(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5992079号
(P5992079)
(45)発行日　平成28年9月14日(2016.9.14)　(24)登録日　平成28年8月26日(2016.8.26)

(51)Int.Cl.　F I
G06F 21/56 (2013.01)　G06F 21/56

請求項の数 1　(全 52 頁)

(21)出願番号　特願2015-160560(P2015-160560)
(22)出願日　平成27年8月17日(2015.8.17)
(62)分割の表示　特願2015-514699(P2015-514699)の分割
原出願日　平成27年3月3日(2015.3.3)
(65)公開番号　特開2016-42361(P2016-42361A)
(43)公開日　平成28年3月31日(2016.3.31)
審査請求日　平成27年8月17日(2015.8.17)
審判番号　不服2016-2277(P2016-2277/J1)
審判請求日　平成28年1月27日(2016.1.27)
(31)優先権主張番号　特願2014-158612(P2014-158612)
(32)優先日　平成26年8月4日(2014.8.4)
(33)優先権主張国　日本国(JP)

早期審理対象出願

(73)特許権者　598037422
根来　文生
神奈川県鎌倉市山ノ内５９１番地
(74)代理人　100110559
弁理士　友野　英三
(72)発明者　根来　文生
神奈川県鎌倉市山ノ内５９１番地

合議体
審判長　高木　進
審判官　辻本　泰隆
審判官　須田　勝巳

最終頁に続く

(54)【発明の名称】ウィルス侵入検知及び無力化方法

(57)【特許請求の範囲】
【請求項１】
コンピュータに本業の業務処理を実行させるシナリオ関数の形式で書かれたプログラムであって、前記シナリオ関数の形式で書かれたプログラムは、主語となるデータ領域に対して内容を決定するための最小叙述構造体であるベクトル構造が任意順序で集積されたパレット４の臨界状態が達成されるまで循環する構造を有する座標関数４に係るプログラムと、前記ベクトル構造が任意順序で集積されたパレット２の臨界状態が達成されるまで循環する構造を有する座標関数２に係るプログラムと、前記ベクトル構造が任意順序で集積されたパレット３の臨界状態が達成されるまで循環する構造を有する座標関数３に係るプログラムとを内蔵し、前記パレット４が臨界状態になれば前記パレット２に、前記パレット２が臨界状態になれば前記パレット３に、前記パレット３が臨界状態になれば前記主語を成り立たせるための変数主語の第４領域の所在に応じて最上ランクに係る座標関数３、同一ランク係る座標関数４、１層下層ランク係る座標関数４のいずれかに制御を移す同期関数に係るプログラムを最上位の制御論理体として備え、
前記ベクトル構造はウイルス観察アルゴリズム（ＶＷＡ）と、前記ウイルス観察アルゴリズム（ＶＷＡ）の次の処理として配置される第１規約と、前記第１規約の次の処理として配置される第２規約と、前記第２規約の次の処理として配置される第３規約と、前記第３規約の判定が是の場合の次の処理として配置される第４規約と、前記第３規約の判定が否の場合の次の処理として配置される第５規約と、前記第５規約の判定が是の場合の次の処理として配置される第６規約と、前記第５規約の判定が非の場合の次の処理として配置さ

れる第７規約とを有し、前記ベクトル構造においては、
前記第１規約では前記ベクトルの正当性判定が行われ、
前記第２規約では前記ベクトルの本来の処理が行われ該第２規約を通過した証である第２フラグをオンにセットし、
前記第３規約では前記ベクトルの本来の処理に関する判定が行われ主語の脈略の正統性を判定する為の領域である第４領域の正統性を判定し、
前記第４規約では前記第３規約の判定が是であれば前記ベクトルで統治される命令文により求められた主語が前記第４領域に移され、かつベクトル自体の再起動を要請するための第６フラグをオフにセットし、
前記第５規約では前記第３規約の判定が是となる可能性の有無が判定され、かつ前記第４領域が初期値化され、
前記第６規約では前記第５規約の判定が是となる可能性がある場合の処理として、前記第６フラグをオンにセットし前記第２フラグをオフにセットし、
前記第７規約では前記第５規約の判定が是となる可能性がない場合の処理として、ベクトル自体の再起動の一時停止を宣言するための第７フラグをオンにセットし前記第２フラグをオフにセットするものであり、
前記シナリオ関数の形式で書かれたプログラムは、
前記シナリオ関数の形式で書かれたプログラムの初期値化状態である、前記第４領域の内容がオフ、前記第２フラグがオフ、前記第６フラグがオン、前記第７フラグがオフの状態における前記プログラムに侵入した任意のウィルスを、前記侵入されたプログラムのデータ領域ごとの汚染として第１の普遍的規則、即ちベクトルの正統性を捉える為の第４領域、第２フラグ、第６フラグ、第７フラグの相対関係で捉え、
前記シナリオ関数の形式で書かれたプログラムの実行状態である、前記第４領域の内容がオン、前記第２フラグがオン、前記第６フラグがオフ、前記第７フラグがオフの状態において第２の普遍的規則、即ち前記汚染が捉えられた前記データ領域を初期値化することで前記汚染を除染し、
前記シナリオ関数の形式で書かれたプログラムの実行状態である、前記第４領域の内容がオン、前記第２フラグがオン、前記第６フラグがオフ、前記第７フラグがオフの状態において第３の普遍的規則、即ち前記除染に続き前記ベクトルの第３、５，６，７規約が該ベクトルを再起させる構造を持つことで前記シナリオ関数の形式で書かれたプログラムの自らの前記実行状態を正常化させる、
ことを前記コンピュータに実行させることを特徴とするプログラム。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明はプログラムの構造論に係り、特に、コンピュータウイルス（以下、単に「ウイルス」ともいう。）問題を自律的に解法するプログラムの定義構造、及び同構造を備えたプログラム、同プログラムを搭載した記憶媒体、並びにウイルス問題を自律的に解法する方法・装置に関する。
【背景技術】
【０００２】
プログラム問題は、従来のプログラムの本質的な危うさに於いて、コンピュータウイルス問題に限らず、既に利便性の向上という観点だけでは、その品質の悪さ、開発維持のわずらわしさと費用の高騰化で弁明しきれない状況になっている。
【０００３】
周知の通り、プログラムはその効果の利便性の裏面で、ソフト危機と呼ばれる様相、に晒されている。
【０００４】
このような現状にありながら、プログラム開発維持の実状は、改善の志さえ排除される

様な魔物化状態にある。更に、コンピュータウイルス（以下、単に「ウイルス」ともいう。）の発生はコンピュータ社会をひとたまりもなく崩壊させることができる潜在力を有し、且つ誰もこの問題を解法できないという点で、この分野の足元がすくわれた状況になっている。これまでの様にソフト危機の問題を真摯に解法せずとも効果の利便性に事寄せてやり過ごせた様なわけには行かなくなったのである。

【０００５】

これ迄はソフト危機の問題を真摯には解法せずに、利便性の一点で、強力な体制による強制的規格化(プロクルースチン・ベッド)がこの分野の技術者、研究者、利用者、要するにこの分野の殆どのひとびとを人類規模で牛耳って来た。

【０００６】

結果的に、プログラム問題の解法の重要性に気付かない人々がプログラム問題を魔物化させている様に、ウイルス問題に対しても現状の施策はこれと同じであって、ウイルスの侵入を入口で阻む（ウイルス・バスター）、という発想に終始している。ウイルス問題の怖さはその才能を持つウイルス創作者なら誰もが密かにそして、決して所在を悟られない様に、ウイルス作成に参画できることである。即ち、その深刻さは量的問題に化けることである。

【０００７】

ウイルス問題に関し、これまでは、プログラムが存在する限り，プログラムへの侵入は阻止することができないと考えられているが、その考えを改めて、ウイルスに侵入されたプログラムが自力でウイルスを無力化する方法を見つける以外に人類はウイルス問題を克服することができない。

【０００８】

こういった問題に対して現在存在するウイルス対応商品群（以下単に、「商品群」ともいう。）はプログラムへのウイルス侵入をその手前で阻止するという考えによっている。すなわち、商品群がよって立つ地盤は、密かに入手したウイルスタグ、違法的な情報分析侵入残骸の解析をする等をしなければ成立しないものである。ウイルスの完全な侵入の完全阻止などはあり得ない。すなわち、本質的に、現行の商品群の施策ではウイルス問題に対して本質的な解法にはなり得ない。

【０００９】

本願は上記のように誤った方向に進化してきたウイルス侵入問題への本来的対処を真剣に考察しようとするものであるという点で、本願に有効な先行技術となるものは存在しない。したがって、下記にはこれまでウイルス問題に対してとってこられたという意味での共通項を有するものを挙げてある。

【先行技術文献】

【特許文献】

【００１０】

【特許文献１】特開２０１３－２４３８６４号公報

【特許文献２】特開２０１２－２３４５７９号公報

【発明の概要】

【発明が解決しようとする課題】

【００１１】

本来、進化したウイルスなら１個のウイルスで電子計算機システムを瓦解させることが可能となってしまうことを鑑みれば、ウイルスは１個でもプログラムに侵入させてはならない、というのがウイルス問題に対する正しい認識であるべきである。

換言すれば、この点がウイルス問題解法の絶対条件である。

【００１２】

したがって、ウイルス問題の解法は、現状のウイルス・バスター的なアイデアでは成就できない。即ち、その為の解法の原理を発見しなければ、ウイルス問題は解法することができない。

【００１３】

本発明は、こうした従来技術の問題点を解決するもので、コンピュータウイルス問題に対する根本的な解法を与えること、すなわち、ウイルス問題を自律的に解法するプログラムの定義構造及び同構造を備えたプログラム、同構造を搭載した記憶媒体、並びにウイルス問題を自律的に解法する方法を提供することを目的とする。

【課題を解決するための手段】

【００１４】

かかる目的を達成するために、本発明は、これまでにない新たなアプローチによってウイルス問題への解法（すなわち本質的な解決）を与えるものである。以下、本発明を具体的に説明するために、次のような順を追って説明を進める。

Ａ．課題の解法に至る経緯

Ｂ．ウイルス問題の捉え方

Ｃ．本願独自の方法論の構築

【００１５】

（Ａ．課題の解決に至る経緯）

本発明に至る研究は、叙述法を改善すればソフト危機は解法できるとの仮説に基づき、本発明者が１９７３年から開始したのが嚆矢である。本研究前段の１５年間は、プログラムの本質を考える為に世界の知見(論文)の収集と様々な分野の数百本に及ぶプログラムの動性分析が行われた。プログラムの動性分析はプログラムの良い静的定義（ソースコード）を見出す為には、不可欠となる。プログラムに限らず、動性体の良い静的定義を行う為には、その動性を良く知ることが重要である。

【００１６】

しかし、この分野ではプログラムの動性分析は、当時から今日に至るも十分には行われていない。因みに、図１２は本研究が捉えたプログラムの動性の様相を示している。本図は世界で初めてのものである。プログラムの動性に関する図１２の光景を観ずして、プログラム言語、ＯＳ，応用プログラムの良い設計の在り方が議論できる筈はないと考える。本発明者にはプログラムの動性分析が行われた１５年間は、プログラムに関するわれわれの叙述の仕方が真摯に論考できた時期であったといえる。

【００１７】

われわれ人間が考えることは自分で意識することはできる。しかし、それを文字や言葉に変える直前の生命状態は場合には、ひらめき的に自覚することがあっても、正確にはその状態は不明である。本発明に至る研究においてはこの生命的状態が叙述を決める仕組になっていると考えた。この上で、その形而上学的模型が論考され仮説された。この形而上学的模型は「調和構造」と名付けられる。そして、「自覚関数」として叙述された。図１２を導くアルゴリズムの起源は自覚関数の解として求められたものである。自覚関数はプログラムの知見を用いて、プログラムとして解釈し直された。そして、そのプログラムはシナリオ関数と呼ばれることになった。

【００１８】

シナリオ関数は他に例のないプログラムの叙述法になっている。シナリオ関数は、「名詞を主語化し、且つそれを脈略する仕組」として成立している。付言すれば、プログラムに限らず、われわれがするどの様な振る舞いも、自覚関数がその模型であると結論付けられる。

【００１９】

調和構造の最小単位は存在論の実在に対応付けられている。実在は実体と属性とで定義される。シナリオ関数では実在の識別子は最小クラスの名詞、実在の実体は名詞の内容、名詞の内容が主語、属性は主語化される名詞の集合として解釈された。自覚関数の段階で、その定義叙述には複写性の成立することが感得された。因みにプログラムの複写性とは、「プログラム言語とプログラム仕様に属す名詞の名称、名称の数、主語化する記憶が同じであれば、そのプログラムのソースコードは誰が作成しても複写機で複写された様に一致する」ことをいう。

【００２０】

シナリオ関数（の原型）は本研究を始めて２３年後の１９９６年に求められた。以上のことは本研究の論文などで公開されている。シナリオ関数の構想の基本部分は１９９９年から日米欧で特許されている。シナリオ関数は日本ではこれ迄３６個の開発プロジェクトで使用されている。その後もシナリオ関数の特質が解析され、２００９年には、従来のプログラムでは解法することができない以下の問題がシナリオ関数では方法論として成立することが検証された。

（１）複写性の成立
（２）意味の仕組を捉える自動化アルゴリズム（意味の仕組）の発見
（３）プログラムの自動生成の仕組
（４）プログラム保守作業の自動対応の仕組
（５）品質概念の消滅

【００２１】

プログラムをプログラム言語の観点でしか捉えることができなくなったこの四半世紀のプログラム作成の在り方はプログラム問題の解法を放棄して来た証しでもあった。プログラムが叙述物であるということに於いて、その作成には叙述法（方法論）が不可欠である。プログラムの叙述法は複写性が成立させられるプログラムの基で成立する。

【００２２】

（Ｂ．ウイルス問題の捉え方）

ここで、ウイルス問題に対して考察を深める。ウイルス問題の解の求め方には以下の考え方がある。

（１）　ウイルスがプログラムに侵入する前に阻止する。
（２）　上記１に公権力を加えてウイルス問題を犯罪化して排除する。
（３）　ウイルス問題を侵入されるプログラム（侵入媒体）に解法させる。

【００２３】

ウイルスの侵入方法は、原理上、稼動中のプログラムの命令の数だけあることになる。世界中でどれだけの命令が稼動しているかは別として、把握しきれないほどの大変な数の命令が動いている状態にあるのが現状である。

【００２４】

ウイルスの侵入を阻止するという、上記（１）（２）の考えではまともな考えとはいえない。これらは、ソフト危機のプログラム問題が解法に至ることができなかった様に、ウイルス問題の部分的解決に止まり、混乱に拍車を掛ける以外の何物でもない。本質的にウイルス問題はプログラム作成法の欠陥（この点については後述する）による技術上の単純問題であるにも関わらず、その元責任を問うことをせずに、単にウイルスを作るな、作れば公権力で罰する、とする考えでは本質的にこの問題を解決することにはならないことは自明である。換言すれば、ウイルスの侵入の阻止ではウイルス問題は解法すること（根本的な解決策を与えること）が不可能であるといえる。既述の様に、ウイルスは１組のウイルスで電算機システムを瓦解することができてしまう。完全な侵入阻止が不可能な技術も、同じ様に排除しなければ意味がない。したがって、上記（１）、（２）の解決方法ではウイルス問題は解法され得ない。

【００２５】

　となると、ウイルス問題の解は完璧にウイルスを無力化する最終解でなければ意味がないことになる。したがって、残された解の求め方は上記（３）しかなく、これにより、最終解を求めなければならない、ということが論理的帰結である。

【００２６】

（Ｃ．本願独自の方法論の構築）

　本発明では、ウイルスの無力化が狙点である。本発明では解法という言葉がしばしば登場するが、これは解決という言葉と語彙を区別する為に採用した語句である。即ち、本発明では「解法」とは問題を原理的に抹消すること、「解決」とは本来的には封印できない問題を臨床的に封印することを意味するものとして用いる。

【００２７】

翻って、ウイルス問題とは電算機システムに生じるウイルスによる悪影響のこと一般を意味する語句である。ウイルスとは実行型のプログラムである。ウイルスは何らかの方法で稼動中のプログラムに宿る為に侵入する。ウイルスが宿るとプログラムは汚染される。プログラム汚染はウイルス意図とは無関係なウイルス現象である。ウイルス汚染は記憶領域を共用し、同期的に稼働する他のプログラムにも波及することになる。他方、ウイルス症状とはウイルスが意図的に発症させるウイルス現象である。

【００２８】

侵入媒体がウイルスの実体を捉えることはできない。ウイルス症状は、譬え、それを知ったとしても、後の祭りとなってしまう、というのが本質である。換言すれば、ウイルス症状の様子を知った所で解法には何の役にも立たず、また、ウイルス症状は汚染後に現れる。即ち、汚染と症状は発症のタイミングが異なる、ということになる。

【００２９】

ウイルス問題とは電算機システムに生じるウイルスによる汚染の悪影響のことである。ウイルス問題の解法はウイルスを除くことではなく、侵入を阻止することでもない。汚染を自力で除くことをいう。ここでいう電算機システムとはプログラムのことであり、ウイルス意図とはウイルス創作者の決意をいう。

【００３０】

ウイルスはウイルス意図に基づき作成される。ウイルス意図については後述される。ウイルスとは実行型のプログラムのことである。ウイルスは何らかの方法で稼動中のプログラムに宿る為に侵入する。ウイルスが宿るとプログラムは汚染される。本発明ではウイルスに汚染されるプログラムを侵入媒体という。プログラム汚染はウイルス意図とは無関係なウイルス現象である。

【００３１】

プログラム汚染は記憶領域を共用し、同期的に稼働する他のプログラムにも波及する。他方、ウイルス症状とはウイルスが意図的に発症させるウイルス現象である。留意すべきはウイルスの実体を捉えても、ウイルスの侵入の仕方や症状を詳しく知ったとしても、そこからウイルス問題を解法する策が得られる訳ではない、という点である。ウイルス症状は汚染の後に現れる。即ち、汚染と症状は発症の時宜（タイミング）が異なる。

【００３２】

プログラム汚染が生じる理由を以下に示す。

（１）ウイルス侵入をシグナルとして発症する汚染

（２）ウイルス症状を成立させる過程で発症する汚染

（３）ウイルス症状として発症する汚染

【００３３】

プログラム汚染は、結果的に、プログラムの命令文の上書に帰着する。命令文がどの様に上書きされるかは後述する。ウイルス症状とは侵入媒体を基にウイルスが意図的に発症させる現象である。即ち、ウイルス症状とはウイルス意図の現象化である。ウイルス症状例については後述する。侵入媒体はウイルスにはなくてはならないウイルスの寄生木（やどりぎ）である。稼動中のプログラムは全てウイルスの寄生木候補である。ウイルスはそれ自体では成立せず侵入媒体の基でのみ成立する構造をとる。ウイルスによる侵入媒体の汚染はウイルス問題を解法する観点では、ウイルスの不可避的な弱点といえる。本発明ではウイルスによる汚染をその発症させない時宜で本プログラムに生じる叙述矛盾として捉える。そして、除染しそのことによりウイルス問題を解法するものである。

【００３４】

本プログラムに生じる叙述矛盾の起因はウイルスによる汚染であるが、叙述矛盾は本プログラムの構造上に生じる様相として捉えられる。故に、本プログラムが捉える叙述矛盾は起因がウイルス汚染ではあってもウイルス情報に基づくものではない。要約すれば、本プログラムはウイルスを本プログラムに自由に侵入させることでウイルス問題を解法する、ということになる。すなわち、ウイルス意図がなんであれ、ウイルスがどの様に工夫されて作られていても、ウイルスが不可避的に生じさせてしまう侵入媒体となる本プログラム

の汚染を本プログラムはウイルス症状を発症さる前段の時宜で、無力化させることにより、症状問題を解法するものである。これは、ウイルスは存在してもその意図が達成できないということになる。「ウイルスを無力化する」との意味はこの点に存在する。結果、本発明ではウイルスに侵入されても、プログラム本業の為の意図は阻害されない、ということになる。

【００３５】

図１Ｄは、本発明の研究過程に於ける確立された主たる構想を示す図である。同図に示されるように、ウイルスに侵入されたプログラムがウイルスをどの様に認識するかについて、本発明ではプログラム汚染という概念で捉えることとし、そして、汚染されるプログラムがそれを自力で捉えなければウイルス問題は消滅させることができないと考えた。これはウイルス問題を解法する為の転機となった。２０００年のことである。１Ｄ－１はそのことを示すものである。解法の構想を実現させる為にはシナリオ関数の特徴を更に深化して捉える必要があった。シナリオ関数の解を求めるアルゴリズムの完全性を求める切口で、２００９年に解法の仕組が捉えられた。１Ｄ－２はそのことを示すものである。次に、２０１１年に解法の原理が捉えられた。１Ｄ－３はそのことを示すものである。２０１３年に開放を実現させるプログラムの構造が求められた。１Ｄ－４はそのことを示すものである。

【００３６】

（ウイルス意図およびウイルス症状）

ウイルス意図と症状（例）を以下に示す。ウイルス意図は今後進化すると思われるので、水準を分けて示す。

水準１：既存システムの活用化（入出力データの書き換え、ＤＢ情報の盗用）

水準２：システムの反乱化（入出力情報の揺動）

水準３：システムの沈黙化（入出力情報の廃棄）

水準４：システム破壊（条件文の破壊）

水準５：システム占拠（ウイルスの起動時宜の制御）

【００３７】

（命令文の汚染の解説）

ウイルスの汚染は結果的にプログラムの命令文の汚染に帰着する。命令文の汚染は命令文の上書き現象に帰着する。命令文の構造が命令文の汚染の内容になる。即ち、以下のようになる。

（１）命令文が使用するデータ領域

（２）命令文の命令コード

（３）命令文が直接使用する定数（定値）

（４）命令文が直接使用する文字列（定置）

【００３８】

データ領域の汚染は同じデータ領域を用いて同期的に稼働している別のプログラム達にも波及する。侵入媒体のどの命令文が汚染されるか、いくつ汚染されるか、命令文のどの部分が汚染されるかは侵入された時宜でそれを観察しなければわからない。本発明ではプログラムに属す命令文が実行されるとき、その命令文が使用するデータ領域の汚染をその命令文を統治する仕組によって観察される仕組になっている。この仕組はベクトルと呼ばれる。

【００３９】

本プログラムでは命令コードの汚染はＯＳとベクトルＥ４２で捉えられる。ＯＳは命令コードの汚染を命令破壊として捉える。Ｅ４２は命令破壊を命令コードの汚染として本プログラムに生じる論理矛盾として捉える。論理矛盾は主語成立数のスタックを用いて捉える。その詳細は後述する。定置、定値、領域名の汚染は命令コードの破壊と見做すことができる。ＯＳとプログラム言語とを改善すれば、定置問題は自動回復を可能にする命令破壊として、Ｅ４２よりも優れた時宜と簡便さで解法することもできる。Ｅ４２の仕組は電算機の仕組やＯＳ、プログラム言語の改善で代行することは不可能である。本発明ではＯＳ

