(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6528465号
(P6528465)
(45)発行日　令和1年6月12日(2019.6.12)　(24)登録日　令和1年5月24日(2019.5.24)

(51)Int.Cl.　F I
G06F 11/36 (2006.01)　G06F 11/36 108

請求項の数 15 (全 15 頁)

(21)出願番号　特願2015-41801(P2015-41801)
(22)出願日　平成27年3月3日(2015.3.3)
(65)公開番号　特開2015-204109(P2015-204109A)
(43)公開日　平成27年11月16日(2015.11.16)
審査請求日　平成30年1月15日(2018.1.15)
(31)優先権主張番号　14/253,342
(32)優先日　平成26年4月15日(2014.4.15)
(33)優先権主張国　米国(US)

(73)特許権者　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　100099759
弁理士　青木　篤
(74)代理人　100119987
弁理士　伊坪　公一
(74)代理人　100133835
弁理士　河野　努
(74)代理人　100135976
弁理士　宮本　哲夫

最終頁に続く

(54)【発明の名称】ソフトウェアテストのためのシンボリック実行における状態パラメータ化

(57)【特許請求の範囲】
【請求項１】
パラメトリック状態を使用して実行可能コードをシンボリック実行するコンピュータ実行方法であって、
記憶媒体にアクセス可能なプロセッサが、当該記憶媒体に記憶された前記実行可能コードに関連する、シンボリックパラメータを用いて表される複数の実行状態において共有される共有状態要素と前記複数の実行状態において互いに相異する相異状態要素とを特定し、
前記プロセッサが、状態抽象化を用いて前記相異状態要素を表し、
前記プロセッサが、前記状態抽象化に基づいて、パラメトリック制約を有し、かつ前記複数の実行状態の何れかを含む実行状態の群を無損失に表すパラメトリック状態を定義し、前記実行状態の群における実行状態が前記パラメトリック状態のインスタンスであり、
前記プロセッサが、前記パラメトリック状態を使用して前記実行可能コードをシンボリック実行する、
ことを含むコンピュータ実行方法。
【請求項２】
状態抽象化を用いて前記相異状態要素を表すことは、
前記相異状態要素の少なくとも一部を簡約化し、
少なくとも一部が簡約化された前記相異状態要素を前記記憶媒体に格納された共通パターンとマッチングすることにより、前記相異状態要素における共通パターンを特定する、

ことを含む請求項１に記載のコンピュータ実行方法。
【請求項３】
前記パラメトリック状態を定義することは、
前記特定された共通パターンに基づいて、前記パラメトリック状態を定義する、
ことを含む請求項２に記載のコンピュータ実行方法。
【請求項４】
前記実行状態の群は、ループの繰返を表し、
前記ループのループ本体から前記パラメトリック状態のパス条件と値ストアとを定義することをさらに含み、
前記パラメトリック状態を定義することは、さらに前記ループのループヘッダに基づいて、前記パラメトリック制約を定義することを含む、
請求項１に記載のコンピュータ実行方法。
【請求項５】
前記パラメトリック状態を使用して前記実行可能コードをシンボリック実行することは、
前記パラメトリック状態を用いてテストケースを生成し、
前記テストケースを用いて、前記実行状態の群においてバグチェックを実行する、
ことをさらに有する、
請求項１に記載のコンピュータ実行方法。
【請求項６】
前記プロセッサが、前記実行状態の群を前記パラメトリック状態に置換して、前記実行可能コードに関連する前記実行状態の数を低減させる、
ことをさらに有する請求項１に記載のコンピュータ実行方法。
【請求項７】
前記プロセッサが、前記実行状態の群を前記パラメトリック状態に置換して、前記シンボリック実行によって消費されるメモリ資源を低減させる、
請求項６に記載のコンピュータ実行方法。
【請求項８】
パラメトリック状態を使用して実行可能コードをシンボリック実行するためのプロセッサ実行可能命令であって、記憶媒体にアクセス可能なプロセッサによって実行されたときに、
前記記憶媒体に記憶された前記実行可能コードに関連する、シンボリックパラメータを用いて表される複数の実行状態において共有される共有状態要素と前記複数の実行状態において互いに相異する相異状態要素とを特定し、
前記プロセッサが、状態抽象化を用いて前記相異状態要素を表し、
前記プロセッサが、前記状態抽象化に基づいて、パラメトリック制約を有し、かつ前記複数の実行状態の何れかを含む実行状態の群を無損失に表すパラメトリック状態を定義し、前記実行状態の群における実行状態が前記パラメトリック状態のインスタンスであり、
前記プロセッサが、前記パラメトリック状態を使用して前記実行可能コードをシンボリック実行する、
ことを前記プロセッサに実行させるための前記命令を格納する一又は複数の非一時的なコンピュータ可読媒体。
【請求項９】
状態抽象化を用いて前記相異状態要素を表すための前記命令は、
前記相異状態要素の少なくとも一部を簡約化し、
少なくとも一部が簡約化された前記相異状態要素を前記記憶媒体に格納された共通パターンとマッチングすることにより、前記相異状態要素における共通パターンを特定するための命令を含む請求項８に記載の一又は複数の非一時的なコンピュータ可読媒体。
【請求項１０】
前記パラメトリック状態を定義するための前記命令は、前記特定された共通パターンに

