明　細　書

発明の名称 ：

論理結合型プログラムが実行時に発生する非同期型アルゴリズムの正統性の検証を実施可能にする方法

技術分野

[0001]　論理結合型プログラムが実行時に発生する非同期型アルゴリズム（ＮＳＡ）が不正値を発症させる論理結合型プログラムのソース（ＬＣＰＳ）の構文（バグ構文）を特定する方法に関する。

背景技術

[0002]　実行時に同期型アルゴリズム（ＳＡ）を発生させるシナリオ関数（ＳＦ）の全ソース（ＳＦＳ）を決定するためのＬＹＥＥ理論を開示している文献として、特許文献１乃至３及び非特許文献１乃至３がある。また、ＳＦＳを開示している文献として、特許文献４乃至６及び非特許文献４及び５がある。

[0003]　ＬＹＥＥ理論は意味の最小単位を論理原子、その構造を「ベクトル」として、ＳＦＳを決定するための生命作用のモデル論である。

先行技術文献

特許文献

[0004] 特許文献1：米国特許第６５３２５８６号明細書

特許文献2：特許第３１３３３４３号公報

特許文献3：欧州特許第０９４７９１６号明細書

特許文献4：特許第５９９２０７９号公報

特許文献5：特許第６０８６９７７号公報

特許文献6：米国特許第１０２３５５２２号明細書

非特許文献

[0005] 非特許文献1：Fumio Negoro, ”Lyee’s Hypothetical World”, ”New Trends in Software Methodologies Tools and Techniques”84 of Frontiers in Artificial Intelligence and Applications, pp. 3-22, IOS Press, Septembe

r 2002

非特許文献2：根来文生、「コンピュータウイルスを無力化するプログラム革命（ＬＹＥＥ）」日本地域社会研究所、２０１４年１０月

非特許文献3：根来文生、「ＬＹＥＥ理論の要約」、［online］、エムティインターナショナル株式会社、［令和１年９月１２日検索］、インターネット＜URL： https://mtiinc.jimdo.com/no190907/ｌｙｅｅ理論の要約/＞

非特許文献4：根来文生、「シナリオ関数の原理模型」、［online］、エムティインターナショナル株式会社、［令和１年９月１２日検索］、インターネット＜URL： https://mtiinc.jimdo.com/no190907/シナリオ関数の原理模型/＞

非特許文献5：根来文生、「シナリオ関数の全景」、［online］、エムティインターナショナル株式会社、［令和１年９月１２日検索］、インターネット＜URL： https://mtiinc.jimdo.com/no190907/シナリオ関数の全景/＞

発明の概要

発明が解決しようとする課題

［0006］（ＬＣＰＳが発生させるＮＳＡの正統性診断を可能にする方法）

ＳＦの研究課程の中で、ＳＦのＳＡの正統性をＳＡに恒常的に維持させるために、ＬＣＰＳが発生させる非同期型アルゴリズム（ＮＳＡ）の正統性診断を可能にする仕組が論考された。そして、求められた仕組が、本発明の「ＬＣＰＳの超言語脈絡」である。

［0007］　ＬＣＰＳの超言語脈絡を捉えるためにＬＣＰＳに加えられる、後述される「動化パラメータ」はＳＦでは本来的に具備される情報なので、ＳＦでは改めて動化パラメータを取り込む必要はない。本発明のＬＣＰＳの超言語脈絡は結果的にＳＦとの関係に於いて求められている。そして、この仕組が、ＳＦが実行時に発生させるＳＡの中でＳＡが遭遇するバグ事象問題、ウイルス問題をＳＡが自律的に解法を可能にする方法として採用され、且つＳＦに組み入れられた。そして、並行的に、本発明のＬＣＰＳが実行時に発生するＮＳＡの正統性成否を判定するための方法として確立された。２００３年のこ

とである。

[0008]　以下、（１）から（１０）で、本発明に係る知見を示し、ＬＣＰＳの超言語脈絡の仕組は（１１）以降で示す。ＬＣＰＳの超言語脈絡をＬＣＰＳから導出する手続は実施形態の項で詳述される。

[0009]（１）（ＬＣＰＳ、ＳＦＳ）

プログラムが実行時に発生させるアルゴリズムにはＮＳＡ（非同期型）とＳＡ（同期型）がある。後者は１９８６年に本発明者がその存在を予告し、２００８年にＳＡを発生させるプログラムであるＳＦの静的構造のソース（ＳＦＳ）とそのためのＬＹＥＥ理論の最終版に到達している。実行時にＮＳＡを発生させるＬＣＰＳが論理結合型と呼ばれる構造になるのに対し、ＳＦＳはデータ結合型構造に帰着する。ＳＦＳが実行時に発生させるＳＡは世界で最初であることに於いて、ＳＦＳは世界で最初で唯一の静的構造になっている。ＳＦＳのコードは既に公開されている（「ＳＦＳの全景」を参照）。既述の様にＳＦＳは体系化されたＬＹＥＥ理論に因り成立するのに対し、既に、伝統的様相として普及しているＬＣＰＳには体系化されたＬＹＥＥ理論の様な論拠はない。ＮＳＡ、ＳＡの認識の仕方はそれぞれの定義法として後述されている。

[0010]（２）（バグ構文）

ここでは、プログラムとして成立する応用アルゴリズム（ＡＡ）、ＬＣＰＳ、ＳＦＳを総称して「ＡＡ」と記す。ＡＡの成立を阻害する構文の主語名の主語（最終情報）を発症させる構文を本発明では論考の結果、バグ構文と呼ぶことにする。構文が生成する主語を保持する記憶領域は習慣的に構文ごとに特定されている。ＬＹＥＥ理論の研究ではＬＣＰＳを解析的に考察する上で、曖昧さを除くために、構文が生成する情報を主語、主語を保持する構文ごとに特定される記憶領域を主語名として捉える。本発明はこの命名法を踏襲している。例えば、ＬＣＰＳで生成されるＡＡが実行時に発生させる情報が主語名の主語となるとき、この主語名の主語は構文が生成している。そして、この主語を発生させる構文（必要条件）に係る十分条件の全体をＮＳ

Ａが満たしていなければ、本発明ではこの主語を生成する構文を「バグ構文」と定義する。そして、この様な主語は不正値と総称される。この観点に立てば、例えば、ウイルス情報、バグ情報が構文の主語名の主語として捉えられれば、本発明ではおのずとこの主語名の主語は不正値となる。

[0011]　構文の主語名の主語が不正値となることを本発明では「ＮＳＡが構文をバグ構文に変える」と表現する。どの構文も主語が不定値になる迄の間はコンパイル済であることに於いて、どの構文も正統である。ＳＦのバグ構文となる不正値はＳＦのＳＡが悪影響を他の構文に及ぶ前にＳＦで主語名（構文）ごとに用いられる「主語ベクトル」の全体の効果でその不正値を自律的に捉え、且つ自律的にリセットし、且つ自律的に再生される。他方、ＬＣＰＳの構造は論理結合型であるために主語ベクトルを用いることが出来ない。故に、ＬＣＰＳのＮＳＡではＳＡの様な自律性を成立させることが出来ない。

[0012]（３）（ＬＣＰＳの宿命的な欠陥）

ＬＣＰＳを分析するとＡＡが占める構文数の割合は平均的にＬＣＰＳ（全体）の約９０％で、それら構文には論理結合のために実施時に於ける順序性が不可欠になる。

[0013]　ＳＦを分析するとＡＡが占める構文数の割合はＳＦ（全体）の約３０％である。但し、ＳＦの構文はデータ結合であるために実施時に於ける構文の順序性は不要である。そして、ＳＡを成立させる過程の中で、ＡＡの成立を阻害する情報、例えば、バグ、ウイルス情報等を含む全ての不要情報（不正値）が発症する都度、それらを自律的に捉えかつ、ＡＡに悪影響を及ぼす瞬前にそれらを排除する仕組を自律的に成立させている。

[0014]　ＬＣＰＳにＳＡの様な仕組を求めることは不可能である。本発明者の検証では、残存バグ構文候補は本発明が捉えるバグ構文数の約３０％に及ぶ。換言すれば、ＬＣＰＳとはこの量のバグ構文候補を抱えて稼働しているプログラムのことである。

[0015]（４）（ＮＳＡの正統性診断を可能にする仕組）

ＮＳＡの正統性診断とは１個の主語名とそれ以外の主語名の脈絡関係の成

否が全主語名について求められれば、ＮＳＡの正統性診断は可能になる。しかし、この仕組をＮＳＡに求めることは現実的ではない。ここでする話はＬＣＰＳから全主語名について正統性診断を可能にする仕組（ＬＣＰＳの超言語脈絡）のことである。不可能を可能にする話である。

[0016]（５）（ＳＡ，ＮＳＡの定義）

ＬＣＰＳが実行時に発生させるアルゴリズムはＬＣＰＳに属す全主語名を複数個の部分集合に分け、それぞれを単元として、それらの単元を論理的に網羅し、全主語名の主語を捉えるためのアルゴリズムのことである。このアルゴリズムは非同期型アルゴリズム（ＮＳＡ）と呼ばれる。ＳＦが実行時に発生させるアルゴリズムはＳＦに属す全主語名を唯一の単元とし、全主語名の主語が成立する迄、この唯一の単元を繰り返す仕組のアルゴリズムのことである。このアルゴリズムは同期型アルゴリズム（ＳＡ）と呼ばれる。

[0017]（６）（主語ベクトル）

ＳＡを捉えるためには、「主語ベクトル」が不可欠となる。主語ベクトルはＬＹＥＥ理論で発見されＳＦの構成素子となっている。図８参照。

[0018]（７）（プログラムの解）

プログラムの解とは、プログラムが実行時に発生させるアルゴリズムで、プログラムに属す全主語名の主語が成立する状態として定義される。この定義はＬＹＥＥ理論により明らかにされたものである。この定義によれば、ＬＣＰＳの解は不明であるが、ＳＦの解は存在する。ＳＦの解は「主語系譜」と名付けられている。

[0019]（８）（プログラム解の役割）

ＬＣＰＳの部分分割をＬＣＰＳに属す全主語名の部分分割として捉えれば、部分分割は可能である。しかし、部分分割をアルゴリズムの部分分割として捉えると、アルゴリズムは部分分割を不可能として成立するので、この部分分割は不可能である。もし、プログラムの解が成立すればアルゴリズムの部分分割は可能になる。通常、ＬＣＰＳの解は成立しないので、ＬＣＰＳのアルゴリズムの部分分割は不可能である。プログラムの解の成否はアルゴリ

ズムの部分分割が可能か否やを判定するパラメータになっている。ＬＣＰＳのＮＳＡが発症させるバグ事象はこの問題に起因するＬＣＰＳ部分化問題に因り発症している。コンパイルが完結していても、ＬＣＰＳの必要条件である仕様の妥当性の確認しか果たせない伝統的なＬＣＰＳの検証法でこの起因を捉えることは不可能である。故に、バグ事象問題は実質的には未解決のまま放置されている状況にある。因みに、プログラムの解をもつＳＦでは部分化されるアルゴリズムの構造欠陥問題は主語ベクトルと実行時に発生するＳＡを用いて実行時に解決される。公開されているシナリオ関数の全景図、ＳＦの解である主語系譜を熟視すれば、このことが理解できる様になっている。

[0020]（９）（応用アルゴリズム（ＡＡ）の成立の仕方）

ＬＣＰＳのＮＳＡはＬＣＰＳに属す全主語名の主語を論理結合的に成立させることに因りＬＣＰＳのＡＡを成立させる。ＳＦのＳＡはＳＦに属す全主語名の主語を同期的に成立させることに因りＳＦのＡＡを成立させている。

[0021]（１０）（ＳＡのＡＡとＮＳＡのＡＡの違い）

ＳＦのＡＡは不正値を排除しながら正統なＡＡを成立させるためのアルゴリズムを成立させている。他方、ＬＣＰＳのＡＡは本来的に正統なＡＡを成立させるためのアルゴリズムを成立させる構造にはなっていない