が捉える命令破壊は今後、電算機、ＯＳ，プログラム言語の改善で、自動回復できることを示唆している。

【００４０】

（データ領域）

本発明でいうデータ領域の内容とは以下の通りである。

（１）ベクトルごとに設けられる第４領域

（２）ベクトルごとに設けられる３種のフラグ

（３）シナリオ関数に唯ひとつ設けられる主語成立数カウンタ

（４）シナリオ関数に唯ひとつ設けられる主語成立数スタック

上記（１）の汚染は本プログラムの仕組(後述)で捉え自動解回復させる。上記（２）の汚染はベクトルを基に成立するウイルス観察アルゴリズム（後述）で捉え自動解回復させる。上記（３）（４）の汚染はＥ４２（後述）の成立の背因として捉える。意図的な命令文の汚染も、結果的に、命令汚染となる。注釈文(定置文)の汚染は実行時にその注釈文(定置文)と既知である定義上の注釈文(定置文)を比較定数として両者をＸＯＲすれば捉えられる。この仕組は後述されるベクトルで成立させることができる。本発明ではこの解法をワクチン法と呼ぶこととする。本発明では注釈文(定置文)の汚染は定置問題として位置付けられる。定置問題はＷｅｂプログラムの世界では、動画や絵を汚染する原因に繋がる。

【００４１】

（本発明の要点）

図１Ａ，１Ｃに、本発明の要約を示す。図１Ａにおいて、１Ａ－１は電算機の記憶領域、１Ａ－１０は従来のプログラム、１Ａ－１１は本発明のプログラムを示し、従来のプログラムはウイルスに汚染されるが、本発明のプログラムはウイルスに汚染されても自力で除染するので汚染されることはないことを示している。図１Ｃにおいて、１Ｃ－１は電算機の記憶領域、１Ｃ－１０は従来のプログラム、１Ｃ－１１は本発明のプログラム，１Ｃ－２はウイルスを示す。従来のプログラムはウイルスに汚染されるが、本発明のプログラムはウイルスに汚染されても自力で除染する。それはウイルスに上書きされないことである。本図はそのことを示す図である。

【００４２】

同両図に示されるように、本発明は概略次のような考え方に則っている。

（１）稼動中の本プログラムはウイルスの侵入を許容する。

（２）本発明のプログラムは本業の片手間に侵入したウイルスを自力で無力化する。

（３）結果、本プログラムにより、ウイルスは存在してもウイルス問題は存在しないことになる。

（４）どのプログラムも現行のＯＳの基では命令破壊に遭遇すれば停止するが、本プログラムも同じ。本発明においては、緊急停止後、継続再起を自動的に行わせるプログラムを準備することを提案している。そのプログラムをＳＬＰ(Ｓｏｆｔ　Ｌａｎｄｉｎｇ　Ｐｒｏｒａｍ)と呼ぶ。

（５）結果的に、本プログラムはウイルスに侵入されても命令破壊以外のウイルス症状を、発症させない仕組になっている。

本発明はプログラムとして、正統な主語の脈略を作る仕組が求められれば、ウイルス問題を含むプログラム問題は結果的に解法されるということを発見したことにもなる。

【００４３】

（ルネッサンス的変革の必要性）

上述のように、プログラム問題、ウイルス問題の解法はシナリオ関数で成立する。しかし、シナリオ関数の普及が阻まれる背因をここで触れておきたい。プログラム問題をプログラム言語だけをより所にシステムの利便性に置き換え、プログラムを作ることの危うさは１９７０年代の初めからソフト危機として知られていたものである。しかし、この分野の半世紀に及ぶ、今日までの歩みを回想すると、この問題は１９９０年までには、解法されるべき課題であった。

【００４４】

１９９０年以降、戻るべき途を見失ったと思われるこの分野の暴走の手法は着実にこの分野を未解決問題に埋もれさせ、その状態を内在させて来たものである。この分野が現在足元を掬われているウイルス問題は、この分野が正常化される最後の機会だと考える。本発明では、ウイルス問題をプログラム構造の問題として、これに対する単純明確な解法を示した。この解法の仕方はこの分野がこれ迄何を見落としてきたかを明らかにしいると考える。

【００４５】

プログラムをプログラム言語の観点でしか捉えなくなった１９９０年代以降の四半世紀のプログラム作成の仕方は、皮肉にも電算機商品の著しい普及を遂げさせたといえる。しかし、その裏面は矛盾に満ちているとしかいいようがない。その最大の問題は専門家が育たなかったことである。例えば、ウイルス問題がプログラムに侵入される前に防止するという暴論が横行するのはその証であるといえる。

【００４６】

翻って考えるに、プログラムは叙述物であることに於いて、その作成には叙述法（方法論）が不可欠だという基本的認識が重要である。プログラムの叙述法は複写性が成立させられるプログラムの基でのみ成立するということにも気付く必要がある。しかし、人材が育たず且つ論理結合型プログラムを基調とする限り、この分野がその様な認識に気付くパラダイムにシフトすることは容易なことではないと考えられる。つまり、論理結合型叙述法の意味を捉えられない宿命的欠陥にも気付く必要がある。この問題は、仮に他の分野では許容できることであったとしてもプログラムではその本数が増えることによって看過できない深刻な事態が生じてしまう、ということにもつながってしまう。

【００４７】

他方、論理結合型のプログラムのこの欠陥故に、宗教、思想、体制の違いを超えて同じ作り方が普及することにも気付く必要がある。故に、たかがプログラムとはいえ、その欠陥が、ウイルス問題で明らかな様に、人類規模の危機を作り出すことになった、という理解も可能なのである。その様なプログラムの人類規模の普及が結果的に人類規模の危機を発症させることになった。この問題を回避する為には意識革命の自己ワクチンが必要である。その為に、プログラムの作り方にルネサンス的改革を示すことが必要である。

【００４８】

（本願に係る課題解決手段の概略）

本願では、実行中のプログラムに侵入するウイルスが発症させるプログラムのデータ領域の汚染はプログラムを本発明のプログラム（以下、「本プログラム」ともいう。）に置き換えると本プログラムでは、その汚染を実行時に発症する叙述矛盾として、本プログラムに関わる如何なる箇所の汚染も例外なく、自律的に捉え且つ其れを自律的に排除して、本プログラムの本業の働きを継続する。そのような本プログラムの定義構造は本願特有の課題解決手段である。

【００４９】

また、本プログラムの構成要素である複数のベクトルの全ての各第１規約で例外なく観察されるベクトルの正統性判定、並びに同第３規約で例外なく観察される主語の正統性判定に於いて、ウイルスによる汚染が関われば、それら判定には叙述矛盾が生じる。その叙述矛盾を捉える仕組は本願特有の課題解決手段である。

【００５０】

さらに、本プログラムの汚染箇所を自律的に初期値化して汚染を自動的に取り除く仕組（「ウイルス観察アルゴリズム」ともいう。）は本願特有の課題解決手段である。

【００５１】

より根源的には、プログラムは例外なく主語の脈略であるとする捉え方に関する発見を具体化する本プログラムの基盤となる仕組、すなわちシナリオ関数も本願特有の課題解決手段を構成するものである。

【００５２】

また、従来のプログラムの延長では想像ができない主語の脈略（図１２）を決定する普遍

的な仕組（座標関数、同期関数も含む）が本願特有の課題解決手段を構成するものである。

【００５３】

さらに、本プログラムは正統な主語の脈略を生成する仕組であることに於いて、結果的に、ウイルス情報を一切具備することなくウイルス問題を解法する仕組に帰着するものである。

【００５４】

また、本プログラムは電算機システムの特別な部分のプログラムではなく全ての分野のどのプログラムにも有効となる。

【００５５】

（本願に係る具体的な課題解決手段）

具体的に、上記課題を解決するために、本発明に係るウイルス自律的解法プログラム定義構造は、実行状態にあるプログラムの為の所定の記憶領域が何らかの理由で前記プログラムについての意図に反する反意図情報に汚染されると前記汚染を自力で捉える為の汚染把握機構と、前記汚染把握機構にて捉えられた汚染を自力で除染する為の除染機構と、前記記憶領域を正常状態に自動回復させる為の正常状態回復機構とを具備する。

【００５６】

ここで「汚染」とは、プログラムの記憶領域に対して本来の意図されるところに反して情報を変えるすべての行為をいい、改ざん、書き換え、もしくは破壊される態様を含む概念である。

【００５７】

かかる構造を有することにより、本発明においては、記憶領域に本来望んでいないコンピュータウイルスを含む外部からのデータや命令文が侵入したとしても、それをウイルスの問題としてではなくかかる記憶領域の汚染としてとらえ、しかもかかる侵入を把握した途端に汚染された該記憶領域が自律的に除染される。

【００５８】

この場合において、前記汚染把握機構は、前記第１の情報領域に関係する論理矛盾を発見する構造とすることができる。この構造によれば、コンピュータウイルスの侵入という、パターン認識手法では限界がある事態を客観的にもれなく把握できるアルゴリズムに置き換えることが可能となる。この場合において、汚染把握機構は第１乃至第７の規約を有するベクトルの構造を持つようにすれば好適である。

【００５９】

さらに上記の場合において、前記除染構造は、除染機構は前記ベクトルを初期値化する為の初期化機構を持つとすることができる。さらに具体的には、除染機構は前記汚染が把握された前記反意図情報が該反意図情報に対して意図された症状を発症させる前のタイミングで前記ベクトルを初期値化するとすることができる。この構造によれば、コンピュータウイルスの侵入という、ウイルスに汚染された可能性のあるデータ領域が自律的・自動的に初期化されてしまうので、ウイルスがいわゆる「悪さ」を働く機会が発生しないようにすることが可能となる。

【００６０】

また上記の場合において、前記除染構造は、さらに具体的には、除染機構は前記汚染が把握された前記反意図情報が該反意図情報に対して意図された症状を発症させる前のタイミングで前記ベクトルを初期値化するとすることができる。これによれば、ウイルスがいわゆる「悪さ」を働く機会が発生しないうちに除染してしまうので、いわゆるコンピュータウイルスの侵入問題自体を消滅せしめることが可能となる。

【００６１】

さらに、前記ベクトルは前記汚染把握機構及び／もしくは前記除染機構に対して最適の時宜を付与する構造を持つとすることができる。

【００６２】

またさらに、前記正常状態回復機構は前記プログラムに成立させる再起機構を持つとす

ることができる。
【００６３】
また、前記汚染把握機構に係る前記ベクトルは少なくとも、前記第２規約を通過したことを示す第２フラグ、前記ベクトル自体の再起動を要請するための第６フラグ、前記ベクトル自体の再起動の一時停止を宣言するための第７フラグ、前記第３規約で判定され前記第４規約で決定される領域である第４領域を有し、前記汚染把握機構は前記第２、第６、第７のフラグと前記第４領域との相対関係を判定する３種フラグ・第４領域相対関係判定機構を有するとすることができる。
【００６４】
さらに、前記ベクトルの前記第３規約で前記第４領域の正統性を判定する為に該第４領域の脈略の正統性を判定する為の第４領域脈略正統性判定機構をさらに備えるとすることができる。これによれば、仕組１２と仕組５は第４領域の正当性を判定する仕組であるが、判定の時宜が異なる為に判定の仕方を同じ仕組にすることができない。
【００６５】
またさらに、前記ベクトルに係る前記第５規約において、前記第４領域の成立数スタックを用いて該第４領域の脈略の成否の未来予測をする第４領域脈略成否予測機構をさらに備えるとすることができる。
【００６６】
また、前記プログラムに係るＯＳでは捉えられない命令汚染を捉える命令汚染把握機構をさらに備えるとすることができる。
【００６７】
また、上記課題を解決するために、本発明に係るプログラム定義構造は、ＯＳ（オペレーションシステム）上で起動される動作プログラムもしくは該動作プログラムに係るデータ領域に侵入するウイルスの起こし得る問題をプログラム構造として解法するためのウイルス自律的解法プログラム定義構造であって、主語となるデータ領域に対して内容を決定するための最小叙述構造体であるベクトル構造が任意順序で集積されたパレット４の臨界状態が達成されるまで循環する構造を有する座標関数４と、主語となるデータ領域に対して内容を決定するための最小叙述構造体であるベクトル構造が任意順序で集積されたパレット２の臨界状態が達成されるまで循環する構造を有する座標関数２と、主語となるデータ領域に対して内容を決定するための最小叙述構造体であるベクトル構造が任意順序で集積されたパレット３の臨界状態が達成されるまで循環する構造を有する座標関数３と、前記パレット４が臨界状態になれば前記パレット２に、前記パレット２が臨界状態になれば前記パレット３に、前記パレット３が臨界状態になれば前記主語を成り立たせるための変数主語の第４領域の所在に応じて最上ランクに係る座標関数３、同一ランク係る座標関数４、１層下層ランク係る座標関数４のいずれかに制御を移す同期関数とを有する。
【００６８】
かかる構造を有することにより、本発明においては、第１の情報領域に本来望んでいないコンピュータウイルスを含む外部からのデータや命令文が侵入したとしても、ウイルス検出アルゴリズムがそれをウイルスの問題としてではなく第１の情報領域の汚染としてとらえてしまう構造をシナリオ関数上で動かすことで、ウイルス侵入が自律的に検出される。
【００６９】
この場合において、前記前記ベクトルは、前記データ領域が何らかの理由で汚染されると前記汚染を自力で捉える為の汚染把握機構と、前記汚染把握機構にて捉えられた汚染を自力で除染する為の除染機構と、前記記憶領域を正常状態に自動回復させる為の正常状態回復機構ととすることを採用してもよい。この構造がシナリオ関数の再帰構造と組み合わさって、コンピュータウイルスの侵入という、パターン認識手法では限界がある事態を客観的にもれなく把握できるアルゴリズムに置き換えることが可能となる。
【００７０】
さらに、前記ベクトルは前記汚染把握機構及び／もしくは前記除染機構に対して最適の

時宜を付与する構造を持つとすることができる。
【００７１】
またさらに、前記正常状態回復機構は前記プログラムに成立させる再起機構を持つとすることができる。
【００７２】
また、前記汚染把握機構に係る前記ベクトルは少なくとも、前記第２規約を通過したことを示す第２フラグ、前記ベクトル自体の再起動を要請するための第６フラグ、前記ベクトル自体の再起動の一時停止を宣言するための第７フラグ、前記第３規約で判定され前記第４規約で決定される領域である第４領域を有し、前記汚染把握機構は前記第２、第６、第７のフラグと前記第４領域との相対関係を判定する３種フラグ・第４領域相対関係判定機構を有するとすることができる。
【００７３】
さらに、前記ベクトルの前記第３規約で前記第４領域の正統性を判定する為に該第４領域の脈略の正統性を判定する為の第４領域脈略正統性判定機構をさらに備えるとすることができる。これによれば、仕組１２と仕組５は第４領域の正当性を判定する仕組であるが、判定の時宜が異なる為に判定の仕方を同じ仕組にすることができない。
【００７４】
またさらに、前記ベクトルに係る前記第５規約において、前記第４領域の成立数スタックを用いて該第４領域の脈略の成否の未来予測をする第４領域脈略成否予測機構をさらに備えるとすることができる。
【００７５】
また、前記プログラムに係るＯＳでは捉えられない命令汚染を捉える命令汚染把握機構をさらに備えるとすることができる。
【００７６】
こうした構成を有する本願によれば、コンピュータウイルスの侵入という、パターン認識手法では限界がある事態を客観的にもれなく把握できるアルゴリズムに置き換えることが可能となり、また、ウイルスに汚染された可能性のあるデータ領域が自律的・自動的に初期化されてしまうので、ウイルスがいわゆる「悪さ」を働く機会が発生しないようにすることが可能となる。
【００７７】
また上記の場合において、前記ウイルス検出アルゴリズムが前記ウイルスが侵入した可能性のあるデータ領域を初期化するに当たっては、前記侵入が推定された前記ウイルスが意図された症状を発症させる前のタイミングで前記データ領域を初期化するとすることができる。これによれば、ウイルスがいわゆる「悪さ」を働く機会が発生しないうちに除染してしまうので、いわゆるコンピュータウイルスの侵入問題自体を消滅せしめることが可能となる。
【００７８】
上述した本願発明に係る技術思想は、上記のようなウイルス自律的解法プログラム定義構造としてのみでなく、実質的に同様の発明特定事項を備える、ウイルス自律的解法プログラム、ウイルス自律的解法装置、ウイルス自律的解法方法、として、もしくはかかるプログラムが搭載された記憶媒体として、実現することができる。
【発明の効果】
【００７９】
本発明によれば、ウイルス問題に対して本質的な解法が与えられる。すなわち、ウイルスが稼動中の本発明のプログラム（以下、「本プログラム」ともいう。）に如何なる時宜、如何なる手段、何度でも繰り返し侵入したとしても、本プログラムはそのウイルスを自力で本プログラムが使用する記憶領域の汚染として捉えて、除染し、本プログラムの正常な稼働を継続できる様に迅速に回復させる。
【００８０】
本プログラムは汚染（ウイルス）を本プログラムの意図に反する誤情報として捉える。本

プログラムは誤情報が存在すれば、本プログラムに叙述矛盾が生じる仕組になっている。本プログラムはこの仕組みを用いて汚染を捉える。付言すれば、本プログラムは侵入するウイルスをウイルスとしてではなく、本プログラに生じる本発明による叙述矛盾の仕組で捉える。そして、本プログラムはその汚染を本発明による仕組で除染する。この除染は侵入するウイルスの意図を解体することと同義になる。

【００８１】

汚染を捉える時宜、そして、それを除染する時宜は本発明による時宜で行われる。結果、侵入したウイルスがその意図する所の症状を発症させる前に無力化されることになる。即ち、本プログラムの基では、ウイルスの侵入問題、ウイルス症状問題はウイルス問題として解法するわけではない。本プログラムはウイルスに侵入されてもウイルス問題を本プログラムには発症させない様にする仕組である。故に、本プログラムのこの仕組はウイルス問題の解法ということになるものである。

【００８２】

よって、各産業を困らせていたウイルス問題からこれら産業を一挙に解放することが可能となる。

【図面の簡単な説明】

【００８３】

【図１Ａ】本発明の一実施形態に係るプログラムの効果を示す概念図である。

【図１Ｂ】従来のプログラムから本発明のプログラムを導き出す手順を示す概念図である。

【図１Ｃ】本発明の一実施形態に係る本プログラムの効果を示す概念図である。

【図１Ｄ】本発明の研究過程に於ける確立された主たる構想を示す図である。

【図１Ｅ】プログラムの動性解析による従来型プログラムの名詞の成立軌跡（左側）と本願に係るシナリオ関数によるプログラムの名詞の成立軌跡（右側）とを比較分析した図である。

【図２】本発明の一実施形態に係る本プログラムが使用するベクトルの種別を示す図である。

【図３】本発明の一実施形態に係る、７個の規約と４個の出口で成立するベクトルの概念図である。

【図４Ａ】本発明の一実施形態に係る本プログラムの為のベクトルの概念図である。

【図４Ｂ】本発明の一実施形態に係る本プログラムが使用するベクトルの正統性を捉える為の第４領域、第２フラグ、第６フラグ、第７フラグの相対関係を示す図である。

【図５】本発明の一実施形態に係るウイルス観察アルゴリズムの概念図である。

【図６】本発明の一実施形態に係る本プログラムが使用する主語成立数のスタック構造を示す図である。

【図７】本発明の一実施形態に係る３種の座標関数の基本概念図である。

【図８】本発明の一実施形態に係る同期関数の概念図である。

【図９】本発明の一実施形態に係る３種の本プログラムのランク構造の概念図である。

【図１０】事例として掲げる従来プログラム（論理結合型）の部分の流れ図である。

【図１１Ａ】図１０に掲げられる事例に対する本発明の一実施形態に係るＬｙｅｅ空間として規定されるフレームの一部左側を示す図である。

【図１１Ｂ】図１０に掲げられる事例に対する本発明の一実施形態に係るＬｙｅｅ空間として規定されるフレームの一部右側を示す図である。

【図１２】図１０に掲げられる事例に対する本発明の一実施形態に係る意味の仕組を表した図である。

【図１３】本発明の一実施形態に係る本プログラムに属す主語が本プログラムのランク構造ではどの位置の本プログラムに属すかを示す為の図である。

【図１４】本発明の一実施形態に係る本プログラムが搭載され得る一形態を示した全体構造図である。

【発明を実施するための形態】

【００８４】
以下、図面を参照して本発明を実施するための形態について説明する。なお、以下では、本発明の目的を達成するための説明に必要な範囲を模式的に示し、本発明の該当部分の説明に必要な範囲を主に説明することとし、説明を省略する箇所については公知技術によるものとする。
【００８５】
（本発明の為の成立の由来）
従来のプログラムは論理結合型叙述法に因っていることが、プログラムとしての完全性が成立させられない原因であるが、論理結合型叙述法はわれわれが本能的に身に付けている習慣である。それ故に、そのプログラムの在り方にはプログラム問題で見られる様に多くの問題が内在することになるのであるが、その叙述法の馴れ親しみ易さが、その問題を是とする暗黙の前提になってしまっていると言う問題がある。
【００８６】
ここで、本発明によって到達したプログラムの普遍的構造を説明するために前提的に到達した公理である「意味が成立する構造」について説明する。しかる後に、意味の構造の公理に基づき、意味が導出される仕組みとして本願が到達した考え方を説明し、その考え方に立ってウイルスをどう無力化するのかという本願の直接的課題の解決手段について説明する。
【００８７】
（意味の概念）
意味は全体（内包的光景）的である。自然上の存在たるわれわれは部分なので、「意味」、「全体」等の言葉を持っているにも拘らず、その実体の内包的光景をわれわれ自身の記憶として叙述することはできない。われわれができることは、部分を機能的に脈略化（ロジック化）して、意味や全体を連想するに留まるものである。
【００８８】
（意味は何処に所在するのか）
意味はわれわれの脳裏の中に存在する、と仮説する。
【００８９】
（意味の仕組）
意味の仕組はシナリオ関数の解（Ｓ）である。意味の仕組とはわれわれの脳裏の中に存在する意味の模型である。シナリオ関数はその動性（データ結合）で主語の脈略を成立させる。それが意味の仕組の模型である（図１２参照）。主語の脈略はロジックに較べて、全体の概念により近似する。故に、この特性から主語の脈略を意味の仕組と呼ぶのである。
【００９０】
意味の仕組はシナリオ関数の５種のベクトルＬ４、Ｌ２、Ｌ３、Ｒ２、Ｗ４の第４領域（主語）の脈略である。意味の仕組は論理結合型思考法のわれわれが認識できる様に捉え直されたものである。意味の仕組はシナリオ関数を論理結合型プログラムに変換して、ＬＹＥＥ空間（図１１Ａ、図１１Ｂ）を求め、ＬＹＥＥをグラフ化ツールに入力すれば、求められる。技術的には、意味の仕組は超言語化された単元文から求められる。超言語化された単元文はＬＹＥＥ空間で見ることができる。グラフ化ツールは既に市販されている。シナリオ関数を論理結合型プログラムに自動変換するアルゴリズムは本発明の研究のなかで、既に求められている。論理結合型プログラムからＬＹＥＥ空間を自動的に求めるアルゴリズムは本発明の研究のなかで、既に求められている。論理結合型プログラムの部分例（図１０）から求められる意味の仕組（図１２）で意味の仕組の在り様を解説する。部分例（図１０）のＬＥＥ空間は図１１Ａ、図１１Ｂを参照されたい。
【００９１】
（意味の仕組の構造）
従来のプログラムの動性、即ち、領域を節とする命令の実行軌跡は所謂スパゲティ型となるのに対し、シナリオ関数の動性は櫛型となる。主語の脈略で捉えると両者とも同じ意味の仕組となる。これが意味の仕組みの特徴である。意味の仕組は主語、定値がウイルスで