基づいて、前記パラメトリック状態を定義するための命令を含む、
請求項８に記載の一又は複数の非一時的なコンピュータ可読媒体。
【請求項１１】
前記実行状態の群は、ループの繰返を表し、
前記ループのループ本体から前記パラメトリック状態のパス条件と値ストアとを定義するための命令をさらに含み、
前記パラメトリック状態を定義するための前記命令は、さらに前記ループのループヘッダに基づいて、前記パラメトリック制約を定義するための命令を含む、
請求項８に記載の一又は複数の非一時的なコンピュータ可読媒体。
【請求項１２】
前記パラメトリック状態を使用して前記実行可能コードをシンボリック実行するための前記命令は、
前記パラメトリック状態を用いてテストケースを生成し、
前記テストケースを用いて、前記実行状態の群においてバグチェックを実行する
ための命令をさらに含む、
請求項８に記載の一又は複数の非一時的なコンピュータ可読媒体。
【請求項１３】
前記プロセッサが、前記実行状態の群を前記パラメトリック状態に置換して、前記実行可能コードに関連する前記実行状態の数を低減させるための命令をさらに含む、
請求項８に記載の一又は複数の非一時的なコンピュータ可読媒体。
【請求項１４】
前記実行状態の群を前記パラメトリック状態に置換するための前記命令は、実行されたときに、前記シンボリック実行によって消費されるメモリ資源を低減させる、
請求項１３に記載の一又は複数の非一時的なコンピュータ可読媒体。
【請求項１５】
前記実行状態の群を前記パラメトリック状態に置換するための前記命令は、実行されたときに、前記シンボリック実行のための前記プロセッサの計算量を低減させる、
請求項１３に記載の一又は複数の非一時的なコンピュータ可読媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本開示は、ソフトウェアテストのためのシンボリック実行、特に、シンボリック実行における状態パラメータ化に関する。
【背景技術】
【０００２】
ソフトウェアプログラムの遍在的存在が日常生活の多くの側面に浸透するにつれて、堅牢かつ信頼性のある実行可能プログラムを提供するためのソフトウェアコードの検査及び検証によるソフトウェアテストは、ソフトウェア開発プロセスの重要な部分となってきている。従来、面倒で難しく、かつ多くの場合、ソフトウェアコードを不十分にしか範囲をカバーできない手作業のソフトウェアテストによって、ソフトウェアの品質を保証していた。最近では、正規のソフトウェアテストのための自動化技術が開発されている。そのような技術の一つがシンボリック実行である。
【０００３】
シンボリック実行は、ソフトウェアプログラムへの入力をシンボリック変数として取り扱う非明示的な状態のモデル検査技術である。シンボリック実行は、シンボリック変数を用いたソフトウェアプログラムにおける有限パスの実行による複雑な方程式の作成と、そして決定手順として一般的に知られているソルバを用いて、その複雑な方程式を解いて、存在するならばエラーシナリオを得ることとを含んでいる。明示的な状態のモデル検査とは対照的に、シンボリック実行は、可能入力値と、分析中のソフトウェアプログラムにおける可能入力値の可能使用ケースとを求めることができる。シンボリック実行を使用して

、テスト中のソフトウェアについて構造的に広範囲なテスト入力を自動生成することができる。しかしながら、シンボリック実行中、計算資源の高消費のせいでシンボリック実行の適用性を制限し得る非常に複数の実行パス及び／又は実行状態が生成される可能性がある。

【発明の概要】

【０００４】

一つの観点において、パラメトリック状態を使用して実行可能コードをシンボリック実行する開示の方法は、実行可能コードに関連する実行状態において共有状態要素と相異状態要素とを特定することを含む。その方法は、状態抽象化を用いて相異状態要素を表すことを含む。その方法は、パラメトリック制約を有し、かつ実行状態の群を無損失に表すパラメトリック状態を状態抽象化に基づいて定義することを含む。実行状態の群における実行状態は、パラメトリック状態のインスタンスであってもよい。その方法は、パラメトリック状態を使用することを含む、記憶媒体へのアクセスを有するプロセッサを使用して実行可能コードをシンボリック実行することを含む。

【０００５】

パラメトリック状態を使用して実行可能コードをシンボリック実行するための付加的開示の観点は、記憶媒体へのアクセスを有するプロセッサを含むシステム、及びプロセッサによって実行可能な命令を含む非一時的なコンピュータ可読媒体を含んでもよい。

【図面の簡単な説明】

【０００６】

【図１】シンボリック実行システムの実施形態の選択要素のブロック図である。

【図２】パラメータ化処理の実施形態の選択要素のブロック図である。

【図３】パラメータ化状態を併合する方法の実施形態の選択要素を示すフローチャートである。

【図４Ａ】パラメータ化状態の併合例の実施形態の選択要素の制御フローグラフである。

【図４Ｂ】パラメータ化状態の併合例の実施形態の選択要素の制御フローグラフである。

【図４Ｃ】パラメータ化状態の併合例の実施形態の選択要素の制御フローグラフである。

【図４Ｄ】パラメータ化状態の併合例の実施形態の選択要素の制御フローグラフである。

【図５】シンボリック実行システムの実施形態の選択要素のブロック図である。

【図６】パラメータ化状態を使用してシンボリック実行する方法の実施形態の選択要素を示すフローチャートである。

【発明を実施するための形態】

【０００７】

以下の明細書では、開示の主題の説明を容易にするために、例示的に詳細を記載している。しかしながら、開示の実施形態が例示的なものであり、全ての可能な実施形態を網羅するものではないことは、当該技術分野における当業者にとって自明である。

【０００８】

本開示を通じて、参照番号をハイフンで結んだ形式は、特定の例の要素を指し、参照番号をハイフンで結ばない形式は、要素を一般的又は集合的に指している。したがって、一例として（図示せず）、ウィジェット“１２－１”は、ウィジェットクラスの一例を指し、ウィジェットクラスを集合的にウェジェット“１２”と呼ぶことができ、またウィジェットクラスのいずれかを一般的にウェジェット“１２”と呼ぶことができる。図面及び明細書において、同様の番号は、同様の要素を表すことを意図している。

【０００９】

本明細書において、“シンボリック実行”は、具体的入力ではなくシンボリック入力を用いてテスト中のソフトウェアの実行をシミュレート（又はエミュレート）するソフトウェアテストについての方法及び動作を指す。シンボリック実行は、シンボリック式を用いてテスト中のソフトウェアにおけるシンボリック入力の効果を計算することができる。シンボリック実行を使用して、各種ソフトウェアプログラム及び／又はモジュールを正規にテストし、かつ検査することができる。本明細書において、“エグゼキュータ”は、“テ

