

(19) 日本国特許庁(JP)

(12) 特 許 公 報(B2)

(11) 特許番号

特許第6086977号
(P6086977)

(45) 発行日 平成29年3月1日(2017.3.1)

(24) 登録日 平成29年2月10日(2017.2.10)

(51) Int.Cl. F 1
G 0 6 F 2 1 / 5 6 (2 0 1 3 . 0 1) G 0 6 F 2 1 / 5 6 3 8 0

請求項の数 1 (全 53 頁)

<p>(21) 出願番号 特願2015-514699 (P2015-514699)</p> <p>(86) (22) 出願日 平成27年3月3日(2015.3.3)</p> <p>(86) 国際出願番号 PCT/JP2015/056268</p> <p>(87) 国際公開番号 W02016/021220</p> <p>(87) 国際公開日 平成28年2月11日(2016.2.11)</p> <p>審査請求日 平成27年3月19日(2015.3.19)</p> <p>審判番号 不服2016-8437 (P2016-8437/J1)</p> <p>審判請求日 平成28年6月8日(2016.6.8)</p> <p>(31) 優先権主張番号 特願2014-158612 (P2014-158612)</p> <p>(32) 優先日 平成26年8月4日(2014.8.4)</p> <p>(33) 優先権主張国 日本国(JP)</p> <p>早期審理対象出願</p>	<p>(73) 特許権者 598037422 根来 文生 神奈川県鎌倉市山ノ内591番地</p> <p>(74) 代理人 100110559 弁理士 友野 英三</p> <p>(72) 発明者 根来 文生 神奈川県鎌倉市山ノ内591番地</p> <p>合議体 審判長 高木 進 審判官 辻本 泰隆 審判官 須田 勝巳</p>
--	--

最終頁に続く

(54) 【発明の名称】 本来の業務処理を正統な主語の脈略として成立させるための手順をコンピュータに実行させるシナリオ関数として規定されるプログラム

(57) 【特許請求の範囲】

【請求項1】

本来の業務処理を正統な主語の脈略として成立させるための手順をコンピュータに実行させるシナリオ関数Sとして規定されるプログラムであって、

- ・ 主語：領域の内容；
- ・ 0（同期関数）：3種の座標関数と指標されるプログラム 4、 2、 3を統治し、パレット4が臨界状態になればパレット2に、前記パレット2が臨界状態になればパレット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、同一ランクに係る座標関数4、1層下層ランクに係る座標関数4のいずれかに制御を移す制御論理体であるプログラム；
- ・ 4（座標関数4）：L4ベクトル、W4ベクトル、E41ベクトル、E42ベクトル、T4ベクトルを統治しパレット4内の主語の同期を成立させるべく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前記パレット4の臨界状態が達成されるまで循環する構造を有するプログラム；
- ・ 2（座標関数2）：L2ベクトル、R2ベクトル、T2ベクトルを統治しパレット2内の主語の同期を成立させるべく前記ベクトル構造が任意順序で集積された前記パレット2の臨界状態が達成されるまで循環する構造を有するプログラム；
- ・ 3（座標関数3）：L3ベクトル、T31ベクトルを統治しパレット3内の主語の同期を成立させるべく前記ベクトル構造が任意順序で集積された前記パレット3の臨界状態が達成されるまで循環する構造を有するプログラム；

10

20

- ・ L 4 (L 4 ベクトル) : 代入文、出力文、呼び出し文において名詞を主語化するための前記ベクトル構造に係るプログラム ;
- ・ W 4 (W 4 ベクトル) : 正統な主語を外部記憶領域に出力する前記ベクトル構造に係るプログラム ;
- ・ E 4 1 (E 4 1 ベクトル) : シナリオ関数 S が終了状態にあることを同期関数 0 に告知するための前記ベクトル構造に係るプログラム ;
- ・ E 4 2 (E 4 2 ベクトル) : シナリオ関数 S に生じる論理矛盾の発症を座標関数 4 に告知するための前記ベクトル構造に係るプログラム ;
- ・ T 4 (T 4 ベクトル) : 4 (座標関数 4) を 2 (座標関数 2) へ切り替える条件が整ったことを同期関数 0 に告知するための前記ベクトル構造に係るプログラム ;
- ・ L 2 (L 2 ベクトル) : 定値文、入力文において最小クラスの名詞を主語化するための前記ベクトル構造に係るプログラム ;
- ・ R 2 (R 2 ベクトル) : 外部記憶領域から READ するための前記ベクトル構造に係るプログラム ;
- ・ T 2 (T 2 ベクトル) : 2 (座標関数 2) を 3 (座標関数 3) へ切り替える条件が整ったことを同期関数 0 に告知するための前記ベクトル構造に係るプログラム ;
- ・ L 3 (L 3 ベクトル) : 条件文において名詞を主語化するための前記ベクトル構造に係るプログラム ;
- ・ T 3 1 (T 3 1 ベクトル) : 3 (座標関数 3) を 4 (座標関数 4) へ切り替える条件が整っていることを同期関数 0 に告知するための前記ベクトル構造に係るプログラム

10

20

、
としたとき、

$$S = 0 (4 (\{ L 4 \}、\{ W 4 \}、E 4 1、E 4 2、T 4) + 2 (\{ L 2 \}、\{ R 2 \}、T 2) + 3 (\{ L 3 \}、T 3 1、T 3 2、T 3 3))$$

として規定されるプログラム構造を有し、

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、

30

前記第 1 規約では前記ベクトルの 正統性 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、

40

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 L 4 (L 4 ベクトル)、前記 L 2 (L 2 ベクトル)、前記 L 3 (L 3 ベクトル)、前記 W 4 (W 4 ベクトル)、前記 R 2 (R 2 ベクトル) のそれぞれは、各ベクトルの正統性を捉えるための前記第 4 領域、前記第 2 フラグ、前記第 6 フラグ、前記第 7 フラグの状

50

態を用いてXOR演算をするプログラムであるウイルス観察アルゴリズムプログラムを有する、

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。

10

【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。

20

【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。

30

【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。

40

【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

50

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】

10

【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決の絶対条件である。

【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し遂げられない。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】

20

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

30

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。

40

【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、ひらめきの自覚することがあっても、正確にはその

50

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、自覚関数とその模型であると結論付けられる。

10

【0019】

調和構造の最小単位は存在論の實在に対応付けられている。實在は実体と属性とで定義される。シナリオ関数では實在の識別子は最小クラスの名詞、實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】