汚染されれば、その波及がどこまで及び、そして、汚染の無力化範囲を限定する為の論考に用いられた。結果、定値、主語、変数主語の汚染を無力化すれば、汚染問題は解法できるとの結論が得られた。このことは（後述する）プログラム模型で見ることができる。

【００９２】

（経路）

図１２は図１０の意味の仕組を表す図である。図１２に於ける太い実線は図１０の流れ図の線であり、細線は流れ図には表れないもので流れ図線を補完する関係線で意味の仕組を表すことになる。図１０に現れる線（以下、「経路」ともいう。）の本数は２０個、図１２の図１０と同質の経路数は２２個となる。図１２の方が経路数が２個多いのは図１２の方が、厳密性が高いためである。この経路は調和座標で成立するので、調和脈略と記される。他方、図１２では更に１６個の経路が出現している。この経路は超言語座標で捉えられるので超言語脈略と記される。超言語脈略は図１２に現れ、図１０には出現しない。これが意味の仕組（主語の脈略）の特徴である。そして、図１２の方が図１０よりも明らかに全体化されていることの証である。

【００９３】

調和脈略は機能的な動性の経路である。しかし、超言語脈略は動性の経路ではない。主語がどの様な経緯で成立しているか主語の由来を表す経路である。換言すれば、これは主語の意味を捉える経路である。もし、超言語脈略が意図を反映するものでなければ、例えば、超言語脈略に於ける主語の順序が意図を反映していなければ、調和脈略に問題が生じる。換言すれば、これ迄調和脈略、即ち、流れ図に問題があるとすれば、われわれは調和脈略上、即ち、流れ図上でその原因を捉えようとするのであるが、それは、プログラム・テストで経験する様に簡単なことではない。意味の仕組を用いれば、調和脈略に生じる問題の原因は超言語脈略上の主語の順序の誤りとして観察することができる。主語の順序の誤りを調和脈略上で探すことは困難である。

【００９４】

（ＬＹＥＥ空間の項目）

図１０は論理結合型プログラムを示す流れ図である。図１１Ａ、図１１ＢはこのプログラムのＬＹＥＥ空間である。意味の仕組を求める為に必要なＬＹＥＥの項目の説明をする。

１）事例に図１０のプログラムを用いる。

２）文種とは単元化されたプログラム構文の種類のことである。

単元文の種類は１０種である。

３）１２種のベクトル種別は領域文を除く文種に呼応する。

４）主語とは単元文の解である。

５）連動文とは主語を持たない単元文のことである。これらは主語を持つ単元文に属すことに於いて文脈を構成する。文脈とは意味の仕組の最少単位のことである。

【００９５】

（調和座標）

６）行番号は単元文のそのプログラムに於ける位置である。

７）調和座標は単元文が実行される様子を捉える為の６種の座標である。

a.ＴＣＸは行番号に連動する．単元文のプログラム上の位置である。

b.ＴＣＹは単元文が無条件で次に進む単元文のＴＣＸである。

c.ＴＣＺ１は単元が文条件文で真のとき、次に進む単元文のＴＣＸである。

d.ＴＣＺ２は単元文が条件文で偽のとき、次に進む単元文のＴＣＸである。

e.ＴＣＺ３は単元文が条件文でその及ぶ範囲の終わりの単元文のＴＣＸである。

f.ＴＣＺ４はＴＣＺ３の単元文が次に進む単元文のＴＣＸである。

８）超言語化された単元文とは単元文の構成項目（主語、変数主語、定値,定置、指標）に調和座標が付与された単元文のことである。

【００９６】

（調和脈略と超言語脈略）

単元文は調和座標（図１１Ａ、図１１Ｂ）に従って順序づけられる。これはプログラム例

の流れ図（図１０）と同等の様相である。意味の仕組（図１２）ではこれを調和脈略と記す。流れ図の単元文は意味の仕組では超言語化された単元文に置き換えられる。図１１Ａ、図１１ＢのＬＹＥＥ空間で分かる様に、超言語化された単元文ではその構成項目には全て調和座標が付与されている。これを用いて構成項目の脈略が求められる。これを超言語脈略と記す。ＬＹＥＥ空間（図１１Ａ、図１１Ｂ）参照。

【００９７】

例えば、行番号８６の単元文Ｇ＝１０では主語となる名詞はＧである。そして、その１０は定値と記される。この単元文が超言語化されれば、Ｇ[８６,１１]＝１０[８６]と記される。８６はこの構文が置かれるＴＣＸが８６であることを示す。このＴＣＸ８６は行番号８６と一致しているが、単元文が複数機能を持つ構文があり、複数の単元文に置き換えられたものである場合には、その複数機能を持つ構文は行番号を持ち、単元化された単元文達はその複数機能を持つ構文の行番号に枝番号が付され、それぞれの単元文のＴＣＸとなる。Ｇ[８６，１１]の１１はＧの領域定義文がＴＣＸ：１１に置かれていることを示す。領域定義文はこのＬＹＥＥ空間では省略されているので、見ることができない。１０[８６]は定値が１０で、この形式の定値はプログラム上領域を持たないので、Ｇ[８６，１１]のＴＣＸ：８６の位置の１０であることを示す。

【００９８】

行番号９８の単元文はＣ＝Ｇ－Ａである。この単元文の超言語化はＣ[98,16]＝Ｇ[86,11]－Ａ[,20]である。Ａ[,20]にＡのＴＣＸがないのはＡのＴＣＸには９５、９６があり、成立している方が使用されるので、この段階では特定できないことが表示できない理由である。

【００９９】

この単元文にはＧ[86,11]が使用されている。このＧ[86,11]の由来はＴＣＸ：８６である。故に、この場合、意味の仕組ではＣ[98,16]＝Ｇ[86,11]－Ａ[,20]のＧ[86,11]からＧ[８６,１１]＝１０[８６]のＧ[８６,１１]のＧ[８６,１１]に向かう脈略が超言語脈略として成立する。

【０１００】

Ｃ[98,16]＝Ｇ[86,11]－Ａ[,20]のＣ[98,16]は、[100、]ＩＦ（Ｂ[99,17]＋Ｃ[98,16]）＜０で使用されている。即ち、[100、]ＩＦのＣ[98,16] の由来はＴＣＸ：９８である。故に、この場合、意味の仕組では[100、]ＩＦ（Ｂ[99,17]＋Ｃ[98,16]）＜０のＣ[98,16]からＣ[98,16]＝Ｇ[86,11]－Ａ[,20]のＣ[98,16]に向かう脈略が超言語脈略として成立する。［100、］ＩＦはＴＣＸ：１００に置かれていることを示す。ＩＦは主語ではないので、その領域は存在しないことから、[100、]ＩＦと記される。

【０１０１】

（意味の仕組のガイダンス）

以上の様に意味の仕組、即ち名詞の脈略は座標を用いて生成することができる。結果、従来のプログラムであろうと本プログラムであろうと、プログラムに属す名詞は主語として網羅的に捉えられることになる。主語としてとは既述の通り、調和座標と超言語座標とで名詞が脈略化されることである。即ち、これら座標が名詞を主語化する関係を捉えていることに留意さえすれば、名詞がこれら座標で脈略化されていても、それは主語として脈略化されているということになる。これが意味の仕組のポイントである。

【０１０２】

本発明においては、調和座標、超言語座標がこのことを実現させている。図１２で見る様に調和脈略の経路は論理結合の脈略である。これは従来の流れ図に合致する。同様に、超言語脈略はデータ結合の脈略である。これは従来の流れ図には出現しない。流れ図を見る人の脳裏に流れ図の意味として浮かぶものである。換言すれば、超言語脈略は脳裏に浮かぶ意味を捉えていることになる。

【０１０３】

意味の仕組の利用価値は様々考えられる。

（１）例えばひとつの主語（名詞）が他の主語とどの様な成立関係で成立しているかを目

視することができる。この成立関係は本意味の仕組でなければ、ひとでは完全に脳裏に浮かべることができないものである。その意味で、意味の仕組を求めれば、それは人が初めて目にするプログラムの光景である。
（２）意味の仕組の名詞（主語）のネットワーク（脈略）に於ける順位（主語順序列）はその主語の成立由来を捉えている。故に、これを用いれば、要求者の意図がプログラムで正当に捉えられているかを目視的に観察することができる。
（３）意味の仕組でプログラムのどこをテストするべきかを観察することができる。
以上で、本発明が前提的に到達した公理である「意味が成立する構造」についての説明を終了する。次に、本発明に係るプログラムとシナリオ関数との関連について説明する。

【０１０４】
本発明に係るプログラム（以下、「本プログラム」ともいう。）の成立の由来は論理結合型叙述法に対してデータ結合型叙述法として位置付けられる。この叙述法はわれわれのこれ迄の本能的な有り様とは異なるので、はじめからいきなり馴れ親しみ易く感じられるというわけにはいかない。発想をかえなければ本プログラムに馴れ親しみ易さを感じることはできない。論理結合型叙述法では主語の脈略の仕方が名詞を主語化する知見の再利用（論理）で行われるので、叙述者はその意図を十分ではないにしても認識することができる。他方、データ結合型叙述法では主語の脈略の仕方は、その叙述の解を求めなければ叙述者にも認識することができない（図１２参照）。

【０１０５】
シナリオ関数においては、叙述者にその意図を知る必要がないことが明確に示されている。この点が、本プログラムがプログラムとしての完全性を成立させる仕組になっている。従来のプログラムは本プログラムとは違い、プログラム問題やウイルス問題を解法することはできない。これに対して、本プログラムはプログラム問題やウイルス問題を解法することができる。それだけでなく、本プログラムは、その根拠を普遍的に説明し得ると言う点で、世界で初めてのプログラム構造になっている。

【０１０６】
例えば、本明細書では「ウイルスを無力化する為の仕組」は１５個のスキームに纏められている。これらはシナリオ関数を基にすることにより初めて成立するものである。しかし、従来プログラムではこの内のひとつでも成立させることはできない。ウイルス問題の常識に鑑みればこの根拠は明白であるが、本稿の「ウイルスの無力化」でも解説される。ここでは、併せて、本プログラムの基盤となるシナリオ関数についても把握する必要があるが、これについては、本稿の「シナリオ関数の定義式」で解説される。

【０１０７】
図１Ｂは従来のプログラムから本発明のプログラムを導き出す手順を示す概念図である。同図において、１Ｂ－１は従来のプログラムのソースコード、１Ｂ－１１は従来のプログラムのＬＹＥＥ空間（図１１Ａ，図１１Ｂ参照）、１Ｂ－１１１は主語ベクトルで統治される命令文をＬＹＥＥ空間から抽出する作業を示す。１Ｂ－１１１１は抽出された命令文を統治するベクトルの作成を示す。１Ｂ－１１２はＬ４型制御ベクトルで統治される命令文をＬＹＥＥ空間から抽出する作業を示す。１Ｂ－１１２１は抽出された命令文を統治するベクトルの作成を示す。１Ｂ－４１は主語ベクトルのプログラム模型が本明細書に添付されていることを示す。１Ｂ－４２はウイルス観察アルゴリズム（ＶＷＡ）のプログラム模型が本明細書に添付されていることを示す。１Ｂ－４３は制御ベクトルのプログラム模型が本明細書に添付されていることを示す。１Ｂ－４４は本発明のプログラムの基本プログラムとなる３種の座標関数、同期関数のプログラム模型が本明細書に添付されていることを示す。１Ｂ－２は作成されたベクトルを基にＶＷＡの作成を示す。１Ｂ－２１は作成されたＶＷＡを用いて主語ベクトルを完成させる作業を示す。１Ｂ－２２は作成されたＶＷＡ、制御ベクトルの模型を用いて制御ベクトルを完成させる作業を示す。１Ｂ－２３は基本プログラムの模型を用いて基本プログラムを完成させる作業を示す。１Ｂ－３は本発明のプログラムを完成させる為に１Ｂ－２１、１Ｂ－２２、１Ｂ－２３の結果を用いて本発明のプログラムを編集する作業を示す。同図では、

（１）本プログラムはウイルス問題を解法する理論である、
（２）ウイルス問題はプログラムで解法できる、
ことが示されている。
【０１０８】
本発明に係るシナリオ関数を把握するならば以下のことが認識できる。
（１）本プログラムの構成要素のベクトルを規約する７個の規約のひとつを人為的に定義すれば、本プログラムを構成する全てのコードが決まる。
（２）上記（１）は本プログラムが普遍的であることの証である。
（３）図１Ｂの関係は従来プログラムが存在すれば本プログラムも存在することの証しである。
（４）本明細書に添付される本プログラムの模型は上記（１）（２）（３）の証明である。
（５）本プログラムを決定する命令文においてはその文種を区別する為の情報を事前に定義すれば、この定義を用いて従来プログラムから自動的に取り出すことができる（図１１Ａ、図１１Ｂ（ＬＹＥＥ空間）参照）。
（６）本プログラムを決定する命令文は本プログラムを構成する１２種のベクトルの内のＬ４型制御ベクトルを含む５種の主語ベクトルで使用される。どのベクトルがどの命令文を使用するかはベクトルごとに例外なく決まる。これについては本稿のベクトルの項で解説される。
【０１０９】
ベクトルは命令文を決める為の仕組になっていることにも留意するべきである。本プログラムを構成する座標関数、同期関数、制御ベクトル、ＶＷＡ（ウイルス観察アルゴリズム）と言うプログラムはベクトルが決まれば決まるものである。添付されている解説、構造図、プログラム模型を参照されたい。こうした仕組みによれば、本プログラムは
（１）従来のプログラムと同じ本業をやり遂げる、
（２）並行してウイルス問題を解法する、
ものである。本プログラムを追跡（ウオークスルー）すれば、上記（１）（２）の役割が果たされていることが把握できる。
【０１１０】
プログラムは名詞を主語化し、その主語を脈略する仕組であることに留意すれば、シナリオ関数の動性を理解することができる。プログラムの機能とは主語を脈略する仕組にほかならない。従来のプログラムもこの点は同じである。プログラムはデータ処理をしていると言うより主語の脈略を生成していると言うほうが妥当である。結果的に、本発明はウイルス問題を解法するプログラムが存在することの証明論にもなっている。
【０１１１】
（シナリオ関数の定義式）
次に、プログラムの基盤となるシナリオ関数を解説する。シナリオ関数も所謂電算機のプログラムにほかならない。プログラムの機能的役割からすれば、シナリオ関数は従来のプログラムとなんら変わることのないプログラムである。シナリオ関数は以下の定義式で表される。即ち、
Ｓ＝Φ０（Φ４（｛Ｌ４｝、｛Ｗ４｝、Ｅ４１、Ｅ４２、Ｔ４）＋Φ２（｛Ｌ２｝、｛Ｒ２｝、Ｔ２）＋Φ３（｛Ｌ３｝、Ｔ３１、Ｔ３２、Ｔ３３））
シナリオ関数は定義式で決定表現できる世界で初めてのプログラム構造である。右辺の全記号はそれぞれ決定論で導かれるプログラムである。シナリオ関数の定義式の各記号を以下に解説する。
【０１１２】
（シナリオ関数の解）
シナリオ関数には従来のプログラムにはない解という概念が成立する。シナリオ関数の解（Ｓ）は、名詞、或は命令文の実行順序ではなく、主語のデータ結合の様相である。これは意味の仕組とも呼ばれる（図１２参照）。意味の仕組はシナリオ関数の動性の全景であ

る。シナリオ関数の解の求め方は意味の仕組の項で解説される。これ迄の論理の流れ図はシナリオ関数の解の部分に相当する。本発明では動性の全景は存在すると仮説される全体の輪郭に向かう究極の（外延的）様相のとして位置付けられる。ベクトル、座標関数、同期関数はシナリオ関数の解となる主語のデータ結合を可能な限界まで外延的に成立させる為の仕組になっている。

【０１１３】

（ベクトルの構造）

図３（ベクトルの基本構造）を参照する。同図に示されるように、ベクトルは７種の叙述規約で構成される。それぞれ、第１，２，３，４，５，６，７規約と記される。３Ｓ１０１は第１規約で、第２規約に進むかここで終了するかが判定される。３Ｓ１０１１は第１規約の出口である。３Ｓ２０１は第２規約で、ベクトルの本来の処理が行われる場である。３Ｓ３０１は第３規約で、ベクトルに於ける本来の処理に関する判定が行われる場である。３Ｓ４０１は第４規約で、第３規約の是で、ベクトルに於ける本来の処理を完了させる場である。３Ｓ４０１１はベクトルに於ける本来の処理を完了による出口である。３Ｓ３０２は第５規約で、第３規約の否で行われる処理の場である。３Ｓ３０３は第６規約で、第５規約の是で行われるベクトルが再起を要請する場である。３Ｓ３０３１は第６規約の出口である。３Ｓ３０４は第７規約で、第５規約の非で行われるベクトルが再起停止を要請する場である。３Ｓ３０４１は第７規約の出口である。

【０１１４】

同図に示されるように、ベクトルの始点は１か所で第１規約である。ベクトルの終点(出口)は４ヶ所で第１、第４、第６、第７規約である。４ヶ所の終点(出口)で役割を終えるベクトルの状態をベクトルの正統性という（ベクトルの正統性の項参照）。ベクトルは２種のフラグ（第６フラグ、第７フラグ）を固有する。第２フラグはウイルス対応の本プログラムの為に追加されたものである。第２フラグは第２規約を通過した証をオンで示す。第６フラグはオンで自分の再起動を要請する。第７フラグはオンで自分の再起動の一時停止を宣言する。ベクトルの第３、５，６，７規約はベクトルの再起構造の仕組である（再起構造の項参照）。

【０１１５】

（第４領域の正統性）

第４領域はベクトルが統治する命令で示唆される。ベクトルの第３規約は第４領域の正統性を判定し、正統でなければベクトルの再起を促す為に、第５規約に向かう指示を出す。正統な第４領域は第３規約で判定され第４規約で決定される。正統な第４領域とは第４領域が汚染されていないことを意味する。ベクトルが統治する命令文で示唆された第４領域はその命令文の成立に関わる全変数主語が正統であれば、正統であるといえる。この判定は第３規約で行われる。この判定を第４領域の正統性と記す。

【０１１６】

（ベクトルの解）

正統な第４領域がベクトルの解である。ベクトルの第５規約では正統な第４領域の成立可能性の有無が判定されるが、その方法については後述される。ベクトルの第６規約は正統な第４領域の成立の可能性が同じ座標周期にあることを宣言する（座標周期の項参照）。ベクトルの第７規約は正統な第４領域の成立の可能性が近い未来にはないことを宣言する。ベクトルの第１規約では、自分の正統性を用いて、自分の作用をここで終えるか第２規約に進むかを判定する。ベクトルの構造の由来は本発明者の本（コンピュータウイルスを無力化するプログラム革命）で解説されている。図４Ａ、図５にて示されるように、本発明のプログラムのベクトルはベクトルで決まるウイルス観察アルゴリズムと既述の様に第２フラグを加えて成立している。

【０１１７】

（ベクトル区分）

ベクトルは以下の３種に区分できる。

（１）命令文の主語を解とするベクトル

（２）１個以上の命令文で決まる機能の成否を解とするベクトル
（３）シナリオ関数の制御の成否を解とするベクトル
上記（１）は主語ベクトル、（２）はＬ４型制御ベクトル、（３）は制御ベクトルと総称される。
【０１１８】
（ベクトルの種別）
図２（ベクトル種別）を参照する。同図に示されるように、ベクトルの種別はＬ４、Ｗ４、Ｅ４１、Ｅ４２、Ｔ４、Ｌ２、Ｒ２、Ｔ２、Ｌ３、Ｔ３１、Ｔ３２、Ｔ３３の１２種である。Ｌ４、Ｗ４、Ｌ２、Ｒ２、Ｌ３は主語ベクトルである。主語ベクトルは最小クラスの名詞を主語化する。最小クラスの名詞は単元文に属す。
【０１１９】
（単元文）
どの様なプログラム言語でもその命令文を単元化すれば以下１０種の命令文種で叙述されている。単元化とは構文を１機能１構文の命令文に解釈し直すことである。ここで、単元化された構文を単元文という。単元文は１．領域文、２．注釈（定置）文、３．翻訳文、４．代入文、５．定値文、６．条件文、７．入力文、８．出力文、９．呼出文、１０．制御文の１０種である。ベクトルが統治する命令文は単元文である。単元文はベクトルに統治される。ベクトルは時制を帯同している。故に、ベクトルに統治される単元文には時制が付与されることになる。
【０１２０】
単元文のその時制は以下の様になる。
（１）領域文、注釈文、翻訳文、制御文は時制を超越する。
（２）代入文、出力文、呼び出し文の時制は「今」である。
（３）定値文、入力文の時制は「過去」である。
（４）条件文の時制は「未来」である。
ここにおいて、時制を持つ単元文は主語を持つ。主語を成立させる単元文を統治するベクトルを主語ベクトルという。ベクトルの第４領域はベクトルの解が治まる領域である。主語ベクトルの第４領域は主語とも呼ばれる。シナリオ関数には固有の制御ベクトルＥ４１、Ｅ４２、Ｔ４、Ｔ２、Ｔ３１、Ｔ３２、Ｔ３３がある。これらは従来のプログラムとは関係がない。
【０１２１】
（名詞、主語、変数主語）
名詞とは領域の名称、主語とは領域の内容である。主語ベクトルは主語化される名詞(領域名)を付与して識別される。
例；：Ｌ４，名詞。
【０１２２】
名詞が主語化される仕組は命令文で決定される。故に、主語ベクトルでは主語を持つ命令文が統治される。そして、その命令文の主語が主語ベクトルの解となる。条件文には変数主語はあるが、主語がない。しかし、条件文は主語を持つ命令文に呼応されるので、その命令文の名詞を用いて、Ｌ３，名詞として識別され、主語ベクトルとして存在する。しかし、Ｌ３，名詞の領域は条件文の成否をオン、オフのいずれかで表す為の第４領域となる。
【０１２３】
Ｅ４１、Ｅ４２、Ｔ４、Ｔ２、Ｔ３１、Ｔ３２、Ｔ３３はシナリオ関数の制御ベクトルである。制御ベクトルは所属するパレットの種別を付与して識別する。
例；Ｅ４１，Ｐ４という形式になる。
Ｌ４型制御ベクトルとは主語を持たない命令文が果たす機能を統治するベクトルである。
識別はＬ４型制御ベクトルを代表する命令文に登場番号を付して示す。
【０１２４】
（Ｌ４型制御ベクトルの補足）