スト中のソフトウェア（SUT）”においてシンボリック実行を行う実行エンジンを指す。テスト中のソフトウェアは、完全なアプリケーションプログラムであってもよいし、一又は複数の選択されたコードモジュールであってもよい。

【００１０】

特定の実施形態において、シンボリック実行は、論理式の結合として定義可能なパス条件を用いて、テスト中のソフトウェアにおける各実行パスを特徴付ける。パス条件における各論理式は、テスト中のソフトウェアの個別パスのシンボリック実行中に生成される一の分岐決定を示すことができる。シンボリック実行が終了すると、複数のパス条件が生成され、各パス条件は、シンボリック入力に関連するプログラムコードの実行可能な実行パスに対応している。テスト中のソフトウェアが特定の定義された実行パスに沿って実行されるように、パス条件に対する解をテスト入力として使用することができる。特定の実施形態では、SMT（充足可能性モジュロ理論）ソルバのような決定手順を使用して、パス条件に対する解を見つけ出して偽パスを削除する。そのような解がテスト中のソフトウェアにおいて実際の実行パスについて得られた場合には、その後、プログラム又はモジュールの全数テストが可能となり得る。加えて、シンボリック実行中には、メモリへの境界を越えたアクセス、ゼロ除算、及び特定種類のユーザ定義のアサーションのような、いわゆる“サニティ特性”をチェックすることができる。

【００１１】

シンボリック実行は、テスト中のソフトウェアを検査することができ、従来のテストアプローチよりも大きなプログラム動作のテスト範囲を実現することができるが、計算集約的であり、また、シンボリック実行の経済的実現可能性に悪影響を与え得る相当量の、プロセッサ時間、メモリ空間、及びプロセッサ数などといった計算資源を消費する可能性がある。例えば、シンボリック実行は、テスト中のソフトウェアがエグゼキュータによる処理について計算困難なシンボリックモデルをもたらし得るパス急増及び／又は状態急増の問題に悩まされる可能性がある。パス急増／状態急増の問題の結果として、エグゼキュータは、複数の実行パス及び関連する状態を処理する可能性があり、合理的な時間内に一連の有用なテストケースを出力できない可能性がある。このため、あまりに多くの実行パス及び関連する状態は、シンボリック実行についての主要な障害となる可能性があり、また、テスト中の特定のソフトウェアにシンボリック実行の拡張性及び／又は適用性の好ましくない制限をもたらす可能性がある。

【００１２】

本明細書に示されるように、テスト中のソフトウェアの実行状態及び対応するパスを、状態を表す複数のノード及びパス条件を表すノード間の複数のエッジを含む状態ツリーを用いて表すことができる。状態ツリーにおける各ノードは、命令カウンタ、パス条件、及び値ストアを包含する実行状態を表すことができる。各エッジは、ノードとノードとを接続する制限を表すことができる。

【００１３】

米国特許公開２０１２／０３１１５４５号公報では、いかなる中間の実行ジャンプ又はジャンプターゲットを欠く基本コードブロックを有するテスト中のソフトウェアの実行パスの数と状態の数とを無損失に低減する方法及びシステムを用いて、シンボリック実行中の過度の実行パスに関連する特定の問題に対処している。本明細書においてさらに詳細に説明するように、本発明の発明者は、パラメータ化方法を用いた状態併合技術を提供可能なシンボリック実行におけるパラメータ化状態を管理する方法及びシステムを発見した。本明細書に記載のシンボリック実行中のパラメータ化状態を管理する方法及びシステムは、シンボリック実行の資源消費を改善するために有効であり、したがって、自動ソフトウェアテストにとって有益なシンボリック実行を用いた有益なテストケースの、より迅速かつ効率的な生成をもたらすことができる。本明細書に記載の方法及びシステムは、シンボリック実行に使用される以前の改善及び解とともに使用可能であることに留意されたい。

【００１４】

本明細書で使用する場合、シンボリック実行におけるパラメトリック状態（“p状態”

）は、シンボリックパラメータを用いて表される具体的状態（“c状態”）群を定義することができる。このように、p状態を使用して、個々のc状態の羅列を回避することによってシンボリック実行における複雑さを低減することができる。p状態は、羅列された一連のc状態の正確な符号化を表すことができる。p状態は、繰返及びケースが互いに独立しているシンボリック実行におけるループ及びスイッチ文（又はケース文）に特に適用可能であってもよい。

【００１５】

ここで、図面を参照すると、図１は、本実施形態のシンボリック実行システム１００の選択要素のブロック図を示している。図示するように、シンボリック実行システム１００は、本明細書に記載されるように、シンボリック実行を行うためのツールの集合を表すことができる。特定の実施形態において、エグゼキュータ１２０のような、シンボリック実行システム１００の少なくとも特定の部分は、シンボリック実行（図５も参照）を行うために、プロセッサによって実行可能な実行可能コード及び／又は命令を表している。図１において、エグゼキュータ１２０は、テスト中のソフトウェアを表す実行可能コード（SUT）１１０においてシンボリック実行を行う実行エンジンを表すことができる。シンボリック実行システム１００は、異なる言語における実行可能コード及び／又は異なる種類又はバージョンのコンパイラを使用してコンパイルされた実行可能コードを含む各種実行可能コード（SUT）１１０について使用可能であることに留意されたい。図１に示すように、エグゼキュータ１２０は、シンボリック実行の出力としてテストケース１５０と統計量１５２とを生成することができる。テストケース１５０は、例えば、バグチェック及び／又は他のランタイムエラーについて、実行可能コード（SUT）１１０を分析及び評価するために使用可能な実際の実行可能コードの結果を表すことができる。特定の実施形態では、エグゼキュータ１２０が実行可能コード（SUT）１１０をシンボリック実行することによって得られるシンボリック式を、例えば、SMTソルバを用いて解くことができる。もしあれば解を使用して、実行可能コード（SUT）１１０をテスト及び検査するためのテストケース１５０を生成することができる。統計量１５２は、実行可能コード（SUT）１１０内の特定の実行パスの範囲を追跡するための統計値のような、テストケース１５０を記述する統計値を表すことができる。

