すると、NULLでない分岐(A: キャスト可)では、arrの型集合のうちNULLでない型のみを残

し、NULLな分岐(B: キャスト不可)では、NULLな型のみを残す（（１）条件分岐によるふるい落とし）。

[0069] また、検知装置１０が、分岐中の変数のみを操作すると、ありえない組み合わせが発生する可能性があるため（例えば、(A)でobj: Number*, arr: Array*等）、依存するobjの型集合も計算する。

[0070] また、検知装置１０は、C++の継承と仮想メソッドを解析し、様々な型に応じた処理へ正しく振り分けることができる。また、検知装置１０は、オブジェクトから仮想メソッドのポインターを取り出し、引数でオブジェクトを渡す。例えば、検知装置１０は、objの型を１つずつ選び、仮想メソッドへのポインターを列挙し、処理を振り分けることで、依存関係を考慮した、型集合の計算を正確に行うことができる。

[0071] [システム構成等]

また、図示した各部の各構成要素は機能概念的なものであり、必ずしも物理的に図示のように構成されていることを要しない。すなわち、各装置の分散・統合の具体的形態は図示のものに限られず、その全部又は一部を、各種の負荷や使用状況等に応じて、任意の単位で機能的又は物理的に分散・統合して構成することができる。さらに、各装置にて行われる各処理機能は、その全部又は任意の一部が、CPU及び当該CPUにて実行されるプログラムにて実現され、あるいは、ワイヤードロジックによるハードウェアとして実現され得る。

[0072] また、前記した実施形態において説明した処理のうち、自動的に行われるものとして説明した処理の全部又は一部を手動的に行うこともでき、あるいは、手動的に行われるものとして説明した処理の全部又は一部を公知の方法で自動的に行うこともできる。この他、上記文書中や図面中で示した処理手順、制御手順、具体的名称、各種のデータやパラメータを含む情報については、特記する場合を除いて任意に変更することができる。

[0073] [プログラム]

前記した検知装置１０は、パッケージソフトウェアやオンラインソフトウ

ウェアとして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、情報処理装置を各実施形態の検知装置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、デスクトップ型又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) 等の移動体通信端末、さらには、PDA (Personal Digital Assistant) 等の端末等がその範疇に含まれる。

[0074] また、検知装置10は、ユーザが使用する端末装置をクライアントとし、当該クライアントに上記の処理に関するサービスを提供するサーバ装置として実装することもできる。この場合、サーバ装置は、Webサーバとして実装することとしてもよいし、アウトソーシングによって上記の処理に関するサービスを提供するクラウドとして実装することとしてもかまわない。

[0075] 図11は、検知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、CPU1020を有する。また、コンピュータ1000は、ハードディスクドライバインタフェース1030、ディスクドライバインタフェース1040、シリアルポートインタフェース1050、ビデオアダプタ1060、ネットワークインタフェース1070を有する。これらの各部は、バス1080によって接続される。

[0076] メモリ1010は、ROM (Read Only Memory) 1011及び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バインタフェース1030は、ハードディスクドライブ1090に接続される。ディスクドライバインタフェース1040は、ディスクドライブ1100に接続される。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、ディスクドライブ1100に挿入される。シリアルポートインタフェース1050は、例えばマウス1110、キーボード1120に接続される。ビデオアダプタ1060は、例えばディスプレイ1130に接続される。

- [0077] ハードディスクドライブ1090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、プログラムデータ1094を記憶する。すなわち、上記の検知装置10が実行する各処理を規定するプログラムは、コンピュータにより実行可能なコードが記述されたプログラムモジュール1093として実装される。プログラムモジュール1093は、例えばハードディスクドライブ1090に記憶される。例えば、検知装置10における機能構成と同様の処理を実行するためのプログラムモジュール1093が、ハードディスクドライブ1090に記憶される。なお、ハードディスクドライブ1090は、SSDにより代替されてもよい。
- [0078] また、上述した実施形態の処理で用いられる各データは、プログラムデータ1094として、例えばメモリ1010やハードディスクドライブ1090に記憶される。そして、CPU1020が、メモリ1010やハードディスクドライブ1090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に読み出して実行する。
- [0079] なお、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1090に記憶される場合に限らず、例えば着脱可能な記憶媒体に記憶され、ディスクドライブ1100等を介してCPU1020によって読み出されてもよい。あるいは、プログラムモジュール1093及びプログラムデータ1094は、ネットワークされたーク（LAN（Local Area Network）、WAN（Wide Area Network）等）を介して接続他のコンピュータに記憶されてもよい。そして、プログラムモジュール1093及びプログラムデータ1094は、他のコンピュータから、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。