Ｌ４型制御ベクトルに複数の命令文が属す場合、そして、その命令文の中に主語ベクトル化される命令文が属す場合、それは主語ベクトル化することができる。或いは、その命令文が既に主語ベクトル化されているのであれば、Ｌ４型制御ベクトルのその命令文の位置にはその命令文に代わり、その命令文の第４領域の成否を判定する叙述することができる。即ち、第４領域オン∧第２フラグオン∧第６フラグオフ∧第７フラグオフであることと記すことができる。もし、この条件を満たしていなければ、この命令文は成立していない。即ち、そのＬ４型制御ベクトルは成立していないことになる。

【０１２５】

（ベクトルが統治する命令文）

主語ベクトルでは主語を成立させる１個の命令文が統治される。即ち、代入文、定値文、呼出文、入力文はそれぞれのベクトルの第２規約、条件文は第３規約、そして、出力文は第４規約で統治される。制御文を統治するベクトルはない。領域文はベクトル化されない。領域文で定義される領域情報は従来のプログラムと同じ様に定義されて基本的には同期関数の先頭に置かれる。翻訳文はＬ４型制御ベクトルで統治される。

【０１２６】

（制御ベクトルの補足）

Ｅ４１は本プログラムの終了状態を捉える。Ｅ４２は論理矛盾を捉える。Ｔ４は座標関数４を２に切り替える条件を捉える。Ｔ２は座標関数２を３に切り替える条件を捉える。Ｔ３１は座標関数３を自分の座標関数４に切り替える条件を捉える。Ｔ３２は座標関数３を自分の下位の座標関数４に切り替える条件を捉える。Ｔ３３は座標関数３を自分の最上位の座標関数３に切り替える条件を捉える。図９（シナリオ関数のランク構造）を参照すれば示されるように、自分の下位、最上位は本プログラムのランク構造で成立する概念である。

【０１２７】

（ベクトル記号）

ベクトル種別Ｌ４、Ｗ４、Ｅ４１、Ｅ４２、Ｔ４、Ｌ２、Ｒ２、Ｔ２、Ｌ３、Ｔ３１、Ｔ３２、Ｔ３３の数値はそのベクトルが存在する時制を示すものである。「４」はそのベクトルが時制上の今、「２」は過去、「３」は未來で存在するとの識別である。この概念は全時制が同期することに因り、意味が成立するとの本研究の仮説から生じたものである。

【０１２８】

（Ｌ３ベクトルの補足）

Ｌ３は真偽それぞれに定義される。それに呼応するベクトルは、主語ベクトルである。主語ベクトルは主語を持つ構文を制している。それに対し、Ｌ３は主語を持たない構文を制している。それ故、Ｌ３に呼応する主語ベクトルはそのＬ３に自分の主語を写す。Ｌ３はその主語を得て識別を成立させることになる。

【０１２９】

（パレット）

パレットとはベクトルの３種の部分集合のことである。

（１）｛｛Ｌ４｝、｛Ｗ４｝、Ｅ４１、Ｅ４２、Ｔ４｝はパレット４、Ｐ４と記される。

（２）｛｛Ｌ２｝、｛Ｒ２｝、Ｔ２｝はパレット２、Ｐ２と記される。

（３）｛｛Ｌ３｝、Ｔ３１、Ｔ３２、Ｔ３３｝はパレット３、Ｐ３と記される。

パレット４に搭載されるベクトルは時制「今」で成立する命令文を統治するベクトルである。例えば代入文、出力文である。パレット２に搭載されるベクトルは時制「過去」で成立する命令文を統治するベクトルである。例えば定値文、入力文である。パレット３に搭載されるベクトルは時制「未來」で成立する命令文を統治するベクトルである。例えば条件文である。

【０１３０】

（パレット上のベクトルの並べ方）

従来のプログラムでは命令の並べ方の順位は、実行の前にその順位を決めて並べてなければならない。その理由は従来のプログラムは論理結合型思考法で成立するからである。他

方、シナリオ関数ではパレットに属すベクトルの並べ方の順位は自由である。これは、シナリオ関数ではパレット上のベクトルの全体が繰り返し実行されることにより、ベクトルの主語の間にデータ結合が成立することで、次第にベクトルの順位が決まる仕組になっているからである。そして、繰り返し実行されることにより主語のデータ結合が成立し、シナリオ関数の解となる流れ図を凌駕する意味の仕組が導出される構造になっている（意味の仕組の項を参照）。

【０１３１】

（同期関数）

図８（同期関数（Φ０）の基本構造）を参照する。同図において、同期関数が起動すると、主語成立数、再起カウンタの値がゼロであれば、初期起動なので、主語成立数のスタック構造を初期値化する。初期起動でなければ、８Ｓ１０１はこのスタックを初期値化しない。主語成立数のスタック構造は図６参照。８Ｓ２０１では制御ベクトルＥ４１の成否が調べられる。Ｅ４１は本プログラムが終了状態にあることを告げるベクトルである。Ｅ４１が成立していれば、同期関数は終了手続きを行うプログラム（ＳＥＰ）を起動する。Ｅ４１が成立していなければ、同期関数は統治する座標関数を起動する為の判定を行う。８Ｓ３０１では制御ベクトルＴ３１の成否を調べる。Ｔ３１が成立していれば、８Ｓ３０２でＴ３１を初期値化してから自分が統治する座標関数４を起動する。Ｔ３１が成立していなければ８Ｓ４０１で制御ベクトルＴ４の成否を調べる。Ｔ４が成立していれば、８Ｓ４０２でＴ４を初期値化してから自分が統治する座標関数２を起動する。Ｔ４が成立していなければ、８Ｓ５０１で制御ベクトルＴ２の成否を調べる。Ｔ２が成立していれば、８Ｓ５０２でＴ２を初期値化してから自分が統治する座標関数３を起動する。Ｔ２が成立していなければ、８Ｓ６０１で制御ベクトルＴ３２の成否を調べる。Ｔ３２が成立していれば、８Ｓ６０１１でＴ３２を初期値化してから自分の隣下にいる本プログラムの座標関数４を起動して終了する。Ｔ３２が成立していなければ、８Ｓ６０３で制御ベクトルＴ３３の成否を調べる。Ｔ３３が成立していれば、８Ｓ６０２１でＴ３３を初期値化してから自分の最上位にいる本プログラムの座標関数３を起動して終了する。Ｔ３３が成立していなければ、８Ｓ６０３で自分の座標関数４を起動する。

【０１３２】

図８に示されるように、シナリオ関数には１個の同期関数が存在する。同期関数はシナリオ関数に於ける最上位の制御論理体である。同期関数は３種の座標関数を統治する。同期関数は再起構造体である。同期関数の再起構造は同期周期を成立させる。シナリオ関数が使用する領域定義文は同期関数に置かれる。

【０１３３】

（座標関数）

図７（座標関数の基本構造（Φ４，２，３））を参照する。同図において、７Ｓ１０１は同期関数で起動されるとＯＳが捉える命令破壊信号を調べる。７Ｓ１２０１は命令破壊信号が捉えられれば、この座標関数は本システムを停止させる為に準備されるプログラム（ＳＬＰ）を起動する。７Ｓ２０１は命令破壊が行われていなければ、座標関数決まるパレットに搭載されているベクトルの搭載順位１のベクトルを指定する。ベクトルの搭載順には規則はない。ここが座標周期のはじまり点である。７Ｓ３０１では指定されたベクトルが起動される。７Ｓ４０１では起動されたベクトルがＲＥＴＵＲＮするとこの座標関数はＯＳが捉える命令破壊信号を調べる。そして、命令破壊信号が捉えられれば、ＳＬＰを起動する。７Ｓ５０１では命令破壊がなかったので、ベクトルの搭載順位が１個更新される。７Ｓ６０１では更新された搭載順位がベクトルの搭載数を超えたか否かが調べられる。搭載数を超えてなければ指定された順位のベクトルが７Ｓ３０１で起動される。７Ｓ７０１はベクトルの搭載数が超えた場合の処理である。ここでは指定されているパレットのベクトルの中に第６フラグがオンのものがあるかないかが調べられる。もし、第６フラグがオンのものがあれば、このパレットはまだ臨界状態には足していないのでパレットに搭載されているベクトルを搭載順位１のものから再起させる必要がある。この場合、７Ｓ７０１で再起カウンタを更新して７Ｓ２０１に移る。７Ｓ７０１で第６フラグがオンのものが

なければ、このパレットは臨界状態には達している。この状態はこの座標関数の座標周期の終了である。７Ｓ８０１では第７フラグは全てオフにされる。これは次の座標周期で第４領域が成立する可能性があるので、その可能性を阻害しないためである。７Ｓ９０１では主語成立数のスタックを更新する。７Ｓ１００１ではベクトルＥ４２の成否が調べられる。Ｅ４２はＯＳでは捉えられない命令の上書き汚染を捉える為のものである。もし、成立していれば、この座標関数はＳＬＰを起動する。Ｅ４２が成立していなければ７Ｓ１１０１ではこの座標関数の制御を同期関数に返す。

【０１３４】

図７に示されるように、：シナリオ関数には３個の座標関数が存在する。３個の座標関数（Φ４、Φ２、Φ３）は統治するパレット以外は同形である。座標関数はパレットに属すベクトルを統治する。座標関数はパレット内の主語の同期を成立させる仕組である。座標関数は再起構造体である。座標関数の再起構造には座標周期が成立する。図６（スタック構造）を参照すれば示されるように、座標周期は主語成立数のスタック構造を決める時宜である。

【０１３５】

（本プログラムのウイルスを解法する仕組の規範）

プログラムが自分でウイルス汚染を解法するには、以下の仕組を備えることが必要である。

（１）実行時にウイルス汚染を捉える仕組

（２）除染する仕組

（３）除染後再起させる仕組

【０１３６】

プログラムが再起構造を持てば、汚染された領域は初期値化すれば除染されることになる。本プログラムはシナリオ関数で分かる様にシナリオ関数はあたかも最初からやり直す様な再起構造になっている。故に、汚染された領域をその定義属性で初期値化すれば除染されることになる。この再起構造はシナリオ関数の動性がデータ結合型として成立する故に成立する仕組である。従来のプログラムの動性は論理結合で成立するので、シナリオ関数の様な領域単位ごとに再起を成立させる構造を成立させることができない。本プログラムではウイルス汚染は、結果的に、本プログラムに生じる構造上の叙述矛盾として捉えられる（ベクトルの正統性、第４領域の正統性を参照）。

【０１３７】

故に、本プログラムではウイルス情報を一切用いずにウイルス問題を解法することができる（ウイルスの無力化を参照）。付言すれば、ウイルス問題はウイルス情報を用いては解法できないという点に気付く必要がある。その理由は、ソフト危機について詳しくなればなる程、その解決策が複雑になり、結果的に埒外のプログラム言語に委ね、解法を放棄した様に、ウイルスについても詳しくなるとその解決策は臨床的になり、解法には及ばなくなるからである。換言すれば、問題に詳しくなることと問題を解法することは往々にして別問題になることが多いのである。故に、ウイルス作成者を活用してウイルス問題を解決するという発想では、事態を複雑化させるだけとなる。改めていえば、ソフト危機をプログラム言語で解決しようとして、コード化率を上げようとするだけのプログラム言語の普及で、プログラム言語の表面的な利便性が図られた結果、ソフト危機は改善されず、事態を更に複雑化させて来た、というのがこれ迄の歴史であったと指摘せざるを得ない。

【０１３８】

叙述法とはプログラム言語（文法）に制約を与える律性のことである。ここで言及しなければならないのは、プログラム言語と叙述法の概念は似て非なるものだということである。この制約の律性はプログラム構造で決まる。プログラム言語による多くの制約を、究極的には普遍性を成立させるに至るプログラム構造が、プログラム問題、ウイルス問題を解法することになる。プログラム言語文法を超越する叙述法の概念は、本発明者による１５年間の独自の動性分析で得られたものである。この概念が本願に係るシナリオ関数に反映されている。本プログラムの模型を参照すれば、その簡潔さの中にありながらもこの叙述

法的な感覚が存するのを気付くことが可能であると考える。プログラム問題はプログラム構造を変えれば解法できることを実証したのがシナリオ関数である、ともいえる。

【０１３９】

常識的なウイルスなら１組のウイルスで電子計算機システムを瓦解させる力を持っている。故に、ウイルスというのは１個でもプログラムに侵入させてはならない。これは「ウイルス問題解法の絶対条件」である。であるが故に、ウイルス問題の解法は単にアイデアでは成就することはできない。即ち、解法の原理を発見しなければ、ウイルス問題は解法することができない、といえる。主語（第４領域）は同じ主語でも実行時には様々に変化するので、主語が汚染されているや否やは実行時に判定するしか方法がないのである。そして、主語だけを観察しても汚染の有無は解らない。主語が汚染されているや否やは主語の成立経緯の妥当性を観察するのでなければ、機械的に判定することができないのである。

【０１４０】

ベクトルには普遍的なフラグ(第６フラグ、第７フラグ)がある。これが主語の叙述矛盾を捉える仕組の基盤になっている。この２種のフラグに３番目の第２フラグを追加すれば、第４領域（主語）の有無と３種のフラグとの相対関係で主語の成立経緯の妥当性を判定することが可能となる。この仕組が主語のウイルス汚染を機械的に判定する本プログラムの仕組の原理になっている。シナリオ関数に第２フラグを追加するのは、主語の時宜を捉える為に設けられたものである。主語が成立する時宜が不明では主語の汚染は捉えられない。本発明のプログラムの主語の成立経緯の妥当性を捉える仕組はベクトルの第３規約で成立する。

【０１４１】

（ウイルス問題解法の公理）

本発明では、ウイルス問題は従来プログラムの動性の不完全さが発症させる現象として透徹されている。本発明は、１９７３年のソフト危機の段階で本来改革されるべきであった論理結合型プログラムのソースコードの定義、並びにその動性の不完全さが、実際には改善されることなくそのままの状態で電算機システムの利便性を増加させてきた結果、それらプログラムが温床となって、ウイルス問題が発症している、という認識に立つものである。また、ウイルス症状の発症を阻止する防御壁を、侵入媒体の手前に置くとする、ウイルス対応の現商品群がとっている方法も基本的に欠陥があり、ウイルス問題の解法には至らないということも自明であって、問題がある。

【０１４２】

上述したように、理論上ウイルスの侵入窓口は侵入媒体の命令文の数だけある。故に、侵入媒体の手前でウイルスを阻止するという発想は問題を真摯に解法する姿勢とは考え難いものとしかいいえない。換言すれば、課題を解法せずに事業の種にしているこれ迄のこの分野の事業手法がなお懲りることなくウイルス問題にも適用されていると指摘せざるを得ない。こうしたものと対局をなす本願によれば、ウイルス問題を単純化して解法する為の公理が立論でき、それを以下に示す。

（１）ウイルス汚染は侵入媒体の叙述矛盾として捉えられる。

（２）侵入媒体の汚染を解法すれば侵入媒体のウイルス問題は消去される。

【０１４３】

（公理の補足）

公理１に準じれば、ウイルス問題を解法する為には、ウイルスを侵入媒体に侵入させて侵入媒体に汚染を発症させなければならない。公理１はウイルスの侵入方法はウイルス問題から除外できるということを意味している。侵入媒体に生じる叙述矛盾は侵入媒体の構造で決まるから、侵入媒体の叙述矛盾はウイルスとは無関係に捉えることができるということになる。一方、公理２はウイルスの仕組から必然の帰結である。ウイルスの無力化とはウイルスの存在を無意味にすることである。

【０１４４】

（本プログラムの構造）

本プログラムの為のベクトルは図３から図４Ａに置き換えられる。図４Ａではウイルス観

察アルゴリズム（図５）が使用されている。図４Ａにおいて、４ＡＳ１０１はベクトルに添付されるウイルス観察アルゴリズム（ＶＷＡ）である。４ＡＳ２０１はベクトルの第１規約で、ここではベクトルの正当性判定が行われる。ベクトルの正当性判定は図４Ｂ（後述する）を参照されたい。４ＡＳ２０１１は第１規約の出口である。４ＡＳ３０１はベクトルの第２規約で、ここでは第２フラグがオンにセットされる。第２規約で統治される命令文はここに置かれる。４ＡＳ４０１はベクトルの第３規約で、ここでは第４領域の正当性が判定される。４ＡＳ５０１はベクトルの第４規約で、第３規約の判定が是ならば、ベクトルで統治される命令文により求められた主語が求められた時点の再起カウンタと共に第４領域に移される。第６フラグはオフにされる。４ＡＳ５０１１は第４規約の出口である。４ＡＳ４０２はベクトルの第５規約である。第３規約の判定が否ならば、第５規約でその判定が是となる可能性の有無が判定される。この判定の為に主語成立数のスタック構造が用いられる。ここで第４領域が初期値化される。４ＡＳ４０３はベクトルの第６規約で、第５規約で可能性が有れば、ベクトルの再起要請の為にここで、第６フラグをオンに、第２フラグをオフにする。４ＡＳ４０３１は第６規約の出口である。４ＡＳ４０４はベクトルの第７規約で、第５規約で可能性が無ければ、ベクトルの再起停止の為にここで第７フラグをオンに、第２フラグをオフにする。４ＡＳ４０４１は第７規約の出口である。第６、第７規約にはもし、サービスメッセージなどの補完的措置が必要ならばそれを取り込むことは可能である。４ＡＳ６０１では主語成立数カウンタに１が加算される。４ＡＳ７０１では再起カウンタの値が保持される。

【０１４５】

図４Ｂにおいては、本プログラムが使用するベクトルの正統性を捉える為の第４領域、第２フラグ、第６フラグ、第７フラグの相対関係が示されている。この相対性はベクトルの出口２、３、４に於けるベクトルの状態として求められている。

【０１４６】

図５は本発明の一実施形態に係るウイルス観察アルゴリズムの概念図である。同図において、５Ｓ１０１は第２フラグの汚染の有無が観察される。５Ｓ２０１は第６フラグの汚染の有無が観察される。５Ｓ３０１は第７フラグの汚染の有無が観察される。５Ｓ４０１は第４領域の汚染の有無が観察される。５Ｓ５０１は上記のひとつでも汚染されていればこのベクトルを初期値化する。５Ｓ５０１１は主語成立数カウンタを１減じる。５Ｓ９０１は本ＶＷＡの終了出口である。５Ｓ６０１は主語成立数カウンタの汚染の有無が観察される。５Ｓ７０１は再起カウンタの汚染の有無が観察される。５Ｓ８０１はベクトルの初期値化で済まされない汚染なので本システムを停止させる為に準備されるプログラム（ＳＬＰ）を起動させる。

【０１４７】

３種の座標関数は図７を基本とする。ここにおいて、主語成立数カウンタは本プログラムには１個である。本プログラムはシナリオ関数の定義式に準拠する。本プログラムの定義構造は従来のプログラムと異なることは明白である。結果的に、本発明はプログラムの精密な構造論に帰着している。

【０１４８】

（ベクトルの正統性の補足）

ベクトルは座標関数により、再起することに於いて、ベクトルが自分の出口２、３，４のいずれかをパスしていること、或はウイルス観測プログラム（ＶＷＡ）で初期値化されている状態であること、がベクトルの正統性を満たす要件である。因みに出口３の状態はＶＷＡの出口状態と同じである（図３、図５参照）。ベクトルの正統性は第１規約で第２規約に進むか否やの判定条件となる。この条件は次の様に定義される。

【０１４９】

（第４領域と３種のフラグの相対性）

第４領域と３種のフラグの間には普遍的な相対性が成立する（図４Ｂ参照）。ベクトルの正統性とはこの相対性のことをいう。

（１）ＶＷＡで初期値化されたベクトルの状態は出口３の状態と同じである。

この場合、ベクトルは第２規約に進む。
（２）出口２のベクトルの状態は、
（第４領域オン∧第２フラグオン∧第６フラグオフ∧第７フラグオフ）である。
この状態でＶＷＡを経て再起されるベクトルは第１規約でＲＥＴＵＲＮする。
（３）出口３のベクトルの状態は、
（第４領域オフ∧第２フラグオフ∧第６フラグオン∧第７フラグオフ）である。
この状態でＶＷＡを経て再起されるベクトルは第２規約に進む。
（４）出口４のベクトルの状態は、
（第４領域オフ∧第２フラグオフ∧第６フラグオフ∧第７フラグオン）である。
この状態で、ＶＷＡを経て再起されるベクトルは第１規約でＲＥＴＵＲＮする。
ベクトルの正統性、第４領域の正統性は意味が異なるので混同しない様に注意が必要である。第４領域の正統性は第３規約で使用する。ベクトルの正統性は第１規約で使用する。

【０１５０】
（ウイルス汚染対応のベクトルの構造）
ベクトルは自分の第４領域の正統な成立を果たす為に自分の第４領域の成立に関わる全変数主語の主語が汚染されずに成立しているや否や、を自分の第３規約で観察する。そして、変数主語の主語がひとつでも汚染されていれば、自分の第４領域の成立を断念し、第５規約に進む。第５規約では正統な主語が近未來に成立する可能性を主語の成立数の変化を用いる独特な方法で調べる。主語成立数のスタック項目を参照されたい。