【００１６】

図１に示すように、エグゼキュータ１２０は、ランタイム環境１２２と、オプティマイザ１２４と、エグゼキュータマネージャ１２６と、状態マネージャ１２８と、シンボリック式ハンドラ１３０と、具体的インタプリタ１３０とを含むことができる。ランタイム環境１２２は、特定種類のプログラミング言語又はコンパイラに合わせられていてもよく、実行可能コード（SUT）１１０のランタイム実行のシミュレーションを可能とすることができる。オプティマイザ１２４は、シンボリック実行において使用される状態及び／又はパスを最適化する機能を含むことができる。エグゼキュータマネージャ１２６は、エグゼキュータ１２０の実行環境の管理を担当することができる。状態マネージャ１２８は、c状態及びp状態を含むシンボリック実行における状態のインスタンスを生成及び管理するための機能を含むことができる。シンボリック式ハンドラ１３０を使用して、シンボリック実行におけるシンボリック式を処理して解くことができる。具体的インタプリタ１３０を使用して、シンボリック実行における具体的状態の実行特性を分析することができる。

【００１７】

シンボリック実行システム１００の動作では、パラメトリック状態を使用して、情報を失うことなく一連の具体的状態を表すことができる。換言すれば、一連の具体的状態を無損失に併合して、シンボリック実行のための一又は複数のパラメトリック状態にすることができる。パラメトリック状態の生成は、c状態間の相異がパラメトリック公式化に適しているか否かの決定を含む、c状態の併合が可能であるか否かの決定を含んでもよい。特定の例では、簡約化及び最適化をc状態に適用して、p状態への併合を可能としてもよい。p状態を定義するための、多くの可能性がある抽象化スキームから、最善の抽象化スキームを決定してもよい。最後に、p状態の使用からの利益がp状態のパラメータ化のコストよ

りも大きくなるように、シンボリック実行の全体的な計算効率を向上させる方法で、c状態をp状態にパラメータ化してもよい。

【００１８】

図２を参照すると、パラメータ化処理２００の実施形態の選択要素のブロック図が示されている。図示するように、パラメータ化処理２００は、本明細書に記載されるように、シンボリック実行においてパラメータ化状態を併合するために使用される動作及びデータを表すことができる。特定の実施形態では、シンボリック実行システム１００（図１を参照）を用いて、例えば、状態マネージャ１２８によって、パラメータ化処理２００を実行することができる。具体的には、パラメータ化処理２００を使用して、一連のc状態を完全に記述する一又は複数のパラメータ化状態を表すp状態２０４を生成することができる。

【００１９】

従来のシンボリック実行において、具体的実行状態（すなわち、c状態）は、命令ポインタ値（IP）、パス条件（PC）、及び値ストア（図示せず）を含むことができる。IPは、シンボリック実行中の現在の実行ポイントを示すことができる。本明細書では、二つのc状態が同一のIPを有する場合にのみ併合されてp状態になるものと仮定する。PCは、実行状態が満足する一連のシンボリック制約を表すことができる。値ストアは、具体値又はシンボリック値となる得る値にアドレスを割り当てることができる。パラメトリック状態又はp状態は、共有パターンを有するc状態群を表すことができる。併合されてp状態になり得るc状態群における相異は、シンボリック式を用いて記述されてもよい。p状態を形成するために使用されるc状態群における相異は、パス状態における相異及び／又はシンボリック値における相異であってもよい。したがって、パラメトリック状態の併合によって生成されるp状態は、本明細書に記載されるように、c状態群の特徴を示すシンボリック論理式である追加部分、すなわちパラメトリック制約（“p制約”）を含むことができる。換言すれば、p制約は、p状態を形成するために使用されるc状態群における相異をまとめたものである。したがって、c状態を対応するp状態のインスタンスとみなすことができる一方で、p状態は、全ての関連するc状態を完全に記述することができる。エグゼキュータの観点から、パラメータ化状態の併合後、所与のp状態は、c状態群における各c状態と全く同一に動作することができる。

【００２０】

図２において、パラメータ化処理２００は、c状態２０２を併合してp状態２０４にするパラメータ化状態の特定の実施形態における動作を説明する。c状態２０２は、c状態２０２－１及びc状態２０２－２など、c状態２０２－Ｎまで示されるＮ個一組のc状態２０２によって表される。c状態２０２は、簡約化２１０によって受信可能であり、簡約化２１０は、c状態２０２を分析して、式の簡約化及び正規化のための機会を決定することができる。一の実施形態において、簡約化２１０は、c状態２０２について状態ツリーの少なくとも特定の部分にマッチさせるように試みて、その相異が一般化及び簡約化された形式に抽象化され得るか否かを確認することができる。簡約化２１０は、シンボリック推論を採用して、c状態２０２に関連する式を解いて簡約化することができる。そしてパターン特定２１２は、パターンデータベース２５０にアクセスして、簡約化２１０の後のc状態２０２に関連する式におけるパターンを検索することができる。いくつかの実施形態において、パターン特定２１２は、状態併合についての可能性を示すことが可能なc状態２０２の値ストア及びPC間の不変条件の検出を試みることができる。パターンデータベース２５０は、パターン特定２１２によるマッチングのためにアクセス可能な共通の不変パターンの知識ストアを表すことができる。パターン特定２１２の後、パラメータ化併合２１４は、パターン特定２１２によって特定されたパターン及び／又は既知の状態抽象化スキーム（図示せず）を使用して、シンボリック実行に関連させるために、全てのc状態２０２を表す一又は複数の実行状態を表現可能なp状態２０４を生成することができる。