20

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

30

(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。

40

(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

50

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ぬ、ということが論理的帰結である。

【0026】

（C．本願独自の的方法論の構築）

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働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働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。

10

20

30

40

50

- (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

【 0 0 3 3 】

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。命令文がどの様子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。

10

【 0 0 3 4 】

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せる前段の時宜で、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。

20

【 0 0 3 5 】

図 1 D 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中、2009年に解法の仕組みが捉えられた。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。

30

【 0 0 3 6 】

(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。

40

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)

【 0 0 3 7 】

(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

50

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)

【 0 0 3 8 】

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。

10

【 0 0 3 9 】

本プログラムでは命令コードの汚染はOSとベクトルE 4 2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。E 4 2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、E 4 2よりも優れた時宜と簡便さで解法することもできる。E 4 2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。

20

【 0 0 4 0 】

(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)で捉え自動回復させる。上記(3) (4)の汚染はE 4 2(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。

30

【 0 0 4 1 】

(本発明の要点)

図1 A , 1 C に、本発明の要約を示す。図1 A において、1 A - 1 は電算機の記憶領域、1 A - 1 0 は従来のプログラム、1 A - 1 1 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1 C において、1 C - 1 は電算機の記憶領域、1 C - 1 0 は従来のプログラム、1 C - 1 1 は本発明のプログラム、1 C - 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。

40

【 0 0 4 2 】

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) 稼働中の本プログラムはウイルスの侵入を許容する。

50

(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 Program)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、発症させない仕組みになっている。

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みが求められれば、ウイルス問題を10
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見出したことにもなる。

【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び、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ている20
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30
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つ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40
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ッサンス的変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。

10

【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。

20

【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。

30

【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。

40

【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

50

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

10

20

30

40

50

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、同一ランクに係る座標関数4、1層下層ランクに係る座標関数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

10

【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せさせて、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。

20

【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。

30

【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とすることができる。

40

【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】

50

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。

10

【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、ウイルスが稼働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みになっている。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、本プログラムの生じる本発明による叙述矛盾の仕組みで捉える。そして、本プログラムはその汚染を本発明による仕組みで除染する。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。

20

【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みである。故に、本プログラムのこの仕組みはウイルス問題の解法ということになるものである。

30

【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。

40

【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。

50

【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L y e e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。

10

【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L y e e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みを表した図である。

【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】

20

【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。

30

【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。

40

【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】

(意味の仕組み)

意味の仕組みはシナリオ関数の解(S)である。意味の仕組みとはわれわれの脳裏の中に存在

50

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LYEE空間は図11A、図11Bを参照されたい。

10

【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ぶかの論考、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。

20

【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。

30

【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。主語がどのよう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。

40

【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。

50

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は単元文が条件文で真のとき、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。

10

20

30

40

50

故に、この場合、意味の仕組では $[100、] I F (B [99,17] + C [98,16]) < 0$ 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。 $[100、] I F$ は $T C X : 1 0 0$ に置かれていることを示す。 $I F$ は主語ではないので、その領域は存在しないことから、 $[100、] I F$ と記される。

【 0 1 0 1 】

(意味の仕組のガイダンス)

以上のように意味の仕組、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。

10

【 0 1 0 2 】

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。

【 0 1 0 3 】

意味の仕組の利用価値は様々考えられる。

20

(1) 例えばひとつの主語(名詞)が他の主語とどのよう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。

30

【 0 1 0 4 】

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)。

【 0 1 0 5 】

40

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。

【 0 1 0 6 】

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

50

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を作成することを示す。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-33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。

10

20

【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。

30

(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。

40

【0109】

ベクトルは命令文を決める為の仕組み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】

50

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。

【 0 1 1 1 】

(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は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、

$$S = 0 (4 (\{ L 4 \} , \{ W 4 \} , E 4 1 , E 4 2 , T 4) + 2 (\{ L 2 \} , \{ R 2 \} , T 2) + 3 (\{ L 3 \} , T 3 1 , T 3 2 , T 3 3))$$

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。

【 0 1 1 2 】

(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 1 2 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。

【 0 1 1 3 】

(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。3 S 1 0 1 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。3 S 1 0 1 1 は第 1 規約の出口である。3 S 2 0 1 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。3 S 3 0 1 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。3 S 4 0 1 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。3 S 4 0 1 1 はベクトルに於ける本来の処理の完了による出口である。3 S 3 0 2 は第 5 規約で、第 3 規約の否で行われる処理の場である。3 S 3 0 3 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。3 S 3 0 3 1 は第 6 規約の出口である。3 S 3 0 4 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。3 S 3 0 4 1 は第 7 規約の出口である。

【 0 1 1 4 】

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 ケ所で第 1、第 4、第 6、第 7 規約である。4 ケ所の終点 (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)。

【 0 1 1 5 】

(第 4 領域の正統性)

10

20

30

40

50

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。

10

【0117】

(ベクトル区分)

ベクトルは以下の3種に区分できる。

20

(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。

30

【0119】

(単元文)

どのよう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。

40

【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

50

関係がない。

【 0 1 2 1 】

(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 ; : L 4 , 名詞。

【 0 1 2 2 】

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、L 3 , 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、L 3 , 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。

10

【 0 1 2 3 】

E 4 1、E 4 2、T 4、T 2、T 3 1、T 3 2、T 3 3 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 ; E 4 1 , P 4 という形式になる。

L 4 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 L 4 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。

【 0 1 2 4 】

(L 4 型制御ベクトルの補足)

L 4 型制御ベクトルに複数の命令文が属する場合、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る場合、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、L 4 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文を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 L 4 型制御ベクトルは成立していないことになる。

20

【 0 1 2 5 】

(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 L 4 型制御ベクトルで統治される。

30

【 0 1 2 6 】

(制御ベクトルの補足)

E 4 1 は本プログラムの終了状態を捉える。E 4 2 は論理矛盾を捉える。T 4 は座標関数 4 を 2 に切り替える条件を捉える。T 2 は座標関数 2 を 3 に切り替える条件を捉える。T 3 1 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。T 3 2 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。T 3 3 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。

40

【 0 1 2 7 】

(ベクトル記号)

ベクトル種別 L 4、W 4、E 4 1、E 4 2、T 4、L 2、R 2、T 2、L 3、T 3 1、T 3 2、T 3 3 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 」は未来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。

50

【 0 1 2 8 】

(L 3 ベクトルの補足)