[0022]（１１）（調和座標、調和脈絡）

調和座標とはＬＣＰＳの全構文に付与される構文の実行順位のことである。調和座標は本研究の中で定義された座標系である。図４の点線で示される構文の実行順位は調和脈絡と呼ぶ。重要なことは超言語座標は付与された調和脈絡の基で成立するという点である。

[0023]（１２）（超言語座標）

超言語座標とはＬＣＰＳの全構文の主語名、変数主語名それぞれに付与される二つの調和座標の対（ＴＣＸ１，ＴＣＸ２）である。

[0024]　ＴＣＸ１は構文の主語名の主語を保持する領域を定義する領域定義文のＴＣＸである。ＴＣＸ２は主語を生成する構文のＴＣＸである。故に、主語が

主語名の主語の場合にはＴＣＸ２はその構文のＴＣＸとなる。そして、主語が変数主語名の主語の場合にはＴＣＸ２はその変数主語名を主語名とする構文のＴＣＸとなる。

[0025]（１３）（超言語脈絡）

（ａ）超言語脈絡の発端主語名とは、ＬＣＰＳに属し、且つ主語ベクトル種別がＬ４、Ｗ４で統治され、且つその構文のＴＣＸがＬＣＰＳの最小の構文の主語名である。

（ｂ）超言語脈絡の終端主語名とは、ＬＣＰＳに属し、且つ主語ベクトル種別がＲ２、Ｌ２で統治される構文の主語名の全てである。

（ｃ）ＬＣＰＳの超言語脈絡とは、二つの超言語座標を結ぶ経路が終端主語名から発端主語名に収斂す経路のことである。

（ｄ）発端主語名が属す構文、終端主語名が属す構文以外の構文は終端主語名と発端主語名を結ぶ脈絡の節点となる。

（ｅ）超言語座標１と超言語座標１を受容する側（発端主語名に近い側）の超言語座標２の関係は構文の位置座標ＴＣＸは異なっていても超言語座標が同じになる。

[0026]（１４）（図２、図３、図４、図５）

ＬＣＰＳのＮＳＡの正統性診断を可能にするＬＣＰＳの超言語脈絡を成立させるための情報には、ＬＣＰＳの情報と、ＳＡを成立させるＳＦの情報から「動化パラメータ」として求められているＬＣＰＳには存在していない情報がある。

[0027]　本発明ではこの情報をＬＣＰＳを阻害しない様にＬＣＰＳに組み入れている。そして、この情報を治める器として、図２の形式が決定されている。本発明では図２に例示の形式の情報を「ＬＣＰＳの超言語脈絡情報」と名付けている。

[0028]　図４は絵解きされた図２の様相である。図４は事例の図２の情報から、２０年余本発明者の研究をトレースし続けている齢８５歳病身の平山氏が描いたものである。因みに、図４は無料のグラフ化ツールで図２の情報をパラメ

ータとして、自動生成することも可能である。ＮＳＡの正統性判定と、ＮＳＡが発症させるＬＣＰＳに掛かるバグ構文の情報は、図２の解析を行う専用プログラムを準備すれば、図３の形式で求めることが出来る。図４、図５は図２の解析を行う専用プログラムの仕様決定に貢献する参考資料である。

[0029]（１５）（ＮＳＡの正統性を特定する方法）

図２から図３に掛かる情報が抽出できなければ、ＬＣＰＳが実行時に発生するＮＳＡは正統である。

[0030]（１６）（ＬＣＰＳの構文をバグ構文に変える起因）

ＬＣＰＳの超言語脈絡を用いれば、構文をバグ構文に変える起因は以下の二つに特定することが出来る。

（ａ）ＬＣＰＳの超言語脈絡に途切れを発症させる構文の存在。

（ｂ）ＬＣＰＳの超言語脈絡に錯誤を発症させる構文の存在。

[0031] 即ち、起因（ａ）は構文１の変数主語名が受容する筈の主語名の構文がＬＣＰＳの超言語脈絡上に存在しない場合である。この場合、本発明では構文１を脈絡を途切れさせる構文と呼ぶ。構文１は図２の情報の超言語座標のＴＣＸ２が空白であることから視覚的に捉えることが出来る。この構文１を特に［データ揺動構文］と呼ぶ（図３参照）。

[0032] 起因（ｂ）は構文１の変数主語名が受容する正解の主語名が正解ではない主語名を受容する場合である。正解の主語名とは、変数主語名１が属す構文のＴＣＸを起点とすれば、時制的にこの起点のＴＣＸの直近の過去側に位置する構文に属す主語名のことである。正解ではない主語名とは、正解以外の構文の主語名を受容する場合である。

[0033] 起因（ｂ）の構文１が受容する主語名が属す構文は図２を解析する専用のプログラムを準備して、それにより構文１が受容している主語名が属す構文を特定しなければ、その脈絡の正誤を判定することはできない。起因（ｂ）の構文をＬＣＰＳから暗算的に捉えることは余程の変人でなければ不可能である。伝統的なＬＣＰＳの検証法でこの構文が特定できる道理がない。

[0034] 起因（ｂ）の構文位置を誤らせる原因は本発明の観点では調和座標の付与

の錯誤である。調和座標の付与は結果的に人による作業である。そして、そこに錯覚が生じる場合、起因（ｂ）が発症する。本発明者は大学で教鞭をとっていた頃、スマートコンパイラを成立させるための基本的な課題としてこの課題を克服するコンパイラを作成し日本の情報処理学会で発表している。５０年前の話である。

[0035]（１７）（バグ構文を特定する方法）

ＬＣＰＳの超言語脈絡でバグ構文を発症させる起因が特定できれば、不正値を具体値で特定しなくても、ＬＣＰＳの超言語脈絡の構造から、その発端主語名の主語は不正値となることは明らかである。そして、発端主語名の主語が不正値になれば、発端主語名の配下の発端主語名を変数主語名とする構文はその影響を受けることに於いて、バグ事象を発症させる構文に変わることになる（図５、図４を参照）。これは、ＬＣＰＳをテストデータを用いて電算機で処理しなくても、バグ構文に変わるＬＣＰＳの構文を特定できることを意味する。

[0036]（１８）（バグ事象区分）

ＬＣＰＳの超言語脈絡に於いて、起因（ａ）（ｂ）の構文位置と発端主語名が属す構文位置の脈絡の間でバグ構文は発症する（図５参照）。そして、この脈絡の間に１個以上の主語ベクトルＬ３で統治される条件文が存在し、その内の条件文の判定式が発端主語名の主語の不正値の影響を受けていれば、発端主語名から最遠の位置のこの条件文と起因（ａ）（ｂ）の構文位置の間の全構文は部分破壊される。部分破壊を伴なうバグ構文は図３のバグ事象区分では「２」と記される。

[0037]　部分破壊を伴なわないバグ構文は図３のバグ事象区分では「１」と記される。バグ事象区分２のバグ構文の修正は容易ではない（図５、図３参照）。

[0038]（１９）（起因（ｂ）の構文が発症する理由）

コンパイル済のＬＣＰＳの構文が実行時バグ構文に変わるのはＮＳＡの仕儀である。このＮＳＡの仕儀が生じるのは以下の理由である。即ち、ＬＣＰＳにはプログラムの解が成立しない。これは実行時のＬＣＰＳでは部分と全

体のアルゴリズムの関係を曖昧にする原因になることを示唆する（図７参照）。

[0039]　そして、ＬＣＰＳの同じ部分プログラムが複数個所で使用される場合、同じ部分プログラムの稼働条件が全て同じになるとは限らない。そして、ＬＣＰＳの部分プログラムにはこの違いに対するコーデング上の対応措置が不可欠であるにも関わらず習慣的にこの措置が講じられてはいない。結果、このことが変数主語を受容する正解の主語名の構文が正しくは不明となる事態であるにも関わらず、不明と解釈できない想いが、ＬＣＰＳの調和座標の付与を規則通りに成立させてしまう。前項（１６）の起因（ｂ）の構文はこの様な背景により生じている。

[0040]（２０）（ＬＣＰＳの役割）

ＯＳの仕事はプログラムの実行生産性の向上に止まる。以下の課題はＯＳでは解決できない。

（ア）ＬＣＰＳの生産性

（イ）ＬＣＰＳの保守生産性

（ウ）プログラム言語問題

（エ）コンパイラ問題

（オ）バグ構文の排除

（カ）ウイルスの排除

[0041]　ＳＦではこれら課題はＳＦの開発作業時並びに、特に（オ）（カ）はＳＦの実行時に自律的に解法される。他方、ＬＣＰＳではこれら課題は何一つ解法することが出来ない。

[0042]（２１）（主語名の臨界状態）

本発明者のＳＦ、ＬＹＥＥ理論の研究課程の中で、ＬＣＰＳに属す主語名数とＬＣＰＳが実行時に発生するアルゴリズム（ＮＳＡ）の錯誤の関係が論考されている。この論考の中で同じ名詞が重複している場合にＮＳＡを錯誤させる原因（前記（１６））が生じていることが明らかにされたのである。本発明では同じ名詞が重複して用いられている状態を主語名の臨界状態と呼

ぶ。

[0043]［ＬＹＥＥ理論］

（１）（無限象）

ＬＹＥＥ理論では、生命作用が認識を行うとき、その瞬前に生命作用は正統な認識を成立させるための「全貌」を受容していると仮説する。ＬＹＥＥ理論ではこの全貌を「無限象」と呼ぶ。ＬＹＥＥ理論はこの無限象を捉える仕組の論考である。この仕組はＬＹＥＥ理論では「意識関数」と呼ばれる。そして、意識関数の意味をプログラム世界の用語に写し替えられた意識関数がシナリオ関数（ＳＦ）である。

[0044]（２）（同期化）

無限象の全貌を捉える仕組は、部分を極限まで全体化する仕組として成立する。部分とは動性様相であり、全体とはその動性が静性に遷移した様相である。換言すれば、部分を極限まで全体化するとは部分の動性を同期化することである。何故なら、同期化された様相は静的様相に帰着するからである。因みに、同期化が成立しないプログラムは部分である。故に、構文数の規模に関係なく、ＬＣＰＳの部分プログラム、並びにＬＣＰＳの全体は恒常的に部分である。即ち、それらの様相は動性である。故に、ＳＦではプログラムの解は特定されても、同期化が成立しないＬＣＰＳのプログラムの解は不明のままである。部分を極限まで全体化するための仕組はＬＹＥＥ理論により求められた。即ち、無現象の単元（フラクタル）の内容を認識可能にする仕組１（ベクトル）と、仕組１の全集合を同期化させる仕組２が求められた。無現象の単元は主語名に置き換えられている。ＳＦとはＬＣＰＳを極限まで全体化するための仕組のプログラムである。

[0045]（３）（可視化されたアルゴリズム）

ラバール大学による可視化された「ＮＳＡの全景」はトウモロコシ型の混沌たる脈絡に、そして、「ＳＡの全景」は整然とした繰り返し型の脈絡に治まっている。

[0046]（４）（バグ事象を捉える手段）

本来的に正統性を捉えているＳＡをＮＳＡに写し両者の差分を論考する。その差分がバグ事象を発症させＮＳＡの原因である。正統性が成立しないＮＳＡにバグ事象が発症するのは必然である。そのバグ事象をＮＳＡで捉えようとするのは言を待つまでもなく矛盾である。ＬＣＰＳを基にＬＣＰＳの超言語脈絡を求め、それを解析するのでなければ、バグ事象は捉えられない。ＬＣＰＳの超言語脈絡とはＬＣＰＳから求められる最後の同期構造の様相である。

[0047]（５）（ＮＳＡ成否判定の原理パラメータ）

ＬＣＰＳの完成度は他の分野の製品なら返品レベルにある。かつＩＴ業界のビジネス第一主義はＮＳＡに係る技術、そして、その知性（良心）の空洞化はとめどなく加速している様に思われる。本発明はＮＳＡの成否判定を行う方法である。アルゴリズムの性質からＬＣＰＳ仕様の成否判定を行う伝統的な検証法と本発明の検証法では以下の理由で異なる。