【０１５１】
（第４領域の汚染の解法）
ＶＷＡはフラグが汚染されていれば、そのベクトルを初期値化して、本プログラムの再起の仕組を用いて正統な第４領域を求め直す様に再起させる。しかし、ここで示すのは汚染されていないフラグの相対性が第４領域が正統であると示唆しているにも関わらず、即ち、第２フラグオン∧第６フラグオフ∧第７フラグオフにも関わらず、第４領域が、ウイルスで上書き汚染されている問題である。この第４領域は再起回数ｋｎで成立している。その第４領域は第４規約で第４領域ｋｎとして保持される。ＶＷＡは実行の都度、第４領域ｋｎと実行時の第４領域をＸＯＲし、同じであれば、第４領域ｋｎは汚染されていないことになる。同じでなければ、汚染されていることになる（図５参照）。汚染されていればこのベクトルはこのＶＷＡで初期値化される。この措置の為に、
（１）座標関数で更新される再起カウンタが設置される。
（２）再起カウンタは本プログラムに１個である。
（３）設置場所は座標関数の臨界判定がＮＯとなった直後の位置である。
（４）ベクトルは第４領域が成立した時点でその時点の再起カウンタＫｎを付して第４領域を保持する。
（５）再起カウンタは主語成立数、主語成立数のスタックと同様に汚染されない特権領域に置かれる。

【０１５２】
（特権領域）
再起カウンタ、主語成立数、スタック構造はＶＷＡ(図５)のプログラム模型の３種のフラグの汚染観察の仕組でみることができる様にＸＯＲの定数を用いて汚染の有無が観察される。本発明ではその様な領域（３種のフラグ、再起カウンタ、主語成立数、スタック構造）を特権領域と記す。特権領域の汚染は全てＶＷＡで３種のフラグの汚染観察と同じ様にＸＯＲ定数を用いて観察される。この為のＸＯＲ定数例を以下に示す。
（１）フラグ領域は４桁の最上位でオンオフが示される。
フラグ領域が汚染されれば、最上位桁だけでなく、その下３桁も汚染される。故に、４桁の下３桁ゼロが、この場合のＸＯＲ定数として成立する。
（２）再起カウンタは４桁の最上位から計数値が示される。
再起カウンタが汚染されれば、４桁の下１桁も汚染される。故に、４桁の下１桁ゼロが、この場合のＸＯＲ定数として成立する。

（３）主語成立数カウンタ４桁の最上位から計数値が示される。
主語成立数カウンが汚染されれば、４桁の下１桁も汚染される。故に、４桁の下１桁ゼロが、この場合のＸＯＲ定数として成立する。
（４）主語成立数スタックが汚染されれば、その性質から命令破壊と同じになる。この状態はＸＯＲ定数では捉えられないので、Ｅ４２で捉える。
本プログラムの厳密で奥深い仕組が汚染を捉え自動回復させる仕組になっている。本プログラムのこの様な仕組を破壊するウイルス汚染は命令破壊以外にはあり得ないことになる。

【０１５３】
（３種のフラグのオンオフの場所）
・第２フラグの初期値はオフである。
・第２フラグは第６規約がオフにする。
・第２フラグは第７規約がオフにする。
・第２フラグは座標関数がオフにする。
・第６フラグの初期値はオンである。
・第６フラグは第６規約がオンにする。
・第６フラグは第４規約がオフにする。
・第６フラグは第５規約がオフにする。
・第７フラグの初期値はオフである。
・第７フラグは第７規約がオンにする。
・第７フラグは座標関数がオフにする。

【０１５４】
（ウイルス観察アルゴリズム（ＶＷＡ））
ウイルス観察アルゴリズムは３種のフラグの汚染を観察し、汚染されていればその第４領域も汚染されているものと判断する。この場合、ベクトルは初期値化される（図５参照）。ウイルス観察アルゴリズム（図５）は図４Ａで見る様にベクトルごとその先頭に置かれる。３種のフラグが汚染されずに第４領域が汚染される状況は起こり得ないと判断する。もし、この事態が起きれば論理矛盾が起きることになり、その場合にはＥ４２がカバーすることになる。

【０１５５】
（プログラム動性の付言）
２０００年代の動性解析法の権威であったラバール大学（カナダ）のプログラムの動性解析法はプログラムに属す名詞（変数名）の成立軌跡を採る方法であった。これによれば、従来のプログラムの成立軌跡は、いわゆるスパゲッティ構造になる。一方、本願に係るシナリオ関数を解析すると、名詞の成立軌跡はいわゆる櫛形となり、いわゆるスパゲッティ構造にはならないことが確証され、従来型ソフトウエアの問題構造が可視的に示された（図１Ｅ参照）。すなわち、本発明による主語の軌跡を採る動性解析法を用いれば、従来のプログラムもシナリオ関数も本発明が名付ける意味の仕組（図１２）の構造に帰着する。これはシナリオ関数がプログラムの叙述法として、最高のランクにあることの証の一つであると考えられる。意味の仕組の使途はシナリオ関数よりも従来プログラムの動性解析ツールとして有効である。

【０１５６】
（本プログラムの動性）
本プログラムの電算機内の動性は図１２を主語のデータ結合で成立させる為の仕組である。従来のプログラムの電算機内の動性は図１２を論理結合で成立させる仕組である。ウイルスは稼動中の本プログラムに侵入する。本プログラムが稼動中とは、本プログラムが電算機を用いて、本プログラムの解である、例えば、図１２を作成中の状態をいう。ウイルスは複数の本プログラムの中から例えば、本プログラムＡを侵入媒体として選んだ関係で本プログラムＡに侵入する。本プログラムＡからいえば、ウイルスの侵入とは、仕事中に訳もなく、自分が汚染されている事態である。この場合に、汚染される命令文の数は誰に

も分からない。故に本発明ではベクトルに統治される命令は全て実行時にその汚染の有無がベクトルにより観察される。本プログラムには大切なフラグ、主語成立数カウンタ、そのスタックがあるが、それらも汚染されている可能性もある。汚染されていれば、本プログラムは結果的にこれらの汚染も命令破壊に至らない命令汚染を捉えるＥ４２で捉えることになる。

【０１５７】

（本発明の妥当性）

本稿の解説、添付されているプログラム模型、そして、以下に示す課題の３者を紐付ければ、本発明の妥当性を見出すことができる。この場合の課題とは以下の４種の仕組である。

（１）本業の仕事をする仕組

（２）ウイルスを無力化する仕組

（３）本プログラムの完全性

（４）従来のプログラムを本プログラムに自動変換するアルゴリズムの成立

【０１５８】

（本業の仕事をする仕組）

３種の座標関数はそれぞれが統治するパレットに搭載されている主語ベクトルを起動し、主語を成立させる。本プログラムは再起構造により主語を脈略関係を用いて可能な限り成立させる仕組になっている。以上は本プログラムが本業を行っていることの必要かつ十分な証明である。

【０１５９】

（本プログラムの汚染される対象）

本プログラムの汚染される対象は以下に示されるとおりである。

（１）ベクトル、座標関数、同期関数を規約する命令文

（２）データ領域

（３）３種のフラグ、主語成立数カウンタ、主語成立数のスタック領域

【０１６０】

（ウイルスを無力化するとは）

ウイルスを無力化するとは以下の措置を講じることである。

（１）データ領域の初期値化

（２）プログラムの実行停止

【０１６１】

（ウイルスを無力化する為の仕組）

本プログラムはで使用される１５個の普遍的（例外のないこと）な仕組を以下に示す。

（１）ベクトルの構造

（２）３種のフラグ

（３）ＶＷＡとその置かれる位置

（４）再起構造の仕組

（５）座標周期の仕組

（６）同期構造の仕組

（７）ベクトルの第１規約で使用するベクトルの正統性を捉える仕組

（８）ベクトルの第３規約で使用する第４領域の正統性を捉える仕組

（９）ＶＷＡが使用する第４領域の汚染を捉える仕組

（１０）ＶＷＡが使用する特権領域の汚染を捉えるＥＯＲ定数の仕組

（１１）主語成立数スタックを用いて論理矛盾をＥ４２で捉える仕組

（１２）主語成立数スタックを用いて第４領域の成立可能性を第５規約で捉える仕組

（１３）主語成立数スタックの為に主語成立数を計数する為の位置

（１４）パレットの臨界を捉える仕組

（１５）プログラムの臨界を捉える仕組

【０１６２】

（本プログラムの完全性）
正統な主語の最大数で成立する脈略の仕組は本プログラムの本業の完全性の証明である。因みに、従来のプログラムの本業の完全性をこの方法で証明すれば、プログラムのテストは不要になる。
【０１６３】
（従来のプログラムを本プログラムに自動変換する仕組）
従来のプログラムからＬＹＥＥ空間(図１１Ａ、図１１Ｂ)を生成するアルゴリズム、並びに、ＬＹＥＥ空間情報から本プログラムを生成するアルゴリズム（図１Ｂ）により従来のプログラムを本プログラムに自動変換することができる。複数のシナリオ関数で構成される本発明のプログラム構造はランク構造と呼ばれる。図９はシナリオ関数のランク構造図の一例を示す図である。同図において、Φ４（１，１）はランク構造（１，１）の座標関数４、Φ２（１，１）はランク構造（１，１）の座標関数２、Φ３（１，１）はランク構造（１，１）の座標関数３を示す。ランク構造の（Ｘ，Ｙ）はランク構造を構成する本プログラムの識別子である。制御ベクトルＴ４，２，３１は同じランクの本プログラムの座標関数４，２，３を連接する役割を果たす。制御ベクトルＴ３２，３３は異なるランクの座標関数の座標関数を連接する役割を果たす。ランク構造のシナリオ関数はベクトルＴ３２、Ｔ３３で連接される。従来のシステムでは理論上も実際上も複数のプログラムで構成しなければならないのに対し、シナリオ関数では理論上唯ひとつのシナリオ関数で構成することができる。この理由は、シナリオ関数ではシステムを構成するのに必要なプログラムはベクトルで対応することができるということにある。
【０１６４】
ランク構造はシステムの運営管理上の問題に因る理由で設けられる。従来のプログラムは論理結合型であることから、座標関数、同期関数に相当するロジックが成立しないので、そのシステムでは複数のプログラムが必要になる。シナリオ関数では座標関数、同期関数は普遍的なので、ベクトルの量とは無関係に成立する。シナリオ関数を複数本に小分けしなければならない格別な理由は存在しない。シナリオ関数を本プログラムと置き換えて考えることができる。
【０１６５】
ここで、本プログラムに属す主語が本プログラムのランク構造ではどの位置の本プログラムに属すかについて説明する。図１３は本発明の一実施形態に係る本プログラムに属す主語が本プログラムのランク構造ではどの位置の本プログラムに属すかを示す為の図である。同図に示されるように、ランク構造（１，１）に属す主語Ａ（１，１）の変数主語、例えばＢ、Ｃの主語がどのランク構造の本プログラムに属すかが示されている。この例では変数主語Ｂ、Ｃの主語は同じランクに属していることを示している。変数主語Ｂ、Ｃの主語はＢ、Ｃである。
【０１６６】
（論理結合型プログラムとシナリオ関数の関係）
図１Ｂに示されるように、論理結合型プログラムは専用のツールで本発明のシナリオ関数に自動変換することができる。
【０１６７】
（主語と変数主語）
シナリオ関数では命令は主語を求める為の作用である。例えば、命令「Ａ＝Ｂ＋Ｃ」のＡ、Ｂ、Ｃは名詞、Ａが主語、そして、Ｂ、Ｃは変数主語となる。変数主語Ｂ、Ｃはシナリオ関数の何処かで主語Ｂ、主語Ｃになっている。シナリオ関数を本プログラムと置き換えて考えればよい。
【０１６８】
（ウイルス問題を解法する本プログラムの基盤的特徴）
ラバール大学のプログラムの動性解析で明らかな様に従来のプログラムの定義並びにその動性の（いわゆる）スパゲテイ様相に普遍性を見出すことはできない。一方、シナリオ関数の動性はデータ結合型であるが、その構成要素であるベクトル、座標関数、同期関数の

叙述法は論理結合型であり、これらのソースコードの定義には複写性という普遍性が成立する。これはベクトルの定義に普遍性が成立する為である。データ結合にはラバール大学のプログラムの動性解析で明らかな様に、櫛形光景の普遍性が成立する。本プログラムがウイルス問題を叙述矛盾として解法できるのはソースコードの定義、並びにその動性に見られる普遍性が成立しているからである。本発明に係る同期関数、座標関数は固有のデータ領域を所有していない。本願において固有のデータ領域を所有しているのはベクトルだけである。故に、ベクトルがデータ領域の汚染を解法すれば、同期関数、座標関数はその影響を受けないことになる。ベクトルは従来の部分プログラムの様に機能を定義しているわけではない。唯、第４領域（主語)を決める仕組が定義されているだけである。その結果、座標関数は臨界まで第４領域を求める仕組に、同期関数は求められた第４領域を同期させる仕組に、それぞれ帰着している。従来的発想に必要な様々な機能がシナリオ関数ではすっかり不要になっている。その結果、シナリオ関数は従来のプログラムに較べれば、見違える様に単調になっている。ウイルス汚染を捉え解法する仕組はシナリオ関数のこの単調さ故に成立する。

【０１６９】

（再起構造の意義）

シナリオ関数の再起構造はパレットに属す主語とその主語達の脈略化を可能的限界迄成立させる為のものである。即ち、意味の仕組を可能的限界まで追い求める外延的仕組の究極構造になっている。この再起構造は、Ａ．ベクトル、Ｂ．３種の座標関数、Ｃ．１個の同期関数、が共通して持つ仕組である。ベクトルは自分の第４領域（主語）を成立させる為に自分を可能的限界（臨界）まで稼動する再起構造を持っている。座標関数は統治するパレットの中で、ベクトルの第４領域（主語）を脈略化してひとつでも多くの第４領域を成立させる為に可能的限界まで稼動する再起構造をもっている。

【０１７０】

同期関数は３種のパレット間の関係に於いて、ベクトルの第４領域（主語）をひとつでも多く成立させる為に可能的限界まで同期関数を稼動する再起構造を持っている。これら再起構造の全ての仕組は、Ａ．ベクトルの第４領域、Ｂ．ベクトルの３種のフラグ、Ｃ．５種のＴ型ベクトル（Ｔ４、Ｔ２、Ｔ３１、Ｔ３２、Ｔ３３）、Ｄ．２種のＥ型ベクトル（Ｅ４１、Ｅ４２）、で制御されている。従来のプログラムが不可避的に発生させるがゆえの弱点である複雑さ問題を、本発明においては、プログラムの本質的な洞察とそれを基に得られたプログラムの再起構造とによって、シナリオ関数として、その複雑さを単調構造に変えて回避させている。本プログラムはそのシナリオ関数を基盤にウイルス問題を解法するプログラム構造として成立している。

【０１７１】

（同期構造の意義）

本発明においては、従来のプログラムの動的分析を通じて、われわれがいう問題とは非同期の様相であることを突き止め、それに対する解法をそれを同期構造に変える手続きとして位置付けられるものである。シナリオ関数の動性はデータ結合で成立し、従来のプログラムの動性は論理結合で成立する。それらはいずれも解法を意味する。シナリオ関数は電算機を用いて同期構造を成立させる仕組であるが、従来のプログラムはソースコードの定義段階で既に解法の道筋を決めている。そのことが従来プログラムの動性をいわゆるスパゲテイ構造として複雑化させている原因になっている。図１２に示される意味の仕組、が解法の理想的な構造である。従来のプログラムの世界で意味の仕組を成立させるアルゴリズムがこれ迄発見されていないのは、プログラムの非同期様相から同期構造の考えを導くことは不可能であったからであると考えることができる。因みに、同期構造とはプログラムの意味の仕組を求めるアルゴリズムであり、これは世界で初めての発見であったものである。

【０１７２】

（同期構造の補足）

本発明の研究では、意味は時制(今、過去、未来)を同期させる仕組で成立すると仮説し、

その仕組、即ち、解を求める仕組がシナリオ関数として結実している。付言すれば、従来のプログラムには解の概念は存在しない。故に、従来のプログラムには同期構造という概念自体が存在しない。これは従来プログラムが完全に見落としている欠陥である。シナリオ関数では全時制の同期が成立する時宜、つまり、意味の仕組が成立する時宜を同期周期と名付けている。同期関数の上で同期周期を見ることができる。シナリオ関数に登場するベクトルは１２種であるが、それらは３種の時制で分けられている。パレットは時制で分類されたベクトルを収容する器である。故に、パレットも３種になる。３種のパレットはそれぞれ固有の３種の座標関数で統治されている。座標関数４は時制の今を同期させる働きの模型である。座標関数２は時制の過去を同期させる働きの模型である。座標関数３は時制の未来を同期させる働きの模型である。同期関数はその３種の同期を統合する役割を果たしている。

【０１７３】

（パレットの臨界）

座標関数４は統治するパレット４の全ベクトルを起動し、その起動を終える都度、パレット４に属すベクトルの中に第６フラグがオンのベクトルの存在の有無を調べる。そして、ひとつでもオンのベクトルがあれば、座標関数４は、再びパレット４の全ベクトルを初めから起動し直す。この再起動は座標関数４に属すベクトルの中に第６フラグのオンのものがなくなる迄繰り返される。換言すると、これはパレット４の全ベクトルの第４領域がオン且つ第７フラグがオンで充足される迄繰り返すということである。本発明ではこの充足状態をパレット４の臨界と呼ぶ。座標関数４が最初に起動してからパレット４の臨界状態に至るまでが、座標関数４の座標周期である。

【０１７４】

座標関数２，３にも同じ様に座標周期が成立する。座標周期は同期構造のひとつの形態である。座標関数４の座標周期は本プログラムに於ける主語成立数のスタックを生成及び更新する時宜になっている。この時宜で決まる主語成立数のスタックが全てのベクトルの第５規約で用いられる。即ち、ベクトルの解である第４領域が未来に成立するや否や、を判定する未來条件は座標関数４の座標周期の終了点とその時点に於ける主語成立数によるスタック構造で捉えられるということである。この仕組は従来プログラムの発想では求められるものではない。

【０１７５】

（主語の脈略を成立させる仕組）

図１２の意味の仕組とは主語の脈略化図である。これを観れば誰もがその脈略を納得できるが、それを観ない限りこの全てを知ることができない。調和脈略は流れ図として知ることはできても、超言語脈略の大部分は未知の脈略である。そして、プログラムは従来のものであっても意味の仕組みの基で成立していることに気付いて欲しい。ということは、われわれはプログラムがどの様に成立しているかについての論理の殆どの部分が解からずにプログラムを作っているにも関わらず調和脈略だけでプログラムの論理が解かった様に振る舞っているということである。シナリオ関数並びにシナリオ関数を基盤とする本発明のプログラムはわれわれが解かっているのは名詞を主語化する論理だけだとして、その前提でプログラムの解である意味の仕組を成立させるプログラムである。プログラムの完全性やウイルス問題の解法はプログラムの意味の仕組を知ることに因り成立するものである。

【０１７６】

（論理矛盾を捉える仕組）

本プログラムは、シナリオ関数のベクトルに第２フラグとウイルス観察アルゴリズムを加えて正統な主語の脈略を成立させる仕組である。

【０１７７】

本プログラムはプログラムの処理をこれ迄のプログラムと違って、主語の脈略で成立させる仕組になっている。そして、脈略は汚染されていない主語、換言すれば、プログラム意図に沿わない主語で成立しては正常な処理を果たしたことにはならない。故に、本プログ

ラムは汚染されていない主語、即ち正統な主語で脈略が成立する仕組になっている。即ち、ウイルス観察アルゴリズム、ベクトルの第１規約、同第３規約は汚染された主語が脈略に加われない様に観察する為の論理が叙述されている。故に、汚染された主語が脈略に加われればその叙述に矛盾が生じる。この矛盾は論理に反する叙述として捉えることができる。換言すれば、本プログラムのこの仕組はウイルス汚染を主語毎に捉えるということで、汚染を完全に捉えることができるが、この仕組は完全なプログラムの仕組を求める研究に於いて求められたものである。即ち、ウイルスとは無関係に求められたものである。故に、本プログラムではウイルスタグ（ウイルス情報）を用いずにウイルス汚染を捉え、それを無力化することができる。ウイルス問題をウイルス知識で解法することはできない問題である。ウイルス観察アルゴリズムは汚染を捉え汚染されたそのベクトルを初期値化する。ベクトルの第１規約はベクトルの正統性を第４領域と３種のフラグの相対性で観察する。ベクトルの第３規約は第４領域（主語）の正統性を正統な主語の脈略の関係に於いて観察する。本プログラムの観点でいえばウイルスは横目で本プログラムを眺めながら本プログラムに手出しができず、ただ居眠りを装う以外に存在することができない。

【０１７８】

（主語の計数）

正統な主語（第４領域）が成立すれば主語成立数はベクトルの第４規約で主語成立数カウンタに１を加えて計数される。そして、ウイルス観察アルゴリズムでは、主語（第４領域）が正統でないと判定すれば、そのウイルス観察アルゴリズムがベクトルを初期値化する。その時点でウイルス観察アルゴリズムが主語成立数カウンタの値を１減じる。

【０１７９】

（スタック構造）

図６のスタック構造図を参照する。同図は本発明の一実施形態に係る本プログラムが使用する主語成立数（第４領域成立数と同義）のスタック構造を示す図である。本スタックは本プログラムに１個で座標関数４の座標周期を利用して生成される。座標関数２、３では生成する必要がない。本スタックはベクトルで利用される。

利用の仕方を以下に記す。

（１）ベクトルの第５規約での利用

ベクトルの第５規約は自分の第４領域が同じ座標周期で成立する可能性の有無を問う為の規約である。この判定の為に主語成立数のスタックを用いる。

即ち、ＮＳ１＞ＮＳ５なら自分の第４領域は同じ座標周期で成立する可能性があると判断することができる。

【０１８０】

（２）ベクトルＥ４２での利用

Ｅ４２はＯＳでは捉えられない命令破壊を捉える為のベクトルである。Ｅ４２はこの命令破壊を主語成立数の論理矛盾、即ち、ＮＳ１＜ＮＳ５として捉える。本プログラムが正常であれば、ＮＳ１とＮＳ５の関係はかならずＮＳ１＞ＮＳ５である。しかし、この事態が発症すると本プログラムに於けるこの関係はＮＳ１＜ＮＳ５にゆきつく。

原因は以下のいずれかである。

１）ベクトルの誤り

２）命令破壊に至らない命令コードの汚染

実行の段階で上記１）の誤りはないので、原因は上記２）である。

【０１８１】

（３）ベクトルＥ４１での利用

Ｅ４１は本プログラムの主語成立数が臨界に達した状態（本プログラムの正常終了状態）を捉える為のベクトルである。Ｅ４１はこの事態をＮＳ１＝ＮＳ５として捉える。

【０１８２】

（４）ベクトルＴ４、Ｔ２、Ｔ３での利用

各パレットで成立する主語の限界状態はパレットの臨界と記す。パレット４の臨界はＴ４の成立条件となる。パレット２の臨界はＴ２の成立条件となる。パレット３の臨界はＴ３

１、Ｔ３２，Ｔ３３の二つの成立条件のひとつとなる。もう一つの条件は主語分布表で示される主語の所在箇所である。

【０１８３】

（１）従来のプログラムは汚染されれば、その影響はそのプログラムだけでなく、そのプログラムが属すシステムに及ぶ。

（２）従来のプログラムを本発明のプログラムに置き換えれば、本プログラムはウイルス問題を解法する。

（３）従来のプログラムを図１Ｂで解る様に専用ツールで本プログラムに自動変換することが可能である。

（４）従来のプログラム技術者が、本プログラムを能率よく自分で作る為には約６０時間程度の座学が必要である。

【０１８４】

（本プログラムに論理結合型プログラムが存在させられる理由）

本プログラムに関するプログラム模型を本稿に添付する。本プログラムでは、データ領域を持つプログラムは本願に係るベクトルだけである。データ領域は汚染の対象である。故に、ベクトルにはデータ領域の汚染を無力化させる為の仕組が必要である。故に、本発明においては、ベクトルには第２フラグとＶＷＡとが加えられている。本発明のベクトル以外のプログラム、即ち、ＶＷＡ、座標関数、同期関数は固有のデータ領域を持たないことに留意してほしい。そのことに於いて、これらプログラムにはデータ領域の汚染問題は生じない。ということは即ち、これらに生じる汚染は命令破壊だけである、ということになる。これはベクトルの効果であり、本プログラムの特長である。故に、ＶＷＡ、座標関数、同期関数は論理結合型プログラムのままで良い。プログラムが固有のデータ領域を持たなければ、そのプログラムはだれが作成したものでも、譬え、複写性が成立していなくても、共用することができることになる。即ち、ＶＷＡ、座標関数、同期関数は誰が作成したものでも、複写性が成立していなくても共用できる、ということである。これも複写性の１種となる。

【０１８５】

（本プログラムの模型）

本プログラムのアルゴリズム例（以下、「プログラム模型」という。）は本明細書の内容、プログラム化される構造、即ち図３、４Ａ、５、７、８を納得できる様にする為の解説資料として添付されている。故に、複雑による納得の混乱を避ける為に本明細書の内容の全てが模型化されているわけではない。本願発明の技術思想の開示として十分な範囲に留められている。従来のプログラム作成者でも本明細書の記述と本プログラム模型をガイダンスとすれば、本発明に基づくプログラムを作成することができる。本プログラム模型は本プログラムのコーデングを自動化するツールを示唆する仕様書にもなっている（図１Ｂ参照）

【０１８６】

本プログラムはプログラム言語ごとに作成する。画面操作言語、ＤＢ，ＤＣ言語についてはどのようなものを用いても本願発明は適用可能であり、作成者の好みのものが利用できる。本プログラムは以下のプログラム模型に基づいてベクトル、座標関数、同期関数を作り、シナリオ関数の定義式通りに組み立てれば完成するものである。座標関数、同期関数は世界にひと組あれば、それをそのまま利用することができる。

【０１８７】

（ＶＷＡ（Ｌ４，Ａ）の論理結合型プログラムの模型）

図５を参照する。次に、この場合のプログラム模型を示す。

００１　ＳＴＡＲＴ.