L 3 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、L 3 は主語を持たない構文を制している。それ故、L 3 に呼応する主語ベクトルはそのL 3 に自分の主語を写す。L 3 はその主語を得て識別を成立させることになる。

【 0 1 2 9 】

(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) { { L 4 } , { W 4 } , E 4 1 , E 4 2 , T 4 } はパレット 4 、 P 4 と記される。 10

(2) { { L 2 } , { R 2 } , T 2 } はパレット 2 、 P 2 と記される。

(3) { { L 3 } , T 3 1 , T 3 2 , T 3 3 } はパレット 3 、 P 3 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「未来」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。

【 0 1 3 0 】

(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み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みが導出される構造になっている(意味の仕組みの項を参照)。 20

【 0 1 3 1 】

(同期関数)

図 8 (同期関数 (0) 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、8 S 1 0 1 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。8 S 2 0 1 では制御ベクトル E 4 1 の成否が調べられる。E 4 1 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。E 4 1 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 (S E P) を起動する。E 4 1 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。8 S 3 0 1 では制御ベクトル T 3 1 の成否を調べる。T 3 1 が成立していれば、8 S 3 0 2 で T 3 1 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。T 3 1 が成立していなければ 8 S 4 0 1 で制御ベクトル T 4 の成否を調べる。T 4 が成立していれば、8 S 4 0 2 で T 4 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。T 4 が成立していなければ、8 S 5 0 1 で制御ベクトル T 2 の成否を調べる。T 2 が成立していれば、8 S 5 0 2 で T 2 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。T 2 が成立していなければ、8 S 6 0 1 で制御ベクトル T 3 2 の成否を調べる。T 3 2 が成立していれば、8 S 6 0 1 1 で T 3 2 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。T 3 2 が成立していなければ、8 S 6 0 2 で制御ベクトル T 3 3 の成否を調べる。T 3 3 が成立していれば、8 S 6 0 2 1 で T 3 3 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。T 3 3 が成立していなければ、8 S 6 0 3 で自分の座標関数 4 を起動する。 40

【 0 1 3 2 】

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 50

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(4, 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、座標関数で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ため、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7S101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(4, 2, 3)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。

10

【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内容は似て非なるものだという事である。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内容は、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。本プログラムのモデルを参照すれば、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られる。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。

20

【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個でもプログラムで動作させてはならない。これは「ウイルス問題解決の絶対条件」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。

30

【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。

40

【0141】

（ウイルス問題解決の公理）

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も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

50

らないということも自明であって、問題がある。

【 0 1 4 2 】

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。

10

【 0 1 4 3 】

(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の意味している。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。

【 0 1 4 4 】

(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 4 A に置き換えられる。図 4 A 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 4 A において、4 A S 1 0 1 はベクトルに添付されるウイルス観察アルゴリズム (V W A) である。4 A S 2 0 1 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 4 B (後述する) を参照されたい。4 A S 2 0 1 1 は第 1 規約の出口である。4 A S 3 0 1 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。4 A S 4 0 1 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。4 A S 5 0 1 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。4 A S 5 0 1 1 は第 4 規約の出口である。4 A S 4 0 2 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。4 A S 4 0 3 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。4 A S 4 0 3 1 は第 6 規約の出口である。4 A S 4 0 4 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。4 A S 4 0 4 1 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。4 A S 6 0 1 では主語成立数カウンタに 1 が加算される。4 A S 7 0 1 では再起カウンタの値が保持される。

20

30

40

【 0 1 4 5 】

図 4 B 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。

【 0 1 4 6 】

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、5 S 1 0 1 は第 2 フラグの汚染の有無が観察される。5 S 2 0 1 は第 6 フラグの汚染の有無が観察される。5 S 3 0 1 は第 7 フラグの汚染の有無が観察される。5 S 4 0 1 は第 4 領域の汚染の有無が観察される。5 S 5 0 1 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。5 S 5 0 1 1 は主語成立数カウンタを 1 減じる。5 S 9 0

50

1 は本 VWA の終了出口である。5 S 6 0 1 は主語成立数カウンタの汚染の有無が観察される。5 S 7 0 1 は再起カウンタの汚染の有無が観察される。5 S 8 0 1 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 (S L P) を起動させる。

【 0 1 4 7 】

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。

【 0 1 4 8 】

(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。

【 0 1 4 9 】

(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 4 B 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。

【 0 1 5 0 】

(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来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。

【 0 1 5 1 】

(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みを用いて正統な第 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 k n で成立している。その第 4 領域は第 4 規約で第 4 領域 k n として保持される。VWA は実行の都度、第 4 領域 k n と実行時の第 4 領域を XOR し、同じであれば、第 4 領域 k n 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

10

20

30

40

50

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みでみる事ができる様に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。

(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みが汚染を捉え自動回復させる仕組みになっている。本プログラムのこの様な仕組み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で見え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。

10

20

30

40

50

【 0 1 5 5 】

(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、従来型ソフトウェ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。意味の仕組の用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。

10

【 0 1 5 6 】

(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、仕事に

20

【 0 1 5 7 】

(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

30

- (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

【 0 1 5 8 】

(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絡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。

40

【 0 1 5 9 】

(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

【 0 1 6 0 】

(ウイルスを無力化するととは)

ウイルスを無力化するととは以下の措置を講じることである。

50

- (1) データ領域の初期値化
- (2) プログラムの実行停止

【 0 1 6 1 】

(ウイルスを無力化する為の仕組)

本プログラムで使用される 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 10
- (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 20

【 0 1 6 2 】

(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。

【 0 1 6 3 】

(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 L Y E E 空間 (図 1 1 A、図 1 1 B) を生成するアルゴリズム、並びに、L Y E E 空間情報から本プログラムを生成するアルゴリズム (図 1 B) 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、 $4(1, 1)$ はランク構造 $(1, 1)$ の座標関数 4、 $2(1, 1)$ はランク構造 $(1, 1)$ の座標関数 2、 $3(1, 1)$ 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 $T_{4, 2, 31}$ は同じランクの本プログラムの座標関数 4, 2, 3 を接続する役割を果たす。制御ベクトル $T_{32, 33}$ は異なるランクの座標関数の座標関数を接続する役割を果たす。ランク構造のシナリオ関数はベクトル T_{32} 、 T_{33} で接続される。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。 30

【 0 1 6 4 】

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。 40

【 0 1 6 5 】

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 1 3 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。

【 0 1 6 6 】

(論理結合型プログラムとシナリオ関数の関係)