（ｉ）ＬＣＰＳの仕様情報はＬＣＰＳの手続きを決めるための必要条件の部分である。

（ｉｉ）ＬＣＰＳのアルゴリズムは必要条件が発生させる動性様相である。

（ｉｉｉ）アルゴリズムは部分では成立しない。

（ｉｖ）アルゴリズムは必要条件に係る十分条件の全体である。

（ｖ）アルゴリズムの全体はプログラムの実行時に特定される。

（ｖｉ）ＳＦのソース構造は実行時に十分条件の全体を捉える。

（ｖｉｉ）ＮＳＡはＬＣＰＳが発生させるアルゴリズムである。

（ｖｉｉｉ）ＳＡはＳＦが発生させるアルゴリズムである。

[0048]（６）（８項目の意義）

上記アルゴリズムに係る８項目は、仕様の検証法に帰着している伝統的な検証法では、ＮＳＡの正統性の成否が十分に判定できないことを示唆している。付言すれば、ＬＣＰＳの伝統的な検証法をいくら山積みしてもＮＳＡの完全な正統性判定には至らないことを示唆している。故に、このことを前提

に稼働中のＮＳＡを調査するとＮＳＡの欠陥を明らかにすることができる。ＮＳＡが成立させるＡＡは不完全であることが明らかになる。結果、すべてのＬＣＰＳは不完全の状態で稼働していることも明らかになる。故に、ＬＣＰＳにシステム事故が発症するのは回避不可能なＬＣＰＳの必然性だということである。既述のラバール大学での可視化されたＬＣＰＳのＮＳＡはこのことを示唆するに足りるものになっている。

[0049]（７）上記（５）の８項目はＳＦの静的構造を求める本発明者の研究課程の中で、稼働中のＬＣＰＳの調査分析の情報から求められたものである。プログラムのアルゴリズムは生命作用として捉えなければプログラム課題の本質（プログラムの真実）には至らないとの自覚の目途が立った１９８６年である。ＬＹＥＥ理論、そしてＳＦは１９７２年から開始された研究のゴールとして、完結されるのは２００８年である。この間、本発明者により行われた稼働中のＬＣＰＳの分析は通算１０００個に及ぶ。ＬＹＥＥ理論、そしてＳＦが完結に至るのはこの分析情報から得られたＬＣＰＳに係る知見である。本発明はＳＦのＳＡが基盤である。

[0050]（８）（ＬＣＰＳのＮＳＡがバグ構文を発症させる原因）

コンパイル過程が完了しているＬＣＰが実行時に発生するＮＳＡは正統である。そして、ＮＳＡが正統ならバグ構文は発症しないのがプログラムの道理である。故に、ＳＦではコンパイル過程が完了すれば、ＬＣＰＳに対して行われる様な伝統的なＬＣＰＳの検証過程は不要である。ＳＦではコンパイル過程が完了すれば、稼働過程に入ることが出来る。

[0051]　そして、実行時に不要情報が発症すればが不要情報を捉え且つ排除する仕組がＳＡには自律的に成立する様にＳＦＳの定義構造が設計されている。しかし、この様な仕組を成立させるＮＳＡをＬＣＰＳに求めることは不可能である。

[0052]　この問題をＬＣＰＳで回避させるためには、部分プログラムが使用される個所ごとに実行可能条件を設定して、その成否で部分プログラムの使用の可否を判定しなければならない。そうでなければその部分プログラムの定義は

可能でもそのアルゴリズムは成立しない。この実行可能条件はその部分プログラムに属す主語名以外の主語名を必要とする。この様な問題を解法するために、ＳＦではＬＣＰで言う様な部分プログラムはない。主語名ごとに定義される主語ベクトル（図９参照）が用いられる。そして、それら全主語ベクトルでこの問題は自律的に解法される。従来の部分プログラムを使用したい向きにはそれら部品プログラムは主語ベクトルの第２規約として設定すればよい。論理結合型の部分プログラムに実行可能条件を設定することはこのことが理解できる人が見当たらないことに於いて現実的ではない。ＡＩの技術者がいないというレベルの話とはわけが違うのである。

[0053]（９）（正統な構文をバグ構文に変える仕組）

ＮＳＡの正統性を検証するデータは、ＡＡの成否を問う伝統的な検証法で用いられるデータ（仕様情報）ではなく、ＮＳＡを発生させる実行可能なＬＣＰＳである。本発明ではＮＳＡを検証するとは、ＬＣＰＳの全構文を用いてＮＳＡの様相を捉えることである。この様相を本発明ではＬＣＰＳの超言語脈絡と呼ぶ。図２または図４はＬＣＰＳから求められる超言語脈絡のための情報である。

[0054]　超言語脈絡が本発明の主体である。結論的に言えば、本発明は超言語脈絡を用いて、ＮＳＡがＬＣＰＳの正統な構文をバグ構文に変える構文を特定する方法である。超言語脈絡の全ての脈絡の経路は、先走るが、脈絡の定義方法から、図４で見ることが出来る様に、終端主語名に端を発し発端主語名に収斂する。超言語脈絡を用いて、バグ構文の存在性を解釈すれば、バグ構文とは超言語脈絡の経路の途切れに因り発症する構文のことである。

[0055]　ＬＣＰＳには当初から脈絡が途切れているバグ構文、稼働中に脈絡が途切れるバグ構文がある。前者は幼稚なプログラムの錯誤である。この様なバグ構文は言語変換された履歴を持つＬＣＰＳに多く見ることが出来る。このバグ構文が生成するデータは不正値となるのだが、正常値と不正値の差分は往々にして微小なので、システムにもよるが、関係者、利用者の殆どはこの事象を見過ごしている。

[0056]　超言語脈絡に途切れを生じさせる原因は、既述の幼稚なプログラムの錯誤を除けば、上述に係る認識不足が遠因である。関係者がこの認識不足の事象を認識できれば、脈絡を途切れさせる構文を図２または図４から特定することが出来る。そして、その構文に端を発する全てのバグ構文を図３の形式で特定することが出来る。

[0057]（１０）（ＬＣＰＳ世界の再起の指針）

ＬＣＰＳの伝統的な検証法をよく理解できれば、システムの処理時間測定程度にとどまる。もし、それ以外の意義をこの検証法に求めるのであれば、伝統的な検証法にこだわらずに、プログラムの検証法とは何かを考えることを勧めたい。本発明は新たな検証法の指針になるであろう。

[0058]（１１）（本発明の診断対象）

ＳＦではコンパイルが済めばＳＡの正統性が保証される。故に、ＳＡは本発明を必要としない。本発明の診断対象は成否が不明なＮＳＡの原因を明らかにすることである。

[0059]（１２）（ＬＣＰＳの原理的な課題）

ＳＦがデータ結合型プログラム故に実行時のアルゴリズムはＳＡとなる。他方、ＬＣＰＳは論理結合型、換言すれば、人工的プログラム故にその実行時のアルゴリズムはＮＳＡとなる。そして、ＮＳＡが発症させるプログラム問題に気付いたとしてもそれをそのＬＣＰＳ上で、あるいは別のＬＣＰＳで解決しようとしても、それは不可能である。ウイルス対応のＬＣＰＳで明らかな様に、ただプログラム問題の屋上屋を重ねるだけである。ＮＳＡの課題はＬＣＰＳの構造問題に起因するので、ＬＣＰＳの構造を変えない限り、普遍的な解法（例外のない解決）には至らない。

[0060]（１３）（存在し続ける約３０％のバグ構文）

本発明者の調査では、残存バグが占める割合は全体の約３０％である。そして、稼働中この残存バグが目覚めればシステム事故を発症させる。事故後、このバグ構文の発症の仕組は誰も解らない故に、関係者は仕組の一部を原因として、結果的に屋上屋を重ねるしか途のないＬＣＰＳの改修を実施して

いる。

[0061]（１４）（すべてのバグ構文を捉える仕組）

本発明はＬＣＰＳの超言語脈絡を用いてバグ構文の全てを捉える方法である。

[0062]（１５）（ＬＹＥＥ理論の適用実績）

実行時不要情報から解法されるＳＦの開発作業は、作業量的な意味で、学習要領で明記されている通りの内容の作業をその明記どおりに果たす気質（新人）のプログラマであれば実施可能である。例えば、４０１Ｋシステムは世間的には知る人は知るものであるが、２００３年当時、本発明者はまだＳＦの使用許可を与えていなかったので、ＬＹＥＥ理論を用いて、約２００万個の構文でなるプログラムを新人勢力が約１０か月で納品している。これは想定されるＳＦの生産性パラメータ（コード化率、テストレス率、全作業工数、作業工数の工程配分率、作業自動化率、開発作業難度、業務難度）にもっとも合致する典型例となった。因みに、ＬＹＥＥ理論で１９９３年以降２００６年までに開発されたシステムはいずれも大型で３６個である。

[0063]（１６）（ＬＣＰＳ技術課題）

実行時にバグ構文が発症しなければ、ＮＳＡ、ＬＣＰＳには正統性が成立する。しかし、この様なことは、システムの規模にもよるが、ＬＣＰＳでは万にひとつもありえないのである。本発明者の調査では、稼働中の３００個の構文のＬＣＰＳですらバグ構文が発見されている。しかしＬＣＰＳだと稼働を続けるしか方法のない宿命的なプログラムである。

[0064]［ＬＣＰＳの十分条件］

バグ構文を発症させる原因を要約すれば、以下の５点である。

（ア）ＬＣＰは解が不定であること。

（イ）結果、部分と全体の意味があいまいになること。

（ウ）故にＬＣＰＳの全構文の実行順番の付与規則が決定的にはならないこと。このことが付与者がコンパイラであれ、人であれ、誤記を生じさせる。

（エ）実行順番の誤記が「超言語座標」の付与に誤記を生じさせること。

（オ）結果、「超言語脈絡」に途切れを生じさせること。

[0065]　以上が実行時にＬＣＰＳに属す構文をバグ構文に変える仕組の大筋である。バグ構文を発症させる最終的な原因は、超言語脈絡に途切れが生じる上記（オ）のほかに、超言語脈絡が上記（エ）の理由で誤って成立する場合もバグ構文を発症させる最終的な原因となる。上記（オ）は図２の情報を図４的に解析すればその位置の特定が可能になる。上記（エ）は超言語脈絡の生成規則に違反する状態である。超言語脈絡を解析すれば、その箇所が捉えられる。

[0066]［発明の概要］

１９７２年から２００８年迄、本発明者が進めて来た研究は次の様に要約される。

（１）論理結合型プログラムが実行時に創出する非同期アルゴリズムの課題に関する研究

（２）上記課題を解法するためのプログラム（ＳＦ）の存在性に関する研究

[0067]（本発明の動機）

本発明の動機はプログラムの完全体を志向するプログラム（ＳＦ）の静的構造を理論的に求める研究の過程で、ＬＣＰＳが発症させるバグ事象の起因を明らかにする必要があり、バグ事象が１９８６年から２０００年にわたり観察された。そして、その起因には（ア）ＬＣＰＳに紐づけられるもの、（イ）起因が不明のものとに仕分けされる。本発明では（ア）のバグ事象は、後述される図３のバグ事象区分１、そして、（イ）のバグ事象は図３のバグ事象区分２に反映されている。そして、区分２はＬＣＰＳの部分破壊を伴うバグ事象、区分１はＬＣＰＳの部分破壊を伴わないバグ事象である。これらも本発明では図３に反映されている。

[0068]（ＣＰＳがバグ構文を発症させる原因）

この研究は「ソフト危機の警告」に端を発して１９７２年に開始される。因みに、この警告はプログラム規模が１５０万ラインを超えると所謂バグ問

題を解決することが誰にも出来なくなり、プログラマのコード化率（プログラマが１日に決定可能なコード数）がゼロに至るとの予告のことである。因みに、原理的に好ましくないプログラムの部分化が積極的に進められる様になるのは、プログラムの開発方法論、自動プログラミングが望めなくなるとの研究不足の人々が大勢を成す１９９０年初頭からである。この警告がその背景にある。そして、原理的に好ましくないプログラムの部分化がＩＴ業界の潮流となる。

