(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)公表番号
特表2023-506097
(P2023-506097A)
(43)公表日　令和5年2月14日(2023.2.14)

(51)国際特許分類		FI			テーマコード（参考）
G06F 11/36 (2006.01)		G06F	11/36	112	5B042
G05B 19/05 (2006.01)		G05B	19/05	Z	5H220

審査請求　有　予備審査請求　未請求　（全24頁）

(21)出願番号　特願2022-561256(P2022-561256)
(86)(22)出願日　令和2年12月23日(2020.12.23)
(85)翻訳文提出日　令和4年6月9日(2022.6.9)
(86)国際出願番号　PCT/JP2020/049302
(87)国際公開番号　WO2021/199552
(87)国際公開日　令和3年10月7日(2021.10.7)
(31)優先権主張番号　20315085.9
(32)優先日　令和2年3月31日(2020.3.31)
(33)優先権主張国・地域又は機関
欧州特許庁(EP)
(81)指定国・地域　AP(BW,GH,GM,KE,LR,LS,MW,MZ,NA,RW,SD,SL,ST,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,RU,TJ,TM),EP(AL,AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HR,HU,IE,IS,IT,LT,LU,LV,MC,
最終頁に続く

(71)出願人　503163527
ミツビシ・エレクトリック・アールアンドディー・センター・ヨーロッパ・ビーヴィ
ＭＩＴＳＵＢＩＳＨＩ ＥＬＥＣＴＲＩＣ Ｒ＆Ｄ ＣＥＮＴＲＥ ＥＵＲＯＰＥ Ｂ．Ｖ．
オランダ国、１１１９ エヌエス・スヒプホール・レーイク、カプロニラーン ４６
Ｃａｐｒｏｎｉｌａａｎ ４６，１１１９ ＮＳ Ｓｃｈｉｐｈｏｌ Ｒｉｊｋ，Ｔｈｅ Ｎｅｔｈｅｒｌａｎｄｓ
(74)代理人　100110423
弁理士　曾我 道治
最終頁に続く

(54)【発明の名称】　プログラマブルロジックコントローラプログラムの解析方法

(57)【要約】
プログラムが論理フレームワークのプログラムモデルに変換され、そこからプロパティが決定される、ＰＬＣプログラム分析方法が開示される。連動性に結合されたプロパティが、自動化ソルバによって検証される。プロパティの対偶が充足可能である場合、モデルの入力及び内部メモリ値を表す反例が提供される。反例は、モデルのエラー初期構成に変換される。モデルの実行は、モデルエラー初期構成を用いてシミュレートされ、モデルシミュレーションのエラー中間構成が、プロパティ違反まで記録される。元のプログラムのエラー初期構成及びエラー中間構成が、モデルのエラー初期構成及びモデルシミュレーションのエラー中間構成から導出され、表示される。モデルを実行する装置が提供される。

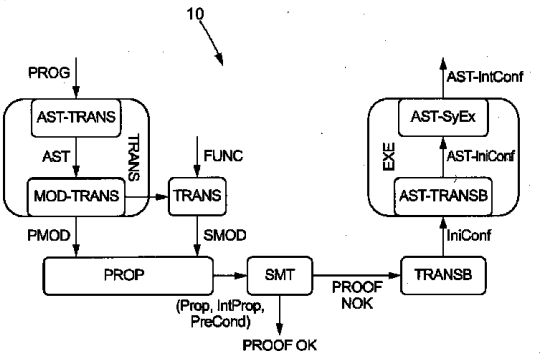


FIG. 1

【特許請求の範囲】
【請求項１】
プログラマブルロジックコントローラプログラム分析方法であって、
プログラマブルロジックコントローラプログラムのタイプの元のプログラムを論理フレームワークのプログラムモデルに変換するステップと、
少なくとも前記プログラムモデルに基づいて、ユーザ仕様を論理フレームワークの仕様モデルに変換するステップと、
少なくとも前記プログラムモデルと事前定義された言語形式とから、前記元のプログラムの内部変数に関する一組のプロパティを決定するステップと、
自動化ソルバによって、前記仕様モデルから得られる連動性と結合された前記一組のプロパティの充足可能性を検証し、前記一組のプロパティについてのプロパティの対偶が充足可能である場合には、前記プロパティの対偶が充足可能であるプログラムモデルの入力及び内部メモリ値を表す一組の反例を提供し、又は、前記一組のプロパティが常に充足されている場合には、その確認を提供するステップと、
反例を前記プログラムモデルのエラー初期構成に変換するステップであって、前記エラー初期構成は入力及び内部メモリの初期値を含む、反例を変換するステップと、
前記プログラムモデルの実行を、前記プログラムモデルのエラー初期構成を用いてシミュレートし、実行の開始から前記プロパティの違反までの前記プログラムモデルのシミュレーションのエラー中間構成を記録するステップであって、前記エラー中間構成は内部メモリの中間値を含む、モデル実行をシミュレートするステップと、
前記プログラムモデルの前記エラー初期構成及び前記プログラムモデルのシミュレーションの前記エラー中間構成を前記元のプログラムのエラー初期構成及びエラー中間構成に変換するステップと、
前記元のプログラムのエラー初期構成及びエラー中間構成を表示するステップと、
を含む、プログラマブルロジックコントローラプログラム分析方法。
【請求項２】
ユーザ仕様を仕様モデルに変換する前記ステップは、前記ユーザ仕様が、機能仕様テンプレートと、前記プログラマブルロジックコントローラプログラムによって使用されるデバイスの選択とを使用して表される中間ステップを含む、請求項１に記載のプログラマブルロジックコントローラプログラム分析方法。
【請求項３】
前記元のプログラムを前記プログラムモデルに変換するステップは、抽象構文木としての前記元のプログラムの表現の第１の中間ステップと、前記抽象構文木（ＡＳＴ）についての前記プログラムモデルの生成の第２の中間ステップとを含む、請求項１又は２に記載のプログラマブルロジックコントローラプログラム分析方法。
【請求項４】
前記モデル実行をシミュレートするステップは、
前記プログラムモデルのエラー初期構成を、前記抽象構文木に対応するエラー初期構成に変換する第１の中間ステップと、
前記対応する初期構成を用いて前記抽象構文木を計算し、前記抽象構文木に対応する内部メモリの中間値を回収する第２の中間ステップと、
を含む、請求項３に記載のプログラマブルロジックコントローラプログラム分析方法。
【請求項５】
前記プログラムモデルは、一階述語論理フレームワークで表される、請求項１～４のいずれか一項に記載のプログラマブルロジックコントローラプログラム分析方法。
【請求項６】
前記元のプログラムを前記プログラムモデルに変換するステップは、静的単一代入変換の中間ステップを含む、請求項１～５のいずれか一項に記載のプログラマブルロジックコントローラプログラム分析方法。
【請求項７】

前記一組のプロパティを決定するステップ中、前記一組のプロパティは、ダイクストラの最弱事前条件計算法を用いて、前記一組のプロパティに対する事前条件を決定するように計算され、前記一組のプロパティの検証は、その事前条件に基づいて実施される、請求項１～６のいずれか一項に記載のプログラマブルロジックコントローラプログラム分析方法。

【請求項８】

連動性と結合された前記一組のプロパティの充足可能性は、充足可能性モジュロソルバを使用して検証される、請求項１～７のいずれか一項に記載のプログラマブルロジックコントローラプログラム分析方法。

【請求項９】

プロセッサによって実行されると、請求項１～８のいずれか一項に記載の方法を実行する命令を含むコンピュータプログラム。

【請求項１０】

請求項１～８のいずれか一項に記載のプログラマブルロジックコントローラプログラム分析方法を実行する装置。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、国際標準規格ＩＥＣ６１１３１－３に記載されている言語で書かれたプログラムを分析する方法及び装置に関する。こうしたプログラムは、特に、産業システムの制御を実行するプログラマブルロジックコントローラ（ＰＬＣ）のために意図されている。

【０００２】

本発明は、より詳細には、そのようなＰＬＣプログラムにおけるエラーを分析、検出及び修正する方法及び装置に関する。

【背景技術】

【０００３】

プログラマブルロジックコントローラ（ＰＬＣ）は、組立ライン又はロボットデバイス等、製造プロセスのオートメーションコントローラとして使用される、産業用デジタルコンピュータである。

【０００４】

ＰＬＣには、入力の値から出力を計算するソフトウェア及び内部メモリが設けられているので、ＰＬＣは、配線によるリレー、タイマ及びシーケンサに取って代わった。標準規格ＩＥＣ６１１３１－３は、ラダーロジック（ラダー）、命令リスト（ＩＬ）、構造化テキスト（ＳＴ）、シーケンシャルファンクションチャート（ＳＦＣ）及びファンクションブロックダイアグラム（ＦＢＤ）等のＰＬＣをプログラミングするのに使用される種々のプログラミング言語を記載している。ラダーロジックとしても知られるラダー言語は、ＰＬＣソフトウェアを開発するために用いられるプログラミング言語である。この言語は、リレーロジックハードウェアの回路図を用いて、ＰＬＣプログラムを図式的に表す。