符号の説明

- [0080] 10 検知装置
11 ローダー部
12 トレーサー部
13 命令解釈部

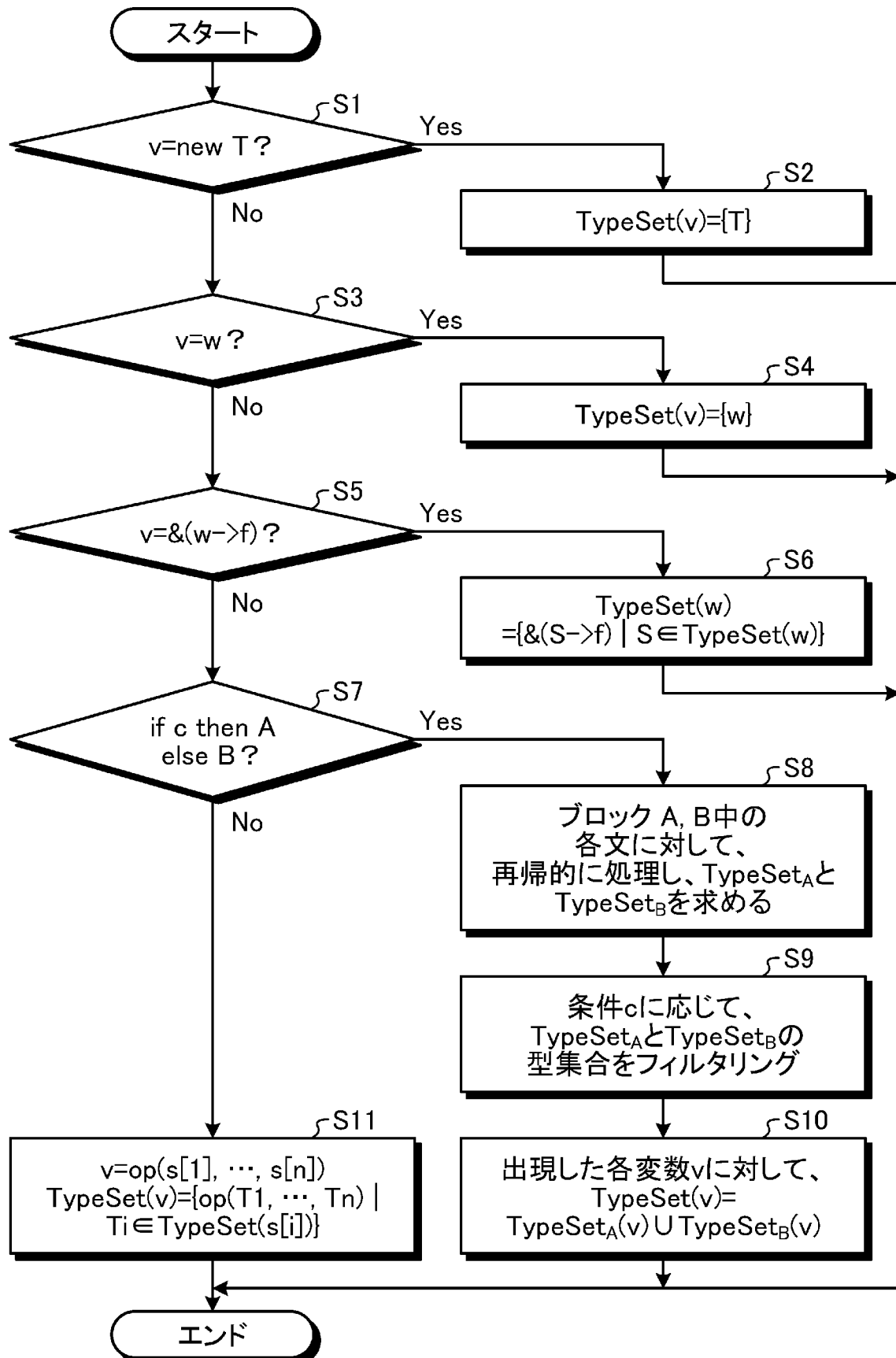
1 4 型検査部

1 5 文脈

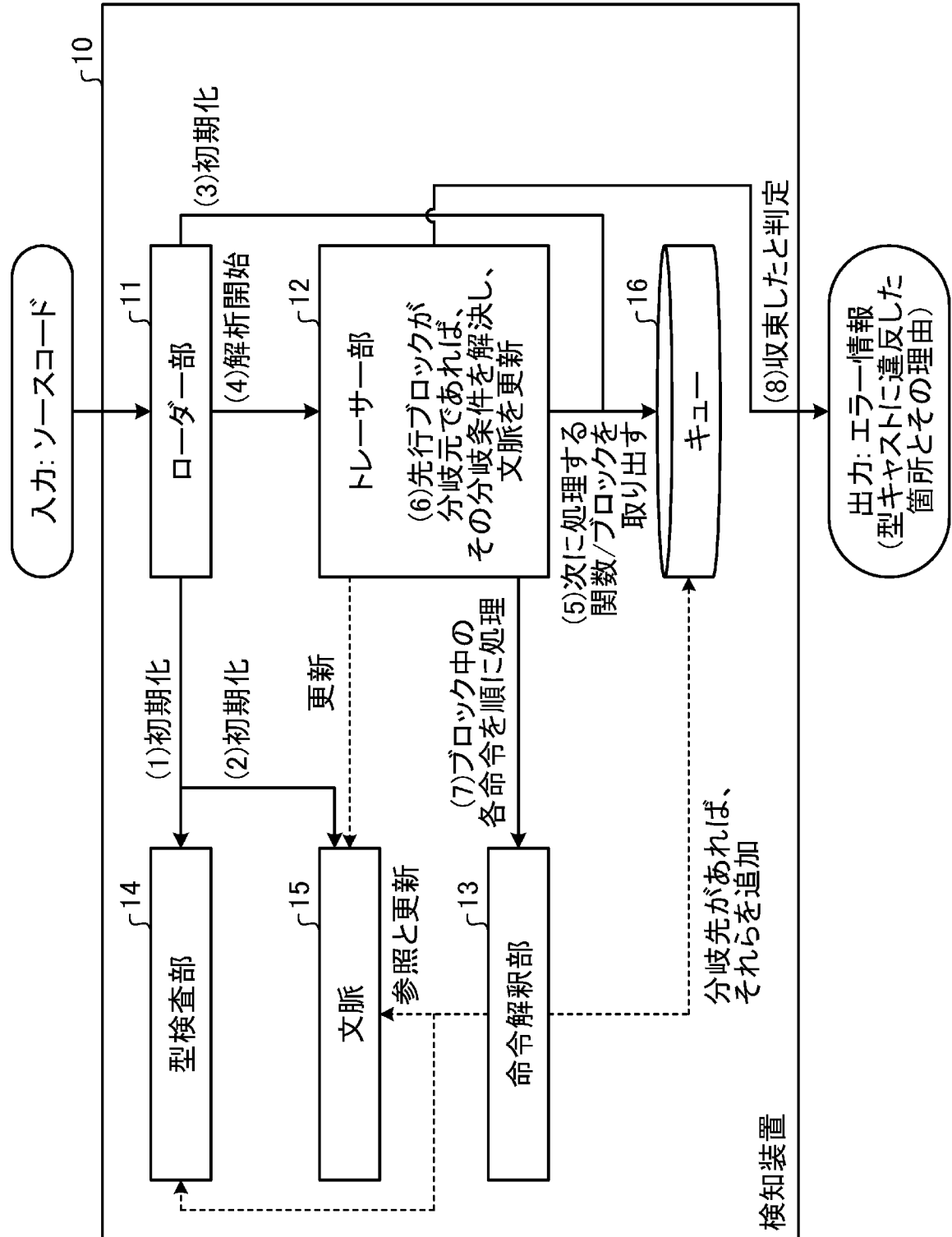
請求の範囲

- [請求項1] プログラムのソースコードを読み込むローダー部と、
 前記読み込んだソースコードに抽象解釈を適用することにより、前記ソースコードの条件分岐で取り得る変数の型を収集し、依存型理論により前記変数間の依存関係を特定し、前記依存関係と前記条件分岐で用いられる条件を用いて、前記収集した型の集合のうち前記条件分岐先における型の集合のフィルタリングを行う命令解釈部と、
 を備えることを特徴とする検知装置。
- [請求項2] 前記型の集合の各要素がキャスト先の具体型へキャスト可能か否かを判定する型検査を行い、前記型検査の結果を出力する型検査部
 をさらに備えることを特徴とする請求項1に記載の検知装置。
- [請求項3] 前記命令解釈部は、
 前記変数間の依存関係を特定する際、さらに、関数ポインターと引数列との間の依存関係を特定する
 ことを特徴とする請求項1に記載の検知装置。
- [請求項4] プログラムのソースコードを読み込む工程と、
 前記読み込んだソースコードに抽象解釈を適用することにより、前記ソースコードの条件分岐で取り得る変数の型を収集し、依存型理論により前記変数間の依存関係を特定し、前記依存関係と前記条件分岐で用いられる条件を用いて、前記収集した型の集合のうち、前記条件分岐先における型の集合のフィルタリングを行う工程と、
 をコンピュータに実行させることを特徴とする検知プログラム。

[図1]



[図2]



[図3]

```

Type :=
  "external" ConcreteType // 追跡対象外(型検査しない)
  | TypeOwned // 追跡対象(所有型)
  | TypeExpr // 追跡対象(型式)

TypeOwned :=
  ConcreteType
  | "(" Value "as" Type ")" // 定数型
  | "fn" Name "*" // グローバル関数へのポインター型
  | "&" TypeOwned // 参照型
  | "poison" "(" NonNegativeInteger ")"
  // 未初期化領域の型

TypeExpr :=
  RegPath // 変数名
  | "*" TypeExpr // デリファレンス操作
  | "&" TypeExpr // アドレス参照操作
  | "&" "(" TypeExpr "as" ConcreteType ")"
  | "[" Type "]" // フィールドアドレス参照操作
  | TypeCond
  | "filter" TypeCond // 条件によるフィルター操作

TypeCond :=
  "!"? "is_null" "(" TypeExpr ")" // NULLチェック述語