[0069]　プログラムの部分化が原理的に好ましくないとは本発明者の指摘なのだが、そのわけは、論理結合型事象の部分化の定義はシステムに属す全主語名を部分化することだから誰にでも出来る。しかし、その部分プログラムが実行時に発生させるアルゴリズムの成否はその部分プログラムが発生させるアルゴリズムで判定できるとは限らない。プログラム言語の文法規則の中にもこの成否問題を解決する構文規則は見当たらない。アルゴリズムの成否はアルゴリズムの存在原理から、全アルゴリズムでのみ問うことが出来る。故に、部分化の定義は誰にでもできるが、その部分アルゴリズムが全体となる様に部分プログラムを定義することは誰にでも出来る話ではない（図６参照）。付言すれば、人が機械物と異なるプログラムなる存在に対して、配慮なく機械物の部品と同じ感覚で部品プログラムを定義している。部品プログラムが実行時に人が気付くことが出来ない錯誤を帯同する部分アルゴリズムを正統なアルゴリズムとして発生させている。ＬＣＰＳの構文をバグ構文に変える可能的原因はこの部分アルゴリズムである。

[0070]（本発明は実施検証がされている）

本発明者は約５０万個の構文で構成される制御プログラムが実行時に発症させるバグ構文を捉えるために本発明の考えを用いて実証検証を行っている。この検証で捉えられたバグ構文に変わる構文は複数個所で共通に使用される部分プログラムに属す構文のひとつであった。本発明の成立性はこの検証で証明されている。

[0071]（十分条件の全体とＬＣＰＳの超言語脈絡の違い）

（１）（十分条件の全体）

ＬＣＰＳに属す構文、例えば構文１の主語名１の主語が実行時に成立するには、主語名１に掛る全変数名がそれぞれ別の構文の主語名となる関係から、構文１が実行される以前にそれら主語名の主語は既に成立していなければならない。そして、変数主語名の主語を成立させるこの関係が、ＬＣＰＳの始祖となる主語迄達していれば、この関係は主語名１の主語の正統性が成立する十分条件の全体と呼ばれる。

[0072]（２）（十分条件の部分）

該当項を参照

[0073]（３）（図２の求め方）

上記（２）の現象を捉えるための情報が「図２」である。

[0074]（４）（図４の求め方）

図２の情報を絵解きすれば図４となる。図４は図２の情報から人的に、または図２の情報をパラメータとして市販されている絵解きツールで描くことが出来る。

[0075]（５）（図３の求め方）

図２の情報を機械的手続きで解析すれば、図３の情報が求められる。これは図３の情報は人的に、または作成可能な専用プログラムで図２から求められることを意味する。

[0076]（６）（ＬＣＰＳの十分条件の全体）

ＬＣＰＳに属す全構文に掛る全主語名それぞれの主語名の主語の正統性を判定するためには主語名の主語に掛る十分条件の全体をＬＣＰＳに属す全主語名ごとに捉える必要がある。しかし、論理結合型であるＬＣＰＳは十文条件の全体を捉えるために必要な情報は持ち得ていないのである。つまり、十文条件の全体を捉えることはＬＣＰＳでは不可能である。換言すれば、これはＬＣＰＳの実行時の正統性検証は原理的には不可能であることを意味する。

[0077]　因みに、ＳＦは実行時に同期型アルゴリズムを成立させる構造として設計

されている。故に、ＳＦではこの構造により主語名の主語に掛る十分条件の全体を全主語名について捉え且つ成立しない主語、つまり不正値となる主語を自律的に排除することにも成功している。

[0078]　本発明では、ＬＣＰＳバグ構文を捉える仕組が論考され、不正値を発症させるバグ構文の定義が明らかにされ、ＬＣＰＳからバグ構文を捉えるための必要な情報が定義された。

[0079]　また、本発明では、バグ構文を捉えるために不可欠な情報が以下の動化パラメータとして定義され追加された。

主語名

変数主語名

主語ベクトル

構文種別

多重構造化される調和座標

調和脈絡

超言語座標

超言語脈絡

発端主語名

終端主語名

[0080]　以上の情報は図２に収集する。図２の情報を絵解きするために発端主語名、終端主語名が求められる。そして、図２の情報から、バグ構文を捉えることに成功する。

[0081]　ＬＣＰＳの脈絡とは図２の情報のことである。即ち、ＬＣＰＳの脈絡とは十分条件の全体の構造を基に図２から求められるただひとつの特異な脈絡である。そして、ＬＣＰＳの脈絡をＬＣＰＳに適用するとその適用効能は図２に対応する図３で示される様に不正値とその不正値に掛る構文の情報を合わせて捉えることが出来る。これにより、伝統的なＬＣＰＳの検証法がＬＣＰＳの完全な動性保証を与えることが出来ない検証法に留まるのに対し、ＬＣＰＳの脈絡はＬＣＰＳの完全な動性保証を与える為の情報を全て明らかにす

ることが出来る。

課題を解決する手段

[0082]　「ＬＣＰＳの超言語脈絡」は課題を解決する手段である。ここでは「ＬＣＰＳの超言語脈絡」をＬＣＰＳから導くために用いられる定義概念を以下に示す（図８参照）。

（０１）同期アルゴリズム（ＳＡ）

（０２）非同期アルゴリズム（ＮＳＡ）

（０３）プログラムの解

（０４）ＬＣＰＳの未解決問題

（０５）１１種の構文種別

（０６）主語名

（０７）変数主語名

（０８）主語名の主語の不正値

（０９）バグ構文

（１０）５種の主語ベクトル（Ｌ４，Ｌ２，Ｌ３，Ｒ２，Ｗ４）

（１１）１１種の構文種別と５種の主語ベクトルの関係

（１２）主語名の主語の十分条件の部分

（１３）主語名の主語の十分条件の全体

（１４）ＬＣＰＳの超言語脈絡

（１５）ＬＣＰＳの超言語脈絡の発端主語名

（１６）ＬＣＰＳの超言語脈絡の終端主語名

（１７）調和座標と調和脈絡

（１８）超言語座標

（１９）超言語脈絡

（２０）ＬＣＰＳの動化パラメータ（調和座標、構文種別、主語名、変数主語名、主語ベクトル種別、超言語座標）

（２１）図２

（２２）データ揺動構文

（２３）主語名の臨界数

（２４）調和座標の錯誤

（２５）超言語脈絡の途切れ構文

（２６）超言語脈絡の誤記

（２７）ＬＣＰＳの部分破壊

（２８）バグ事象の仕分コード１，２

（２９）図３

（３０）図７

（３１）平文法

[0083]（定義概念の要約）

１．（１１種のプログラム構文）

本発明ではプログラム構文を以下の１１種の構文に仕分する。

領域定義文

算術文

条件文

条件文に属す判定文

定値文

翻訳文

呼び出し文

ＧＯＴＯ文

主語の領域入れ替え文

入力文

出力文

[0084] ２．（主語名、変数主語名、定数、定値）

２．１：例えば、算術文Ａ＝Ｂ＋Ｃに於いて、Ａは主語名、Ｂ、Ｃは変数主語名と記す。

２．２：主語名Ａ、変数主語名Ｂ、Ｃは共に領域名である。

２．３：変数主語名ＢはＢ＝の代数式に於いては主語名である。

２．４：主語名ＡはＤ＝Ｅ＋Ａに於いては変数主語名である。

２．５：主語とは領域名に属す実体化された状態である。

２．６：例えば、条件文の判定式Ａ＝ＢのＡ並びにＢは変数主語名である。

２．７：例えば、Ａ＝７のＡは主語名、７はＡの主語である。この場合７は「定数」「直値」「定値」などと記される。

２．８：Ａ（ＸＹＺ）のＡは主語名、文字列ＸＹＺはＡの主語で定値と記される。この場合、文字列ＸＹＺは「定置」と記される。

[0085] ３．（主語ベクトル）（図９参照）

３．１：主語ベクトルはＳＦで用いられる構文を律する仕組である。本願はその転用である。主語ベクトルはＬＣＰＳの部品に当たるが、その構造はＬＹＥＥ理論により求められた究極の普遍部品として位置づけられている。図９はその構造である。主語ベクトルは律する構文を自分の第２規約で統治する。そして、律する構文の種別により、主語ベクトルは５種に仕分けされる。

[0086] ３．２：入力文を律する主語ベクトルは「Ｒ２，主語名」と表記される。この主語名は入力領域を代表する主語名である。Ｒ２の２は入力文が時制上の過去として位置づけられることを示す。

[0087] ３．３：出力文を律する主語ベクトルは「Ｗ４，主語名」と表記される。この主語名は出力領域を代表する主語名である。Ｗ４の４は出力文が時制上の今として位置づけられることを示す。

[0088] ３．４：算術文で代表される構文を律する主語ベクトルは「Ｌ４，主語名」と表記される。この主語名は算術式左辺の主語名を指す。

[0089] ３．５：定値式を律する主語ベクトルは「Ｌ２，主語名」と表記される。この主語名は定値の主語名である。この主語名が存在しなければ、直接定値が記される。Ｌ２の２は定値式が時制上の過去として位置づけられることを示す。

[0090] ３．６：判定式を含む条件文を律する主語ベクトルは「Ｌ３，主語名」と表

記される。この主語名はＬ３が制御する「Ｌ４，主語名」、「Ｗ４，主語名」、「Ｒ２，主語名」「Ｌ２、主語名」の主語名を基に付与する。Ｌ３の３は条件文が時制上の未来として位置づけられることを示す。

[0091] ３．７：主語ベクトルＬ４、Ｗ４の主語名は超言語脈絡の発端候補となる。

[0092] ３．８：発端となる主語ベクトル以外の主語ベクトルＬ４、Ｗ４、Ｌ３は脈絡上の節点となる。

[0093] ３．９：主語ベクトルＬ２、Ｒ２は脈絡の終端となる。

[0094] ３．１０：主語ベクトルの第２規約で統治される条件文は調和座標（ＴＣＹ，ＴＣＺ１，２，３，４）で律される。

[0095] ３．１１：言葉的に時制とは所謂動詞の属性概念で、故に名詞に時制はない。しかし、名詞は実体化されることに於いて背景には動詞性を含意している。実体化された主語名を本願では主語と記すので、故に、主語名には時制があると考える。この為に主語ベクトルでは実体化される構文の主語名に時制が付与される。これは実体化で成立する主語達の同期状態をプログラムの存在性とするＳＦでは必然の措置となるためである。主語達の同期様相では時制の異なる主語は区別される必要があるからである。入力される名詞の主語、出力される名詞の主語と同期様相を成立させる主語は区別されなければならないからである。この原理はＬＣＰＳの主語の不正値をＬＣＰＳの脈絡を用いて捉える場合にも脈絡の跡切れを行う構文を特定する上で必要になる。

[0096] ３．１２：動化パラメータとは、ＬＣＰＳに動性を付与するために課す以下の情報のことである。即ち、主語名、変数主語名、調和座標、超言語座標、１１種の構文種別、５種の主語ベクトル（図２参照）。

[0097] ４．（ＬＣＰＳの十分条件の全体）

ＬＣＰＳの十分条件の全体は実行時のＬＣＰＳの動的様相の基で捉えることが出来る。

[0098] ４．１（ＬＣＰＳの十分条件の部分）

ＬＣＰＳの十分条件の部分は実行時のＬＣＰＳの静的で捉えることが出来る。

[0099] ４．２：（ＬＣＰＳの超言語脈絡の存在性）

ＬＣＰＳの超言語脈絡はＬＣＰＳの十分条件の全体を捉えるための仕組である。

[0100] ４．３：（図２の情報）

ＬＣＰＳから求められる図２の情報１はＬＣＰＳの超言語脈絡を捉えるための情報である。故に、図２の情報の為にＬＣＰＳには動化パラメータが課される。

[0101] ５．（調和座標）

５．１：調和座標とは、本願発明者により発案された６種の座標系である。６種の座標とは、ＴＣＸ、ＴＣＹ、ＴＣＺ１、ＴＣＺ２、ＴＣＺ３、ＴＣＺ４である。

[0102] ５．２：本発明では全ＬＣＰＳの全構文に対して１組の調和座標系が付与される。

[0103] ５．３：６種の調和座標の意味を以下に記す。

（１）ＴＣＸは構文のプログラム上の位置を１から始まる連続する自然数で示す。

（２）ＴＣＹはＴＣＸの構文から見て次に実行される構文のＴＣＸの位置を指す。

（３）ＴＣＺ１はＴＣＸの構文から見てＴＣＸの構文に真偽の真が成立する場合に実行される構文のＴＣＸの位置を指す。

（４）ＴＣＺ２はＴＣＸの構文から見てＴＣＸの構文に真偽の偽が成立する場合に実行される構文のＴＣＸの位置を指す。

（５）ＴＣＺ３はＴＣＸの構文から見てＴＣＸの構文の真偽評価の終了点に位置する構文のＴＣＸを指す。

（６）ＴＣＺ４はＴＣＸの構文から見てＴＣＸの構文の真偽評価の終了点の次に位置する構文のＴＣＸを指す。

[0104] ６．（超言語座標）

ＬＣＰＳの超言語脈絡を参照。

[0105] ７．（ＬＣＰＳの十分条件の全体の補記）

７．１：例えば、算術式Ａ＝Ｂ＋Ｃに於いて、本願ではＡを主語名、Ｂ、Ｃを変数主語名と記す。主語名、変数主語名は共に領域の名称（領域名）であるが、本願では算術式左辺のＡを特に主語名と総称する。故にＢ、Ｃは別の算術式では、可能的に主語名となる関係にある。