【００２１】

ここで、図３を参照すると、パラメータ化状態を併合する方法３００の実施形態の選択

要素のブロック図がフローチャート形式で示されている。方法３００は、エグゼキュータ１２０によって実行されて、パラメータ化処理２００（図１及び２を参照）を実施することができる。方法３００において説明される特定の動作は、任意であってもよく、また、異なる実施形態において再配置されてもよいことに留意されたい。

【００２２】

方法３００は、命令ポインタ（IP）が実行のために残されているか否かの決定（動作３０２）から開始されてもよい。動作３０２においてIPは、シンボリック実行を用いたテスト中のソフトウェアに関連する命令を表してもよい。動作３０２の結果がＮＯである場合、方法３００は、終了してもよい（動作３０４）。動作３０２の結果がＹＥＳである場合には、次のＩＰが実行されて新たなc状態Sを生成することができる（動作３０６）。c状態Sのパス条件（PC）及び／又はシンボリック式は、簡約化されてもよい（動作３０８）。動作３０８は、c状態Sにおける式の正規化を含んでもよい。そして併合点に達したか否かの判定がされてもよい（動作３１０）。動作３１０の結果がＮＯである場合、方法３００は、動作３０２に戻ってもよい。動作３１０の結果がＹＥＳである場合には、c状態Sが既存のp状態P内に吸収可能であるか否かの判定がされてもよい（動作３１２）。p状態Pは、p状態Pのパラメトリック条件の拡張に従ってc状態Sに適用するように拡張可能である場合、c状態Sを吸収することができる。動作３１２の結果がＹＥＳである場合、p状態Pのパラメトリック条件は、更新されてもよい（動作３１６）。動作３１２の結果がＮＯである場合には、c状態Sと同一のIPを有する状態P'が存在しているか否かの判定がされてもよい（動作３１４）。状態P'は、c状態又はp状態であってもよい。動作３１４の結果がＮＯである場合、方法３００は、動作３０２に戻ってもよい。動作３１４の結果がＹＥＳである場合には、c状態SをP'と併合可能か否かの判定がされてもよい（動作３２０）。c状態Sと状態P'とのPCを共通PCに抽象化可能である場合、並びにc状態Sと状態P'とにおける変数を、共通変数に抽象化可能である場合又はif-then-else文のような論理評価で記述可能である場合には、動作３２０における条件を満足することができる。動作３２０の結果がＮＯである場合、方法３００は、動作３１４に戻ってもよい。動作３２０の結果がＹＥＳである場合には、c状態Sと状態P'とを併合して、新たなパラメトリック状態Pnを生成することができる（動作３２２）。特定の実施形態では、新たなパラメトリック状態Pnのために状態P'を再利用してもよい。動作３２２における併合に際して、新たなパラメトリック状態Pnは、c状態Sの抽象化を含んでもよい。動作３２２の後及び動作３１６の後には、存在するならばc状態S及びs状態Pを除去することができる（動作３２４）。動作３２４の後、方法３００は、動作３０２に戻ってもよい。

【００２３】

ここで、パラメトリック状態の併合の第一の例について、さらに詳細に説明する。第一の例では、テスト中のソフトウェアにおいてc状態として出現し得る以下の式１及び２を考慮する。

2a+b+c+1　（式１）

3+b+c+6a　（式２）

【００２４】

第一の方法では、共通パターン及び／又はサブパターンによって、式１及び２についてのp制約を特定することができる。例えば、式の簡約化及び正規化を用いて、可能な抽象化について、式の相異を特定及び評価することができる。例えば、可変次数に従って式２を正規化して、式３を得ることができる。

6a+b+c+3　（式３）

【００２５】

式１及び３の間の相異を、項2a、6a、及び項1、3とみなすことができる。これらの相異から、シンボリック変数iを（1，3）の値を有するところに導入して、元の式１及び２の両方を含むp状態についてのp制約を表す式４における抽象化を得ることができる。

(2i*a)+b+c+i　（式４）

【００２６】

式は、変数値ドメインを拡張することによって、別の式に吸収することができる。例えば、以下の式５は、i=5についての式４のインスタンスであり、式４における抽象化形式が、式５も吸収できることを示している。

10a+b+c+5　（式５）

【００２７】

第一の例における第一の方法の代わりに、式４を得るための第二の方法は、不変ツールの使用と、iの取り得る値を制約するための改善とを含んでもよい。第一の例における第一の方法及び／又は第二の方法の代わりに、第三の方法は、式１及び２を一次多項式のような共通の不変パターンと比較して、式４にマッチさせるためのパターンライブラリの使用を含んでもよい。

【００２８】

ここで、図４Ａ、４Ｂ、４Ｃ、及び４Ｄを参照すると、パラメトリック状態の併合の第二の例の実施形態の選択要素は、各状態ツリー４００、４０１、４０２、及び４０３として示されている。図４Ａ－Ｄに図示される第二の例は、シンボリック入力が変数n，a，及びbである、テスト中の以下の実行可能コードのシンボリック実行に基づいている。図４Ａ－Ｄにおいて、シンボリック実行についての状態ツリーは、1～1000までのnの値についてのテスト中の実行可能コードの完全繰返後の実行状態を表すために図示されている。

【表１】

```
int k = 0, v = 0;
for (int i = 1; i <= 1000; i++) {
        if (i == n) {
           if (k > f (i * b)) v = a + i * b;
        }
        k += 2;
}
assert(v + k != 500);
```

第二の例についてのテスト中の実行可能コード

【００２９】

図４Ａにおいて、状態ツリー４００は、forループの最初の繰返が完了したときの完了した実行状態を表している。状態S1は、最初のコードの整数変数k=0，v=0を表すことができる一方で、状態S2は、条件“ i==n ”が、パス条件がn=1であることを指定しているときの、最初のforループの入力を表すことができる。条件“ i≠n ”に関する場合は、図示されていない。状態S3及びS4は、条件付きif文コードの二つの可能性を表すことができる。したがって、n=1についての完了した実行状態の後、状態S3については、0>f(b)，v=a+b，及びk=2となる一方で、状態S4については、0≦f(b)，v=0，及びk=2となる。これは、シンボリック実行の最初の繰返であるので、p制約は未だ特定されておらず、また、p状態は未だ存在していない。