```

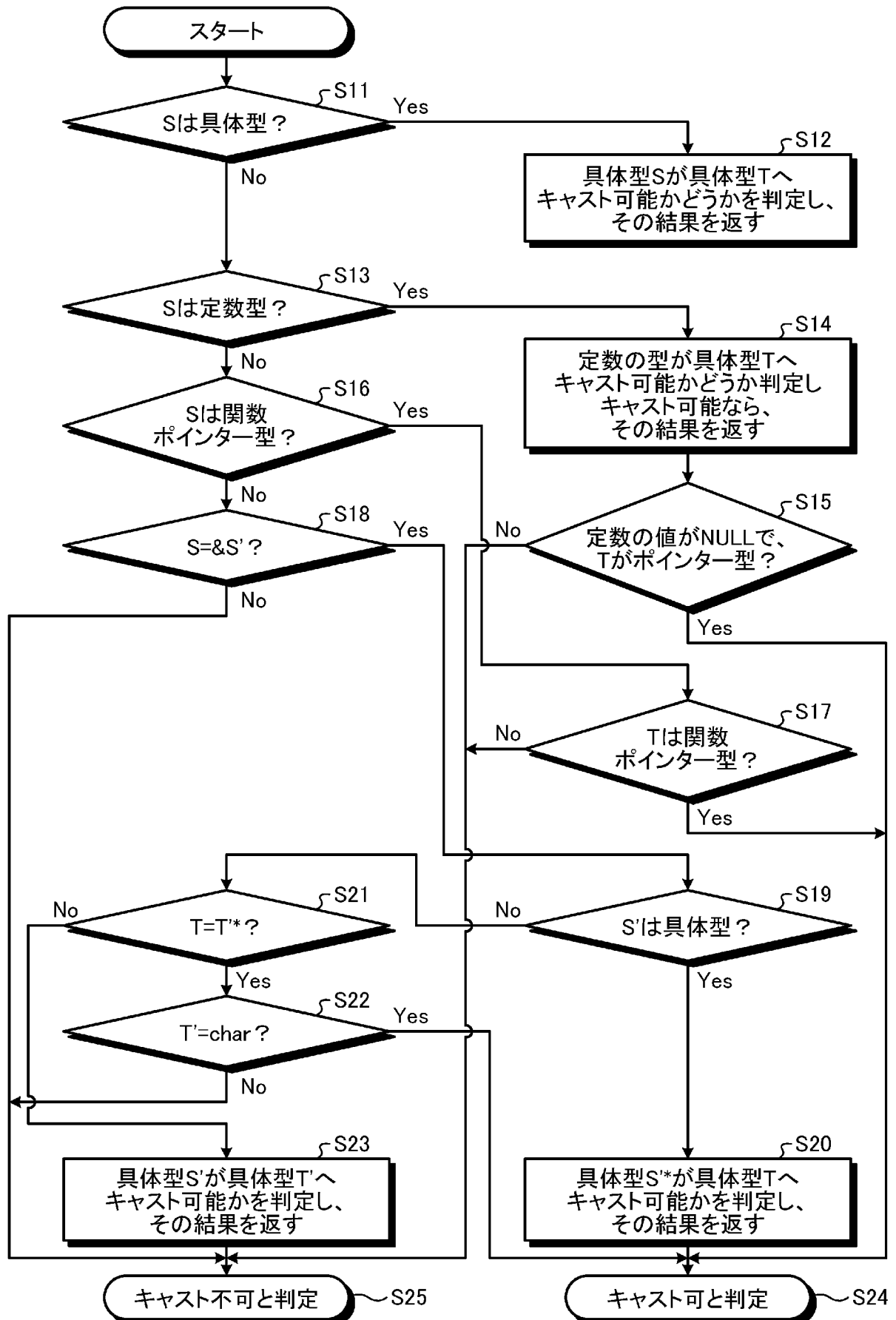
[図4]

```
RegPath := RegName "@" Loc  
  
RegName :=  
  Ret // 戻り値  
  | LocalName // ローカル変数  
  | GlobalName // グローバル変数  
  | LocalName "@" Loc  
    // Locで初期化されたトレース対象のオブジェクト  
  | RegName "[" NonNegativeInteger "]" // フィールド  
  
Loc := FuncName ":" BlockName ":" InsnIdx
```