【０００５】

ソフトウェア開発プロセスの一部は、機能仕様の作成に割り当てられる。本発明の開示では、機能仕様はユーザ仕様とも呼ばれる。機能仕様は、通常、自然言語で記述された文書からなり、プログラムが行うべきことを指定する。機能仕様は、プログラムが実行すると予想される機能を記載している。ソフトウェアは、その後、機能仕様に従って記述されたプログラムを用いて開発される。

【０００６】

ソフトウェア開発プロセスの別の一部は、デバッグに割り当てられる。デバッグは、プログラムが安全に且つ機能仕様文書に記載された仕様に従って動作することを確認することである。工場でバグが発生すると、人的及び物的被害、並びにプラント操業中止など、極めて多大なコストがかかるため、デバッグは、製造環境に導入する前に実施する必要がある。バグは、ダウンタイム並びに場合によっては人的及びハードウェア的な損害の点か

ら、ファクトリーオートメーションシステムにおいても同様に多くのコストを要する可能性がある。

【０００７】

プログラムをデバッグするために構成されたテストには、さまざまなタイプがある。

【０００８】

第１のタイプは、単体テスト及び統合テストに関するものであり、種々の構成要素の動作及び共通の相互作用を調べるものである。これらのテストは、特にランタイムエラーをもたらすプログラミングエラーを検出するように構成される。

【０００９】

第２のタイプのテストは、機能テストとも呼ばれるシステムテストと、承認テストとに関するものであり、プログラムの機能仕様に関するエラーを検出するものである。機能テストは、プログラムが、機能仕様に指定された要件に準拠して記述されているか否かを評価する。したがって、機能テストは、エンドユーザの観点から、及びビジネス要件に従って、プログラムが正しく且つ安全に動作することを検証することができる。

【００１０】

プログラムにテストを行うための通常の方法は、シミュレーション法である。それは、初期構成によって定義されるいくつかのテストを設定し、そのテストにおいてプログラムを実行して、これらの構成下でその動作を確認することである。プログラムは、通常、工場のソフトウェアシミュレーションで実行される。

【００１１】

こうしたデバッグ方法の主な欠点は、このような方法が、網羅的ではなく且つ時間がかかり、したがって、コストがかかるということである。この非網羅性により、選択されたテストが製造環境においてプログラムの可能な限り全ての実行を１回でカバーすることを保証することが極めて困難である。唯一の保証は、プログラムが、テストされた構成に対してバグがないということであるため、テスト段階後にもプログラムにバグが残っている可能性がある。

【００１２】

モデル検査法は、時間及び資源を節約するために、プログラムそのものをテストするのではなく、プログラムのモデルを連続的に実行することである。しかしながら、特に産業応用においては、実行パス関連の複雑さは指数関数的に増大することが多い。テスト効率は、ＣＰＵ時間によって制限されるため、この方法は依然として網羅的ではない。

【００１３】

さらに、テストを生成及び実行するためには、少なくともいくつかの構成は手動で定義する必要があるため、上記テストは、部分的に又は完全に手動で開発され、実行される。テストの生成及び実行の一部は自動化することができるが、こうした方法の別の欠点は、テストの出力値を依然としてプログラマが分析しなければならず、バグに関する唯一の入手可能な情報は、バグを引き起こす入力及び内部メモリの初期構成であるということである。したがって、エラーの基本的な原因と、その初期構成によりプログラムのいずれかの時点でなぜそのエラーが発生するのかとを理解することが困難であることが多い。

【００１４】

この手法のもう１つの欠点は、機能仕様に関するテストの開発方法によってバグが生じる可能性があるということである。

【００１５】

第１に、機能仕様を自然言語で記述することは、不正確であり、曖昧である可能性がある。仕様を自然言語で表現する標準化された方法が既に登場している可能性はあるが、こうした方法では、実装時に開発者が仕様を誤解し、関係のないコードを記述する可能性がある曖昧性と、テスト時に試験者が仕様を誤解して不十分なテストをセットアップする可能性がある曖昧性とをもたらすおそれがある。

【００１６】

第２に、機能仕様を自然言語で記述することは多くの時間を要する。しかし、最も重要

なこととして、この文書は、適切な機能テスト及び認可テストを構築するために、この文書を読んで解釈する必要があるテスト技術者も対象としているということである。それらの全ての段階が多くの時間を要し、したがって、多くの費用を要する。

【００１７】

第３に、こうした方法論は網羅的でない。なぜならば、自然言語の使用に由来する曖昧さが、テスト開発プロセス中にいくつかの明示されていない部分が必然的に発生し得るからである。このように明確な機能仕様を提供することができないことが、テストベースのデバッグプロセスにおいて遭遇する網羅性の課題を増大させている。ソフトウェアの可能な全ての実行をテストすることはできず、テストに合格した後であっても、バグがコード内に残っている可能性がある。

【００１８】

テストステップ中に誤った解釈がされるリスクのないテストを開発するために、機能仕様を明確な方法で表すいくつかの方法が開発されている。これらの方法のうちのいくつかは、非常に正確且つ明確な方法で時間仕様を表すことができる形式的方法に基づいている。しかしながら、これらの方法を適切に使用するには、多くの場合、ロジックの知識が必要とされるため、特に通常の技術者にとってこうした方法を操作することは難しい。一例は、自然言語文の空白を埋めるような、事前定義された仕様のテンプレートを提供することである。しかしながら、こうした方法は、時間仕様にしか適用することができない。時間仕様は、形式モデルには適用されるがプログラムには直接適用されないモデル検査方法のような非網羅的な形式的方法でしか検証することができない。

【発明の概要】

【発明が解決しようとする課題】

【００１９】

本開示の目的は、プログラムの可能な全ての実行がカバーされることを保証する、プログラマブルロジックコントローラプログラムを網羅的に検証する解決策を提供することである。

【課題を解決するための手段】

【００２０】

本発明の１つの態様によれば、プログラマブルロジックコントローラプログラム分析方法であって、

プログラマブルロジックコントローラプログラムのタイプの元のプログラムを論理フレームワークのプログラムモデルに変換するステップと、

少なくともプログラムモデルに基づいて、ユーザ仕様を論理フレームワークの仕様モデルに変換するステップと、

少なくともプログラムモデルと事前定義された言語形式とから、元のプログラムの内部変数に関する一組のプロパティを決定するステップと、

自動化ソルバによって、仕様モデルから得られる連動性と結合された一組のプロパティの充足可能性を検証し、当該一組のプロパティについてのプロパティの対偶が充足可能である場合には、プロパティの対偶が充足可能であるプログラムモデルの入力及び内部メモリ値を表す一組の反例を提供し、又は、一組のプロパティが常に充足されている場合には、その確認を提供するステップと、

反例をプログラムモデルのエラー初期構成に変換するステップであって、初期構成は入力及び内部メモリの初期値を含む、反例を変換するステップと、

プログラムモデルの実行を、当該プログラムモデルのエラー初期構成を用いてシミュレートし、実行の開始からプロパティの違反までのプログラムモデルのシミュレーションのエラー中間構成を記録するステップであって、当該中間構成は内部メモリの中間値を含む、モデル実行をシミュレートするステップと、

プログラムモデルのエラー初期構成及び当該プログラムモデルのシミュレーションのエラー中間構成を元のプログラムのエラー初期構成及びエラー中間構成に変換するステップと、