図 1 B 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。

10

【 0 1 6 7 】

(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。

【 0 1 6 8 】

(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) スパゲティ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。従来の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。

20

30

【 0 1 6 9 】

(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。

40

【 0 1 7 0 】

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 (T 4、T 2、T 3 1、T 3 2、T 3 3)、D . 2 種の E 型ベクトル (

50

E 4 1、E 4 2)、で制御されている。従来のプログラムが不可避免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。

【 0 1 7 1 】

(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みであるが、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ィ構造として複雑化させている原因になっている。図 1 2 に示される意味の仕組み、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み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られる。因みに、同期構造とはプログラムの意味の仕組み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。

【 0 1 7 2 】

(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)を同期させる仕組みで成立すると仮説し、その仕組み、即ち、解を求める仕組み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、意味の仕組み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 1 2 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。

【 0 1 7 3 】

(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。

【 0 1 7 4 】

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、を判定する未来条件は座標関数 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。この仕組みは従来プログラムの発想では求めら

10

20

30

40

50

れるものではない。

【 0 1 7 5 】

(主語の脈略を成立させる仕組)

図 1 2 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ではできても、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より成立するものである。

10

【 0 1 7 6 】

(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。

【 0 1 7 7 】

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はない。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ある。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。

20

30

【 0 1 7 8 】

(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。

40

【 0 1 7 9 】

(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

50

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は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。

10

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T31、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。

20

【0183】

(1) 従来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。

30

(4) 従来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みが必要である。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、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、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。

40

【0185】

(本プログラムの模型)

50

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ドランスとすれば、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ィ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】

10

20

30

40

50

(L 4 , A のベクトル模型)

このベクトルの命令文例を「 A = B + C + 5 1 2 」とする。 L 4 , 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 L 4 , A には L 3 , A がいないものとする。 L 4 , A に L 3 , A がいれば本第 3 規約には L 3 , 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。

```

0 0 1  S T A R T .
0 0 2  N O P .
0 0 3  C A L L  V W A ( L 4 , A ) .
0 0 4  第 1 規約 : L 4 , A の正統性を用いて第 2 規約に進むか、 R E T U R N かを判定 10
。
      * ベクトルの正統性項参照
0 0 5  第 2 規約。
0 0 5 1  M O V E  1  T O  第 2 フラグ .
0 0 5 2  A = B + C + 5 1 2 .
0 0 6  第 3 規約 : A の正統性 ( 第 4 領域の正統性 ) の判定。
0 0 6 1  L 4 , B の第 4 領域は正統か .
0 0 6 1 1  正統なら G O T O  0 0 6 2 .
0 0 6 1 2  正統でなければ G O T O  第 5 規約 .
0 0 6 2  L 4 , C の第 4 領域は正統か . 20
0 0 6 2 1  正統なら G O T O  第 4 規約 .
0 0 6 2 2  正統でなければ G O T O  第 5 規約 .
0 0 7  第 4 規約。
0 0 7 1  M O V E  0  T O  L 4 , A の第 6 フラグ .
0 0 7 2  M O V E  A  T O  L 4 , A の第 4 領域 .
0 0 7 3  A D D  1  T O  主語成立数カウンタ .
0 0 8  R E T U R N .
0 0 9  第 5 規約 : L 4 , A の A が同じ座標周期で成立するや否やの判定。
0 0 9 1  M O V E  0  T O  第 6 フラグ .
0 0 9 2  ( N S 1 = N S 5 ) なら第 7 規約に進む . 30
0 0 9 3  ( N S 1 > N S 5 ) なら第 6 規約に進む .
0 1 0  第 6 規約 : L 4 , A の再起要請の宣言。
0 1 0 1  L 4 , A の第 4 領域を初期値化 .
0 1 0 2  M O V E  1  T O  第 6 フラグ .
0 1 0 3  M O V E  0  T O  第 2 フラグ .
0 1 0 4  R E T U R N .
0 1 1  第 7 規約 : L 4 , A の再起停止宣言。
0 1 1 1  M O V E  1  T O  第 7 フラグ .
0 1 1 2  M O V E  0  T O  第 2 フラグ .
0 1 1 3  R E T U R N . 40
【 0 1 8 9 】

```

(L 3 , D のベクトル模型)