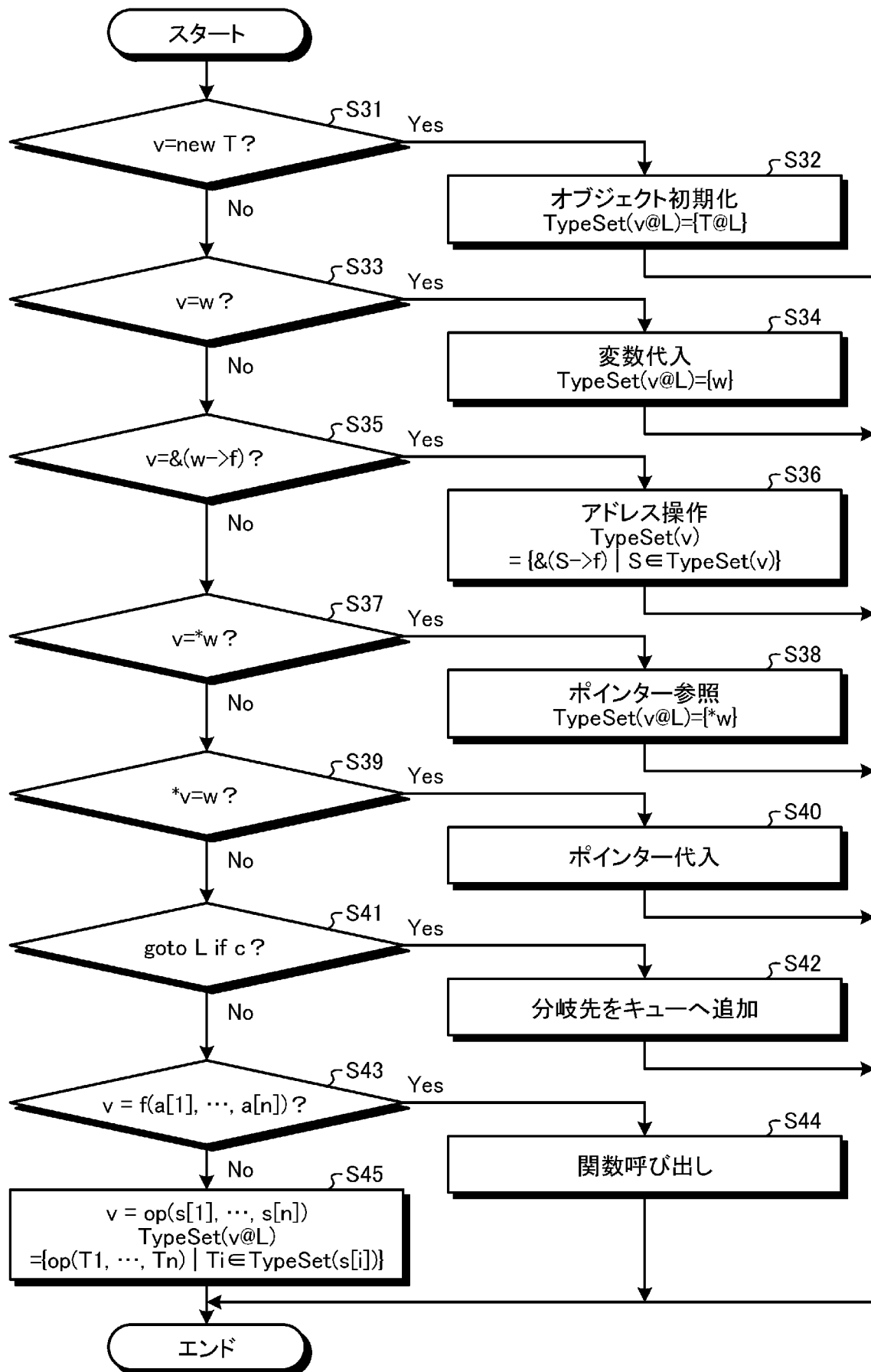
[図5]

```
ProjectorKind :=  
  "at" Loc // 一命令でのみ有効  
  | "true_branch" BlockName // 真の分岐先でのみ有効  
  | "false_branch" BlockName // 偽の分岐先でのみ有効  
  | "since" Loc // それ以降の命令でのみ有効
```

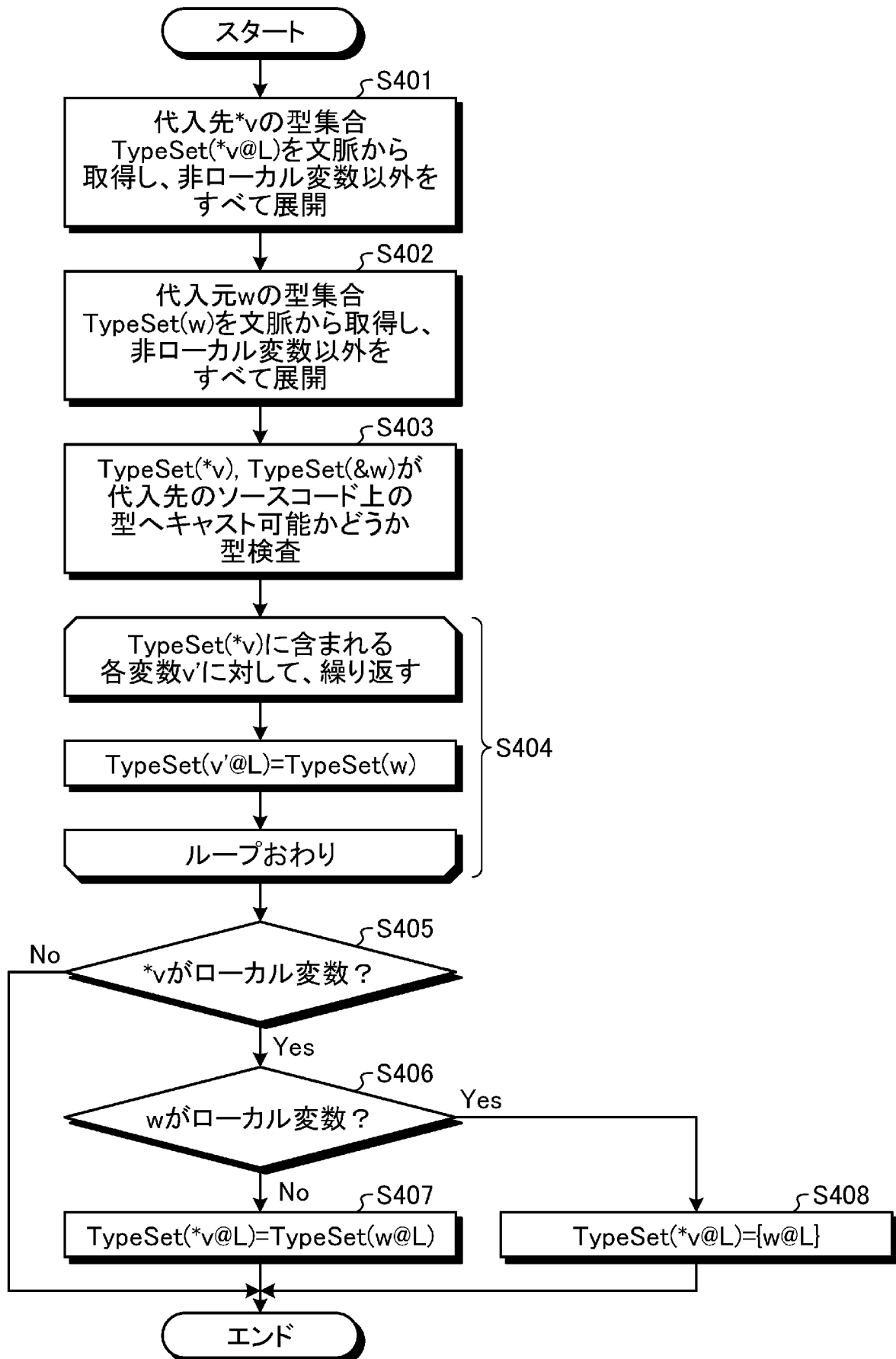
[図6]