プログラムのエラー初期構成及びエラー中間構成を表示するステップと、
を含む、プログラマブルロジックコントローラプログラム分析方法が開示される。
【００２１】
これらの特質の下で、プログラムに対する１つの又はいくつかのプロパティが違反されたとき、バグが発生する。プロパティは、プログラムの入力、出力及びローカルメモリの値に関連する。本発明により、シミュレーション環境又は実環境におけるプログラムの実行の前にこうした違反を検出することができる。本方法は、プログラムの実行中にエラーをもたらす入力及びローカルメモリの初期値を見つけることである。デバッグは自動化されるとともに加速される。本方法は、記載した方法でエラーシナリオが見つからない場合に、実行構成がプログラムプロパティを違反しないことが保証されるため、網羅的である。実際に、従来技術では、テストの実行中にプロパティ違反が検出されない場合であっても、全てのあり得る実行に対してそのプロパティが成立する保証はない。
【００２２】
より詳細には、本発明は、ＰＬＣプログラムの機能仕様を曖昧性なく表すとともに、こうした機能仕様の検証を完全に自動化して高速化する解決策を提供する。
【００２３】
仕様モデルは論理フレームワークで表されるので、仕様モデルと結合された一組のプロパティの充足可能性の検証は、仕様違反が見つからない場合に、プログラムの実行が生成時に仕様に違反する可能性がないことを保証する。
【００２４】
仕様モデルは、その後、充足可能性検証ステップ中に自動的に検証される。この検証ステップ中に演繹的検証を使用することによって、プロパティ違反が見つからない場合に、プログラムの実行が機能仕様に違反する可能性がないことが保証される。ＰＬＣプログラムの機能仕様を検証する通常のテストベースの手法では、機能仕様に従って記述されたプログラムの可能な全ての実行をテストすることができないが、この手法は、通常のテストベースの手法とは対照的に網羅的である。したがって、本発明のＰＬＣプログラム分析方法によって、ＰＬＣプログラムの仕様は、曖昧性なく、完全且つ自動的に指定し、検証することができる。
【００２５】
本方法は、検証されるＰＬＣプログラムの可能な全ての実行を、その機能仕様とともにカバーする。したがって、得られた初期構成及び中間構成によって、エラーシナリオについての有用な情報をプログラムコードとともに本方法の最終ステップにおいて提供して、こうしたエラーがプログラムの実行中にどこで、どのようにして、なぜ発生するのかを示すことができる。プロパティ違反は正確に説明される。プロパティ違反をもたらす初期構成が計算され、メモリ割当等の実行情報が、初期構成からプロパティが違反されるプログラムの箇所まで、記録及び取得される。
【００２６】
本発明は、ＰＬＣプログラムの実行の安全性及び健全性を保証することに加えて、機能仕様を検証する解決策を提供する。本方法は、プログラマが一階述語論理において表現したランタイムエラー又は機能仕様違反のいずれかを連動性として検出する。本発明は、ランタイムエラー又は仕様違反が見つかったときに有用なデバッグ情報を提供する。本発明は、ランタイムエラー又は仕様違反をもたらす初期構成及び中間構成に関する全ての実行情報をＰＬＣプログラムのプログラミング言語に従って提供する。
【００２７】
仕様違反が検出された場合には、仕様がいつ、なぜ違反されたのかをプログラマが理解するのに必要な全ての情報が表示される。こうした情報は、通常、当該仕様違反をもたらすメモリの初期値及び中間値を含む。
【００２８】
プログラムは、プロパティ違反をもたらす初期値及び中間値並びにバグを修正する方法等、実行中のメモリ値に対する追加の情報を表示することができる。したがって、本方法

は、実行すべきテストを決定し、それらの実行を監視し、それらの結果を分析するために、人の介入が不要であるため、自動デバッグを可能にする。

【００２９】

本方法はまた、産業サイズの複雑なプログラムに対して制限がある自動シミュレーションの場合のように、シミュレートすべき複数のテスト構成下でプログラムを連続的に実行するものではなく、プロパティ検証に依存するため、より高速なデバッグも可能になる。したがって、中央処理装置での実行時間が削減される。通常の産業サイズのＰＬＣプログラムは、本発明のＰＬＣプログラム分析方法に通すと数秒で検証することができる。

【００３０】

これらの特質のおかげで、ＰＬＣプログラム展開時間を安全に削減し、ファクトリーオートメーション業界におけるＰＬＣプログラムの信頼を高める、効率的で、網羅的且つ高速なツールが提供される。

【００３１】

プログラムモデルは、好ましくは、数学的に証明可能である一階述語論理フレームワークで表されるため、計算問題を表現するのに有用である。さらに、一階述語論理は、ＰＬＣプログラムモデルから導出されるプロパティと対応する機能仕様とを表すように適合され、上位の論理フレームワークと比較して、一階述語論理プロパティの生成は、自動化がより容易である。

【００３２】

一実施の形態によれば、ユーザ仕様を仕様モデルに変換するステップは、ユーザ仕様が、機能仕様テンプレートと、ＰＬＣプログラムによって使用されるデバイスの選択とを使用して表される中間ステップを含む。

【００３３】

ユーザ仕様は、ＰＬＣプログラムが実行すると予想される機能を記載し、一般に、当該ユーザ仕様に従って開発されるＰＬＣプログラムに先立って記述される。ＰＬＣプログラムの機能は、通常、自然言語で指定されるため、ユーザ仕様は非形式言語で表される。ユーザ仕様の変換ステップのこの中間ステップは、ユーザ仕様を形式言語で表すことを可能にする。

【００３４】

このように、仕様モデルは、機能仕様テンプレートと、ＰＬＣプログラムによって使用されるデバイスの選択とを使用して表されたユーザ仕様から論理フレームワークで生成される。ＰＬＣプログラムによって使用されるデバイスの選択は、プログラムモデルから提供される場合がある。したがって、生成された仕様モデルは、プログラムモデルに関係している。一実施の形態によれば、元のプログラムをプログラムモデルに変換するステップは、抽象構文木としての元のプログラムを表現する第１の中間ステップと、抽象構文木からプログラムモデルを生成する第２の中間ステップとを含む。

【００３５】

こうした中間ステップの実装は、抽象構文木を、それが含有する全ての要素に対するプロパティ及びアノテーション等の情報により強化することができるため、有利である。抽象構文木は、好ましくは、ＯＣＡＭＬ等の関数型言語で記述される。また、抽象構文木は、ＰＬＣプログラムにおける各要素の位置を記憶することもできる。それは、プログラムモデルのシミュレートされた実行中にエラー中間構成を取得するのに有用である。

【００３６】

このため、プログラムモデル実行をシミュレートするステップは、好ましくは、

プログラムモデルエラー初期構成を、抽象構文木に対応するエラー初期構成に変換する第１の中間ステップと、

対応する初期構成を用いて抽象構文木を計算し、抽象構文木に対応する内部メモリの中間値を取得する第２の中間ステップと、

を含む。

【００３７】

抽象構文木表現は、命令を接続する論理ゲートにおける値等、ＰＬＣプログラムの単純な実行によって得ることができない中間値を生成することができるため、有利である。

【００３８】

一実施の形態によれば、元のプログラムをプログラムモデルに変換するステップは、静的単一代入変換の中間ステップを含む。静的単一代入変換は、好ましくは、抽象構文木表現に対して適用され、実行中に内部メモリ値を追跡するのに有利である。

【００３９】

一実施の形態によれば、一組のプロパティを決定するステップ中、一組のプロパティは、ダイクストラの最弱事前条件計算法を用いて、一組のプロパティに対する事前条件を決定するように計算され、一組のプロパティの検証は、その事前条件に基づいて実施される。事前条件は、プロパティに固有である基礎をなす前提である。事前条件が満たされることは、プロパティが検証され、その実行中にエラーが発生しないことを意味する。

【００４０】

一実施の形態によれば、連動性と結合された一組の述語の充足可能性は、充足可能性モジュロソルバを用いて検証され、これは、一階述語論理において論理式として表された決定問題である、充足可能性モジュロ理論問題を解くように構成された、自動化ソルバである。

【００４１】

本発明の別の態様によれば、少なくとも１つのプロセッサによって実行されると、上述したような方法を実行する命令を含むコンピュータプログラムが開示される。

【００４２】

好ましい実施の形態において、コンピュータプログラムは、演繹的プログラム検証プラットフォームにおいて実行される。

【００４３】

本発明は、本発明による、上記に定義したような方法のステップをコンピュータに実行させる、コンピュータプログラムを記憶する、非一時的コンピュータ可読媒体を更に目的とする。

【００４４】

本発明の別の態様によれば、上述したようなプログラマブルロジックコントローラプログラム分析方法を実行する装置が開示される。こうした装置は、一実施の形態において、

上述したコンピュータプログラム命令、及び場合により一時的な計算データ等の他のデータも記憶するメモリＭＥＭと、

メモリＭＥＭの内容を読み出すとともに、本発明による方法のステップを実行するプロセッサＰＲＯＣと、

場合により、プロセッサＰＲＯＣによって処理されるべき／処理されたデータを（ネットワーク又は他の任意のリンクを通して）受信／送信する入出力インターフェースＩＮＴと、

を備える（図７の例に示すような）処理回路ＰＣを備えることができる。

【００４５】

本発明の他の特徴及び利点は、非限定的な例として与えるその実施の形態のうちの１つの以下の詳細な説明から明らかとなり、且つ上述した図７よりも更に添付図面を参照して明らかとなる。