【００３０】

図４Ｂにおいて、状態ツリー４０１は、n=2についての状態が生成されるforループの二回目の繰返が完了したときの完了した実行状態を表している。状態S1は、最初のコードの整数変数k=0，v=0を表すことができる一方で、状態S2及びS5は、それぞれn=1及びn=2についてのforループの入力コードを表すことができる。状態S3及びS4は、n=1についての条件付きif文コードの二つの可能性を表すことができる。S3及びS4がn=2で二回目のループ繰返を実行する場合、条件“ 2==n ”が“ n=1 ”に抵触するため、状態S3及びS4は、if文の実行をスキップし、整数変数kの値を2から4に増加させる。同様に、状態S6及びS7は、n=2に

ついての条件付きif文コードの二つの可能性を表すことができる。したがって、n=2についての完了した実行状態の後、状態S6については、2>f(b)，v=a+2b，及びk=4となる一方で、状態S7については、2≦f(b)，v=0，及びk=4となる。このとき、状態S6は、いかなる既存のp状態のインスタンスでもないが、状態S3及びS4と同一のIPを共有している。したがって、v=a+n*bを用いて、状態S3及びS6を2(n-1)>f(b)に抽象化することができる。パラメトリック状態のために状態S3を再利用することができるとともに、状態S6を除去することができる。このとき、状態S7は、いかなる既存のp状態のインスタンスでもないが、状態S3及びS4と同一のIPを共有している。したがって、v=0を用いて、状態S4及びS7を2(n-1)≦f(b)に抽象化することができる。パラメトリック状態のために状態S4を再利用することができるとともに、状態S7を除去することができる。状態S2が状態S5に対応するので、状態S6及びS7が除去されたときに状態S5も除去される。

【００３１】

図４Ｃにおいて、状態ツリー４０２は、n=3についての状態が生成されるforループの三回目の繰返が完了したときの完了した実行状態を表している。状態S1は、最初のコードの整数変数k=0，v=0を表すことができる一方で、状態S2及びS8は、それぞれn=1，2及びn=3についてのforループの入力コードを表すことができる。状態S3及びS4は、n∈［1，2］についての二つのp状態を表すことができるとともに、状態S9及びS10は、n=3についての条件付きif文コードの二つの可能性を表すことができる。したがって、n=3についての完了した実行状態の後、状態S9については、4>f(3b)，v=a+3，及びk=6となる一方で、状態S10については、4≦f(3b)，v=0，及びk=6となる。このとき、状態S9は、p状態S3のインスタンスであり、状態S10は、p状態S4のインスタンスである。したがって、p状態S3及びS4は、それぞれ状態S9及びS10を吸収することができるとともに、状態S8、S9、及びS10を除去することができる。

【００３２】

図４Ｄにおいて、状態ツリー４０３は、n=1000についての状態が生成されるforループの1000回目の繰返が完了したときの完了した実行状態を表している。状態S1は、最初のコードの整数変数k=0，v=0を表すことができる一方で、状態S2及びS11は、それぞれn=1，2…，999及びn=1000についてのforループの入力コードを表すことができる。状態S3及びS4は、n∈［1，999］についての二つのp状態を表すことができるとともに、状態S12及びS13は、n=1000についての条件付きif文コードの二つの可能性を表すことができる。したがって、n=1000についての完了した実行状態の後、状態S12については、1998>f(1000b)，v=a+1000b，及びk=2000となる一方で、状態S13については、1998≦f(1000b)，v=0，及びk=2000となる。このとき、状態S12は、p状態S3のインスタンスであり、状態S13はp状態S4のインスタンスである。したがって、p状態S3及びS4は、それぞれ状態S12及びS13を吸収することができるとともに、状態S11、S12、及びS13を除去することができる。

【００３３】

nの特定の値についての繰返は、図４Ａ～Ｄにおける第二の例から省略されているが、図４ＤにおいてP状態S3及びS4が、いかなる情報も失うことなく、n∈［1，1000］についてのp状態を表していることは明らかである。併合してp状態S3及びS4にした後には、元の2000個のc状態によって繰り返すのではなく、p状態S3及びS4を用いて、SUTにおけるアサーションを正確にチェックすることによって、生成されるテストの数の低減だけでなく、バグ検出のための計算コストの相当の低減をもたらすことができる。したがって、第二の例では、2000個のc状態を二つのp状態に置換することによって、シンボリック実行に関連する、シンボリック実行を行うプロセッサのメモリ消費及び／又は計算負荷を含む計算資源の消費における、対応する低減とともに、シンボリック実行のための状態の相当の低減をもたらすことができる。

【００３４】

上記の第二の例に示すように、パラメトリック状態の併合は、本明細書に記載されるように、シンボリック実行中のループ構造に特に有効である。ループの繰返は、多くの場合、同様の動作を示すことができ、ループインデックスがどのように使用されるかにおいて

主に相異し得る。この動作を使用して、例えば、ループヘッダにおいてp制約を定義することができる一方で、ループ本体においてPC及び値ストアを含む共通パターンを特定することができる。

【００３５】

様々な実施形態では、以下のループ構造を含むSUTを静的に調査して、p制約及び共通パターンを推測することができる。二つのループヘッダは、p制約候補が、シンボリック値i及びjについて∈［k1，…］∧j∈［i，…］であることを意味している。ループ本体は、パラメトリックパス条件がc(i,j)であり、かつvのパラメータ値がf(i,j)であることを意味している。

【表２】

```
for (int i = k1; …) {
    …
    for (int j = i, …) {
        …
        if (c(i,j)) {
            v = f(i,j);
            …
        }
    }
}
```