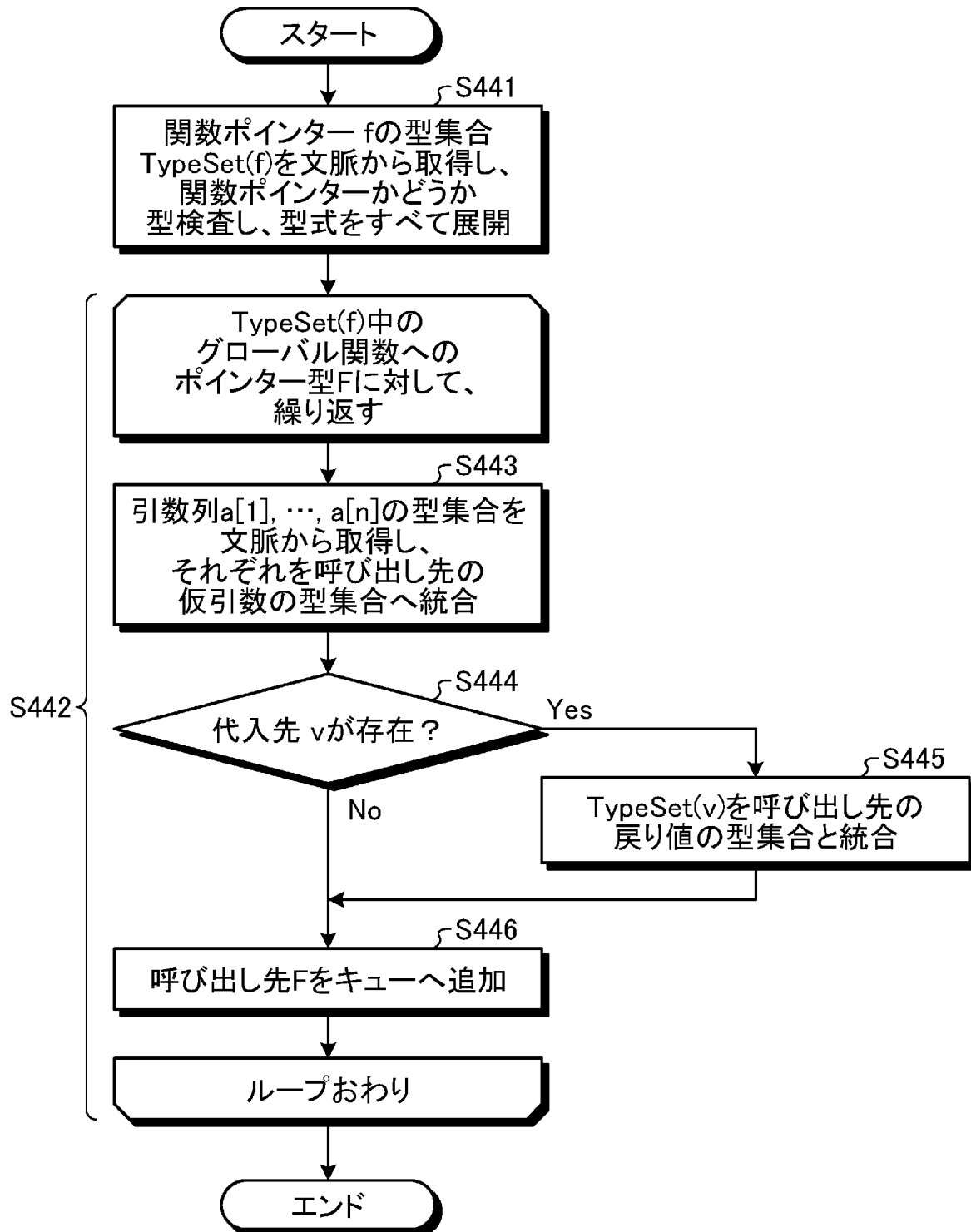
[図7]