【図面の簡単な説明】

【００４６】

【図１】開示する方法に関与する例示的な一組のステップを示す図である。

【図２】開示する方法のステップを実施するグラフィカルインターフェースの一部を表す図である。

【図３】開示する方法の特定の実施形態におけるラダー図の一例を表す図である。

【図４ａ】開示する方法に関与する中間ステップの実行結果を示す図である。

【図４ｂ】開示する方法に関与する中間ステップの実行結果を示す図である。

【図５】開示する方法に関与する中間ステップの別の実行結果を示す図である。
【図６】開示する方法の実行の結果としてエラーシナリオを示す図である。
【図７】開示する方法を実行する装置を示す図である。
【発明を実施するための形態】
【００４７】
図において、同じ参照符号は同一の又は同様の要素を示す。
【００４８】
図１は、参照番号１０によって示す、ＰＬＣプログラム演繹的検証に関与する例示的な一組のステップを示す。この一組のステップは、ＰＬＣプログラムを分析及び検証する方法の一実施態様の全体的なアーキテクチャを表している。
【００４９】
図１に表す例では、本方法は７つのステップを含む。最初の６つのステップは、任意のＰＬＣプログラミング言語で記述されたＰＬＣプログラムに適用可能である。第７のステップは、ＰＬＣプログラムを記述するプログラミング言語に応じて実施される。図１は、本発明による方法の以下のステップを表している。
本方法の第１のステップ（ＴＲＡＮＳ１）の間に、ＰＬＣプログラム（ＰＲＯＧ）が、プログラムモデル（ＰＭＯＤ）に変換される；
第２のステップ（ＴＲＡＮＳ２）の間に、機能仕様が、仕様モデル（ＳＭＯＤ）に変換される。この仕様モデル（ＳＭＯＤ）は、プログラムモデル（ＰＭＯＤ）についての論理フレームワークで表され、仕様テンプレートと、ＰＬＣプログラムによって使用されるデバイスの選択とを使用して生成される。機能仕様は、有利には、当該機能仕様を形式的に表すのに使用される機能仕様テンプレートを表示するグラフィカルインターフェース上に表される。より正確には、機能仕様テンプレートは、ＰＬＣプログラムによって使用されるデバイスの選択とともにグラフィカルインターフェース上に表示される；
第３のステップ（ＰｒｅｄＴ）の間に、検証すべきプロパティが、プログラムモデル（ＰＭＯＤ）と、仕様モデル（ＳＭＯＤ）と、事前定義されたＰＬＣ言語形式とから生成される；
本方法の第４のステップ（ＳＭＴ）の間に、自動化ソルバが使用され、第３のステップにおいて生成されたプロパティが形式的に証明されるか、又は、これらのプロパティに対する反例が見つけられる；
第５のステップ（ＴＲＡＮＳＢ）の間に、存在する場合にはプロパティの反例（ＰＲＯＯＦ　ＮＯＫ）が、モデル構成、より詳細には初期モデル構成に変換される；
第６のステップ（ＥＸＥ）の間に、第５のステップにおいて得られたモデルの初期構成から内部メモリの全ての中間値を計算するために、ＰＬＣプログラムの実行がシミュレートされる。このステップは、全ての中間値が、第４のステップにおいて返されたプロパティの反例において提供された場合の任意選択のステップである；
第７のステップ（ＤＩＳＰ）の間に、モデル中間値が、中間値の情報を用いてＰＬＣプログラムに変換して戻される。使用されるＰＬＣ言語がラダーロジックであるとき、結果は、元のラダープログラムと同様に、グラフィカルにユーザに表示される。
【００５０】
本方法の第１のステップ（ＴＲＡＮＳ１）は、ＰＬＣプログラム（ＰＲＯＧ）をプログラムモデル（ＰＭＯＤ）に変換するものである、変換ステップである。
【００５１】
上記ＰＬＣプログラムＰＲＯＧは、以降ＰＬＣと称するプログラマブルロジックコントローラで実行可能である。ＰＬＣは、産業機械及びプロセスを制御するために、順序、計時、計数、算術、データ操作及び通信等の命令を記憶及び実行することができる。フィールドデバイスへのインターフェース回路は、入力接続及び出力接続の形態で提供される。
【００５２】
特定の実施形態において、ＰＬＣプログラムはラダーロジックで記述され、ラダー図はＰＬＣプログラムのシーケンス制御ロジックをグラフィカル図で表す。ラダー言語は、ラ

ダー図の一例を表す図３に示すようにリレー論理電気回路図を模倣するグラフィカル言語である。ラダーロジックは、実際には、ルールベース言語である。「ラング」とも称されるルールは、一組のデータにおける条件によってアクティベートされるとインスタンス化される。一組のデータは、上記アクティベーションに従って選択され、それらのルールに属するステートメントが実行される。リレー等の電気機械デバイスに実装される場合、プログラムを構成する様々なルールは、ソフトウェアの一部として連続ループで逐次実行される。１秒につき多数回ループを実行することにより、同時且つ即時の実行の効果を達成することができる。ラングは、プログラマブルコントローラの適切な動作を達成するために所与の順序で実行される。より詳細には、ラダープログラムは、連続的に、例えば１００ミリ秒毎に実行される１つのループを含むことができる。

【００５３】

ラング入力は、「接点」とも称される論理チェッカーである。いわゆる「接点」は、一体化された又は外部の入力モジュールを介する、押しボタン及びリミットスイッチ等の物理的デバイスからプログラマブルコントローラへの物理的又はハード入力を指すことができる。接点はまた、プログラムの別の場所で生成される可能性がある内部記憶ビットのステータスも表すことができる。

【００５４】

ラング出力は、「コイル」によって表されるアクチュエータである。「コイル」は、プログラマブルコントローラに接続された何らかのデバイスを動作させる物理的出力を表すことができ、又は、プログラムの別の場所で使用される内部記憶ビットを表すことができる。各接点又はコイルは、プログラマブルコントローラのメモリにおける単一ビットのステータスに対応する。これらの命令は、メモリにおける特定のビットアドレスのオン／オフステータスを検査し、内部出力及び外部出力の状態を制御する能力を提供する。電気機械リレーとは異なり、ラダープログラムは、単一ビットのステータスを任意の回数参照することができ、無限に多数の接点を有するリレーと等価である。

【００５５】

ラダー図では、ラングは、接続された命令のネットワークとして構成される。命令の間の接続は、上記命令の間の論理関係を表す。例えば、ラダーロジックにおいてＯＲロジックは２つの接点の並列接続で実装され、ＡＮＤロジックはラダーロジックで接点の直列接続として実装される。

【００５６】

第１のステップで動作する変換（ＴＲＡＮＳ１）は、変換アルゴリズムを実施することによって実行される。変換アルゴリズムは、ＰＬＣ言語プリミティブの事前定義されたモデル化を用いて、ＰＬＣプログラム（ＰＲＯＧ）を、論理フレームワークで表されるプログラムモデル（ＰＭＯＤ）に変換する。ＰＬＣ言語プリミティブは、接点によって表される論理チェッカー、コイルによって表されるアクチュエータ、機能ブロック、及びより一般的には任意の基本又は拡張ラダー言語命令等、ＰＬＣ論理回路を構成する論理命令要素からなる。

【００５７】

プログラムモデル（ＰＭＯＤ）は、好ましくは、一階述語論理の論理フレームワークで表される。一階述語論理は、命題論理の拡張であり、領域と称される世界の部分的な視野において命題が真であるか又は偽であるかを考慮する。一階述語論理は、アルファベット、一階述語言語、一組の公理及び一組の推論規則からなる。一階述語論理は数学的に証明可能であり得るため、計算問題を表現するのに有用である。一階述語論理は、構文論及び意味論から構成されている。一階述語論理の構文論は、概念を表すために用いられる形式言語であり、一階述語論理の意味論は、任意の一階述語論理式の値を決定する論理式である。

【００５８】

ＰＬＣプログラムのシーケンス構造を考慮すると、変換は、ループ自体に含まれる命令に対してのみ動作する。したがって、プログラムがループの連続的な実行を含まないため

、時相論理でのプログラムモデルの表現は不要である。変換の第１のステップ中、代数的データ型を用いて、プログラムの入力、内部メモリ及び出力がモデル化される一方、多相型を用いて、モデルプリミティブの数が因数分解される。

【００５９】

命令のモデル化は、述語を表現するように一階述語論理式として命令を表すことである。述語は、基本的に、二値変数及び非二値変数の二値関数である。実行時における入力及びローカルメモリの認可された値は、こうした述語から得ることができる。命令モデル化は、実行時の入力及びローカルメモリの認可された値、すなわち、命令が実行されるときにエラーを発生させない値を表す、一階述語論理式として表現されるプロパティに関連付けることができる。これらの論理式は、命令及び論理式が参照するエラー理由のような更なる情報にリンクさせることができる。

【００６０】

ＰＬＣプログラムモデル（ＰＭＯＤ）は、上記述語を含む数学的記述を用いて生成される。言い換えれば、ＰＬＣプログラムの目的は、選択された数理論理学における一組の述語を用いて表される。こうした数学的記述の検証により、プログラムの計算手法が正確であることが確実になる。こうした数学的記述の検証により、計算手法が、計算することが期待されることを実行する理由が明らかとなる。こうした検証は、証明とも称され、数学的記述に関連するプログラムの実行の安全性及び健全性を保証する。プログラム実行中に、ランタイムエラー（メモリへの不正アクセス若しくはオーバーフロー、０によって除算しようとする試みのような不正操作、又は、ループの連続的な実行を含む他のタイプのプログラムの場合は、無限ループのような終了問題等）がないことが証明された場合に、実行の安全性が保証される。プログラム健全性は、機能正確性とも称され、プログラムが行うように想定されることを行うことを検証することである。

【００６１】

形式論理では、論理システムは、システムにおいて証明することができる全ての論理式が、システムの意味論に関して論理的に妥当である場合にのみ、健全性を有する。言い換えれば、システムは、その定理の全てがトートロジーであるとき、健全である。演繹的システムの健全性は、その演繹的システムにおいて証明可能である任意の文が、意味論的理論の、その理論が基づく言語に対する全ての解釈又は構造において真でもあるという特性である。