このベクトルの命令文例を「 I F X < Y (真) 」とする。 L 3 , 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。

```

0 0 1  S T A R T .
0 0 2  N O P .
0 0 3  C A L L  V W A ( L 3 , D ) .
0 0 4  第 1 規約 : L 3 , D の正統性を用いて第 2 規約に進むか、 R E T U R N 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。 10
 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。 20
 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。 30
 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。 50

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) 10
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. 20
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。 30
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。 40
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。 50

0 1 1 3 RETURN .

【0 1 9 2】

(R 2 DB 1 1 のベクトル模型)

このベクトルの命令文例を「READ DB 1 1 , 集計」とする。これは外部記憶領域「DB 1 1」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。R 2 DB 1 1 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。R 2 DB 1 1 の VWA , 第 1、第 3 規約では L 4 , 集計に置き換えて処理される。この R 2 DB 1 1 には L 3 , URIAGE がいないものとする。R 2 DB 1 1 に L 3 , DB 1 1 がいれば本第 3 規約には L 3 , DB 1 1 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。

10

0 0 1 START .

0 0 2 NOP .

0 0 3 CALL VWA (R 2 DB 1 1) .

0 0 4 第 1 規約 : L 4 , 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

0 0 5 第 2 規約。

0 0 5 1 MOVE 1 TO 第 2 フラグ .

0 0 5 2 READ DB 1 1 , 集計 .

0 0 6 第 3 規約 : L 4 , 集計の正統性の判定。

20

0 0 6 1 L 4 , 集計の第 4 領域は正統か .

0 0 6 1 1 正統なら GOTO 0 0 7 .

0 0 6 1 2 正統でなければ GOTO 第 5 規約 .

0 0 7 第 4 規約。

0 0 7 1 MOVE 0 TO 第 6 フラグ .

0 0 7 2 MOVE 集計 TO READ DB 1 1 , 集計の第 4 領域 .

0 0 7 3 ADD 1 TO 主語成立数カウンタ .

0 0 8 RETURN .

0 0 9 第 5 規約 : L 4 , 集計の集計が同じ座標周期で成立するや否やの判定。

30

0 0 9 1 MOVE 0 TO 第 6 フラグ .

0 0 9 2 (NS 1 = NS 5) なら第 7 規約に進む .

0 0 9 3 (NS 1 > NS 5) なら第 6 規約に進む .

0 1 0 第 6 規約 : READ DB 1 1 , 集計の再起要請の宣言。

0 1 0 1 READ DB 1 1 , 集計の第 4 領域を初期値化 .

0 1 0 2 MOVE 1 TO 第 6 フラグ .

0 1 0 3 MOVE 0 TO 第 2 フラグ .

0 1 0 4 RETURN .

0 1 1 第 7 規約 : READ DB 1 1 , 集計の再起停止宣言。

0 1 1 1 MOVE 1 TO 第 7 フラグ .

40

0 1 1 2 MOVE 0 TO 第 2 フラグ .

0 1 1 3 RETURN .

【0 1 9 3】

(E 4 1 , P 4 プログラムの論理結合型模型)

E 4 1 , P 4 は本プログラムが終了状態にあることを告知する。E 4 1 , P 4 を分かり易くする為、論理結合型で示しているが、E 4 1 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。

0 0 1 START .

0 0 2 主語成立数スタックから NS 1 を採る .

50

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。

10

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。

20

【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。

30

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。

40

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。

50

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 .

10

20

30

40

50

0 0 2 P 3 の臨界判定 :

0 0 2 1 パレット 3 の第 6 フラグ列の作成 .

0 0 2 2 パレット 3 の第 6 フラグ列常数の作成 .

0 0 3 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) .

0 0 3 1 XOR が ZERO なら GOTO 0 0 4 .