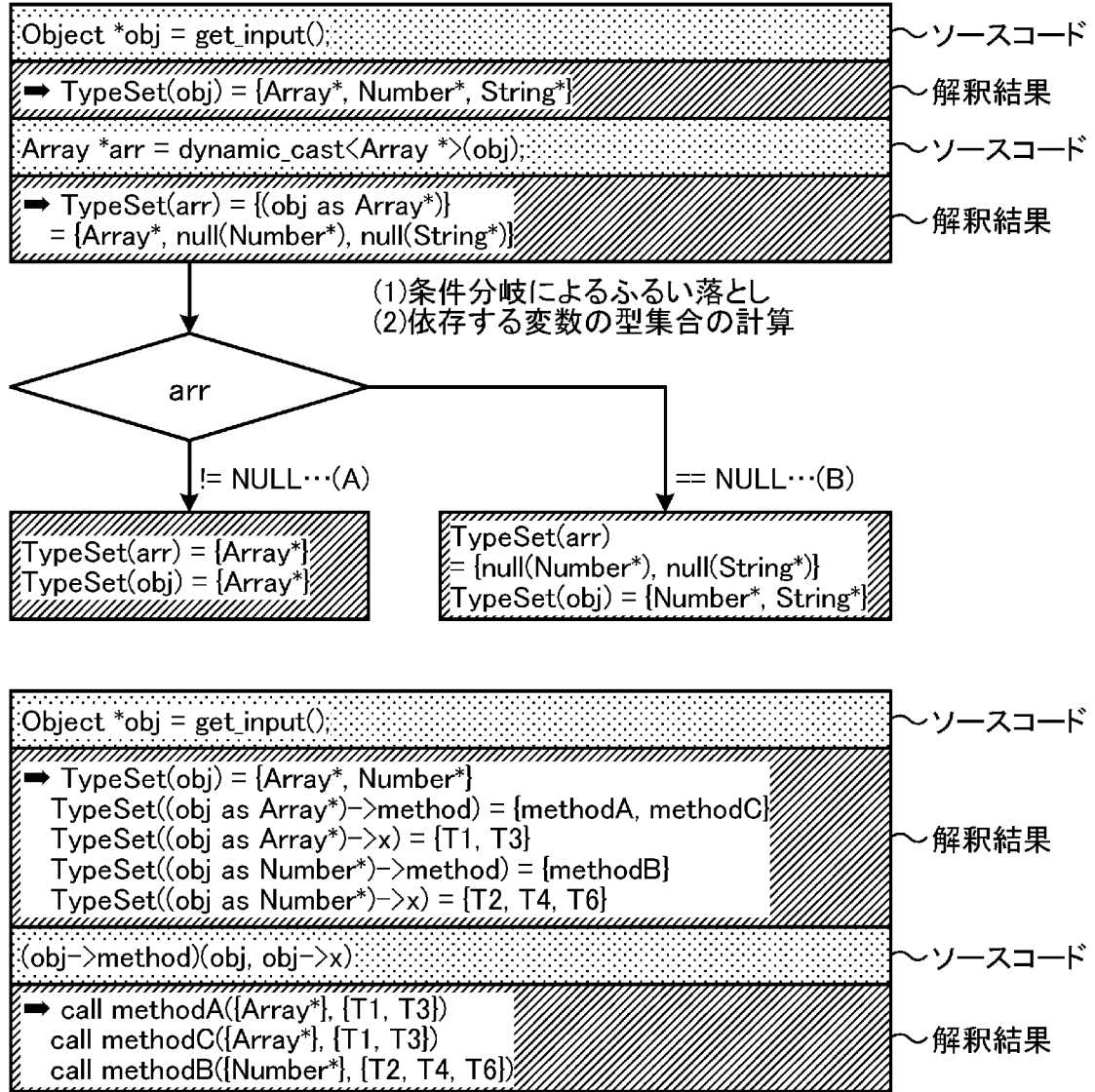
[図8]