[0106] ７．２：本願では領域名の実体は主語と総称される。領域名の主語を想定する必要がある場合には主語名の後に領域定義文のＴＣＸが付与される。Ａ＝Ｂ＋Ｃの意味は、Ａ（ＴＣＸ）＝Ｂ（ＴＣＸ）＋Ｃ（ＴＣＸ）である。

[0107] ７．３：主語名Ａ（）は変数主語名Ｂ（）、Ｃ（）の主語が既に成立し、且つこのＡ（）の算定を許可する条件が充足されていることに於いて成立する。もし、その条件をＸ（）＝Ｙ（）とすれば、Ｘ（）、Ｙ（）が既に成立しており、且つＸ（）＝Ｙ（）なら、このＡ（）の算定（＋）が許可されるということである。つまり、Ｂ（）、Ｃ（）、並びにＸ（）＝Ｙ（）となるＸ（）、Ｙ（）が成立していることに於いて、このＡ（）は成立する。しかし、これらはこのＡ（）が成立するに足りる十分条件の部分に過ぎない。なぜなら、Ｂ（）、Ｃ（）、Ｘ（）、Ｙ（）達だけでＡ（）を成立させるに適うかどうかの判定は不十分だからである。

[0108] ７．４：これら主語名達は更に脈絡の歩を進めて脈絡関係が全て完結する迄、即ち、この脈絡の終端となる構文の主語名に至る迄、辿り尽くさなければＡ（）の成立は保証することが出来ない。特に論理結合型のプログラム、即ち、ＬＣＰＳにはこのことは不可欠である。付言すれば、ここまで辿れる脈絡がＡ（）の十分条件の全体である。

[0109] ７．５：終端となる構文とはＲＥＡＤ文、定値文である。

[0110] ８．（ＬＣＰＳの超言語脈絡）

８．１：ＬＣＰＳの脈絡とは図２の情報のことである。

[0111] ８．２：図２の情報を絵的化すれば、図４である。図４は図２の情報から本発明者がスケッチしたものである。

[0112] ８．３：ＬＣＰＳの脈絡は図４で明らかな様に調和脈絡と超言語脈絡で構

成される。しかし、単にＬＣＰＳの脈絡と記す場合は超言語脈絡を指す。

[0113]　８．４：調和脈絡とは調和座標を用いてＬＣＰＳの全構文の実行順位を示すための構文の脈絡である（図４の点線矢線参照）。

[0114]　８．５：超言語脈絡とはＬＣＰＳに属す構文の主語名、変数主語名に付与される超言語座標のＴＣＸ２を同じ主語名の関係で結ぶ脈絡の事である。図４の実線矢線のことである。

[0115]　８．６：超言語座標とは全構文の主語名、変数数主語名に付与される座標の対（ＴＣＸ１，ＴＣＸ２）のことである。ＴＣＸ１とは主語名、変数主語名の領域を定義する領域定義文の調和座標ＴＣＸのことである。そして、ＴＣＸ２とはＴＣＸ１で示される領域に主語（値）を設定する構文のＴＣＸのことである。

[0116]　８．７：超言語脈絡に於ける発端主語名とはＬＣＰＳの全構文に付与される唯一の調和座標に於ける最小のＴＣＸを有する主語ベクトルＬ４、またはＷ４の第２規約で統治される構文の主語名のことである。

[0117]　８．８：同様に、超言語脈絡に於ける終端主語名とは主語ベクトルＲ２の第２規約で統治される入力構文の主語名、並びに主語ベクトルＬ２の第２規約で統治される定値文の主語名の主語のことである。

[0118]　８．９：超言語脈絡の発端、終端以外の構文は発端と終端を結ぶ超言語脈絡の節点の役割を担う（図４参照）。

[0119]　８．１０：超言語座標とはＴＣＸの対のことである。上記７の変数主語名を実体化する構文２のＴＣＸは構文１の可能的直近、且つその過去に位置しなければならない（詳細は図２の情報、及び、図４を参照）。

[0120]　８．１１：変数主語名の超言語座標は変数主語名を実体化させる主語名に向かう矢線で結ばれる。超言語脈絡とはこの矢線の全体である（図４参照）。

[0121]　８．１２：ＬＣＰＳの脈絡は構文に係る十分条件の全体を内包して、且つＬＣＰＳの十分条件の全体を捉える仕組になっている。ＬＣＰＳの脈絡の発端主語名が唯１個となるのはこのためである（図４参照）。

[0122]　８．１３：ＬＣＰＳの調和脈絡は１個のＬＣＰＳとして編集される。そして、このＬＣＰＳに属す全構文に対して、調和座標を基に実行順序は付与される。ＬＣＰＳの調和脈絡とはこのＴＣＸによる実行順位のことである（図４の点線矢線参照）。

[0123]　８．１４：調和脈絡を基に、全構文の主語名、変数主語名に超言語座標が付与される。

[0124]　８．１５：超言語脈絡の終端に至る過程で節点（ノード）となる構文が存在しなければ、超言語脈絡の経路は途切れる。つまり、節点となる構文を持たない構文は超言語脈絡の途切れとして、その構文のＴＣＸは図３の情報３の左欄に記される。この構文は図２の情報１、図４並びに図５でみることが出来る。

[0125]　８．１６：超言語脈絡の途切れを生じさせる構文はＬＣＰＳの宿命的な構造欠陥が背因となり生じるＴＣＸの乱れ、超言語座標の乱れにより発症している。しかし、ＬＣＰＳに棲息するこの仕組をＬＣＰＳの作成者が従来の検証法で事前に捉え且つ認識することは不可能である。ＬＣＰＳを部品化して構築されるシステムはバベル塔の様な話になる。この様なシステムが横行する世界ではＬＣＰＳに棲息するこの仕組が今後、間違いなく目覚め始めることになる。

[0126]　８．１７：超言語脈絡を途切れさせる構文が存在することは本来的にはあり得ないと考えられている。しかし、その思いはどうであれ、存在していれば、この場合、超言語脈絡の発端となる構文の主語名の主語は不正値となる。本願ではこの場合のこの発端主語名をバグの起因と記し、そのＴＣＸを図３の情報３の右欄に記す。そして、この発端となる構文の主語名を変数主語名とする構文の主語名の主語も不正値となるので、その様な構文を、本願はバグ事象を発症させる構文として、それらのＴＣＸを図３の情報４の左欄に記す。

[0127]　８．１８：発端主語名から図２を用いて超言語脈絡を辿ることが出来る。この過程で超言語脈絡を途切れさせる構文が存在すれば捉えることが出来る

。因みにデータ揺動を発症させる構文は超言語脈絡を途切れさせる構文となる。この構文は図２の情報で発見することが出来る。これら構文のＴＣＸは図３の情報６に記される。

[0128]　８．１９：バグ事象の区分１の構文の不正値の修正はそのマナーを心得ている技術者なら正統な対応は可能である。バグ事象の区分２の場合の対応はバグ事象とは何かを会得していなければ正統な対応は不可能である。

[0129]　８．２０：ＬＣＰＳは不正値の問題を問題として会得していない人々に因り作成されている現実がある。また、ＬＣＰＳにはその様な構文が内包されていても稼働が続けられるという構造欠陥がある。本願発明のＬＣＰＳの脈絡はその様な構文の存在を明らかにすることが出来る。

[0130] ９．（平文法の要旨）

平文法と総称される平文法の作業手続の要旨を以下に記す（図１参照）。

９．１：ＬＣＰＳから図２の情報を生成すること。

９．２：図２の情報から脈絡の発端となる１個の主語名をその定義規約に従い決めること。

９．３：図２の情報から脈絡の終端となる全主語名をその定義規約に従い決めること。

９．４：発端主語名から超言語脈絡を辿り、脈絡の途切れ点となる構文を探すこと。脈絡の途切れ点となる構文が存在すれば、脈絡の正統性は成立しない。

９．５：脈絡の正統性が成立しない場合、脈絡の発端となる主語名はバグの起因となる。

９．６:バグの起因となる構文の主語名が変数主語となる脈絡上の構文を図２の情報から抽出すればバグ事象となるバグ構文である。

９．７：図２の情報からデータ揺動の規約に従いその構文が存在すればそれがデータ揺動構文である。

９．８：図２の情報から図３の情報２、３、４、５、６の情報をそれぞれの規約に従い決定すること。

９．９：図４を描くための情報を図２の情報から編集すること。

[0131]（図の補足説明）

以下に図の補足説明を行う。

[0132]（図１の補足）

平文法とは、入力側情報から出力側情報を生成するための機械的作業手続きの総称である。

平文法４はグラフ化ツールに置き換えることも可能である。

図２は図４を描くためのパラメータ情報である。図７は発端主語名から超言語脈絡を導出するためのパラメータ情報である。

[0133]（平文法の補記）

平文法１は２種の入力側情報、即ち、１．ＬＣＰＳと２．ＬＣＰＳの７種の動化パラメータから図２を生成するための機械的手続きの事である。

平文法２は図２から図７の情報の編集、並びに図３の情報３、４、５を捉えるための機械的手続き事である。

図７はＬＣＰＳの発端主語名から超言語脈絡を辿るためのパラメータ情報である。図２は図４の２種の脈絡、即ち、調和脈絡、超言語脈絡を導くためのパラメータ情報である。

平文法３は図７の超言語脈絡の情報から図１の情報６、７、８、９を求めるための機械的手続きのことである。

平文法４は図２の情報から図４を描くための機械的手続きの事である。

[0134]（図６の補足）

１１．超言語脈絡に途切れを発症させる構文が存在すれば、発端構文の主語名の主語には十分条件の全体が成立しないので、発端主語名の主語は、結果的に不正値である。

１２．途切れ構文の存在条件は発端構文の主語名の主語を変数主語とする構文数を最大にする超言語脈絡上の構文である（図６参照）。

１３．上記１２．で決まる構文の主語名の主語は結果的に不正値となる。

１４．本願発明のＬＣＰＳの脈絡を用いれば、主語名の主語の判定は実値

を求める必要がない。これがテストデータによる従来的な実行検証法との基本的な違いである。

[0135]（図２の補足）

情報１：７種の動化パラメータを課せばＬＣＰＳから求められる「ＬＣＰＳの脈絡」を捉えるための情報。

[0136]（図３の補足）

情報２：「ＬＣＰＳの脈絡」の発端、終端となる主語名のＴＣＸ。情報１から求められる。

情報３：「ＬＣＰＳの脈絡」から導出される情報のＴＣＸ。

情報４：「ＬＣＰＳの脈絡」から導出される情報のＴＣＸ。

情報５：「ＬＣＰＳの脈絡」から導出される情報のＴＣＸ。

情報６：情報１から導出される情報のＴＣＸ。

[0137]（図４の補足）

（１）図２、図７の情報はＬＣＰＳの脈絡をグラフ化ツールで図表現する時のパラメータ。

（２）図４はＬＣＰＳ例の脈絡を図化したものである。図４はＬＣＰＳの脈絡の定義通りに描かれているが、図の概要を容易に理解できるように全ての超言語座標が記されているわけではない。敢えて省略してある。また、図４ではデータ揺動箇所が３か所出現しているが、これはＬＣＰＳ例の脈絡の概念を分かり易くするためにＬＣＰＳの部分を対象としたことに因る。実際に本願発明者が捉えたデータ揺動は２５年余の調査の中では３件である。また、本例のＬＣＰＳ例ではデータ揺動（図３の情報６）しか発症していない。そして、図３の情報３、情報４の左欄は空白である。