００２　第２フラグの汚染観察：

００２１　第２フラグの（２，３，４）桁を抽出する.

００２２　第２フラグの（２，３，４）桁と対応するフラグ常数（０，０，０）をＸＯＲする.

００２２１　ＺＥＲＯなら，第２フラグは汚染されていない、
ＧＯ　ＴＯ　第６フラグの汚染観察.
００２２２　ＮＯＴ　ＺＥＲＯなら，第２フラグは汚染されている、
ＧＯ　ＴＯ　Ｌ４，Ａの初期値化.
００３　第６フラグの汚染観察：
００３１　第６フラグの（２，３，４）桁を抽出する.
００３２　第６フラグの（２，３，４）桁と対応するフラグ常数（０，０，０）をＸＯＲする.
００３２１　ＺＥＲＯなら，第６フラグは汚染されていない、
ＧＯ　ＴＯ　第７フラグの汚染観察.
００３２２　ＮＯＴ　ＺＥＲＯなら，第６フラグは汚染されている、
ＧＯ　ＴＯ　Ｌ４，Ａの初期値化.
００４　第７フラグの汚染観察：
００４１　第７フラグの（２，３，４）桁を抽出する.
００４２　第７フラグの（２，３，４）桁と対応するフラグ常数（０，０，０）をＸＯＲする.
００４２１　ＺＥＲＯなら，第７フラグは汚染されていない、
ＲＥＴＵＲＮ.
００４２２　ＮＯＴ　ＺＥＲＯなら，第７フラグは汚染されている、
ＧＯ　ＴＯ　Ｌ４，Ａの初期値化.
００５　Ｌ４，Ａの初期値化
００５１　第４領域を初期値化する.
００５２　第２フラグをオフにする.
００５３　第６フラグをオンにする.
００５４　第７フラグをオフにする.
００６　主語成立数カウンタをマイナス１にする。
００７　ＲＥＴＵＲＮ.

【０１８８】

（Ｌ４，Ａのベクトル模型）

このベクトルの命令文例を「Ａ＝Ｂ＋Ｃ＋５１２」とする。Ｌ４，Ａが統治する命令文はベクトルの叙述法により第２規約に置かれる。このＬ４，ＡにはＬ３，Ａがいないものとする。Ｌ４，ＡにＬ３，Ａがいれば本第３規約にはＬ３，Ａの正統性判定も加えられることになる。次に、この場合のプログラム模型を示す。

００１　ＳＴＡＲＴ.
００２　ＮＯＰ.
００３　ＣＡＬＬ　ＶＷＡ（Ｌ４，Ａ）.
００４　第１規約：Ｌ４，Ａの正統性を用いて第２規約に進むか、ＲＥＴＵＲＮかを判定。
　　＊ベクトルの正統性項参照
００５　第２規約。
００５１　ＭＯＶＥ　１　ＴＯ　第２フラグ.
００５２　Ａ＝Ｂ＋Ｃ＋５１２.
００６　第３規約：Ａの正統性（第４領域の正統性）の判定。
００６１　　Ｌ４，Ｂの第４領域は正統か.
００６１１　正統なら　ＧＯＴＯ　００６２.
００６１２　正統でなければ　ＧＯＴＯ　第５規約.
００６２　　Ｌ４，Ｃの第４領域は正統か.
００６２１　正統なら　ＧＯＴＯ　第４規約.
００６２２　正統でなければ　ＧＯＴＯ　第５規約.

００７　第４規約。
００７１　ＭＯＶＥ　０　ＴＯ　Ｌ４，Ａの第６フラグ.
００７２　ＭＯＶＥ　Ａ　ＴＯ　Ｌ４，Ａの第４領域.
００７３　ＡＤＤ　１　ＴＯ　主語成立数カウンタ.
００８　ＲＥＴＵＲＮ.
００９　第５規約：Ｌ４，ＡのＡが同じ座標周期で成立するや否やの判定。
００９１　ＭＯＶＥ　０　ＴＯ　第６フラグ.
００９２　（ＮＳ１＝ＮＳ５）なら第７規約に進む.
００９３　（ＮＳ１＞ＮＳ５）なら第６規約に進む.
０１０　第６規約：Ｌ４，Ａの再起要請の宣言。
０１０１　Ｌ４，Ａの第４領域を初期値化.
０１０２　ＭＯＶＥ　１　ＴＯ　第６フラグ.
０１０３　ＭＯＶＥ　０　ＴＯ　第２フラグ.
０１０４　ＲＥＴＵＲＮ.
０１１　第７規約：Ｌ４，Ａの再起停止宣言。
０１１１　ＭＯＶＥ　１　ＴＯ　第７フラグ.
０１１２　ＭＯＶＥ　０　ＴＯ　第２フラグ.
０１１３　ＲＥＴＵＲＮ.

【０１８９】

（Ｌ３，Ｄのベクトル模型）

このベクトルの命令文例を「ＩＦ　Ｘ＜Ｙ（真）」とする。Ｌ３，Ｄが統治する命令文はベクトルの叙述法により第３規約に置かれる。次に、この場合のプログラム模型を示す。

００１　ＳＴＡＲＴ.
００２　ＮＯＰ.
００３　ＣＡＬＬ　ＶＷＡ（Ｌ３，Ｄ）.
００４　第１規約：Ｌ３，Ｄの正統性を用いて第２規約に進むか、ＲＥＴＵＲＮかを判定。
　　＊ベクトルの正統性の項参照
００５　第２規約。
００５１　ＭＯＶＥ　１　ＴＯ　第２フラグ.
００５２　ＮＯＰ.
００６　第３規約：ＩＦ　Ｘ＜Ｙの正統性の判定。
００６１　　Ｌ４，Ｘの第４領域は正統か.
００６１１　正統なら　ＧＯＴＯ　００６２.
００６１２　正統でなければ　ＧＯＴＯ　第５規約.
００６２　　Ｌ４，Ｙの第４領域は正統か.
００６２１　正統なら　ＧＯＴＯ　００６３.
００６２２　正統でなければ　ＧＯＴＯ　第５規約.
００６３　　ＩＦ　Ｘ＜Ｙか.
００６３１　正統なら　ＧＯＴＯ　第４規約.
００６３２　正統でなければ　ＧＯＴＯ　第５規約.
００７　第４規約。
００７１　ＭＯＶＥ　０　ＴＯ　Ｌ３，Ｄの第６フラグ.
００７２　ＭＯＶＥ　１　ＴＯ　Ｌ３，Ｄの第４領域.
００７３　ＡＤＤ　１　ＴＯ　主語成立数カウンタ.
００８　ＲＥＴＵＲＮ.
００９　第５規約：Ｌ３，ＤのＤが同じ座標周期で成立するや否やの判定。
００９１　ＭＯＶＥ　０　ＴＯ　第６フラグ.
００９２　（ＮＳ１＝ＮＳ５）なら第７規約に進む.

００９３　（ＮＳ１＞ＮＳ５）なら第６規約に進む.
０１０　第６規約：Ｌ３，Ｄの再起要請の宣言。
０１０１　Ｌ３，Ｄの第４領域を初期値化.
０１０２　ＭＯＶＥ　１　ＴＯ　第６フラグ.
０１０３　ＭＯＶＥ　０　ＴＯ　第２フラグ.
０１０４　ＲＥＴＵＲＮ.
０１１　第７規約：Ｌ３，Ｄの再起停止宣言。
０１１１　ＭＯＶＥ　１　ＴＯ　第７フラグ.
０１１２　ＭＯＶＥ　０　ＴＯ　第２フラグ.
０１１３　ＲＥＴＵＲＮ.

【０１９０】
（Ｌ２，Ｃのベクトル模型）
このベクトルの命令文例を「Ｃ＝１１」とする。「Ｌ２，Ｃ」が統治する命令文はベクトルの叙述法により第２規約に置かれる。このＬ２，ＣにはＬ３，Ｃがいないものとする。Ｌ２，ＣにＬ３，Ｃがいれば本第３規約にはＬ３，Ｃの正統性判定も加えられることになる。その場合には第５，６，７規約が発生することになる。次に、この場合のプログラム模型を示す。

００１　ＳＴＡＲＴ.
００２　ＮＯＰ.
００３　ＣＡＬＬ　ＶＷＡ（Ｌ２，Ｃ）.
００４　第１規約：Ｌ２，Ｃの正統性を用いて第２規約に進むか、ＲＥＴＵＲＮかを判定。
　　＊ベクトルの正統性の項参照
００５　第２規約。
００５１　ＭＯＶＥ　１　ＴＯ　第２フラグ.
００５２　Ｃ＝１１.
００６　第３規約：Ｃ＝１１のＣの正統性の判定。ＮＯＰ.
００７　第４規約
００７１　ＭＯＶＥ　０　ＴＯ　Ｌ２，Ｃの第６フラグ.
００７２　ＭＯＶＥ　Ｃ　ＴＯ　Ｌ２，Ｃの第４領域.
００７３　ＡＤＤ　１　ＴＯ　主語成立数カウンタ.
００８　ＲＥＴＵＲＮ.
００９　第５規約：ＮＯＰ.
０１０　第６規約：ＮＯＰ.
０１１　第７規約：ＮＯＰ.

【０１９１】
（Ｗ４，ＵＲＩＡＧＥのベクトル模型）
このベクトルの命令文例を「ＷＲＩＴＥ　ＵＲＩＡＧＥ，ＤＢ１１」とする。これはＵＲＩＡＧＥを外部記憶領域「ＤＢ１１」に書く命令文である。この命令文の主語はＤＢ１１に成立する。ＤＢ１１の正統性は不明である。このベクトルの役割は正統なＬ４，ＵＲＩＡＧＥをＤＢ１１に出力するということである。Ｗ４，ＵＲＩＡＧＥが統治する命令文はベクトルの叙述法により第４規約に置かれる。Ｗ４，ＵＲＩＡＧＥのＶＷＡ，第１、第３規約ではＬ４，ＵＲＩＡＧＥに置き換えて処理される。このＷ４，ＵＲＩＡＧＥにはＬ３，ＵＲＩＡＧＥがいないものとする。Ｗ４，ＵＲＩＡＧＥにＬ３，ＵＲＩＡＧＥがいれば本第３規約にはＬ３，ＵＲＩＡＧＥの正統性判定も加えられることになる。次に、この場合のプログラム模型を示す。

００１　ＳＴＡＲＴ.
００２　ＮＯＰ.
００３　ＣＡＬＬ　ＶＷＡ（Ｗ４，ＵＲＩＡＧＥ）.

００４　第１規約：Ｌ４，ＵＲＩＡＧＥの正統性を用いて第２規約に進むか、ＲＥＴＵＲＮかを判定。
　　＊ベクトルの正統性の項参照
００５　第２規約。
００５１　ＭＯＶＥ　１　ＴＯ　第２フラグ.
００５２　ＮＯＰ.
００６　第３規約：Ｌ４，ＵＲＩＡＧＥの正統性の判定。
００６１　　Ｌ４，ＵＲＩＡＧＥの第４領域は正統か.
００６１１　正統なら　ＧＯＴＯ第４規約.
００６１２　正統でなければ　ＧＯＴＯ　第５規約.
００７　第４規約。
００７１　ＭＯＶＥ　０　ＴＯ　Ｗ４，ＵＲＩＡＧＥの第６フラグ.
００７２　ＷＲＩＴＥ　ＵＲＩＡＧＥ，ＤＢ１１.
００７３　ＡＤＤ　１　ＴＯ　主語成立数カウンタ.
００８　ＲＥＴＵＲＮ.
００９　第５規約：　Ｌ４，ＵＲＩＡＧＥのＵＲＩＡＧＥが同じ座標周期で成立するや否やの判定。
００９１　ＭＯＶＥ　０　ＴＯ　第６フラグ.
００９２　（ＮＳ１＝ＮＳ５）なら第７規約に進む.
００９３　（ＮＳ１＞ＮＳ５）なら第６規約に進む.
０１０　第６規約：Ｗ４，ＵＲＩＡＧＥの再起要請の宣言。
０１０１　Ｌ４，ＵＲＩＡＧＥ. の第４領域を初期値化.
０１０２　ＭＯＶＥ　１　ＴＯ　第６フラグ.
０１０３　ＭＯＶＥ　０　ＴＯ　第２フラグ.
０１０４　ＲＥＴＵＲＮ.
０１１　第７規約：Ｗ４，ＵＲＩＡＧＥの再起停止宣言。
０１１１　ＭＯＶＥ　１　ＴＯ　第７フラグ.
０１１２　ＭＯＶＥ　０　ＴＯ　第２フラグ.
０１１３　ＲＥＴＵＲＮ.

【０１９２】
（Ｒ２　ＤＢ１１のベクトル模型）
このベクトルの命令文例を「ＲＥＡＤ　ＤＢ１１，集計」とずる。これは外部記憶領域「ＤＢ１１」を集計に移す命令文である。この命令文の主語は集計に成立する。このベクトルの役割は正統な集計を成立させるということである。Ｒ２　ＤＢ１１が統治する命令文はベクトルの叙述法により第２規約に置かれる。Ｒ２　ＤＢ１１のＶＷＡ，第１、第３規約ではＬ４，集計に置き換えて処理される。このＲ２　ＤＢ１１にはＬ３，ＵＲＩＡＧＥがいないものとする。Ｒ２　ＤＢ１１にＬ３，ＤＢ１１がいれば本第３規約にはＬ３，ＤＢ１１の正統性判定も加えられることになる。次に、この場合のプログラム模型を示す。
００１　ＳＴＡＲＴ.
００２　ＮＯＰ.
００３　ＣＡＬＬ　ＶＷＡ（Ｒ２　ＤＢ１１）.
００４　第１規約：Ｌ４，集計の正統性を用いて第２規約に進むか、ＲＥＴＵＲＮかを判定。
　　＊ベクトルの正統性の項参照
００５　第２規約。
００５１　ＭＯＶＥ　１　ＴＯ　第２フラグ.
００５２　ＲＥＡＤ　ＤＢ１１，集計.
００６　第３規約：Ｌ４，集計の正統性の判定。
００６１　　Ｌ４，集計の第４領域は正統か.
００６１１　正統なら　ＧＯＴＯ　００７.

００６１２　正統でなければ　ＧＯＴＯ　第５規約.

００７　第４規約。
００７１　ＭＯＶＥ　０　ＴＯ　第６フラグ.
００７２　ＭＯＶＥ　集計　ＴＯ　ＲＥＡＤ　ＤＢ１１，集計の第４領域.
００７３　ＡＤＤ　１　ＴＯ　主語成立数カウンタ.
００８　ＲＥＴＵＲＮ.
００９　第５規約：Ｌ４，集計の集計が同じ座標周期で成立するや否やの判定。
００９１　ＭＯＶＥ　０　ＴＯ　第６フラグ.
００９２　（ＮＳ１＝ＮＳ５）なら第７規約に進む.
００９３　（ＮＳ１＞ＮＳ５）なら第６規約に進む.
０１０　第６規約：ＲＥＡＤ　ＤＢ１１，集計の再起要請の宣言。
０１０１　ＲＥＡＤ　ＤＢ１１，集計の第４領域を初期値化.
０１０２　ＭＯＶＥ　１　ＴＯ　第６フラグ.
０１０３　ＭＯＶＥ　０　ＴＯ　第２フラグ.
０１０４　ＲＥＴＵＲＮ.
０１１　第７規約：ＲＥＡＤ　ＤＢ１１，集計の再起停止宣言。
０１１１　ＭＯＶＥ　１　ＴＯ　第７フラグ.
０１１２　ＭＯＶＥ　０　ＴＯ　第２フラグ.
０１１３　ＲＥＴＵＲＮ.

【０１９３】
（Ｅ４１，Ｐ４プログラムの論理結合型模型）
Ｅ４１，Ｐ４は本プログラムが終了状態にあることを告知する。Ｅ４１，Ｐ４を分かり易くする為、論理結合型で示しているが、Ｅ４１は固有のデータ領域（第４領域）を持っているので、実際はベクトル構造でプログラムする。次に、この場合のプログラム模型を示す。
００１　ＳＴＡＲＴ.
００２　主語成立数スタックからＮＳ１を採る.
００３　主語成立数スタックからＮＳ５を採る.
００４　（ＮＳ１）＝（ＮＳ５）の判定.
００４１　ＹＥＳなら　ＧＯ　ＴＯ　００５.
００４２　ＮＯなら　ＧＯ　ＴＯ　００６.
００５　ＭＯＶＥ　１　ＴＯ　Ｅ４１の第４領域.
００６　ＲＥＴＵＲＮ.

【０１９４】
（Ｅ４２，Ｐ４プログラムの論理結合型模型）
Ｅ４２，Ｐ４は命令破壊には至らない命令汚染により、本プログラムに生じる論理矛盾の発症を告知する。Ｅ４２，Ｐ４を分かり易くする為、論理結合型で示しているが、Ｅ４２は固有のデータ領域（第４領域）を持っているので、実際はベクトル構造でプログラムする。次に、この場合のプログラム模型を示す。
００１　ＳＴＡＲＴ.
００２　主語成立数スタックからＮＳ１を採る.
００３　主語成立数スタックからＮＳ５を採る.
００４　（ＮＳ１）＜（ＮＳ５）の判定.：本プログラムに生じる論理矛盾はこの関係で捉えることができる。
００４１　ＹＥＳなら　ＧＯ　ＴＯ　００５.
００４２　ＮＯなら　ＧＯ　ＴＯ　００６.
００５　ＭＯＶＥ　１　ＴＯ　Ｅ４２の第４領域.
００６　ＲＥＴＵＲＮ.