【００６２】

好ましい実施形態において、第１のステップ（ＴＲＡＮＳ１）は第１の中間ステップ（ＡＳＴ－ＴＲＡＮＳ）を含む。そこでは、ＰＬＣプログラムは最初に、ＡＳＴ表現又は構文木としても知られ、以降、頭字語ＡＳＴと称する、抽象構文木として表現される。次いで、第２の中間ステップ（ＭＯＤ－ＴＲＡＮＳ）において、ＰＬＣプログラムのＡＳＴ表現からＰＬＣモデルが生成される。

【００６３】

抽象構文木は、プログラミング言語で書かれるソースコードの抽象構文構造の木表現である。木の各ノードは、本実施形態においてＰＬＣプログラムである、ソースコードで発生する構成体を示す。

【００６４】

ＡＳＴは、それが含有する全ての要素に対するプロパティ及びアノテーション等の情報により編集及び強化することができる。こうした編集及びアノテーションは、ＰＬＣプログラムのソースコードでは、ソースコードを変更することを意味するため、不可能である。

【００６５】

ＰＬＣプログラムと比較すると、そのＡＳＴは、実際の構文に現れる全ての詳細は含まず、波括弧、セミコロン又は丸括弧等、構造的且つ内容に関連する詳細のみを含む。ｉｆ条件ｔｈｅｎ表現のような構文構成体は、３つの分岐とともに単一のノードを用いて示すことができる。

【００６６】
ＡＳＴには、通常、コンパイラによる連続的な分析段階のために、プログラムに関する追加の情報が含まれる。ＰＬＣプログラムをＡＳＴで表現することは、本方法の続くステップにおいて有用である、ＰＬＣプログラムにおける各要素の位置を記憶することを可能にするため、有利である。この中間ステップはまた、ＡＳＴの完全なトラバーサルにより概してプログラムの正確性の検証が可能であるため、興味深い。
【００６７】
本実施形態において、上記ＡＳＴ表現は、好ましくは、ＯＣＡＭＬ等の関数型言語で記述される。
【００６８】
好ましい実施形態において、変換アルゴリズムは第３の中間ステップを含み、そこでは、静的単一代入変換（ＳＳＡＴ）を用いて、プログラムの実行中に内部メモリ値を追跡する。ＰＬＣ言語等の命令型プログラミング言語では、代入により、変数は、それらの寿命及びスコープの間の異なる時点で異なる値を保持することができる。静的単一代入変換は、各実行段階において代入される値を追跡するために有利である。命令型言語で書かれたプログラムに対して演繹的検証を実施するために、関数型モデルへの変換を実施することができ、実施する場合、静的単一代入変換により、モデルに対して、より詳細にはＡＳＴ表現に対して実施される。静的単一代入変換は、ＰＬＣプログラム要素をラダープログラム要素に連結するために、モデル要素に、コード位置のような情報を追加するのを可能にする、連結ステップとして見ることができる。したがって、元のＰＬＣプログラム（ＰＲＯＧ）まで遡ることを、容易に実施することができる。図４ｂは、こうした静的単一代入変換の一例を提供し、そこでは、モデル実行の異なるステップにおける整数Ｄ１の値は、（ｄ１＿１，ｄ１＿２，ｄ１＿３）の値として記憶される。この変換に相当するものが、図４ａのラダー図に表現されている。
【００６９】
図２は、第２のステップを実行するために使用される一例示的なグラフィカルインターフェースの抜粋を示している。
【００７０】
仕様モデルへのユーザ仕様の変換の第２のステップ（ＴＲＡＮＳ２）は、有利には、ＰＬＣプログラムの機能仕様を完全且つ自動的に表して指定するグラフィカルユーザインターフェースを通じて実施される。この第２のステップは、演繹的検証ツールを使用することによって、ＰＬＣプログラムの機能仕様を網羅的に表して指定することを可能にする。
【００７１】
特定の実施形態において、仕様モデルへのユーザ仕様の変換の第２のステップ（ＴＲＡＮＳ２）は、ユーザが機能仕様を表しているときに実行される。有利な実施形態において、ユーザは、グラフィカルインターフェースを通じて機能仕様を表す。機能仕様がグラフィカルインターフェースを通じて入力されているときに、プログラムモデルに基づく機能仕様テンプレートをユーザに提案することができる。機能仕様を表すことが容易になり、曖昧性がなくなる。機能仕様テンプレートは、有利には、ＰＬＣプログラムによって使用され、したがって機能仕様を表すのに使用することができるデバイスのリストとともにユーザに提案される。図２において、部分Ａは、ユーザがクリックを用いて選択することができるデバイスのリストを表している。
【００７２】
機能仕様の表現及び検証の従来の技法と比較して、このステップは、ユーザに提案されるテンプレートを使用することによって、機能仕様を論理プロパティの形で正確に明記することを可能にする。その結果、仕様モデル（ＳＭＯＤ）は、機能仕様に対応する論理プロパティに基づいて生成され、論理フレームワークで表される。
【００７３】
機能仕様が仕様モデルに変換されているときに、ライブフィードバックがユーザに与えられる。ライブフィードバックは、図２の部分Ｂに示すような自然言語で与えられるとと

もに、図２の部分Ｃに示すようなラダー型言語でも与えられる。このように、ユーザは、機能仕様を曖昧性なく表すことができる。元のＰＬＣプログラムに使用されるプログラミング言語にかかわらず、表された機能仕様に関するフィードバックは、ラダー図等、グラフィカルにユーザに与えることができる。

【００７４】

本方法の第３のステップ（ＰｒｅｄＴ）は、ＰＬＣプログラムモデル（ＰＭＯＤ）からプロパティ（Ｐｒｏｐ）を生成することである。第３のステップは、プログラムモデル（ＰＭＯＤ）を事前定義された言語形式（ＬＦｏｒｍ）と結合して、上記プログラムモデル（ＰＭＯＤ）に関連するプロパティ（Ｐｒｏｐ）を得る。次いで、上記プロパティに対する条件（Ｃｏｎｄ）が得られる。

【００７５】

こうしたプロパティは、プログラムモデル実行において実施される動作を表し、各実行段階において検証されるように予期される。それらは、各再帰呼出し時にループ不変条件を検証することに類似している。上記プロパティに対する条件は、２つのカテゴリー、すなわち、事前条件及び事後条件に分類することができる。

【００７６】

事前条件（ＰｒｅＣｏｎｄ）は、プロパティ（Ｐｒｏｐ）に固有であるとともに、その実行の前に検証されるべきであり、そうでなければエラーが発生する可能性がある、基礎をなす前提を示す。より詳細には、ルーチンが呼び出されると、上記ルーチンプロパティに対応する事前条件は、上記ルーチンプロパティが検証されるように満たされると想定される。

【００７７】

例えば、階乗関数プログラムは、変数に対して実施される再帰的計算法を表すループを含む。こうしたループの事前条件は、階乗が計算される変数が、ループの開始時は正の整数でなければならないということである。

【００７８】

事後条件は、事前条件が検証されてルーチンが呼び出されたときに予測される結果を表す。したがって、ルーチンプロパティの事前条件は、プロパティ計算時にルーチンの最後に予測された結果から演繹することができる。ループの場合、事前条件及び事後条件は、特に、ループの連続的な実行を含むプログラムにおいて、密に関連するか又は更には類似し、ループ不変条件を形成する。ループの実行段階において事後条件が満たされる場合、次の段階の事前条件も同様に満たされることが保証される。

【００７９】

そのため、この第３のステップ（ＰｒｅｄＴ）は、ダイクストラの最弱事前条件計算法を用いて、上記プロパティを計算するとともに、モデルプロパティ（Ｐｒｏｐ）に関連する事前条件（ＰｒｅＣｏｎｄ）を決定する。図５に、第３のステップにおいて事前条件を得るために生成及び計算されたプロパティの一例を示す。図５のプロパティは、以下のように読むことができる。すなわち、「Ｘに記憶された値が「オン」／「真」である場合、Ｄ１に記憶された値は、０よりも大きく且つ９９９９よりも小さい」である。第３のステップにおいて生成及び計算することができるプロパティの別の例は以下のように表される。

【数１】

$$||X|| \rightarrow (Y1 \text{ and } (\text{not } Y2)) \text{ or } ((\text{not } Y1) \text{ and } Y2)$$

【００８０】

このプロパティは、「Ｘに記憶された値が「オン」／「真」である場合に、出力Ｙ１及びＹ２のうちの一方のみが、記憶された値「オン」／「真」を有する」ことを表す。

【００８１】

一階述語論理プロパティに対するダイクストラの最弱事前条件計算法の実行により、こ

うした計算が健全でありながら、計算されたプロパティのサイズを最小限にすることが確実になる。事前条件は、各プロパティの定義領域を提供する。このステップにより、事前条件を満たすことが、ルーチンプロパティが持続することと、その実行中にエラーが発生しないこととを意味するため、証明の健全性が保証される。したがって、プロパティが充足される場合、又は言い換えれば、上記プロパティの対偶が充足可能でない場合、対応するＰＬＣプログラムを実行するときにエラーは発生する可能性がない。