（３）点線は調和脈絡。調和脈絡ではＬＣＰＳの全実行構文がカバーされていれば調和脈絡は成立する。

（４）実線は超言語脈絡。超言語脈絡では主語名または変数主語名の成立関係が捉えられている。

（５）例えば、Ｆ＝Ｒ－Ｆの左辺のＦを主語名Ｆと呼ぶ。

（６）右辺のＲ、Ｆを変数主語名Ｒ、Ｆと呼ぶ。

（７）図中の「？」について、変数主語Ｂの超言語脈絡は成立させられない。変数主語Ｂの主語値は不定。ゆえに、以降の実行構文は成立しないことになる。このことはプログラムの部分破壊が生じているとみなされる（バグ事象２）。

[0138]（図７の補足）

１．超言語座標付ＬＣＰＳの全構文（含む領域定義文）：図２を複写

２．構文の主語名

３．構文の変数主語名

４．変数主語名を起点として超言語脈絡を成立させる主語名を脈絡主語名と記す（図４参照）。

５．超言語脈絡が終端に至ればこの超言語脈絡は成立する。

６．超言語脈絡が終端に至らなければこの超言語脈絡は本図を用いて連接される。

７．超言語脈絡を連接する主語名が見つからなければこの超言語脈絡は途切れる。

８．その実行構文のＴＣＸは図３の情報３の左欄に登録される。

[0139]（図８の補足）

２．において、主語名の主語の脈絡の発端主語名を決定するために必要な情報は、以下の通り。

１：調和座標

２：超言語座標

３：構文種別

４：構文種別と主語ベクトルの関係

[0140]（図９の補足）

主語ベクトルはＬＣＰＳの動的アルゴリズムを捉えるパラメータのひとつである。主語ベクトルの構造概念はＬＹＥＥ理論の要約で公開されている。主語ベクトルのプログラム例はＳＦの学習要領で公開されている。

第１規約の「第４領域は成立済か」という１文がベクトルの構造を決める命題になっている。

全ベクトルに同期を成立させるために、第２、３規約は不可欠。

第５規約は生命作用ができる範囲の予知の規約である。因みに、シナリオ関数ではこの規約は実体（主語）のスタックを用いて予知される仕組になっている。この規約を前提として成立するプログラムがＡＩである。ベクトルの基で成立する予知の仕組みは重要である。

第４規約における記憶は全ベクトルの第３規約を成立させるために不可欠である。

第５、６、７規約は可能的に成立する第３規約のために不可欠である。

[0141]（発明の狙点）

本発明はＬＣＰＳの伝統的な検証法に替わる検証法である。ＬＣＰＳから機械的手続きで求められる「ＬＣＰＳの脈絡」（図４参照）を用いて、ＬＣＰＳが実行時にＬＣＰＳの宿命的な構造欠陥を遠因として、構文の主語名にＬＣＰＳの意図に沿わない不正値を発症させるその様な全ての構文を本作業者が従来の検証法の様なＬＣＰＳに掛る知見を必要とせずにＬＣＰＳの脈絡に対する作業の仕方の指示を受ければ、作業者は机上で捉えることが出来る方法である。この作業者の作業手続きはツール化することも可能である。

[0142]（発明の仕組）

本発明は、ＬＣＰＳの意図に沿わない不正値を発症させる構文とこの構文の不正値に関わる構文をＬＣＰＳの編集により求められるＬＣＰＳの超言語脈絡を捉える情報（図２）を用いて図３の形式で特定する方法である。

[0143]（本発明の意義）

本発明のＬＣＰＳから機械的に求められる「図２の内容」（図２参照）、またはこの内容と等価な「ＬＣＰＳの脈絡」（図４参照）はＬＣＰＳが実行時に構文の主語名に不正値を発症させる構文を捉える為の仕組として用いられる。これはＬＣＰＳの十分条件の全体を捉える仕組の応用である。そして、ＬＣＰＳの十分条件の全体はＬＣＰＳの実行時の様相である。故にこれは

これ迄誰も捉えることはできなかった様相である。順序は逆になるが、これを動性アルゴリズムで観察可能にしたプログラムがＳＦである。そして、これを具体的に捉えてみせたのが本願発明の図４であり図２である。

[0144]　ＬＣＰＳの全体とその部分の関係が曖昧になるのはＬＣＰＳの解が求められないことが原因である。故に、ＬＣＰＳの脈絡ではＬＣＰＳの全体に対して１組の調和座標を付与することが求められる。そして、ＬＣＰＳの脈絡の発端となる構文は、ＬＣＰＳの中から脈絡の規則に基づいて選ばれる。

[0145]　ＬＣＰＳのこの曖昧さが調和座標の付与をＬＣＰＳ関係者達に無意識的な、それでいて共通の誤りを、換言すれば、あいまいにさせる原因になっている。結果的に、この原因がＬＣＰＳの脈絡に途切れを生じさせる構文を発症させている。この構文の存在をＬＣＰＳの開発担当者の経験的知見によって説明することは困難である。

図面の簡単な説明

[0146] [図1]本発明の構想図

[図2]ＬＣＰＳの超言語脈絡を捉える為の情報（例）

[図3]図２から捉えられるＬＣＰＳの解析情報（例）

[図4]図２の図化された様相（例）

[図5]超言語脈絡に係る概念

[図6]ＬＣＰＳのアルゴリズムの全体と部分の関係

[図7]本発明の実施手順

[図8]本発明に係る定義概念

[図9]主語ベクトルの構造

発明を実施するための形態

[0147]　本発明が対象とするプログラムは実行時に非同期型アルゴリズムの全体（ＮＳＡ）を発生させるコンパイル済の実行可能な論理結合型プログラム（ＬＣＰＳ）のソース（ＬＣＰＳ）である。

[0148]　本発明は「ＬＣＰＳの超言語脈絡」を捉えるための情報を上記（０１）のＬＣＰＳから図２の形式で生成する方法のことである。ＬＣＰＳの超言語脈

絡の定義は該当項で示されている。

[0149]　ＬＣＰＳの超言語脈絡とは上記（０１）のＬＣＰＳが実行時に発生させる非同期型アルゴリズム（ＮＳＡ）の成否を判定するための仕組である。

[0150]　ＬＣＰＳの超言語脈絡をＬＣＰＳから求める手続を以下に示す（図７参照）。

ＬＣＰＳの全構文に構文の実行順位を示す１組の「調和座標」を付与する。これにより、「ＬＣＰＳの調和脈絡」が決定される。

ＬＣＰＳの全構文に「動化パラメータ」を付与する。

構文の主語名、変数主語名に「超言語座標」を付与する。

以上の手続きにより、「ＬＣＰＳの超言語脈絡」を捉える情報図２が決定される。

調和座標、ＬＣＰＳの調和脈絡、動化パラメータ、超言語座標、ＬＣＰＳの超言語脈絡の定義は該当項で示されている（図２参照）。

[0151]　ＬＣＰＳの超言語脈絡に「脈絡の途切れ」、「脈絡の誤記」が生じていればＬＣＰＳの正統性は成立しない。ＬＣＰＳの超言語脈絡に於ける「脈絡の途切れ」、「脈絡の誤記」の見つけ方は該当項で示されている。

[0152]　ＬＣＰＳの超言語脈絡に脈絡の途切れ、脈絡の誤記が生じていなければ、ＬＣＰＳの正統性は成立している。

[0153]　脈絡の途切れ、脈絡の誤記が生じれば、ＬＣＰＳの超言語脈絡には「バグ構文を捉える仕組」が成立する。この仕組で捉えられる情報は図３の形式で示される。

[0154]　「バグ構文を捉える仕組」は該当項で示される。

産業上の利用可能性

[0155]　明らかなことは、人工物であり且つ機械物と異なるプログラムの完成度は、プログラムが実行時に発生させるアルゴリズム全体の正統性により決まる。しかし、ここ３０年を振り返れば、アルゴリズム的には二次三次的問題の処理時間の早い遅い問題がもっともらしく語られる傾向にある。アルゴリズムの基本問題はその正統性にある。

[0156]　しかし、この問題は棚上げされている。バグだらけのプログラムが迅速に処理結果を出したところで誇れるほどの意義はないからである。そして、処理時間問題は対人間関係に帰着する問題である。故に、その解決法はプログラム問題として解決されるべきが正統である。対人間関係を超える手段で解決されるべき問題ではない。そして、そのプログラム的解決手段はほかにいくらでもあるからである。故に、プログラムにとり、重要なことは、本発明が行う様に、アルゴリズムの正統性診断を可能にすることである。現行のプログラムがバグだらけであるとはこの問題が放棄されていることが背景にある。せめて、プログラム提供側には稼働中のプログラムに内在するバグ事象リストぐらいは利用者に対し公開する義務がある。因みに、本発明を用いれば、稼働中のプログラムのバグ事象リストは図３の形式で明らかにすることが出来る。

[0157]　ＳＦでは事前に意図的に特別な学習を行わせるわけではない。ＳＦは実行時に発生する同期アルゴリズム（ＳＡ）の仕組が、（１）バグ事象を自律的に回避し、（２）ウイルス情報を自律的に無力化して正統なアルゴリズムをその同期アルゴリズムの中に自律的に成立させる方式である。人間関係的な次元を超える処理時間の高速性問題が、もし、ＳＦにあるのだとすれば、その解法のためにＳＦが処理時間の高速性環境を利用する意義はある。他方、ＬＣＰＳが上記（１）（２）の問題が未解決である限り、そして、アルゴリズムがＮＳＡである限り、ＬＣＰＳには処理時間の高速性問題を議論する立場にはない。ＡＩもＬＣＰＳで作成されている限り、課題はＬＣＰＳと同じ立場である。プログラムＳＦを超える世界に位置し、且つ学習を実行時のアルゴリズムが必要としないプログラムでなければＡＩなどとのたまうべきではない。アルゴリズム的には今風のＡＩは同期アルゴリズムを発生させるＳＦ以下のＬＣＰＳに過ぎない。

[0158]　この分野の始祖的立場におられたマッサーニ先生が指摘された専門性の空洞化、空洞化の独占化が始まるとの指摘は２０年前の話ではあるが、現実はまさにご指摘のとおりの状況である。

[0159]　ＬＣＰＳが発生させる非同期型アルゴリズムが同期型アルゴリズムに転換されない限り、プログラム世界には技術的にも思想性に於いても自動運転、ロボットに見られる様に特別な進歩の形跡はどこにもない。本発明はＬＣＰＳから求められる超言語脈絡の発見である。

そして、これを用いて、ＬＣＰＳが発生させるＮＳＡの正統性を検証可能にする。

請求の範囲

［請求項1］　論理結合型プログラムのソースから超言語脈絡を求めるステップと、

前記超言語脈絡を用いて前記論理結合型プログラムが実行時に発生する非同期型アルゴリズムの正統性を検証するステップと

を備える方法。

［請求項2］　前記超言語脈絡を求めるステップは、前記論理結合型プログラムのソースの構文毎に行番号を付与し、前記行番号で識別される構文毎に、超言語座標、調和座標、構文種別、及び、主語ベクトル種別を特定するステップを含む

請求項１に記載の方法。

［請求項3］　前記超言語座標、前記調和座標、前記構文種別、及び、前記主語ベクトル種別により特定される、前記行番号で識別される構文間の関係を図に表すステップ

を備える請求項２に記載の方法。

［請求項4］　前記超言語脈絡を用いて、前記非同期型アルゴリズムに不正値を発症させる前記論理結合型プログラムのソースの構文を特定するステップ

を備える請求項１に記載の方法。

［請求項5］　前記構文を特定するステップは、途切れ構文又は不正構文よりも発端主語名に近い条件文が存在する場合、バク構文の修正が不可能と判定するステップを含む

請求項４に記載の方法。

［請求項6］　前記超言語脈絡を求めるステップは、前記論理結合型プログラムのソースの構文毎に行番号を付与し、前記行番号で識別される構文毎に、超言語座標、調和座標、構文種別、及び、主語ベクトル種別を特定するステップを含み、