パラメトリックパターンの推測のために実行可能なループ構造

【００３６】

ここで、図５を参照すると、シンボリック実行システム５００の実施形態の選択要素を示すブロック図が図示されている。図５に示すように、シンボリック実行システム５００は、システムバス５０２を用いて通信可能なプロセッサ５０１と記憶媒体５１０とを含んでいる。また、ネットワークへの接続を提供するネットワークアダプタ５２０が、システムバス５０２を介してアクセス可能に示されている。

【００３７】

図５に示すように、記憶媒体５１０は、揮発性、不揮発性、固定及び／又は取外可能な媒体を代表することができ、磁気及び／又は半導体メモリを用いて実現することができる。記憶媒体５１０は、命令及び／又はデータを格納することができる。図示するように、記憶媒体５１０は、オペレーティングシステム（OS）５１２を含む命令（例えば、コンピュータプロセッサ又はマイクロプロセッサのようなプロセッサ５０１によって実行可能なコード）、及びエグゼキュータ１２０（図１を参照）を格納する。オペレーティングシステム５１２は、UNIX（登録商標）バリアント、LINUX（登録商標）、Microsoft Windows（登録商標）オペレーティングシステム、又は別のオペレーティングシステムのような、様々な任意のオペレーティングシステムとすることができる。前述のように、エグゼキュータ１２０は、本明細書に記載されるように、c状態及びp状態を含むシンボリック実行における状態のインスタンスを生成及び管理するための機能を提供可能な状態マネージャ１２８を含むことができる。

【００３８】

ここで、図６を参照すると、パラメータ化状態を用いてシンボリック実行する方法６００の実施形態の選択要素のブロック図がフローチャート形式で示されている。エグゼキュータ１２０によって方法６００を実行して、パラメータ化処理２００（図１及び２を参照）を実施することができる。方法６００において説明される特定の動作は、任意のもので

あってもよく、また、異なる実施形態において再配置されてもよいことに留意されたい。

【００３９】

実行可能コードに関連する実行状態において共有状態要素と相異状態要素とを特定する（動作６０２）ことによって、方法６００が開始されてもよい。相異状態要素は、状態抽象化を用いて表されてもよい（動作６０４）。状態抽象化は、パラメトリック状態の複雑さを最小限にするように選択されてもよい。動作６０４は、少なくともいくつかの相異状態要素を数学的に簡約化すること、及び相異状態要素における共通パターンを特定することを含んでもよい。特定された共通パターンを、格納された共通パターンとマッチングすることができる。状態抽象化に基づいて、方法６００は、パラメトリック制約を有し、かつ実行状態の群を無損失に表すパラメトリック状態を定義する（動作６０６）ことを含んでもよい。動作６０６は、共通パターンに基づいて、パラメトリック状態を定義することを含んでもよい。実行状態の群における実行状態は、パラメトリック状態のインスタンスとすることができる。シンボリック実行についてパラメトリック状態を使用してもよい（動作６０８）。動作６０８は、パラメトリック状態を使用してテストケースを生成すること、及びテストケースを使用して状態群におけるバグチェックを行うことを含んでもよい。

【００４０】

特定の実施形態において、実行状態の群は、ループの繰返を表し、方法６００は、ループのループヘッダに基づいてパラメトリック制約を定義すること、並びにループのループ本体（図６に図示せず）からパラメトリック状態のパス条件及び値ストアを定義することを含んでいる。様々な実施形態において、方法６００は、実行可能コード（図６に図示せず）に関連する実行状態の数を低減させるために、実行状態の群をパラメトリック状態に置換することを含み、そのことは、シンボリック実行によって消費されるメモリ資源を低減させることを含む。

【００４１】

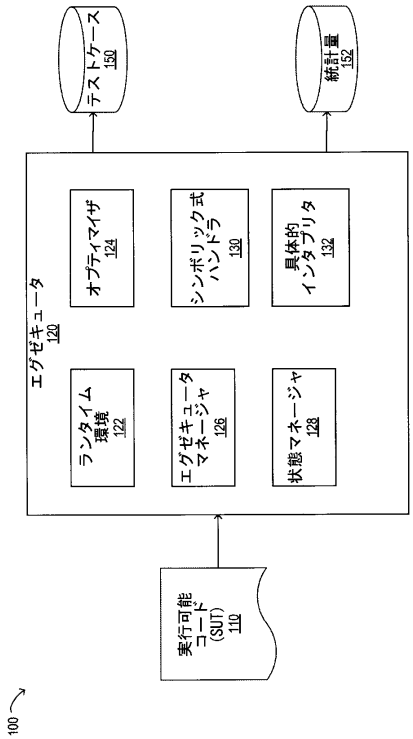
本明細書に開示するように、テスト中のソフトウェアをシンボリック実行する方法及びシステムは、具体的実行状態の群を無損失に表すために、パラメトリック状態の使用を含んでいる。数学的な抽象化は、実行状態間の相異を表すことができ、また、パラメトリック状態についてのパラメトリック制約を定義することができる。計算資源における量を低減させるために、シンボリック実行についてパラメトリック状態を使用可能にすることができる。パラメトリック状態を使用することで、より大きな状態空間及びより多くのプログラムの動作を、シンボリック実行を用いてテスト可能にすることができる。

【００４２】

上記開示された要旨は、例示であって限定的ではないと考えられるべきであり、かつ添付の特許請求の範囲は、本開示の真の精神及び範囲内に入るような全ての変更、拡張、及び他の実施形態を包含することを意図している。したがって、法律により許される最大範囲まで、本開示の範囲は、特許請求の範囲及びその均等物の最も広い許容可能な解釈によって決定されるべきであり、前述の詳細な説明によって制限又は限定されるべきものではない。