【００８２】

有利な実施形態において、ユーザ仕様（ＦＵＮＣ）から、他のプロパティが表される。ユーザ仕様は、機能仕様とも称され、プログラムの予測された動作、すなわち、プログラムが、プログラムユーザによって必要とされるものを満たすように実行するべき機能とともに、入力及び出力の要求されたプロパティを記述する。例えば、産業背景において、ロボットアームは、センサーが所与の範囲内の信号を受信しているときにのみ動作するように要求される場合がある。ユーザ仕様の他の例は、次のタイプの条件、すなわち「２つ以上の特定の出力が一度にアクティベートされるべきでない」、「１つ以上の特定の入力がアクティベートされる場合に、１つ以上の特定の出力がアクティベートされるべきである／されるべきでない」、「特定の出力のうちの１つの出力のみが一時にアクティベートされるべきである」を含む。

【００８３】

書かれているプログラムは、上記仕様に従うように想定されているが、ユーザ仕様から導出される計算プロパティは、書かれたプログラムがランタイムエラーを含む可能性があるため、追加の安全措置である。これらのプロパティは、連動性（ＩｎｔＰｒｏｐ）であり、一階述語論理において機能仕様を表すことによって得られる。より詳細には、連動性（ＩｎｔＰｒｏｐ）は、第２のステップ（ＴＲＡＮＳ２）において得られる仕様モデル（ＳＭＯＤ）から得られる。上記連動性は、満たすべき追加の事後条件としてモデルのプロパティを用いて計算される。したがって、事前条件（ＰｒｅＣｏｎｄ）は、これらの連動性から同様に、ダイクストラの最弱事前条件計算法を用いて導出することができる。

【００８４】

第１のステップ、第２のステップ及び第３のステップにおいて、上位の論理フレームワークも用いることができるが、一階述語論理は、概して、ＰＬＣモデルから導出されるプロパティ（Ｐｒｏｐ）と対応する機能仕様（ＩｎｔＰｒｏｐ）とを表すのに適している。上位の論理フレームワークは、最終的応用に関する不必要な複雑性を導入する可能性がある。さらに、一階述語論理プロパティの生成は、上位の論理フレームワークと比較して自動化がより容易である。

【００８５】

機能仕様から導出されるプロパティの計算により、モデルのプロパティが充足されるとき、実行時に、機能仕様の違反から導出されるエラーが発生しないことが確実になる。

【００８６】

第３のステップは、述語変換としても知られ、好ましくは、入力としてプログラムモデルを受け取るとともに様々なプログラムの形式的証明を生成する適切なツールを含む、演繹的プログラム検証プラットフォーム（ＰＬＡＴ）において計算される。第１のステップもまた、好ましくは、演繹的プログラム検証プラットフォーム（ＰＬＡＴ）において計算される。より詳細には、第１のステップの静的単一代入変換を演繹的プログラム検証プラットフォーム（ＰＬＡＴ）によって行うことができる。命令モデル化は、外部モデル化ツールを通して実施することができ、外部モデル化ツールは、上記演繹的プログラム検証プラットフォームにおいてダウンロードされるとともに、ＰＬＣ言語の事前定義されたテンプレート及びモデルを含む、ライブラリとすることができる。有利な実施形態において、上記プラットフォーム（ＰＬＡＴ）は、ＷＨＹ３のタイプであり、プログラムモデル（ＰＭＯＤ）は、ＷＨＹＭＬのタイプのプログラミング言語で表現される。

【００８７】

本方法の第４のステップ（ＳＭＴ）は、第３のステップにおいて生成されたプロパティ

を形式的に証明するか、又は、これらのプロパティに対する反例を見つけるために、自動化ソルバを用いる。上記生成されたプロパティは、それらが常に充足される場合、又は、その対偶が充足可能でない場合、形式的に証明される。上記プロパティ（Ｐｒｏｐ）又はその対偶の充足可能性は、それらの先行定義された事前条件（ＰｒｅＣｏｎｄ）に従って評価される。

【００８８】

好ましい実施形態において、第４のステップは、充足可能性モジュロ理論（ＳＭＴ）問題に対して自動定理証明器によって実施され、この自動定理証明器を用いて、多数の組込み論理的理論及びそれらの組合せにおいて一階述語論理式の妥当性（又は、二元的に、充足可能性）を証明することができる。こうしたソルバは、論理式として一階述語論理において表される決定問題である、充足可能性モジュロ理論問題を解くように構成されている。ＳＭＴ問題は、命題変数が別の数学的理論の論理式に置き換えられるＳＡＴ問題である。より具体的には、ＳＭＴ問題は、追加の解釈を含む一階述語論理における論理式である一組のＳＭＴインスタンスとして表される。

【００８９】

第３のステップにおいて先行して用いられた演繹的プログラム検証プラットフォームは、概して、自動ソルバを同様に含む。したがって、上記自動ソルバを用いて、第４のステップを同様に計算することができる。用いることができるソルバの一例は、好ましくは、ＣＶＣ４である。別の例は、Ｚ３及びＡｌｔ－Ｅｒｇｏとすることができる。ソルバは、証明を実行するとともに反例を提供するように、プラットフォームと適切に相互作用するように構成されなければならない。上記相互作用は、証明又は反例が必要とされるモデル要素の指示を含む。上記プラットフォーム（ＰＬＡＴ）がＷＨＹ３である好ましい実施形態において、証明又は反例が必要とされるモデル要素は、図４ｂにおいて「model-trace」というラベルで見ることができるように、ＷＨＹＭＬ言語において特定の関数によってラベル付けすることができる。

【００９０】

解は、背景理論に関して発見的に実施され、ＳＭＴインスタンスが充足可能であるか否かを判断することである。自動定理証明器は、有利には、反例を提供するための重要な特徴であるモデル生成能力を有する。第３のステップから得られるプロパティは、少なくとも整数線形算術の理論、好ましくは、レコード、線形実数算術及び弦の理論に関して解かれる。有利な実施形態において、自動定理証明器は、多相型の一階述語論理に基づき、好ましくは、有理数及び整数の線形算術、配列、タプル、レコード、帰納的データ型、ビットベクトル、弦、及び未解釈関数シンボルに関する等式（equality over uninterpreted function symbols）等、組込み基礎理論を含む。より好ましい実施形態において、こうした自動定理証明器は、量子記号のためのサポートを更に含む。

【００９１】

第３のステップで得られたプロパティ（Ｐｒｏｐ、ＩｎｔＰｒｏｐ）が、ＳＭＴソルバを使用することによってその計算時に充足された場合、ＳＭＴソルバにより、プロパティが充足されているという証明を表す応答（ＰＲＯＯＦ　ＯＫ）が生成される。次いで、プログラム（ＰＲＯＧ）の命令が、生成時にエラーをもたらす可能性がないこと、及び／又はユーザによって表された機能仕様（ＦＵＮＣ）が、ＰＬＣプログラム（ＰＲＯＧ）の全てのあり得る実行に対して持続することが確実になる。したがって、プログラム（ＰＲＯＧ）の健全性が論証される。

【００９２】

その反対に、プロパティ（Ｐｒｏｐ、ＩｎｔＰｒｏｐ）の対偶が充足可能である場合、ソルバによってプロパティに対する反例が提供される。ＳＭＴソルバにより、上記反例を表す応答（ＰＲＯＯＦ　ＮＯＫ）が生成され、それは、上記プロパティが充足されていない場合をもたらすモデル構成に対応する。特に、反例の内容は、形式的証明が事前条件（ＰｒｅＣｏｎｄ）に基づいて評価されるため、少なくとも初期構成を指す。

【００９３】

本方法の第５のステップは、得られたプロパティ反例（ＰＲＯＯＦ　ＮＯＫ）をモデル構成に、より詳細には初期モデル構成に変換する（ＴＲＡＮＳＢ）。上記モデル構成は、モデルの入力及び内部メモリ値を含む。この変換は、上記事前定義された言語形式（ＬＦｏｒｍ）に基づいて実施され、第３のステップにおけるプログラムモデル（ＰＭＯＤ）についてのプロパティ生成プロセスの逆のプロセスとして動作する。

【００９４】

得られたプロパティ反例は、初期モデル構成（ＩｎｉＣｏｎｆ）及び中間モデル構成（ＩｎｔＣｏｎｆ）の両方に対応するデータを含み、その理由は、プログラムモデル及び演繹的検証プロセスが関数型であるためである。上記初期モデル構成（ＩｎｉＣｏｎｆ）は、プログラムモデル（ＰＭＯＤ）の実行の開始時に、上記プロパティが充足されない場合をもたらすモデルの入力及び内部メモリの初期値を含む。上記中間モデル構成は、プログラムモデル（ＰＭＯＤ）の実行の開始と、上記プロパティが充足されないモデル位置との間の内部メモリの中間値を含む。

【００９５】

しかしながら、上記反例は、実行中に生成される中間値の全ては含まない。例えば、ラダープログラムの実行中に、ＰＬＣプログラム変数Ｘに対して４つの値を代入することができる。したがって、４つの値は、変数Ｘを表す４つの中間変数（Ｘ１、Ｘ２、Ｘ３、Ｘ４）に代入される。得られた反例は、場合によっては、中間変数（Ｘ１、Ｘ２、Ｘ３、Ｘ４）の全てに対応する値を返さない可能性がある。