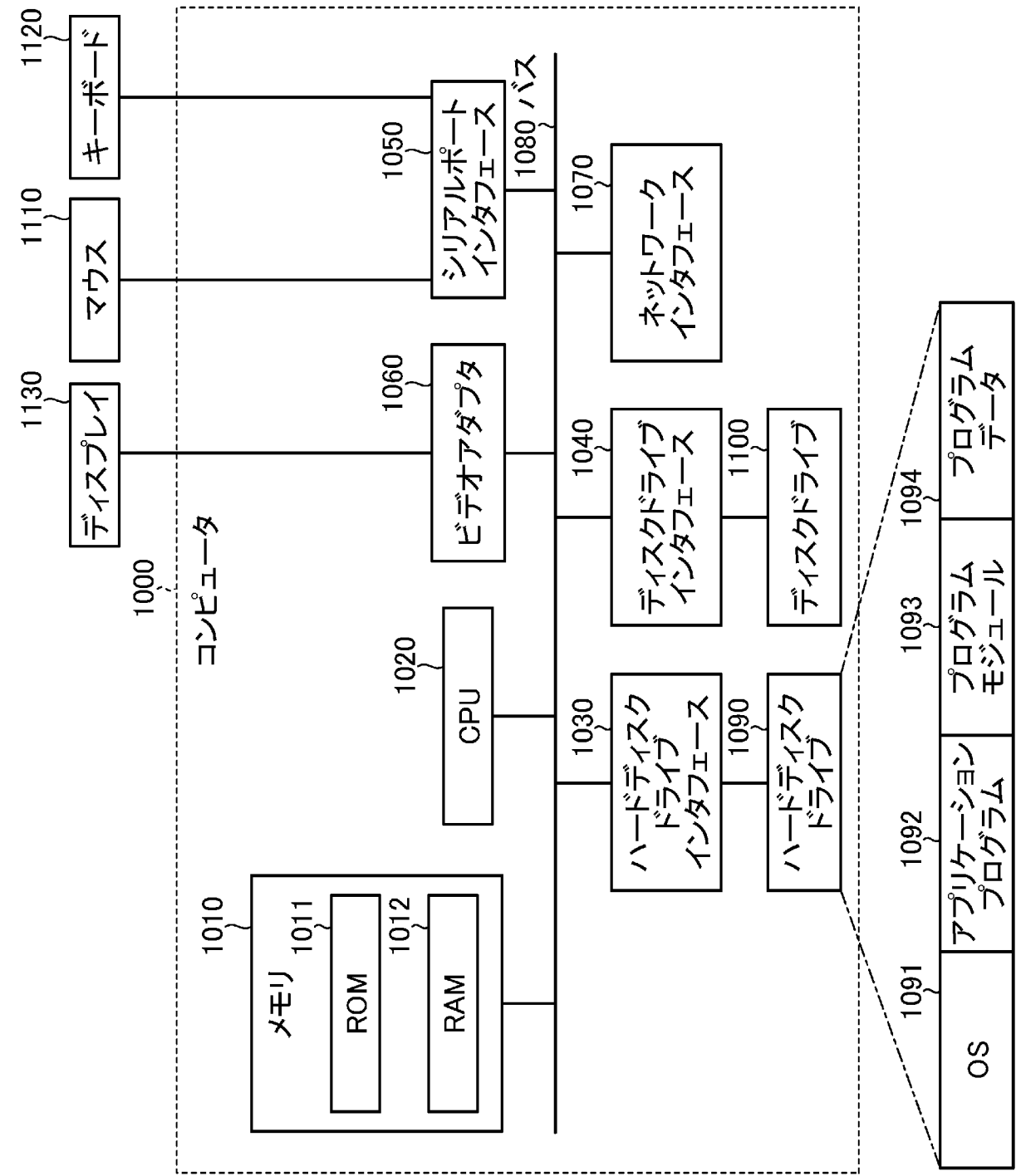
[図9]



[図10]



[図11]



INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2020/038543

A. CLASSIFICATION OF SUBJECT MATTER

Int. Cl. G06F11/36 (2006.01) i

FI: G06F11/36 108

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)

Int. Cl. 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 1971-2020

Registered utility model specifications of Japan 1996-2020

Published registered utility model applications of Japan 1994-2020

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	EICHBERG, M., et al. Hidden truths in dead software paths., In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, 2015, in particular, 3.2 The Abstract Interpretation Analysis, fig. 2	1-4
A	FERRARA, P. Static type analysis of pattern matching by abstract interpretation., In Formal Techniques for Distributed System., 2010, in particular, 1.1 Contribution, 4.2 DT: Dynamic Typing, 5. Experimental Results	1-4



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

26.11.2020

Date of mailing of the international search report

08.12.2020

Name and mailing address of the ISA/

Japan Patent Office

3-4-3, Kasumigaseki, Chiyoda-ku,

Tokyo 100-8915, Japan

Authorized officer

Telephone No.

INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2020/038543

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	UNNO, H., et al., Dependent type inference with interpolants., In Proceedings of the 11th ACM SIGPLAN conference on Principles and practice of declarative programming, 2009, in particular, Abstract	1-4

A. 発明の属する分野の分類（国際特許分類（IPC）） G06F 11/36(2006.01)i FI: G06F11/36 108		
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）） 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 - 2020年 日本国実用新案登録公報 1996 - 2020年 日本国登録実用新案公報 1994 - 2020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	EICHBERG, M., et al., Hidden truths in dead software paths., In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, 2015 特に、3.2 The Abstract Interpretation Analysis, 図2	1-4
A	FERRARA, P., Static type analysis of pattern matching by abstract interpretation., In Formal Techniques for Distributed System., 2010 特に、1.1 Contribution, 4.2 DT: Dynamic Typing, 5 Experimental Results	1-4
A	UNNO, H., et al., Dependent type inference with interpolants., In Proceedings of the 11th ACM SIGPLAN conference on Principles and practice of declarative programming, 2009 特に、Abstract	1-4
☐ 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 26. 11. 2020		国際調査報告の発送日 08. 12. 2020
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） 石川 雄太郎 5B 5090 電話番号 03-3581-1101 内線 3545