0 0 3 2 XOR が NOT ZERO なら GOTO 0 0 6 .

0 0 4 主語分布表 (図 1 3) を用いて行う判定。

0 0 4 1 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)

0 0 4 2 YES なら GOTO 0 0 5 .

0 0 4 3 NO なら GOTO 0 0 6 . 10

0 0 5 MOVE 1 TO T 3 2 の第 4 領域 .

0 0 6 RETURN .

【 0 2 0 0 】

(「主語分布表」 図 1 3 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。

【 0 2 0 1 】

(座標関数 4 プログラムの論理結合型模型)

座標関数 4 のプログラム模型を以下に示す。 20

0 0 1 START .

0 0 2 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

0 0 3 指定された搭載順位 (i) のベクトルを起動 .

0 0 4 ここは起動したベクトルの RETURN 点です .

0 0 5 命令破壊信号 (OS) の有無判定 .

0 0 5 1 有りで、SEP 起動 . : 本プログラムの実行を停止

0 0 5 2 なしで、GO TO 0 0 6 .

0 0 6 起動する次のベクトルを指定 ($i = i + 1$) .

0 0 7 (i) はパレット 4 のベクトルの搭載数 ($N 4$) を超えたか .

0 0 7 1 超えていれば GO TO 0 0 8 .

0 0 7 2 超えていなければ GO TO 0 0 3 . 30

0 0 8 パレット 4 の臨界の成否判定 . : T 4 の第 4 領域のオンオフを利用

0 0 8 1 オフなら GO TO 0 0 2 .

0 0 8 2 オンなら GO TO 0 0 9 . : 座標周期 4 の終了

0 0 9 論理矛盾発症の有無を判定 : E 4 2 の第 4 領域オンオフの利用 .

0 0 9 1 オンで、SLP 起動 . : 本プログラムの実行停止処理

0 0 9 2 オフで、GO TO 0 1 0 . : 本プログラムには論理矛盾は起きていない

0 1 0 同期関数に戻る為に次の処理を行う

0 1 0 1 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 .

0 1 0 2 スタック更新プログラムの起動 .

0 1 1 同期関数に RETURN . 40

【 0 2 0 2 】

(座標関数 2 プログラムの論理結合型模型)

座標関数 2 のプログラム模型を以下に示す。

0 0 1 START .

0 0 2 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

0 0 3 指定された搭載順位 (i) のベクトルを起動 .

0 0 4 ここは起動したベクトルの RETURN 点である .

0 0 5 命令破壊信号 (OS) の有無判定 .

0 0 5 1 有りで、SEP 起動 . : 本プログラムの実行を停止

0 0 5 2 なしで、GO TO 0 0 6 . 50

0 0 6 起動する次のベクトルを指定 ($i = i + 1$) .
 0 0 7 (i) はパレット 2 のベクトルの搭載数 ($N 2$) を超えたか .
 0 0 7 1 超えていれば GO TO 0 0 8 .
 0 0 7 2 超えていなければ GO TO 0 0 3 .
 0 0 8 パレット 2 の臨界の成否判定 . : T 2 の第 4 領域のオンオフを利用
 0 0 8 1 オフなら GO TO 0 0 2 .
 0 0 8 2 オンなら GO TO 0 0 9 . : 座標周期 2 の終了
 0 0 9 論理矛盾発症の有無を判定 : E 4 2 の第 4 領域オンオフの利用 .
 0 0 9 1 オンで、SLP 起動 . : 本プログラムの実行停止処理
 0 0 9 2 オフで、GO TO 0 1 0 . : 本プログラムには論理矛盾は起きていない 10
 0 1 0 同期関数に戻る為に次の処理を行う
 0 1 0 1 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 .
 0 1 1 同期関数に RETURN .
【 0 2 0 3 】
 (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。
 0 0 1 START .
 0 0 2 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
 0 0 3 指定された搭載順位 (i) のベクトルを起動 .
 0 0 4 ここは起動したベクトルの RETURN 点である . 20
 0 0 5 命令破壊信号 (OS) の有無判定 .
 0 0 5 1 有り、SEP 起動 . : 本プログラムの実行を停止
 0 0 5 2 なし、GO TO 0 0 6 .
 0 0 6 起動する次のベクトルを指定 ($i = i + 1$) .
 0 0 7 (i) はパレット 2 のベクトルの搭載数 ($N 3$) を超えたか .
 0 0 7 1 超えていれば GO TO 0 0 8 .
 0 0 7 2 超えていなければ GO TO 0 0 3 .
 0 0 8 パレット 3 の臨界の成否判定 . : T 2 の第 4 領域のオンオフを利用
 0 0 8 1 オフなら GO TO 0 0 2 .
 0 0 8 2 オンなら GO TO 0 0 9 . : 座標周期 5 の終了 30
 0 0 9 論理矛盾発症の有無を判定 : E 4 2 の第 4 領域オンオフの利用 .
 0 0 9 1 オンで、SLP 起動 . : 本プログラムの実行停止処理
 0 0 9 2 オフで、GO TO 0 1 0 . : 本プログラムには論理矛盾は起きていない
 0 1 0 同期関数に戻る為に次の処理を行う
 0 1 0 1 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 .
 0 1 1 同期関数に RETURN .
【 0 2 0 4 】
 (同期関数プログラムの論理結合型模型)
 0 0 1 START . 40
 0 0 2 スタック初期値化プログラム起動 .