【００９６】

後に詳述するように、この場合を克服する好ましい方法は、得られた反例から初期構成を選択することと、反例から選択された上記初期構成（ＩｎｉＣｏｎｆ）によりプログラムモデルを実行することによって中間変数（Ｘ１、Ｘ２、Ｘ３、Ｘ４）の値を再計算することとである。

【００９７】

第６のステップは、第５のステップにおいて得られたモデルの初期構成から内部メモリの中間値を計算するために、ＰＬＣプログラムの実行（ＥＸＥ）をシミュレートすることである。

【００９８】

好ましい実施形態において、第６のステップは、第５のステップで得られたモデル初期構成（ＩｎｉＣｏｎｆ）を、好ましくはＯＣＡＭＬ言語で表されるＡＳＴ表現の対応する初期構成（ＡＳＴ－ＩｎｉＣｏｎｆ）に変換することである、第１の中間ステップ（ＡＳＴ－ＴＲＡＮＳＢ）を含む。こうした変換は、第１のステップで動作したようなＡＳＴ表現のモデルへの変換から導出することができる。

【００９９】

第６のステップは、ＡＳＴ表現内部メモリ（ＡＳＴ－ＩｎｔＣｏｎｆ）の中間値を回収及び記録するために、上記対応する初期構成（ＡＳＴ－ＩｎｉＣｏｎｆ）を用いて、シミュレーションされた実行エンジンによってＡＳＴ表現を計算することである。ＡＳＴ表現実行は、ＰＬＣプログラム実行のシミュレーションとして動作する、第２の中間ステップ（ＡＳＴ－ＳｙＥｘ）を含む。中間値の収集は、エラー又は仕様違反が発生する実行の箇所まで実施される。第６のステップは、上記演繹的プログラム検証プラットフォームにおいて同様に実施することができる。

【０１００】

結果として、ＡＳＴ表現は、ＰＬＣプログラムの実行によって得ることができない中間値の生成を可能にするため、有利である。中間値を得ることができない１つの理由は、プログラムモデルでは、命令を接続する論理ゲートが変数として表現されないということである。したがって、こうしたコード位置において発生するエラーは、返される反例において明確とはならない。第１のステップにおける変換時、これらのコード位置における値は失われ、第４のステップ時に提供される反例値から回収されない。しかしながら、これらのコード位置における値は、エラーをもたらしたものの適切な解釈のために必須である。

第６のステップは、静的単一代入が適用されたＡＳＴ表現の実行により、これらのコード位置における値の回収も可能にする。

【０１０１】

この第６のステップは、モデルエラーシナリオを提供し、これを通して、エラー又は仕様違反をもたらす値が収集される。モデルエラーシナリオは、ＡＳＴ表現構成（ＡＳＴ－ＩｎｉＣｏｎｆ、ＡＳＴ－ＩｎｔＣｏｎｆ）に基づく。

【０１０２】

第７のステップ（ＤＩＳＰ）は、モデルエラーシナリオをＰＬＣプログラムに戻すように変換する。ＡＳＴ表現構成（ＡＳＴ－ＩｎｉＣｏｎｆ、ＡＳＴ－ＩｎｔＣｏｎｆ）は、ＰＬＣプログラムの対応する構成に戻るように変換される。ＰＬＣプログラム分析方法の第７のステップは、言語に焦点を当てたものであり、ＰＬＣプログラムを記述するプログラミング言語に応じて実施される。第１のステップ（ＴＲＡＮＳ１）の間に生成され、後続のステップの種々のプロセス及び変換の間に保持される、ＡＳＴに含まれる元のプログラムについての余分な情報によって、本方法は、エラーシナリオをプログラマに返すことができる。この情報は、好都合なことに、プログラマが機能仕様違反又はランタイムエラーがなぜ、いつ発生するのか並びにどのようにＰＬＣプログラムを修正するかを理解するための表示で強化されたプログラムそのものの形態で表される。ラダープログラミングの特定の場合には、エラーシナリオのこの表現は、グラフィカルに表示することができる。

【０１０３】

図６はエラーシナリオを示し、エラーシナリオは、二値に対する色と、上記エラーシナリオの整数値に対するラベルとを含む。ＰＬＣプログラムがラダープログラミング言語で記述される場合、ラダープログラムは、図６に示すように、グラフィカルに表示され、プログラムにおけるエラー位置、上記エラーの前の、例えば、色強調表示によって示される実行パスと、プログラム実行の開始からエラー又は仕様違反までの入力及び内部メモリの値に関する情報により、拡充されている。エラーシナリオは、第１のステップにおいてプログラムモデルに追加されたエラー位置及び理由情報とともに、見つけられたエラーとそれを修正する方法とに関する非常に充実した情報を与える。

【０１０４】

図７は、上記で説明したようなプログラマブルロジックコントローラプログラム分析方法を実行する装置を示している。こうした装置は、図示した実施形態において、処理回路ＰＣを含み、この処理回路ＰＣは、

本発明による方法を実行する命令を含むとともに、場合によっては一時的な計算データ等の他のデータも含むコンピュータプログラムを記憶するメモリＭＥＭと、

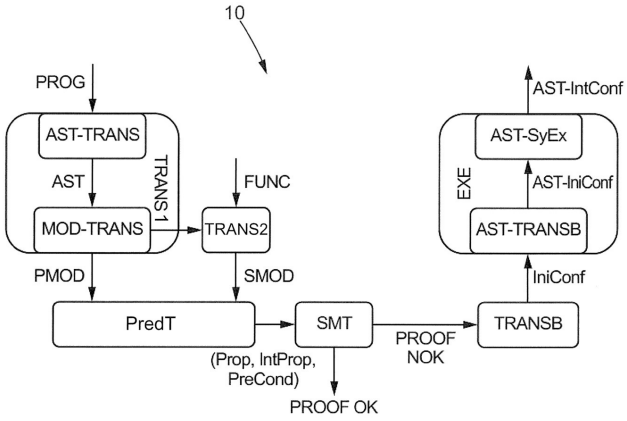
メモリＭＥＭの内容を読み出し、本発明による方法のステップを実行するプロセッサＰＲＯＣと、

場合によっては、プロセッサＰＲＯＣによって処理されるべき／処理されたデータを（ネットワーク又は他の任意のリンクを通じて）受信／送信する入出力インターフェースＩＮＴと、