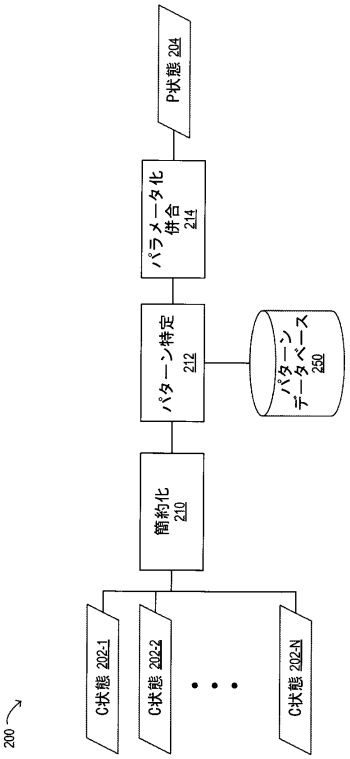
【図１】

図1



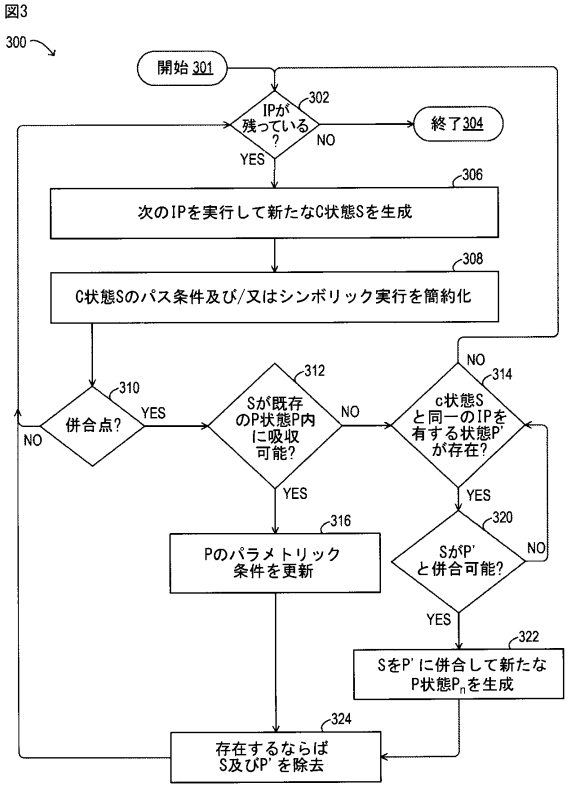
【図２】

図2



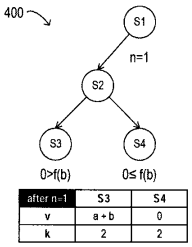
【図３】

図3



【図４Ａ】

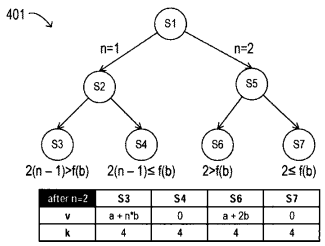
図4A



after n=1	S3	S4
v	a + b	0
k	2	2

【図４Ｂ】

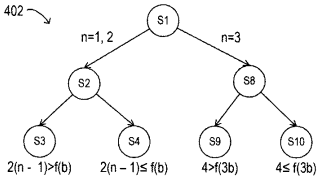
図4B



after n=2	S3	S4	S6	S7
v	a + n*b	0	a + 2b	0
k	4	4	4	4

【図４Ｃ】

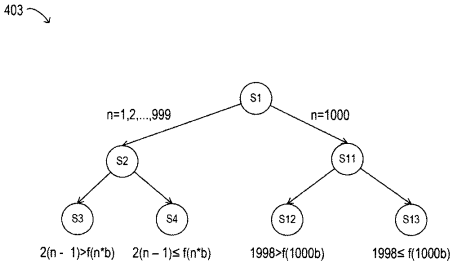
図4C



after n=3	S3	S4	S9	S10
v	a + n*b	0	a + 3b	0
k	6	6	6	6

【図４Ｄ】

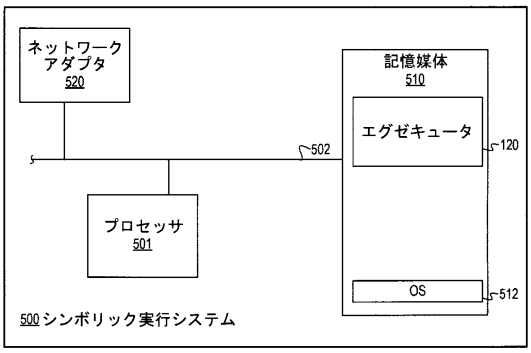
図4D



after n=1000	S3	S4	S12	S13
v	a + n*b	0	a + 1000b	0
k	2000	2000	2000	2000

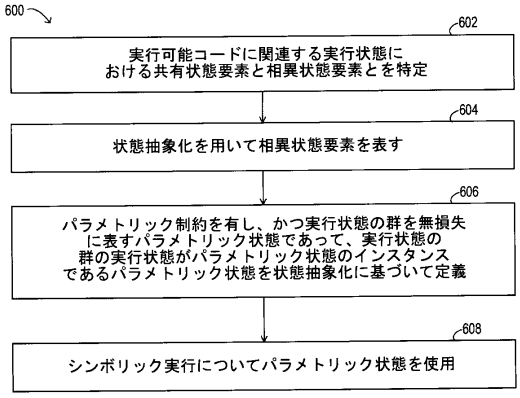
【図５】

図5



【図６】

図6



フロントページの続き

(72)発明者 グオドン リー
アメリカ合衆国, カリフォルニア州 ９４０８５, サニーベール, イースト アークス アベニュー １２４０番 エム／エス ３４５, シー／オー フジツウ ラボラトリーズ オブ アメリカ インコーポレイテッド
(72)発明者 インドラディープ ゴーシュ
アメリカ合衆国, カリフォルニア州 ９４０８５, サニーベール, イースト アークス アベニュー １２４０番 エム／エス ３４５, シー／オー フジツウ ラボラトリーズ オブ アメリカ インコーポレイテッド

審査官 今城 朋彬

(56)参考文献 特開２０１１－１９１９８５（ＪＰ, Ａ）
米国特許出願公開第２００７／０１５７１８０（ＵＳ, Ａ1）

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ １１／３６
Ｇ０６Ｆ ８／７５