【０１９５】

（Ｔ４，Ｐ４プログラムの論理結合型模型）
Ｔ４，Ｐ４はその第４領域で座標関数４を座標関数２に切り替える条件が整ったことを告知する。Ｔ４，Ｐ４を分かり易くする為、論理結合型で示しているが、Ｔ４は固有のデータ領域（第４領域）を持っているので、実際はベクトル構造でプログラムする。次に、この場合のプログラム模型を示す。
００１　ＳＴＡＲＴ.
００２　Ｐ４の臨界判定.
００２１　パレット４の第６フラグ列の作成.
００２２　パレット４の第６フラグ列常数の作成.
００３　パレット４の第６フラグ列と第６フラグ列常数のＸＯＲ.
００３１　ＸＯＲがＺＥＲＯならＧＯＴＯ　００４.
００３２　ＸＯＲがＮＯＴ　ＺＥＲＯならＧＯＴＯ　００５.
００４　ＭＯＶＥ　１　ＴＯ　Ｔ４の第４領域.
００５　ＲＥＴＵＲＮ.
【０１９６】
（Ｔ２，Ｐ２プログラムの論理結合型模型）
Ｔ２，Ｐ２はその第４領域で座標関数２を座標関数３に切り替える条件が整ったことを告知する。Ｔ２，Ｐ２を分かり易くする為、論理結合型で示しているが、Ｔ２は固有のデータ領域（第４領域）を持っているので、実際はベクトル構造でプログラムする。次に、この場合のプログラム模型を示す。
００１　ＳＴＡＲＴ.
００２　Ｐ２の臨界判定：
００２１　パレット２の第６フラグ列の作成.
００２２　パレット２の第６フラグ列常数の作成.
００３　パレット２の第６フラグ列と第６フラグ列常数のＸＯＲ.
００３１　ＸＯＲがＺＥＲＯならＧＯＴＯ　００４.
００３２　ＸＯＲがＮＯＴ　ＺＥＲＯならＧＯＴＯ　００５.
００４　ＭＯＶＥ　１　ＴＯ　Ｔ４の第４領域.
００５　ＲＥＴＵＲＮ.
【０１９７】
（Ｔ３１，Ｐ３プログラムの論理結合型模型）
「Ｔ３１の第４領域」は座標関数３を自分の座標関数４に切り替える条件が整っていることを告知する。Ｔ３１，Ｐ３を分かり易くする為、論理結合型で示しているが、Ｔ３１は固有のデータ領域（第４領域）を持っているので、実際はベクトル構造でプログラムする。Ｔ３１の第４領域の初期値はオンである。次に、この場合のプログラム模型を示す。
００１　ＳＴＡＲＴ.
００２　Ｐ３の臨界判定：
００２１　パレット３の第６フラグ列の作成.
００２２　パレット３の第６フラグ列常数の作成.
００３　パレット３の第６フラグ列と第６フラグ列常数のＸＯＲ（第１条件）.
００３１　ＸＯＲがＺＥＲＯならＧＯＴＯ　００４.
００３２　ＸＯＲがＮＯＴ　ＺＥＲＯならＧＯＴＯ　００６.
００４　主語分布表（図１３）を用いて行う判定。
００４１　本プログラムの全主語は本プログラムに内在するか（第２条件）
００４２　ＹＥＳならＧＯＴＯ　００５.
００４３　ＮＯならＧＯＴＯ　００６.
００５　ＭＯＶＥ　１　ＴＯ　Ｔ３１の第４領域.
００６　ＲＥＴＵＲＮ.
【０１９８】
（Ｔ３２，Ｐ３プログラムの論理結合型模型）

「Ｔ３２の第４領域」は座標関数３をランク構造に於ける本プログラム（１，１）の下位の本プログラム（２，１）の座標関数４に切り替える条件が整っていることを告知する。Ｔ３２，Ｐ３を分かり易くする為、論理結合型で示しているが、Ｔ３２は固有のデータ領域（第４領域）を持っているので、実際はベクトル構造でプログラムする。次に、この場合のプログラム模型を示す。

００１　ＳＴＡＲＴ.
００２　Ｐ３の臨界判定：
００２１　パレット３の第６フラグ列の作成.
００２２　パレット３の第６フラグ列常数の作成.
００３　パレット３の第６フラグ列と第６フラグ列常数のＸＯＲ（第１条件）.
００３１　ＸＯＲがＺＥＲＯならＧＯＴＯ　００４.
００３２　ＸＯＲがＮＯＴ　ＺＥＲＯならＧＯＴＯ　００６.
００４　主語分布表（図１３）を用いて行う判定。
００４１　本プログラムの主語は隣下の本プログラムにも内在するか（第２条件）
００４２　ＹＥＳならＧＯＴＯ　００５.
００４３　ＮＯならＧＯＴＯ　００６.
００５　ＭＯＶＥ　１　ＴＯ　Ｔ３２の第４領域.
００６　ＲＥＴＵＲＮ.

【０１９９】

（Ｔ３３，Ｐ３プログラムの論理結合型模型）

「Ｔ３３の第４領域」は座標関数３をランク構造に於ける本プログラム（１，１）の最上位の本プログラムの座標関数３に切り替える条件が整っていることを告知する。Ｔ３３，Ｐ３を分かり易くする為、論理結合型で示しているが、Ｔ３３は固有のデータ領域（第４領域）を持っているので、実際はベクトル構造でプログラムする。次に、この場合のプログラム模型を示す。

００１　ＳＴＡＲＴ.
００２　Ｐ３の臨界判定：
００２１　パレット３の第６フラグ列の作成.
００２２　パレット３の第６フラグ列常数の作成.
００３　パレット３の第６フラグ列と第６フラグ列常数のＸＯＲ（第１条件）.
００３１　ＸＯＲがＺＥＲＯならＧＯＴＯ　００４.
００３２　ＸＯＲがＮＯＴ　ＺＥＲＯならＧＯＴＯ　００６.
００４　主語分布表（図１３）を用いて行う判定。
００４１　本プログラムの主語は隣上の本プログラムにも内在するか（第２条件）
００４２　ＹＥＳならＧＯＴＯ　００５.
００４３　ＮＯならＧＯＴＯ　００６.
００５　ＭＯＶＥ　１　ＴＯ　Ｔ３２の第４領域.
００６　ＲＥＴＵＲＮ.

【０２００】

（「主語分布表」図１３の解説）

同図は、主語がランク構造に属す本プログラムのどれに属しているかを表す為のものである。例えば（１，１）はランク構造に於けるシナリオ関数の位置座標である。

【０２０１】

（座標関数４プログラムの論理結合型模型）

座標関数４のログラム模型を以下に示す。

００１　ＳＴＡＲＴ.
００２　パレット４の搭載順位１（ｉ＝１）のベクトルを指定.：座標周期４の始まり
００３　指定された搭載順位（ｉ）のベクトルを起動.
００４　ここは起動したベクトルのＲＥＴＵＲＮ点です.
００５　命令破壊信号（ＯＳ）の有無判定.

００５１　有りで、ＳＥＰ起動．：本プログラムの実行を停止
００５２　なしで、ＧＯ　ＴＯ　００６．

００６　起動する次のベクトルを指定（ｉ＝ｉ＋１）．
００７　（ｉ）はパレット４のベクトルの搭載数（Ｎ４）を超えたか．
００７１　超えていれば　ＧＯ　ＴＯ　００８．
００７２　超えていなければ　ＧＯ　ＴＯ　００３．
００８　パレット４の臨界の成否判定．：Ｔ４の第４領域のオンオフを利用
００８１　オフなら　ＧＯ　ＴＯ　００２．
００８２　オンなら　ＧＯ　ＴＯ　００９．：座標周期４の終了
００９　論理矛盾発症の有無を判定：Ｅ４２の第４領域オンオフの利用．
００９１　オンで、ＳＬＰ起動．：本プログラムの実行停止処理
００９２　オフで、ＧＯ　ＴＯ　０１０．
：本プログラムには論理矛盾は起きていない
０１０　同期関数に戻る為に次の処理を行う
０１０１　パレット４の第７フラグオンのベクトルを全て初期値化する．
０１０２　スタック更新プログラムの起動．
０１１　同期関数にＲＥＴＵＲＮ．

【０２０２】

（座標関数２プログラムの論理結合型模型）
座標関数２のログラム模型を以下に示す。
００１　ＳＴＡＲＴ．
００２　パレット２の搭載順位１（ｉ＝１）のベクトルを指定．：座標周期２の始まり
００３　指定された搭載順位（ｉ）のベクトルを起動．
００４　ここは起動したベクトルのＲＥＴＵＲＮ点である．
００５　命令破壊信号（ＯＳ）の有無判定．
００５１　有りで、ＳＥＰ起動．：本プログラムの実行を停止
００５２　なしで、ＧＯ　ＴＯ　００６．

００６　起動する次のベクトルを指定（ｉ＝ｉ＋１）．
００７　（ｉ）はパレット２のベクトルの搭載数（Ｎ２）を超えたか．
００７１　超えていれば　ＧＯ　ＴＯ　００８．
００７２　超えていなければ　ＧＯ　ＴＯ　００３．
００８　パレット２の臨界の成否判定．：Ｔ２の第４領域のオンオフを利用
００８１　オフなら　ＧＯ　ＴＯ　００２．
００８２　オンなら　ＧＯ　ＴＯ　００９．：座標周期２の終了
００９　論理矛盾発症の有無を判定：Ｅ４２の第４領域オンオフの利用．
００９１　オンで、ＳＬＰ起動．：本プログラムの実行停止処理
００９２　オフで、ＧＯ　ＴＯ　０１０．
：本プログラムには論理矛盾は起きていない
０１０　同期関数に戻る為に次の処理を行う
０１０１　パレット２の第７フラグオンのベクトルを全て初期値化する．
０１１　同期関数にＲＥＴＵＲＮ．

【０２０３】

（座標関数３プログラムの論理結合型模型）
座標関数３のプログラム模型を以下に示す。
００１　ＳＴＡＲＴ．
００２　パレット３の搭載順位１（ｉ＝１）のベクトルを指定．：座標周期３の始まり
００３　指定された搭載順位（ｉ）のベクトルを起動．
００４　ここは起動したベクトルのＲＥＴＵＲＮ点である．

００５　命令破壊信号（ＯＳ）の有無判定.
００５１　有りで、ＳＥＰ起動.：本プログラムの実行を停止
００５２　なしで、ＧＯ　ＴＯ　００６.

００６　起動する次のベクトルを指定（ｉ＝ｉ＋１）.
００７　（ｉ）はパレット２のベクトルの搭載数（Ｎ３）を超えたか.
００７１　超えていれば　ＧＯ　ＴＯ　００８.
００７２　超えていなければ　ＧＯ　ＴＯ　００３.
００８　パレット３の臨界の成否判定.：Ｔ２の第４領域のオンオフを利用
００８１　オフなら　ＧＯ　ＴＯ　００２.
００８２　オンなら　ＧＯ　ＴＯ　００９.：座標周期５の終了
００９　論理矛盾発症の有無を判定：Ｅ４２の第４領域オンオフの利用.
００９１　オンで、ＳＬＰ起動.：本プログラムの実行停止処理
００９２　オフで、ＧＯ　ＴＯ　０１０.
：本プログラムには論理矛盾は起きていない
０１０　同期関数に戻る為に次の処理を行う
０１０１　パレット３の第７フラグオンのベクトルを全て初期値化する.
０１１　同期関数にＲＥＴＵＲＮ
【０２０４】
（同期関数プログラムの論理結合型模型）
００１　ＳＴＡＲＴ.
００２　スタック初期値化プログラム起動.
００３　本プログラムの終了判定.：Ｅ４１の第４領域のオンオフを利用
００３１　オンなら、ＧＯ　ＴＯ　００４.：本プログラム終了処理

００３２　オフなら、ＧＯ　ＴＯ　００５.：本プログラムの始動
００４　ＳＥＰ起動.：Ｓｙｓｔｍ　Ｅｎｄｉｎｇ　Ｐｒｏｇｒａｍ(本プログラム終了プログラム)
００５　Ｔ３１第４領域のオンオフ判定.：自分の座標関数４の起動
００５１　オンなら.Ｔ３１第４領域をオフにする.
００５２　自分の座標関数４を起動.
００５３　オフならＧＯ　ＴＯ　００６.
００６　Ｔ３２第４領域のオンオフ判定.：最近傍下位の座標関数４を起動
００６１　オンなら.Ｔ３２第４領域をオフにする.
００６２　最近傍下位の座標関数４を起動.
００６３　オフなら、ＧＯ　ＴＯ　００７.：最近傍下位の座標関数４がなければオフ。
００７　Ｔ３３第４領域のオンオフ判定.：最上位の座標関数３の起動
００７１　オンなら.Ｔ３３第４領域をオフにする.
００７２　最上位の座標関数３を起動.
００７３　オフなら、ＧＯ　ＴＯ　００８.：最上位の座標関数３がなければオフ。
００８　Ｔ４第４領域のオンオフ判定.：自分の座標関数２の起動
００８１　オンなら.Ｔ４第４領域をオフにする.
００８２　自分の座標関数２を起動.
００８３　オフなら、ＧＯ　ＴＯ　００９.：自分の座標関数３の起動。
００９　Ｔ２第４領域のオンオフ判定.：自分の座標関数３の起動
００９１　オンなら.Ｔ２第４領域をオフにする.
００９２　自分の座標関数３を起動.
００９３　オフなら、ＧＯ　ＴＯ　０１０.
０１０　Ｔ３２第４領域のオンオフ判定.：隣下の座標関数４の起動
０１０１　オンなら.Ｔ３２第４領域をオフにする.

０１０２　隣下の座標関数４を起動.
０１０３　オフなら、ＧＯ　ＴＯ　０１１.
０１１　Ｔ３３第４領域のオンオフ判定.：最上位の座標関数３の起動
０１１１　オンなら.Ｔ３３第４領域をオフにする.
０１１２　最上位の座標関数３を起動.
０１１３　オフなら、ＧＯ　ＴＯ　００５２.
０１２　ＥＮＤ.
【０２０５】
（Ｅ４１，Ｐ４プログラムのベクトル型模型）
制御ベクトルの論理結合型模型を２例ベクトルに置き換える。他の制御ベクトルも置き換え方は同じである。論理結合型模型の制御ベクトルをベクトル型模型に変換する原型はＬ４．Ａである。このベクトルの命令文はスタッツクを採ることである。スタックを採る命令文は第２規約に置かれる。Ｅ４１，Ｐ４にはＬ３がない。次にプログラム模型を示す。
００１　ＳＴＡＲＴ.
００２　ＮＯＰ.
００３　ＣＡＬＬ　ＶＷＡ（Ｅ４１，Ｐ４）.
００４　第１規約：Ｅ４１，Ｐ４の正統性を用いて第２規約に進むか、ＲＥＴＵＲＮかを判定。
　　＊ベクトルの正統性項参照
００５　第２規約
００５１　ＭＯＶＥ　１　ＴＯ　第２フラグ.
００５２　主語成立数スタックからＮＳ１を採る.
００５３　主語成立数スタックからＮＳ５を採る.
００６　第３規約：（ＮＳ１）＝（ＮＳ５）の判定.
００６１　　ＹＥＳならＧＯＴＯ　第４規約.
００６２　　ＮＯならＧＯＴＯ　第５規約.
００７　第４規約。
００７１　ＭＯＶＥ　０　ＴＯ　Ｅ４１，Ｐ４の第６フラグ.
００７２　ＭＯＶＥ　Ａ　ＴＯ　Ｅ４１，Ｐ４の第４領域.
００７３　ＡＤＤ　１　ＴＯ　主語成立数カウンタ.
００８　ＲＥＴＵＲＮ.
００９　第５規約：Ｅ４１，Ｐ４が同じ座標周期で成立するや否やの判定。
００９１　ＭＯＶＥ　０　ＴＯ　第６フラグ.
００９２　（ＮＳ１＝ＮＳ５）ならＧＯＴＯ　第７規約.
００９３　（ＮＳ１＞ＮＳ５）ならＧＯＴＯ　第６規約.
０１０　第６規約：Ｅ４１，Ｐ４の再起要請の宣言。
０１０１　Ｅ４１，Ｐ４の第４領域を初期値化.
０１０２　ＭＯＶＥ　１　ＴＯ　第６フラグ.
０１０３　ＭＯＶＥ　０　ＴＯ　第２フラグ.
０１０４　ＲＥＴＵＲＮ.
０１１　第７規約：Ｅ４１，Ｐ４の再起停止宣言。
０１１１　ＭＯＶＥ　１　ＴＯ　第７フラグ.
０１１２　ＭＯＶＥ　０　ＴＯ　第２フラグ.
０１１３　ＲＥＴＵＲＮ.
【０２０６】
（Ｔ３３，Ｐ３プログラムのベクトル型模型）
このベクトルの命令文はスタッツクを採ることである。スタッツクを採る命令文は第２規約に置かれる。Ｔ３３，Ｐ３にはＬ３がない。次にプログラム模型を示す。
００１　ＳＴＡＲＴ.
００２　ＮＯＰ.

００３　ＣＡＬＬ　ＶＷＡ（Ｔ３３，Ｐ３）．
００４　第１規約：Ｔ３３，Ｐ３の正統性を用いて第２規約に進むか、ＲＥＴＵＲＮかを判定。
　　＊ベクトルの正統性項参照
００５　第２規約　Ｔ３３，Ｐ３の成立判定の準備：
００５１　ＭＯＶＥ　１　ＴＯ　第２フラグ．
００５２　パレット３の第６フラグ列の作成．
００５３　パレット３の第６フラグ列常数の作成．
００６　第３規約：Ｔ３３，Ｐ３の成立判定。
００６１　Ｐ３の臨界判定。
００６１１　パレット３の第６フラグ列と第６フラグ列常数のＸＯＲ（第１条件）．
００６１２　ＸＯＲがＺＥＲＯならＧＯＴＯ　００６２．
００６１３　ＸＯＲがＮＯＴ　ＺＥＲＯなら第５規約．
００６２　主語分布表（図１３）を用いて行う判定。
００６２１　主語は隣上の本プログラムにも内在するか（第２条件）
００６２２　ＹＥＳならＧＯＴＯ　第４規約．
００６２３　ＮＯならＧＯＴＯ　ＲＥＴＵＲＮ．
００７　第４規約。
００７１　ＭＯＶＥ　０　ＴＯ　Ｔ３３，Ｐ３の第６フラグ．
００７２　ＭＯＶＥ　Ａ　ＴＯ　Ｔ３３，Ｐ３の第４領域．
００７３　ＡＤＤ　１　ＴＯ　主語成立数カウンタ．
００８　ＲＥＴＵＲＮ．
００９　第５規約：Ｐ３の臨界が同じ座標周期で成立するや否やの判定。
００９１　ＭＯＶＥ　０　ＴＯ　第６フラグ．
００９２　（ＮＳ１＞ＮＳ５）ならＧＯＴＯ　第６規約．
００９３　（ＮＳ１＞ＮＳ５）でなければＧＯＴＯ　第７規約．
０１０　第６規約：Ｅ４１，Ｐ４の再起要請の宣言。
０１０１　Ｔ３３，Ｐ３の第４領域を初期値化．
０１０２　ＭＯＶＥ　１　ＴＯ　第６フラグ．
０１０３　ＭＯＶＥ　０　ＴＯ　第２フラグ．
０１０４　ＲＥＴＵＲＮ．
０１１　第７規約：Ｔ３３，Ｐ３の再起停止宣言。
０１１１　ＭＯＶＥ　１　ＴＯ　第７フラグ．
０１１２　ＭＯＶＥ　０　ＴＯ　第２フラグ．
０１１３　ＲＥＴＵＲＮ．
【０２０７】
（スタック更新プログラムの論理結合型模型）
００１　ＳＴＡＲＴ．
００２　主語成立数カウンタがゼロならスタックを初期値化：スタック初期値化プログラムを起動
００３　ＲＥＴＵＲＮ．
００４　主語成立数カウンタがゼロでなく、ＮＳ１、２，３，４ゼロなら
００５　主語成立数カウンタをＮＳ１に上書．
００６　ＲＥＴＵＲＮ．
００７　主語成立数カウンタがゼロでなく、ＮＳ２，３，４ゼロなら
００８　ＮＳ１をＮＳ２に上書．
００９　主語成立数カウンタをＮＳ１に上書．
０１０　ＲＥＴＵＲＮ．
０１１　主語成立数カウンタがゼロでなく、ＮＳ３，４ゼロなら
０１２　ＮＳ１をＮＳ２に上書．

０１３　ＮＳ２をＮＳ３に上書.
０１４　主語成立数カウンタをＮＳ１に上書.
０１５　ＲＥＴＵＲＮ.
０１６　主語成立数カウンタがゼロでなく、ＮＳ４がゼロなら
０１７　ＮＳ１をＮＳ２に上書.
０１８　ＮＳ２をＮＳ３に上書.
０１９　ＮＳ３をＮＳ４に上書.
０２０　主語成立数カウンタをＮＳ１に上書.
０２１　ＲＥＴＵＲＮ.
０２２　主語成立数カウンタがゼロでなく、全ＮＳがゼロでないなら
０２３　ＮＳ１をＮＳ２に上書.
０２４　ＮＳ２をＮＳ３に上書.
０２５　ＮＳ３をＮＳ４に上書.
０２６　ＮＳ４をＮＳ５に上書.
０２７　主語成立数カウンタをＮＳ１に上書.
０２８　ＲＥＴＵＲＮ.

【０２０８】

（スタック初期値化プログラムの論理結合型模型）

本スタック情報を利用する主語ベクトルの第５規約（ＮＳ１＞ＮＳ５）、制御ベクトルＥ４１（ＮＳ１＝ＮＳ５）、Ｅ４２（ＮＳ１＜ＮＳ５）の為に、本プログラムを同期周期の開始点で起動（ＣＡＬＬ）する。次にプログラム模型を示す。

００１　ＳＴＡＲＴ.
００２　ＮＳ５に１をセット.
００３　ＮＳ４をゼロクリア.
００４　ＮＳ３をゼロクリア.
００５　ＮＳ２をゼロクリア.
００６　ＮＳ１をゼロクリア.
００７　主語成立数カウンタをゼロクリア.
００８　ＲＥＴＵＲＮ.