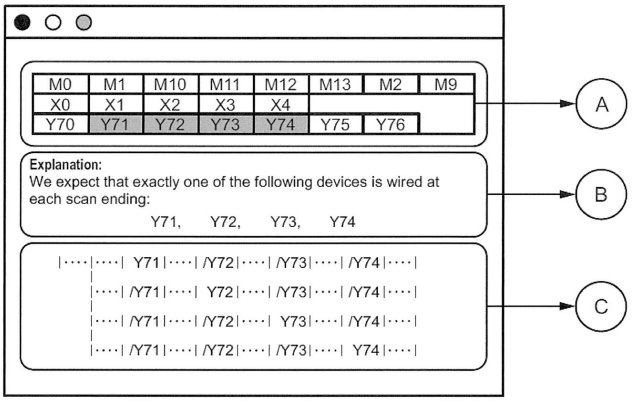
を備える。

【図面】

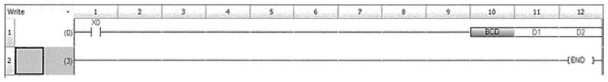
【図１】



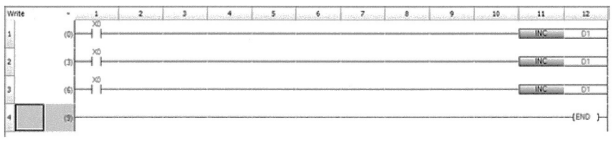
【図２】



【図３】



【図４ａ】



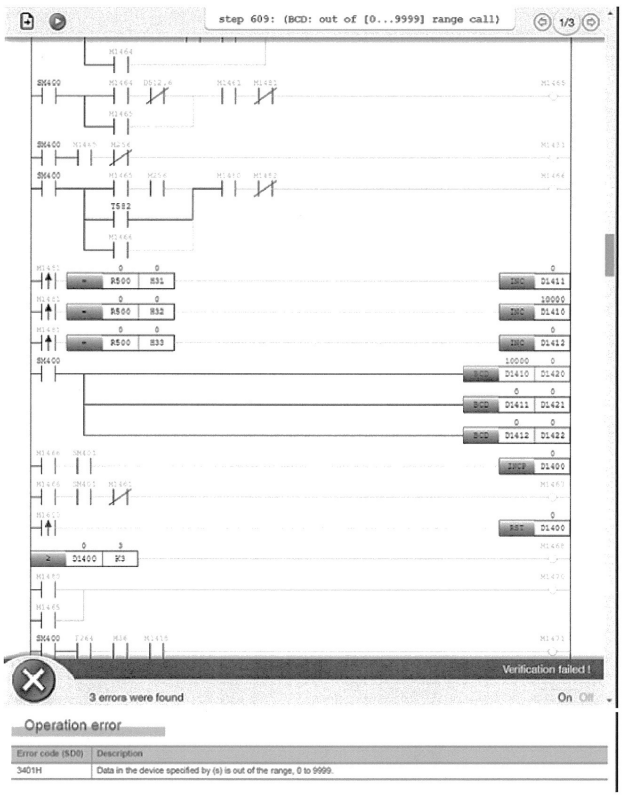
【図４ｂ】

```
module ProgPou

use export ladder_overflow.Ladder

type inputs = {
    x0 "model" "model_trace:.X0" : pulse
  }

type in_outputs = {
    d1 "model" "model_trace:.D1" : int
  }

type outputs = int

type result_t = {
    f_outs "model" "model_trace:.F_outs" : outputs;
    f_in_outs "model" "model_trace:.F_in_outs" : in_outputs
  }

let progPou_scan (last_outs "model" "model_trace:new" : outputs)
                 (last_in_outs "model" "model_trace:last" : in_outputs)
                 (ins "model" "model_trace:last" : inputs)
  requires { "model_vc" ((0 <= last_in_outs.d1) /\ (last_in_outs.d1 <= 65535)
             )
  }
  =
  let common_1 = ins.x0 in
  let d1_1 = "expl:16" inc_16u (False) (common_1) (last_in_outs.d1) in
  let common_2 = ins.x0 in
  let d1_2 = "expl:16" inc_16u (False) (common_2) (d1_1) in
  let common_3 = ins.x0 in
  let d1_3 = "expl:16" inc_16u (False) (common_3) (d1_2) in
  { f_outs = 0; f_in_outs = { d1 = d1_3 } }

end
```

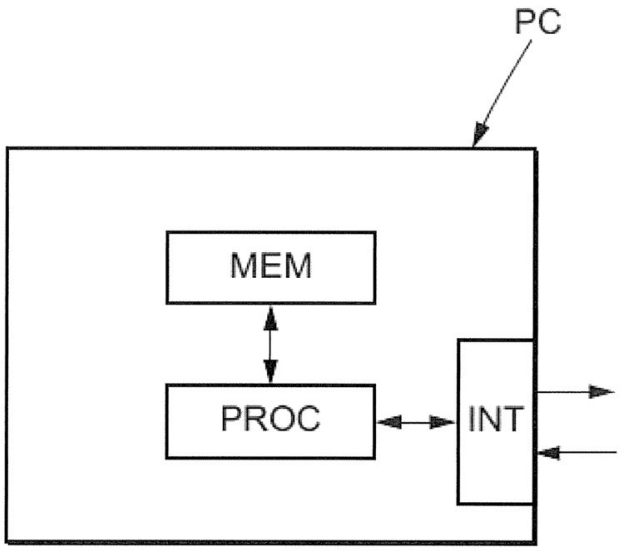
【図５】

```
|| X ||      ->      0 <= || D1 || <= 9999
```

【図６】



【図７】



【手続補正書】
【提出日】令和４年６月９日(２０２２.６.９)
【手続補正１】
【補正対象書類名】特許請求の範囲
【補正対象項目名】全文
【補正方法】変更
【補正の内容】
【特許請求の範囲】
【請求項１】
プログラマブルロジックコントローラプログラム分析方法であって、
プログラマブルロジックコントローラプログラムのタイプの元のプログラムを論理フレームワークのプログラムモデルに変換するステップと、
少なくとも前記プログラムモデルに基づいて、ユーザ仕様を論理フレームワークの仕様モデルに変換するステップと、
少なくとも前記プログラムモデルと事前定義された言語形式とから、前記元のプログラムの内部変数に関する一組のプロパティを決定するステップと、
自動化ソルバによって、前記仕様モデルから得られる連動性と結合された前記一組のプロパティの充足可能性を検証し、前記一組のプロパティについてのプロパティの対偶が充足可能である場合には、前記プロパティの対偶が充足可能であるプログラムモデルの入力及び内部メモリ値を表す一組の反例を提供し、又は、前記一組のプロパティが常に充足されている場合には、その確認を提供するステップと、
反例を前記プログラムモデルのエラー初期構成に変換するステップであって、前記エラー初期構成は入力及び内部メモリの初期値を含む、反例を変換するステップと、
前記プログラムモデルの実行を、前記プログラムモデルのエラー初期構成を用いてシミュレートし、実行の開始から前記プロパティの違反までの前記プログラムモデルのシミュレーションのエラー中間構成を記録するステップであって、前記エラー中間構成は内部メモリの中間値を含む、モデル実行をシミュレートするステップと、
前記プログラムモデルの前記エラー初期構成及び前記プログラムモデルのシミュレーションの前記エラー中間構成を前記元のプログラムのエラー初期構成及びエラー中間構成に変換するステップと、
前記元のプログラムのエラー初期構成及びエラー中間構成を表示するステップと、
を含む、プログラマブルロジックコントローラプログラム分析方法。
【請求項２】
ユーザ仕様を仕様モデルに変換する前記ステップは、前記ユーザ仕様が、機能仕様テンプレートと、前記プログラマブルロジックコントローラプログラムによって使用されるデバイスの選択とを使用して表される中間ステップを含む、請求項１に記載のプログラマブルロジックコントローラプログラム分析方法。
【請求項３】
前記元のプログラムを前記プログラムモデルに変換するステップは、抽象構文木としての前記元のプログラムの表現の第１の中間ステップと、前記抽象構文木（ＡＳＴ）についての前記プログラムモデルの生成の第２の中間ステップとを含む、請求項１又は２に記載のプログラマブルロジックコントローラプログラム分析方法。
【請求項４】
前記モデル実行をシミュレートするステップは、
前記プログラムモデルのエラー初期構成を、前記抽象構文木に対応するエラー初期構成に変換する第１の中間ステップと、
前記対応する初期構成を用いて前記抽象構文木を計算し、前記抽象構文木に対応する内部メモリの中間値を回収する第２の中間ステップと、
を含む、請求項３に記載のプログラマブルロジックコントローラプログラム分析方法。
【請求項５】

前記プログラムモデルは、一階述語論理フレームワークで表される、請求項１～４のいずれか一項に記載のプログラマブルロジックコントローラプログラム分析方法。
【請求項６】
前記元のプログラムを前記プログラムモデルに変換するステップは、静的単一代入変換の中間ステップを含む、請求項１～５のいずれか一項に記載のプログラマブルロジックコントローラプログラム分析方法。
【請求項７】
前記一組のプロパティを決定するステップ中、前記一組のプロパティは、ダイクストラの最弱事前条件計算法を用いて、前記一組のプロパティに対する事前条件を決定するように計算され、前記一組のプロパティの検証は、その事前条件に基づいて実施される、請求項１～６のいずれか一項に記載のプログラマブルロジックコントローラプログラム分析方法。
【請求項８】
連動性と結合された前記一組のプロパティの充足可能性は、充足可能性モジュロソルバを使用して検証される、請求項１～７のいずれか一項に記載のプログラマブルロジックコントローラプログラム分析方法。
【請求項９】
プロセッサによって実行されると、請求項１～８のいずれか一項に記載の方法を実行する命令を含むコンピュータプログラム。
【請求項１０】
請求項９に記載のコンピュータプログラムを実行する装置であって、前記コンピュータプログラムを保存するメモリと、前記メモリと接続された前記プロセッサとを有する、装置。

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No
PCT/JP2020/049302

A. CLASSIFICATION OF SUBJECT MATTER
INV. G05B19/05 G06F8/41
ADD.

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)
G05B G06F

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, INSPEC, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN 110 674 049 A (CEC INTELLIGENT TECH CO LTD) 10 January 2020 (2020-01-10) claims 1-3	1-10
A	CN 109 032 056 A (UNIV EAST CHINA NORMAL; SHANGHAI FORMAL TECH INFORMATION TECH CO LTD) 18 December 2018 (2018-12-18) claim 1	1-10

☐ 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
26 April 2021	07/05/2021

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Messelken, M

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/JP2020/049302

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
CN 110674049 A	10-01-2020	NONE	
CN 109032056 A	18-12-2018	NONE	

Form PCT/ISA/210 (patent family annex) (April 2005)

フロントページの続き

MK,MT,NL,NO,PL,PT,RO,RS,SE,SI,SK,SM,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,KM,ML,MR,NE,SN,TD,TG),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

(74)代理人　100111648
弁理士　梶並 順
(74)代理人　100122437
弁理士　大宅 一宏
(74)代理人　100147566
弁理士　上田 俊一
(74)代理人　100161171
弁理士　吉田 潤一郎
(72)発明者　クージノー、ドゥニ
フランス国、３５７０８ レンヌ・セデックス ７、セーエス １０８０６、アレ・ドゥ・ボーリュー １、ケアオブ・ミツビシ・エレクトリック・アールアンドディー・センター・ヨーロッパ

Ｆターム（参考）　5B042　GB05 HH08 HH41 HH49
5H220　AA04 BB07 CC06 CC07 CX02 DD01 DD04 JJ27 JJ42 JJ57