 0 0 3 本プログラムの終了判定 . : E 4 1 の第 4 領域のオンオフを利用

 0 0 3 1 オンなら、GO TO 0 0 4 . : 本プログラム終了処理
 0 0 3 2 オフなら、GO TO 0 0 5 . : 本プログラムの始動
 0 0 4 SEP 起動 . : System Ending Program (本プログラム終了プログラム)
 0 0 5 T 3 1 第 4 領域のオンオフ判定 . : 自分の座標関数 4 の起動
 0 0 5 1 オンなら、T 3 1 第 4 領域をオフにする .
 0 0 5 2 自分の座標関数 4 を起動 .
 0 0 5 3 オフなら GO TO 0 0 6 . 50

0 0 6 T 3 2 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
 0 0 6 1 オンなら . T 3 2 第 4 領域をオフにする .
 0 0 6 2 最近傍下位の座標関数 4 を起動 .
 0 0 6 3 オフなら、G O T O 0 0 7 . : 最近傍下位の座標関数 4 がなければオフ。
 0 0 7 T 3 3 第 4 領域のオンオフ判定 . : 最上位の座標関数 3 の起動
 0 0 7 1 オンなら . T 3 3 第 4 領域をオフにする .
 0 0 7 2 最上位の座標関数 3 を起動 .
 0 0 7 3 オフなら、G O T O 0 0 8 . : 最上位の座標関数 3 がなければオフ。
 0 0 8 T 4 第 4 領域のオンオフ判定 . : 自分の座標関数 2 の起動
 0 0 8 1 オンなら . T 4 第 4 領域をオフにする . 10
 0 0 8 2 自分の座標関数 2 を起動 .
 0 0 8 3 オフなら、G O T O 0 0 9 . : 自分の座標関数 3 の起動。
 0 0 9 T 2 第 4 領域のオンオフ判定 . : 自分の座標関数 3 の起動
 0 0 9 1 オンなら . T 2 第 4 領域をオフにする .
 0 0 9 2 自分の座標関数 3 を起動 .
 0 0 9 3 オフなら、G O T O 0 1 0 .
 0 1 0 T 3 2 第 4 領域のオンオフ判定 . : 隣下の座標関数 4 の起動
 0 1 0 1 オンなら . T 3 2 第 4 領域をオフにする .
 0 1 0 2 隣下の座標関数 4 を起動 .
 0 1 0 3 オフなら、G O T O 0 1 1 . 20
 0 1 1 T 3 3 第 4 領域のオンオフ判定 . : 最上位の座標関数 3 の起動
 0 1 1 1 オンなら . T 3 3 第 4 領域をオフにする .
 0 1 1 2 最上位の座標関数 3 を起動 .
 0 1 1 3 オフなら、G O T O 0 0 5 2 .
 0 1 2 E N D .
 【 0 2 0 5 】
 (E 4 1 , P 4 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 L
 4 . A である。このベクトルの命令文はスタッ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。E 4 1 , P 4 には L 3 が無い。次にプログラム模型を示す。 30
 0 0 1 S T A R T .
 0 0 2 N O P .
 0 0 3 C A L L V W A (E 4 1 , P 4) .
 0 0 4 第 1 規約 : E 4 1 , P 4 の正統性を用いて第 2 規約に進むか、R E T U R N かを
 判定。
 * ベクトルの正統性項参照
 0 0 5 第 2 規約
 0 0 5 1 M O V E 1 T O 第 2 フラグ .
 0 0 5 2 主語成立数スタックから N S 1 を採る . 40
 0 0 5 3 主語成立数スタックから N S 5 を採る .
 0 0 6 第 3 規約 : (N S 1) = (N S 5) の判定 .
 0 0 6 1 Y E S なら G O T O 第 4 規約 .
 0 0 6 2 N O なら G O T O 第 5 規約 .
 0 0 7 第 4 規約。
 0 0 7 1 M O V E 0 T O E 4 1 , P 4 の第 6 フラグ .
 0 0 7 2 M O V E A T O E 4 1 , P 4 の第 4 領域 .
 0 0 7 3 A D D 1 T O 主語成立数カウンタ .
 0 0 8 R E T U R N .
 0 0 9 第 5 規約 : E 4 1 , P 4 が同じ座標周期で成立するや否やの判定。 50

0 0 9 1 MOVE 0 TO 第6フラグ。
 0 0 9 2 (NS1 = NS5)ならGOTO 第7規約。
 0 0 9 3 (NS1 > NS5)ならGOTO 第6規約。
 0 1 0 第6規約：E41, P4の再起要請の宣言。
 0 1 0 1 E41, P4の第4領域を初期値化。
 0 1 0 2 MOVE 1 TO 第6フラグ。
 0 1 0 3 MOVE 0 TO 第2フラグ。
 0 1 0 4 RETURN。
 0 1 1 第7規約：E41, P4の再起停止宣言。
 0 1 1 1 MOVE 1 TO 第7フラグ。 10
 0 1 1 2 MOVE 0 TO 第2フラグ。
 0 1 1 3 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。T33, P3にはL3がない。次にプログラム模型を示す。
 0 0 1 START。
 0 0 2 NOP。
 0 0 3 CALL VWA(T33, P3)。
 0 0 4 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 20
 判定。
 *ベクトルの正統性項参照
 0 0 5 第2規約 T33, P3の成立判定の準備：
 0 0 5 1 MOVE 1 TO 第2フラグ。
 0 0 5 2 パレット3の第6フラグ列の作成。
 0 0 5 3 パレット3の第6フラグ列常数の作成。
 0 0 6 第3規約：T33, P3の成立判定。
 0 0 6 1 P3の臨界判定。
 0 0 6 1 1 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)。
 0 0 6 1 2 XORがZEROならGOTO 0062。 30
 0 0 6 1 3 XORがNOT ZEROなら第5規約。