【０２０９】

ウイルスが稼動中の本発明のプログラム（以下本プログラム）に如何なる時宜、如何なる手段、そして、それを何度でも繰り返し、侵入しても、本プログラムはそのウイルスを自力で本プログラムが使用する記憶領域の汚染として捉えて、除染し、本プログラムの正常な稼働を継続できる様に迅速に回復させる。

【０２１０】

本プログラムはウイルスを本プログラムの意図に反する誤情報による汚染として捉える。本プログラムは汚染が出現すればそれを叙述矛盾として捉える。本プログラムにはこの仕組が備わっている。しかし、この仕組は侵入するウイルスを捉える為に在るのではなく、プログラムが正統なプログラムとして存在する為の構造条件として求められている。本プログラムは捉えた汚染を本発明による仕組で除染する。この除染の仕組も正統なプログラムとして存在する為の構造条件として求められている。汚染を捉える時宜、そして、それを除染する時宜は本発明による時宜で行われる。この除染の時宜は侵入するウイルス症状を発症させないことに於いて、ウイルスの意図を解体することと同義の作用になる。本願によってのみ奏される本願特有の効果である。

【０２１１】

上記に詳述したように、本発明によれば、ウイルス問題に対して本質的な解法が与えられる。すなわち、本プログラムは、ウイルスをウイルス問題として解法するわけではない。本プログラムは、ウイルスは存在してもウイルス問題（ウイルスの侵入問題、ウイルス症状問題）を存在させない仕組、即ち、解法の仕組になっている。

【０２１２】

なお、本願に係る技術思想は上述した形態に限定されることはなく、本思想の範囲内で、種々の変形、置換、代替、改良、拡大、縮小が可能である。また、本発明に係るプログラム、機能チップ、装置が搭載された二次的複合製品に対しても本願に係る技術思想は及ぶものである。さらに、本発明は、上記に詳述された本発明独自の構造である、ウイルス問題を自律的に解法するプログラムの定義構造としても、同構造を備えたプログラムとしても、同プログラムが搭載される記憶媒体としても、並びにウイルス問題を自律的に解法する方法としても、（たとえばこれらのプログラムと同等の機能を果たす部品として組み込んだ）ウイルス問題を自律的に解法するウイルス問題解法装置としても、それぞれ実現することが可能である。すなわち、図１４は本発明の一実施形態に係る本プログラムが搭載され得る一形態を示した全体構造図であるが、ここに示されるように、本願に係るベクトルＬ４構造を備えたルーチン１４３１、本願に係るベクトルＬ２構造を備えたルーチン１４４１、本願に係るベクトルＬ３構造を備えたルーチン１４５１、本願に係るルーチン１４３１を集積させたＬ４パレット構造を備えたルーチン１４３０、本願に係るルーチン１４４１を集積させたＬ２パレット構造を備えたルーチン１４４０、本願に係るルーチン１４５１を集積させたＬ３パレット構造を備えたルーチン１４５０、のこれらすべて（他のＲ２、Ｗ４等のベクトルについては記載を省略しているが含まれることは当然である）を全体構造プログラムとしてシステム１４００に組み込む形態であっても、これらを単独プログラムとしていわゆるＡＳＰ（アプリケーション・サービス・プロバイダ）形式として提供する形態であっても、或いはこれらの機能をそれぞれ部品化した装置として提供する形態であっても、これら各プログラムをプログラム単体として提供する形態であっても、当該プログラムを搭載した記憶媒体として提供する形態であっても、いずれでも本願は実現・適用することができる。これらのあらゆるメカニズムに対して、上述した本技術思想の中核部分は適用可能である。

【産業上の利用可能性】

【０２１３】

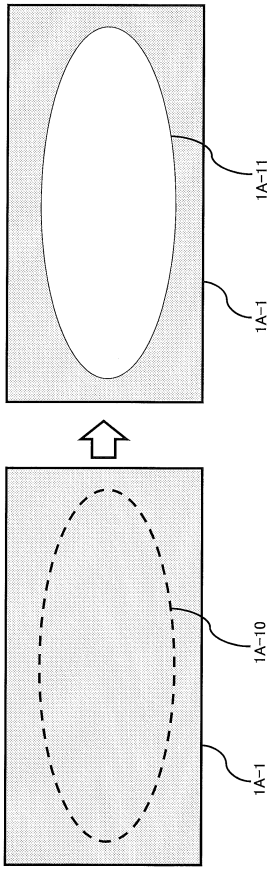
既に、コンピュータ・プログラムは人類の、あらゆる分野に普及している。故に、其のプログラム達は、既に脅かされている分野もある様に、そして、今後は全ての分野のプログラムが例外なく、且つ止まることなく、ウイルス問題に脅かされることになる。本発明は、ウイルス問題を一挙に解法することに於いて人類に貢献する。そして、本発明は他に見当たらないことに於いて本発明は、コンピュータ産業にとどまらず、自動車、航空機、原子力、一般家電を含む各種産業において巨大な利用可能性を有するものである。

【符号の説明】

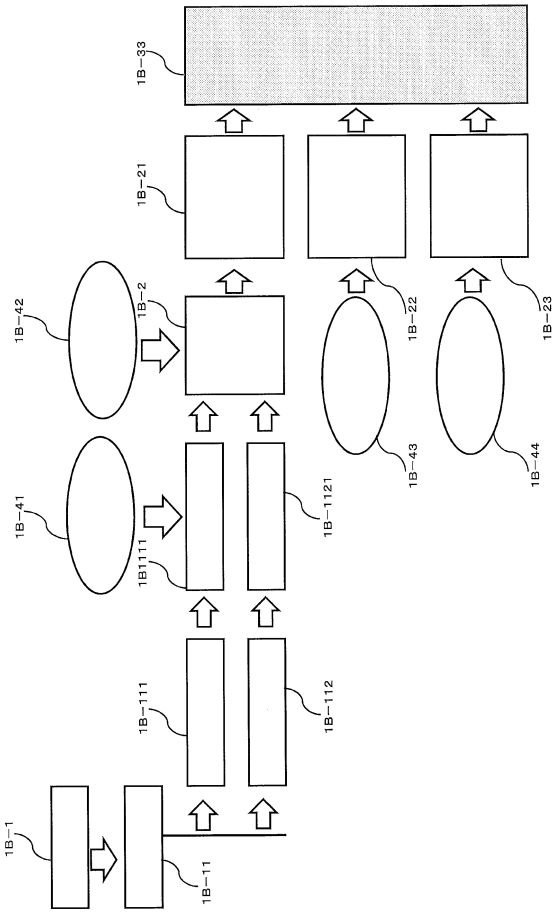
【０２１４】

１Ａ－１…電算機の記憶領域、１Ａ－１０…従来のプログラム、１Ａ－１１…本発明のプログラム、１Ｃ－１…電算機の記憶領域、１Ｃ－１０…従来のプログラム、１Ｃ－１１…本発明のプログラム，１Ｃ－２…コンピュータウイルス

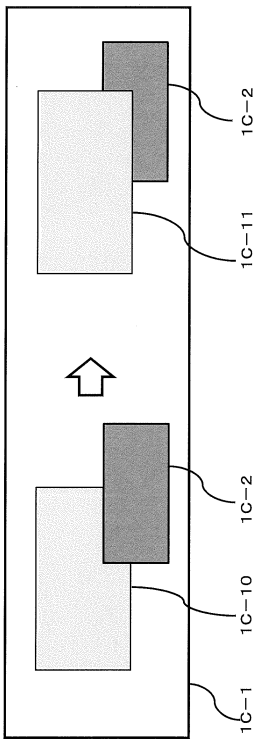
【図１Ａ】



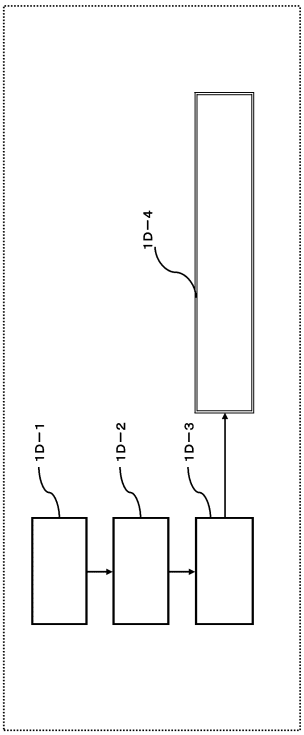
【図１Ｂ】



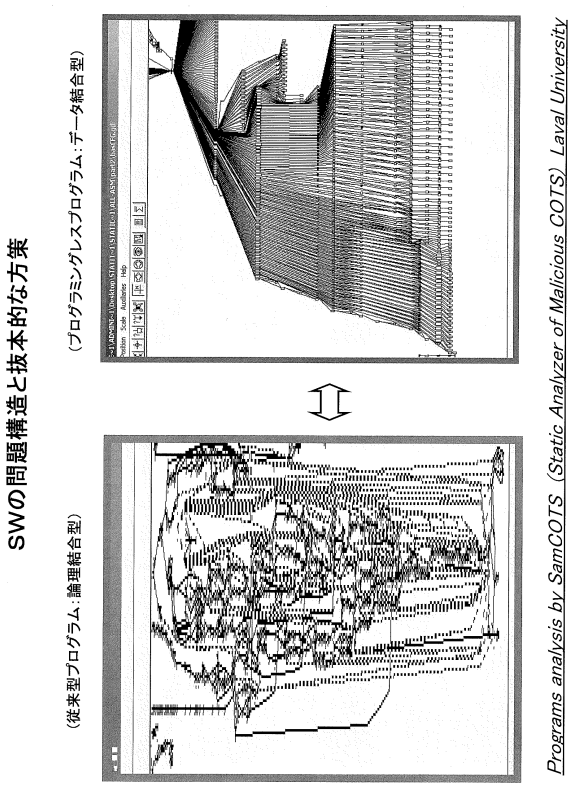
【図１Ｃ】



【図１Ｄ】



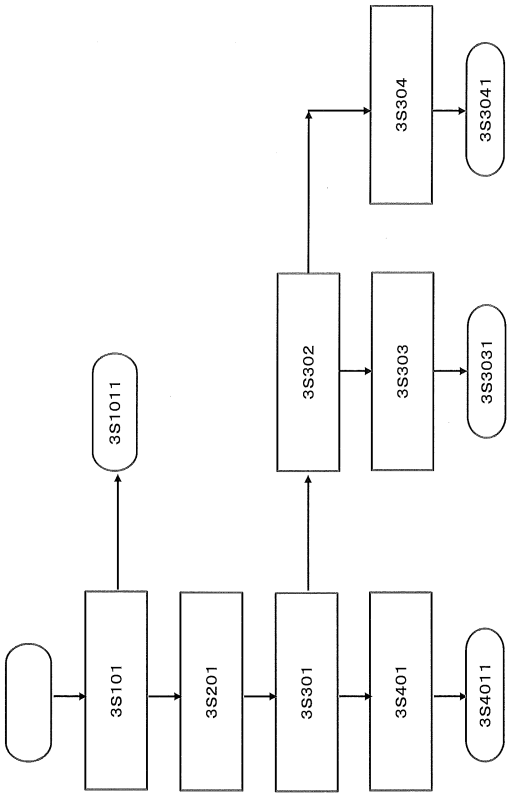
【図１Ｅ】



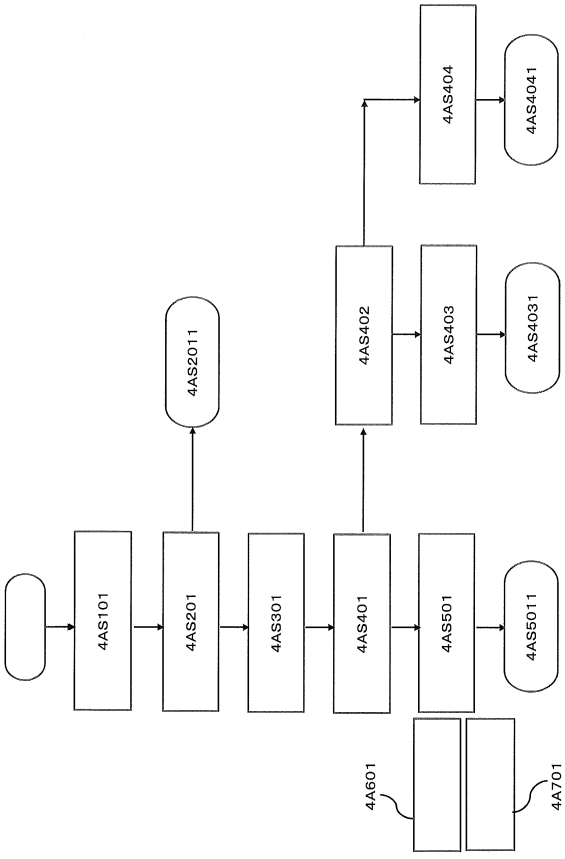
【図２】

No	ベクトル種別	搭載パレット種別	緩治する命令文例	命令文の位置
1	L4, 名詞	P4	代入文	第2緩約
2	L2, 名詞	P2	定義文	第2緩約
3	L3, 名詞	P3	条件文	第3緩約
4	W4, 名詞	P4	出力文	第4緩約
5	R2, 名詞	P2	入力文	第2緩約
6	E41, P4	P4	なし	—
7	E42, P4	P4	なし	—
8	T4, P4	P4	なし	—
9	T2, P2	P2	なし	—
10	T31, P3	P3	なし	—
11	T32, P3	P3	なし	—
12	T33, P3	P3	なし	—

【図３】



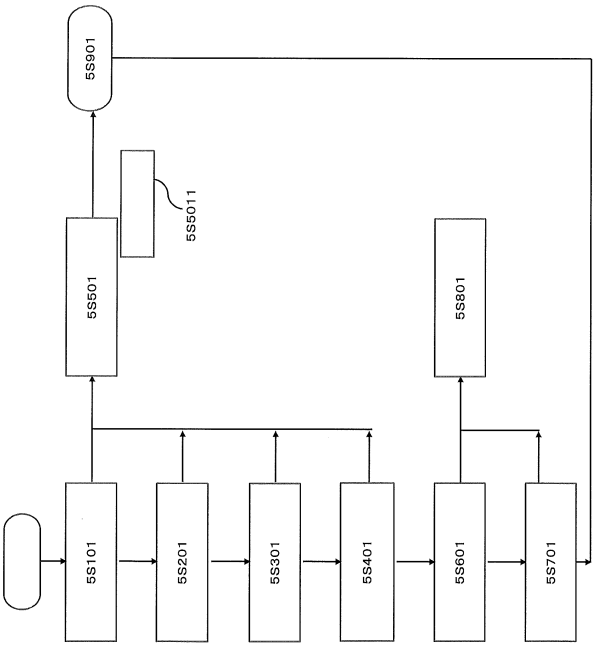
【図４Ａ】



【図４Ｂ】

第4領域	第2フラグ	第6フラグ	第7フラグ	ベクトル
オン	オン	オフ	オフ	正統主語
オフ	オフ	オン	オフ	再起要請
オフ	オフ	オフ	オン	再起停止要請

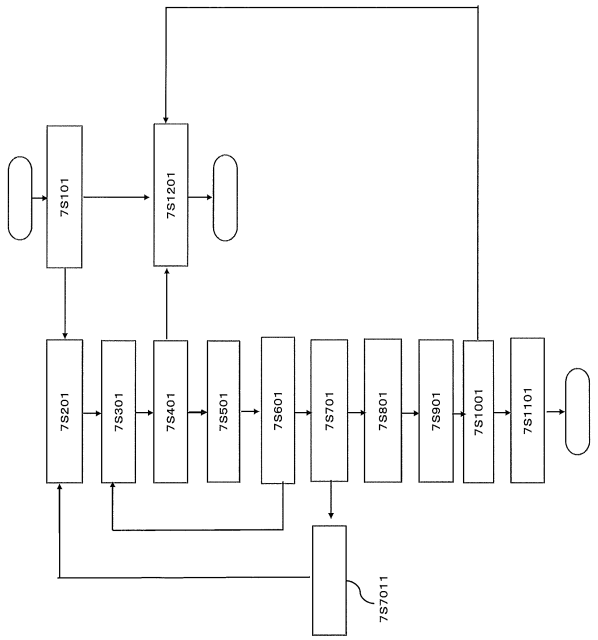
【図５】



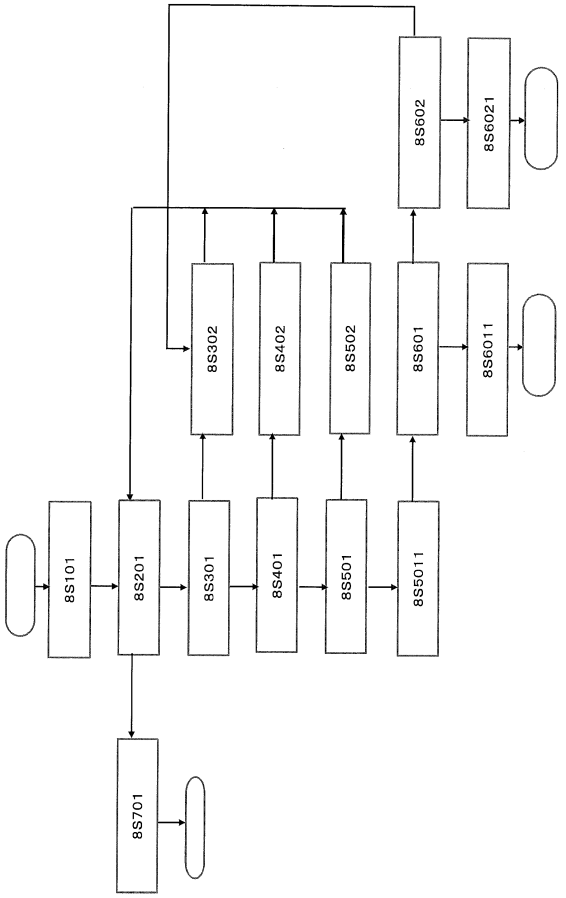
【図６】

主語成立数カウンタの値	スタック順位
最古の主語成立数	NS5
主語成立数	NS4
主語成立数	NS3
主語成立数	NS2
最新の主語成立数	NS1

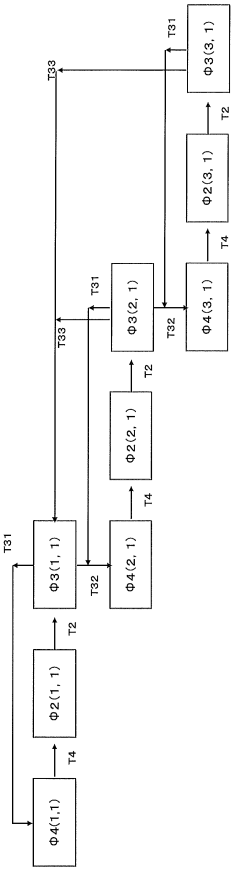
【図７】



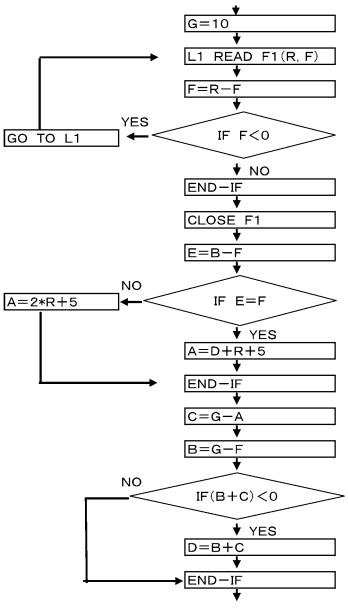
【図８】



【図９】



【図１０】



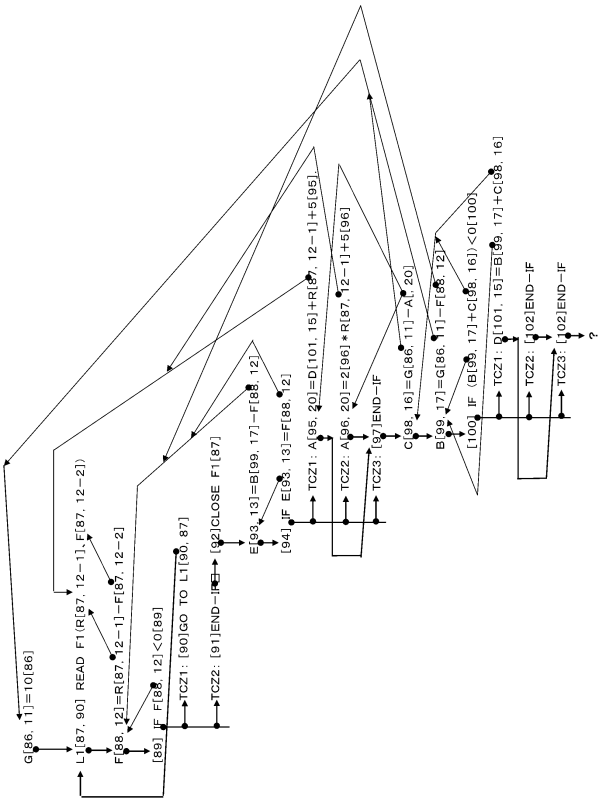
【図１１Ａ】

プログラム	文種	ベクトル種別	主語	連動文	行No	調和座標 TCX	TCY	TCZ1	TCZ2	TCZ3	TCZ4
G=10	定義	L2	G		86	86	87				
L1 READ F1(R, F)	入力	R2	F1(R, F)		87	87	88				
F=R−F	代入	L4	F	●	88	88	89				
IF F<0	条件	L3		●	89	89		90	91	91	92
GO TO L1	制御			●	90	90		87			
END−IF	翻訳			●	91	91	92				
CLOSE F1	翻訳	L4	F1	●	92	92	93				
E=B−F	代入	L4	E	●	93	93	94				
IF E=F	条件	L3		●	94	94		95	96	97	98
A=D+R+5	代入	L4	A		95	95	97				
A=2*R+5	代入	L4	A		96	96	97				
END−IF	翻訳			●	97	97	98				
C=G−A	代入	L4	C		98	98	99				
B=G−F	代入	L4	B		99	99	100				
IF(B+C)<0	条件	L3		●	100	100		101	102	102	?
D=B+C	代入	L4	D		101	101	102				
END−IF	翻訳			●	102	102	?				

【図１１Ｂ】

TCZ4	超言語化された単元文	超言語座標 主語	変数主語	指標	定値	定置
	G[86,11]=10[86]	G[86,11]			10[86]	
	L1[87,90]READ F1(R[87,12−1],F[87,12−2])	F1(R[87,12−1],F[87,12−2])		L1[87,90]		
	F[88,12]=R[87,12−1]−F[87,12−2]	F[88, 12]	R[87,12−1], F[87,12−2]			
92	[89,]IF F[88,12]<O[89]		F[88,12]		O[89]	
	[90,]GO TO L1[90,87]			L1[90,87]		
	[91,]END−IF					
	[92,]CLOSE F1[87]	F1[87]				
	E[93,13]=B[99,17]−F[88,12]	E[93,13]	B[99,17], F[88,12]			
98	[94,]IF E[93,13]=F[88,12]		E[93,13], F[88,12]			
	A[95,20]=D[101,15]+R[87,12−1]+5[95]	A[95,20]	D[101,15], R[87,12−1]		5[95]	
	A[95,20]=2[96]*R[87,12−1]+5[96]	A[96,20]	R[87,12−1]		2[96], 5[96]	
	[97,]END−IF					
	C[98,16]=G[86,11]−A[20]	C[98,16]	G[86,11], A[20]			
	B[99,17]=G[86,11]−F[88,12]	B[99,17]	G[86,11], F[88,12]			
?	[100,]IF (B[99,17]+C[98,16])<O[100]		B[99,17], C[98,16]		O[100]	
	D[101,15]=B[99,17]+C[98,16]	D[101,15]	B[99,17], C[98,16]			
	[102,]END−IF					

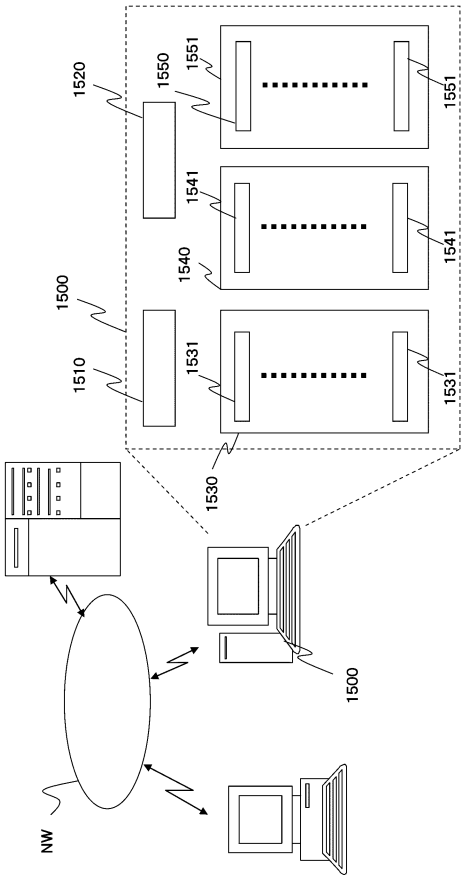
【図１２】



【図１３】

主語No	主語	変数主語	変数主語の主語が属す本プログラム
1	A(1, 1)	B(1, 1)	(1, 1)
		C(1, 1)	(1, 1)
2			

【図１４】



フロントページの続き

(56)参考文献 特開２０１３－１６４７３２（ＪＰ，Ａ）
特開２００３－１０８２５３（ＪＰ，Ａ）
国際公開第２００５／０２９３２２（WO，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F21/56