前記構文を特定するステップは、前記論理結合型プログラムのソー

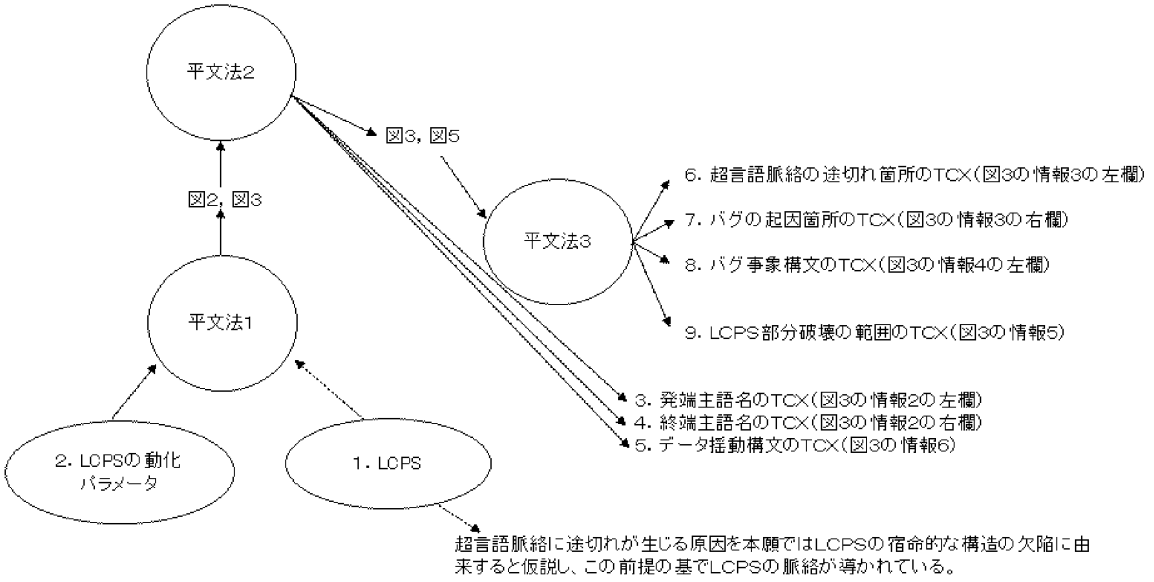
スの前記行番号で識別される構文毎に、発端主語名、終端主語名、脈絡の途切れ又は錯誤の有無、発端主語名がバグの起因となっているか否か、バグ事象を構成するバグ構文、バグ事象仕分けコード、部分破壊範囲、及び、データ揺動構文を特定するステップを含む

請求項５に記載の方法。

要　約　書

本発明は、伝統的なＬＣＰＳの検証法の様に、意図的に検証用データを準備し、それを検証対象のＬＣＰＳで電算機処理させて、ＬＣＰＳが発生させるアルゴリズムの正統性を意図的に成立させるための途を求める方法とは異なる方法を提案する。本発明は検証用データを一切使用せずに、検証対象のＬＣＰＳから本発明の普遍化された方法でＬＣＰＳの超言語脈絡を求め、且つそれを本発明の普遍化された方法で解析して、ＬＣＰＳが稼働時に発生させるアルゴリズムの正統性の成否を判定する方法を提案する。この方法の中で、ＬＣＰＳが稼働時に発症させる全てのバグ事象は本発明の普遍化された方法でバグ構文として明らかにされる。ＬＣＰＳにバグ構文が存在しなければ、ＬＣＰＳが稼働時に発生させるアルゴリズムは正統である。上述の方法は人的作業により行われても、専用プログラムに従いコンピュータにより行われてもよい。

[図1]



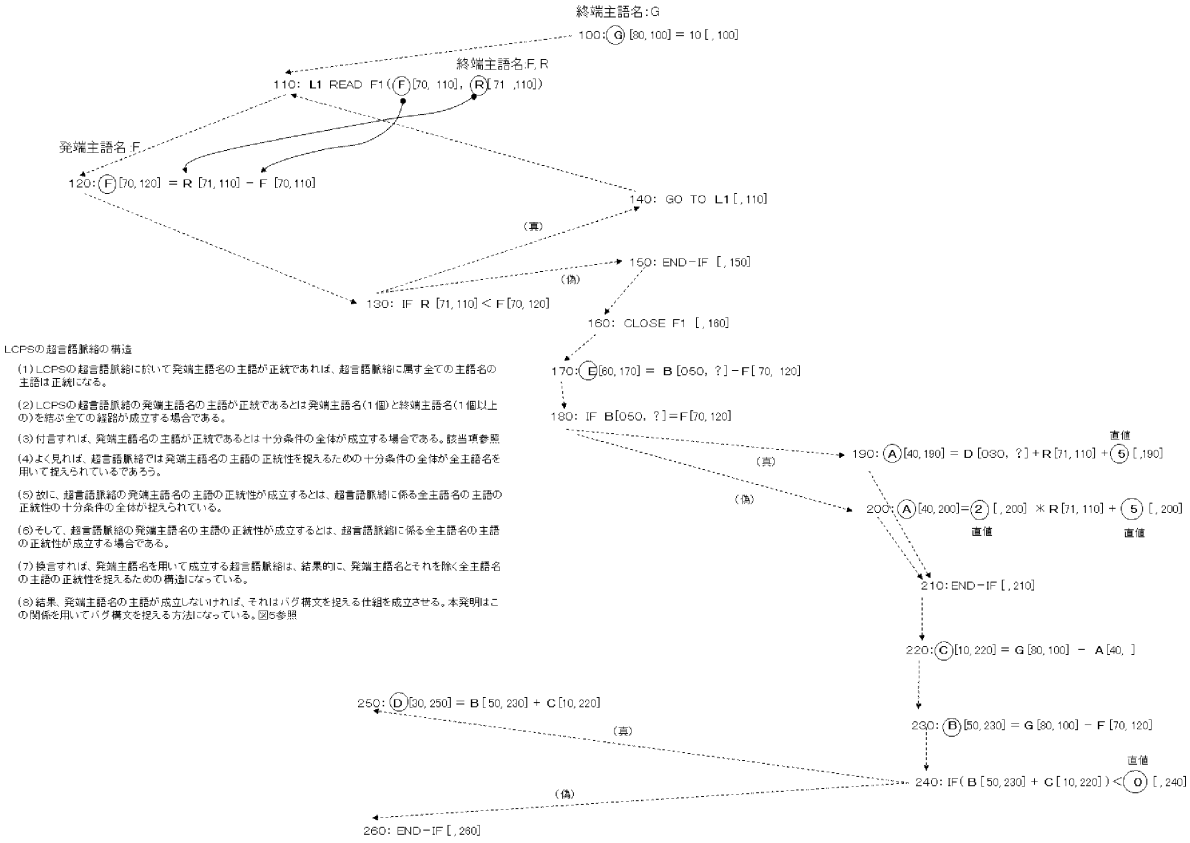
[図2]

動化パラメータ								
行番号(TCX)	情報1							
	主語名，変数主語名に対する超言語座標の付与	調和座標の付与					構文種別	主語ベクトル種別の付与
		TCX	TCY＝TCZ1	TCZ2	TCZ3	TCZ4		
*以下領域定義文								
010	C[010,]						領域定義文	—
020	R[020,]						領域定義文	—
030	D[030,]						領域定義文	—
040	A[040,]						領域定義文	—
050	B[050,]						領域定義文	—
060	E[060,]						領域定義文	—
070	F1(F)[070,]						領域定義文	—
071	F1(R)[071,]						領域定義文	—
080	G[080,]						領域定義文	—
*以下実行文								
100	G[080,100]=10[,100]	100	110				定置文	L2(G:100)
110	L1[110,] READ F1(F[070,110],R[071,110])	110	120				入力文	R2(F1:110)
120	F[070,120] =R[071,110]-F[070,110]	120	130				代入文	L4(F:120)
130	IF R[071,110]<F[070,110]	130	140	150	150	160	判定文	L3(140)140,150,150,160
140	GO TO L1[,110]	140	110				制御文	—
150	END-IF[,150]	150	160				翻訳文	—
160	CLOSE F1[,160]	160	170				翻訳文	—
170	E[060,170]=B[050,?]-F[070,120]	170	180				代入文	L4(E:170)
180	IF B[050,?]=F[070,120]	180	190	200	210	220	判定文	L3(A)190,200,210,220
190	A[040,190]=D[030,?]+R[71,110]+5[,190]	190	210				代入文	L4(A:190)
200	A[040,200]=2[,200]*R[71,110]+5[,200]	200	210				代入文	L4(A:200)
210	END-IF[,210]	210	220				翻訳文	—
220	C[010,220]=G[080,100]-A[040,?]	220	230				代入文	L4(C:220)
230	B[050,230]=G[080,100]-F[070,120]	230	240				代入文	L4(B:230)
240	IF(B[50,230]+C[10,220])<0[,240]	240	250	260	(TCZ3)	(TCZ4)	判定文	L3(D:250)250,260,(TCZ3),(TCZ4)
250	D[030,250]=B[050,230]+C[010,220]	250	260				代入文	L4(D:250)
260	END-IF[,260]						翻訳文	—
(以下略)								

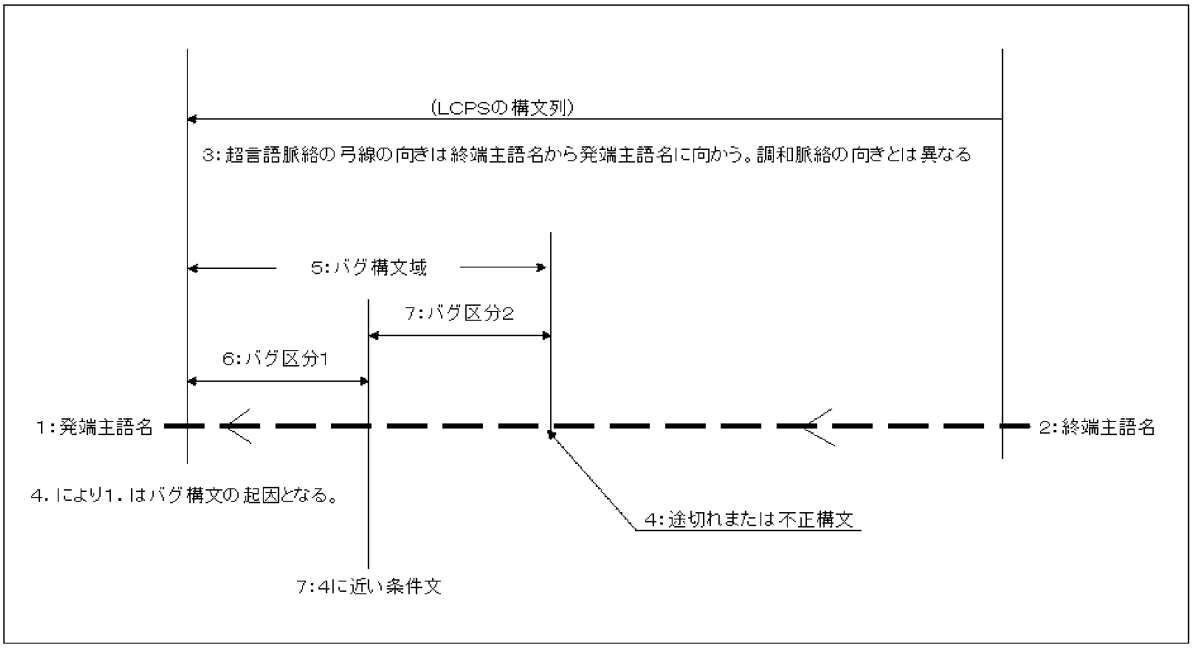
[図3]

	LCPSの解析情報							
	情報2		情報3		情報4		情報5	情報6
行番号(TOX)	発端主語名	脈絡の終端主語名	{脈絡の途切れ}	バグの起因	{バグ事象}	{バグ事象仕分けコード}	{部分破壊範囲}	{データ揺動構文}
*以下領域定義文								
010								
020								
030								
040								
050								
060								
070								
071								
080								
*以下実行文								
100		100						
110		110						
120	120							
130								
140								
150								
160								
170						2	170-230	170
180								
190						2	190-250	190
200								
210								
220						2	190-220	220
230								
240								
250								
260								
(以下略)								

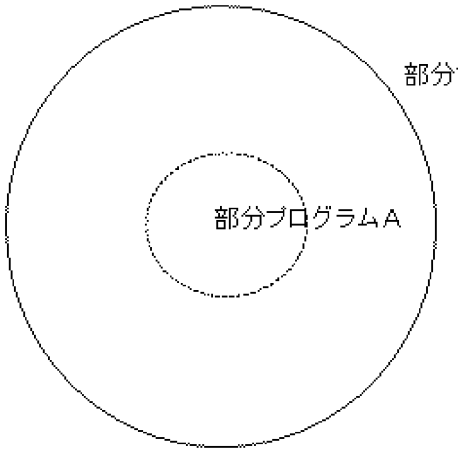
[図4]


終端主語名:G
100: G [80, 100] = 10 [, 100]
終端主語名 F, R
110: L1 READ F1 (F [70, 110], R [71 ,110])
発端主語名 F
120: F [70, 120] = R [71, 110] − F [70, 110]
140: GO TO L1 [, 110]
(真)
150: END−IF [, 150]
(偽)
130: IF R [71, 110] < F [70, 120]
160: CLOSE F1 [, 180]
LCPSの超言語脈絡の構造
(1) LCPSの超言語脈絡において発端主語名の主語が正統であれば、超言語脈絡に属す全ての主語名の主語は正統になる。
(2) LCPSの超言語脈絡の発端主語名の主語が正統であるとは発端主語名(1個)と終端主語名(1個以上の)を結ぶ全ての経路が成立する場合である。
(3) 付言すれば、発端主語名の主語が正統であるとは十分条件の全体が成立する場合である。該当項参照
(4) よく見れば、超言語脈絡では発端主語名の主語の正統性を捉えるための十分条件の全体が全主語名を用いて捉えられているであろう。
(5) 故に、超言語脈絡の発端主語名の主語の正統性が成立するとは、超言語脈絡に係る全主語名の主語の正統性の十分条件の全体が捉えられている。
(6) そして、超言語脈絡の発端主語名の主語の正統性が成立するとは、超言語脈絡に係る全主語名の主語の正統性が成立する場合である。
(7) 換言すれば、発端主語名を用いて成立する超言語脈絡は、結果的に、発端主語名とそれを除く全主語名の主語の正統性を捉えるための構造になっている。
(8) 結果、発端主語名の主語が成立しなければ、それはバグ構文を捉える仕組を成立させる。本発明はこの関係を用いてバグ構文を捉える方法になっている。図5参照
170: E [60, 170] = B [050, ?] − F [70, 120]
180: IF B [050, ?] = F [70, 120]
(真)
190: A [40, 190] = D [030, ?] + R [71, 110] + 5 [, 190]
直値
(偽)
200: A [40, 200] = 2 [, 200] * R [71, 110] + 5 [, 200]
直値
直値
210: END−IF [, 210]
220: C [10, 220] = G [80, 100] − A [40,]
250: D [30, 250] = B [50, 230] + C [10, 220]
230: B [50, 230] = G [80, 100] − F [70, 120]
(真)
直値
240: IF (B [50, 230] + C [10, 220]) < 0 [, 240]
(偽)
260: END−IF [, 260]

[図5]

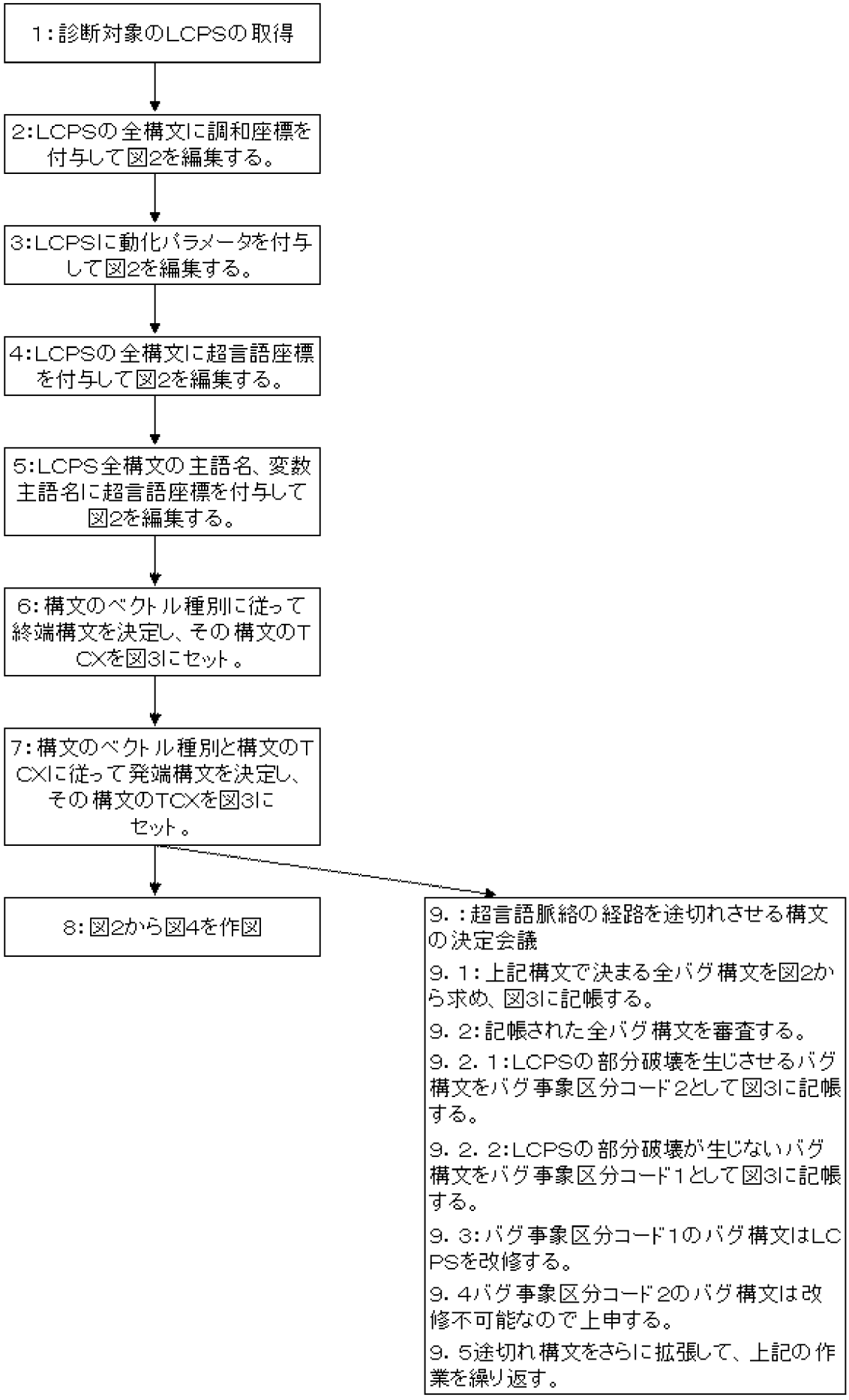

(LCPSの構文列)
3:超言語脈絡の弓線の向きは終端主語名から発端主語名に向かう。調和脈絡の向きとは異なる
5:バグ構文域
7:バグ区分2
6:バグ区分1
1:発端主語名
2:終端主語名
4. により1. はバグ構文の起因となる。
4:途切れまたは不正構文
7:4に近い条件文

[図6]



1．実行時に非同期型アルゴリズムの部分を発生させる論理結合型プログラの部分プログラムA
2．実行時に非同期型アルゴリズムの全体を発生させる論理結合型プログラの全体プログラムB
3．部分プログラムAの十分条件の全体の成否はAを内包するプログラムBに従属する。
部分Aは単独でAの十分条件の全体を決定することが出来ない。

[図7]


1：診断対象のLCPSの取得
2:LCPSの全構文に調和座標を付与して図2を編集する。
3:LCPSに動化パラメータを付与して図2を編集する。
4:LCPSの全構文に超言語座標を付与して図2を編集する。
5:LCPS全構文の主語名、変数主語名に超言語座標を付与して図2を編集する。
6:構文のベクトル種別に従って終端構文を決定し、その構文のTCXを図3にセット。
7:構文のベクトル種別と構文のTCXに従って発端構文を決定し、その構文のTCXを図3にセット。
8:図2から図4を作図
9.：超言語脈絡の経路を途切れさせる構文の決定会議
9.1:上記構文で決まる全バグ構文を図2から求め、図3に記帳する。
9.2:記帳された全バグ構文を審査する。
9.2.1:LCPSの部分破壊を生じさせるバグ構文をバグ事象区分コード2として図3に記帳する。
9.2.2:LCPSの部分破壊が生じないバグ構文をバグ事象区分コード1として図3に記帳する。
9.3:バグ事象区分コード1のバグ構文はLCPSを改修する。
9.4バグ事象区分コード2のバグ構文は改修不可能なので上申する。
9.5途切れ構文をさらに拡張して、上記の作業を繰り返す。

[図8]

1. LCPS(コンパイル済みの従来プログラムのソース)

2. LCPSを動化させるパラメータを以下に示す。(11種の構文種別、主語名、変数主語名、主語、調和座標、超言語座標、5種の主語ベクトル)
LCPSの全体に対して1組の調和座標を付与する。

3. プログラムの十分条件の全体はそのプログラムの実行状態の様相である。
3. 1:SFの十分条件の全体はSFに属す全主語ベクトルの関係を用いて捉えられる。
3. 2:LCPSでは十分条件の部分しか捉えられない。 (本発明への回廊) → 3. 2. 1:本発明の研究過程では「LCPSの脈絡」を捉える方法が論考された。その内容を以下に記す。
(1)LCPSの脈絡とは実行時のCPSに属す構文の主語名の主語に発症する不正値を捉えるための仕組である。
(2)LCPSの脈絡の仕組はSFの動性アルゴリズム(SFの十分条件の全体と同義)を基に設計されている。
(3)LCPSの脈絡は調和脈絡、超言語脈絡として定義される2種の脈絡で成立する。
(4)調和脈絡とは調和座標の行番号で順序付けられる構文の脈絡である。
調和座標はLCPSの全体を統治する唯一組の座標である。図4の点線矢線参照
(5)調言語脈絡とは構文に属す主語名、変数主語名の調和座標のTCX2の脈絡である。
下記(6)、図4の点線矢線を参照
(6)超言語座標とは構文に属す変数主語名に付与される二つの行番号の対(TCX1, TCX2)である。
TCX1は変数主語名の領域を定義する領域定義文の行番号、TCX2は変数主語名の領域を実体化する構文の行番号である。主語名に付与される超言語座標のTCX1は変数主語名の場合と同じ、TCX2は主語名が属す構文の行番号である。
(7)超言語脈絡とは調和座標の行番号TCX2で結ばれる主語名、変数主語名の脈絡のことである。
図4の実線弓線参照
(8)LCPSに2. の動化パラメータを課せば、そのLCPSからLCPSの脈絡を捉えるための全情報(図2)が求められる。
図2参照
(9)図2から以下の情報が求められる。求められる情報は図3に記録される。図3参照
(10)図2から求められる情報
10. 1:LCPSの脈絡のためのただひとつの発端主語名
10. 2:LCPSに存在する全ての脈絡の終端主語名
10. 3データ揺動を発症させる構文
(11LCPSの脈絡の途切れ箇所
11. 1:LCPSの脈絡に途切れを発症させる原因は超言語座標の誤りである。
その遠因はLCPSに解が存在しないことである。
11. 2:LCPSの脈絡の途切れ箇所を発見する方法は①LCPSの脈絡(図4)を発端主語名を起点として人為的に超言語座標を用いて辿る方法、②方法①を専用プログラムで辿らせる方法。

[図9]