 0 0 6 2 主語分布表(図13)を用いて行う判定。
 0 0 6 2 1 主語は隣上の本プログラムにも内在するか(第2条件)
 0 0 6 2 2 YESならGOTO 第4規約。
 0 0 6 2 3 NOならGOTO RETURN。
 0 0 7 第4規約。
 0 0 7 1 MOVE 0 TO T33, P3の第6フラグ。
 0 0 7 2 MOVE A TO T33, P3の第4領域。
 0 0 7 3 ADD 1 TO 主語成立数カウンタ。
 0 0 8 RETURN。 40
 0 0 9 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。
 0 0 9 1 MOVE 0 TO 第6フラグ。
 0 0 9 2 (NS1 > NS5)ならGOTO 第6規約。
 0 0 9 3 (NS1 > NS5)でなければGOTO 第7規約。
 0 1 0 第6規約：E41, P4の再起要請の宣言。
 0 1 0 1 T33, P3の第4領域を初期値化。
 0 1 0 2 MOVE 1 TO 第6フラグ。
 0 1 0 3 MOVE 0 TO 第2フラグ。
 0 1 0 4 RETURN。
 0 1 1 第7規約：T33, P3の再起停止宣言。 50

0 1 1 1 MOVE 1 TO 第7フラグ .
 0 1 1 2 MOVE 0 TO 第2フラグ .
 0 1 1 3 RETURN .
 【0207】
 (スタック更新プログラムの論理結合型模型)
 0 0 1 START .
 0 0 2 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
 0 0 3 RETURN .
 0 0 4 主語成立数カウンタがゼロでなく、NS 1、2、3、4ゼロなら 10
 0 0 5 主語成立数カウンタをNS 1に上書 .
 0 0 6 RETURN .
 0 0 7 主語成立数カウンタがゼロでなく、NS 2、3、4ゼロなら
 0 0 8 NS 1をNS 2に上書 .
 0 0 9 主語成立数カウンタをNS 1に上書 .
 0 1 0 RETURN .
 0 1 1 主語成立数カウンタがゼロでなく、NS 3、4ゼロなら
 0 1 2 NS 1をNS 2に上書 .
 0 1 3 NS 2をNS 3に上書 .
 0 1 4 主語成立数カウンタをNS 1に上書 . 20
 0 1 5 RETURN .
 0 1 6 主語成立数カウンタがゼロでなく、NS 4がゼロなら
 0 1 7 NS 1をNS 2に上書 .
 0 1 8 NS 2をNS 3に上書 .
 0 1 9 NS 3をNS 4に上書 .
 0 2 0 主語成立数カウンタをNS 1に上書 .
 0 2 1 RETURN .
 0 2 2 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
 0 2 3 NS 1をNS 2に上書 .
 0 2 4 NS 2をNS 3に上書 . 30
 0 2 5 NS 3をNS 4に上書 .
 0 2 6 NS 4をNS 5に上書 .
 0 2 7 主語成立数カウンタをNS 1に上書 .
 0 2 8 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 (NS 1 > NS 5)、制御ベクトルE
 4 1 (NS 1 = NS 5)、E 4 2 (NS 1 < NS 5)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。
 0 0 1 START . 40
 0 0 2 NS 5に1をセット .
 0 0 3 NS 4をゼロクリア .
 0 0 4 NS 3をゼロクリア .
 0 0 5 NS 2をゼロクリア .
 0 0 6 NS 1をゼロクリア .
 0 0 7 主語成立数カウンタをゼロクリア .
 0 0 8 RETURN .
 【0209】
 ウイルスが稼働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す、侵入しても、本プログラムはそのウイルスを自 50

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

10

【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。

20

30

40

【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。

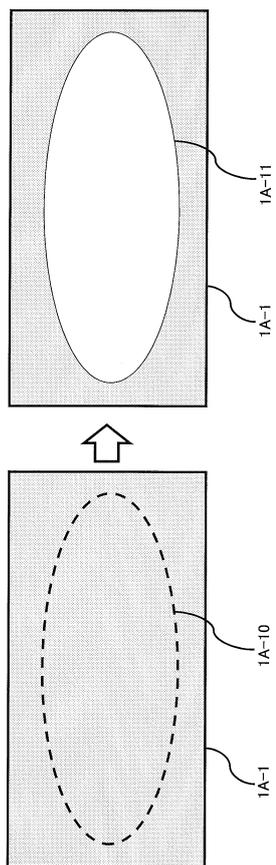
【符号の説明】

50

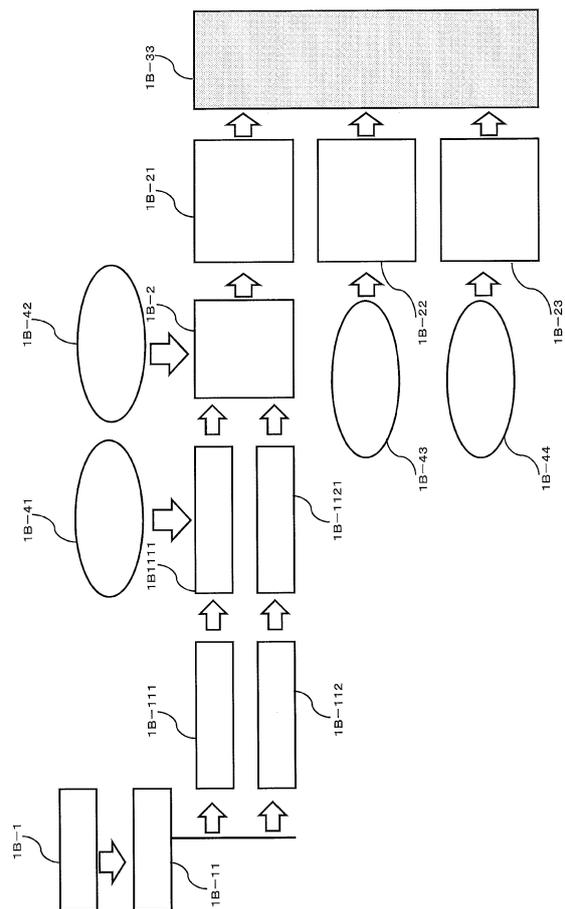
【 0 2 1 4 】

1 A - 1 ... 電算機の記憶領域、1 A - 1 0 ... 従来のプログラム、1 A - 1 1 ... 本発明のプログラム、1 C - 1 ... 電算機の記憶領域、1 C - 1 0 ... 従来のプログラム、1 C - 1 1 ... 本発明のプログラム、1 C - 2 ... コンピュータウイルス

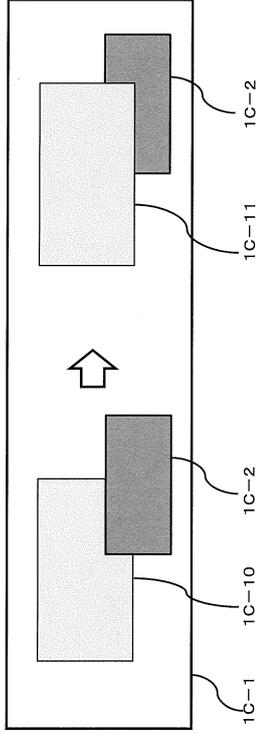
【 図 1 A 】



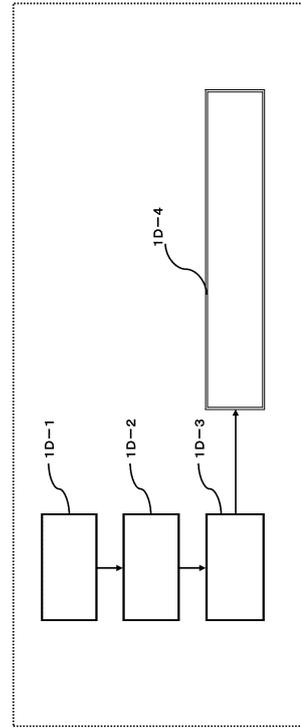
【 図 1 B 】



【図 1 C】



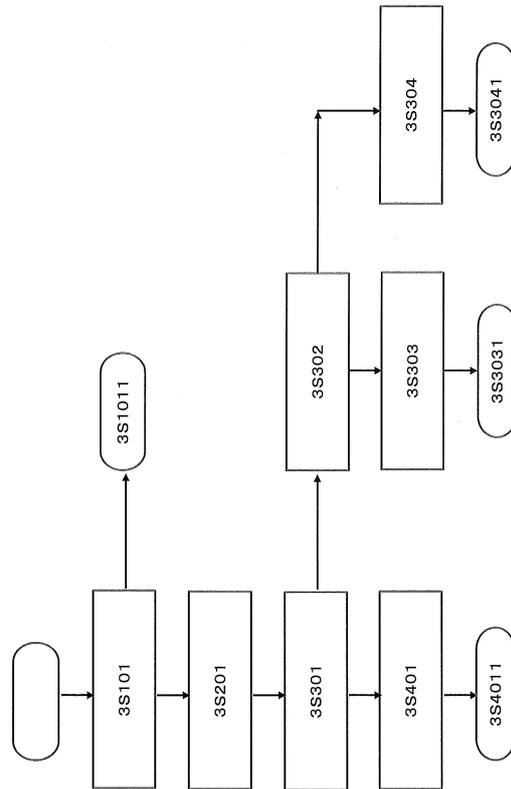
【図 1 D】



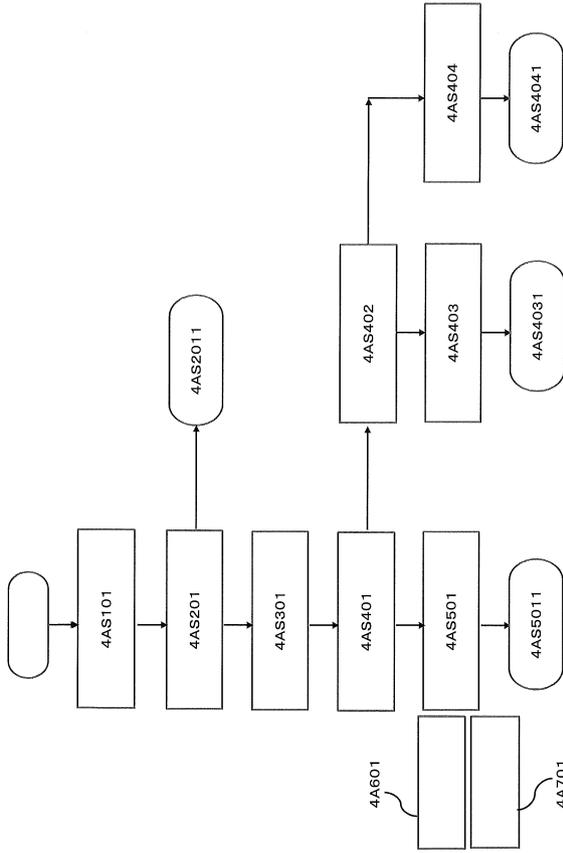
【図 2】

No	ベクトル種別	搭載バレット種別	結合する命令文例	命令文の位置
1	L4, 名詞	P4	代入文	第2線約
2	L2, 名詞	P2	定数文	第3線約
3	L3, 名詞	P3	条件文	第4線約
4	W4, 名詞	P4	出力文	第5線約
5	R2, 名詞	P2	入力文	第6線約
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】



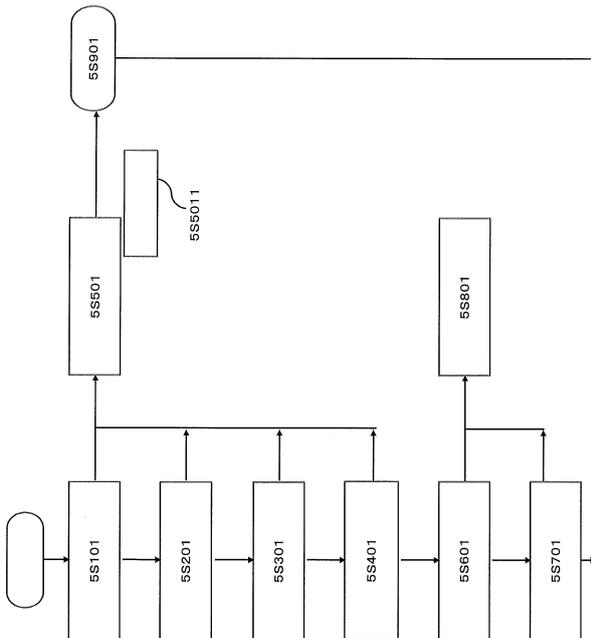
【図4A】



【図4B】

第4領域	第2フラグ	第6フラグ	第7フラグ	備考
オン	オン	オフ	オフ	ベクトルが正統な主語を成立させている。
オフ	オフ	オン	オフ	ベクトルが再起要請をしている。
オフ	オフ	オフ	オン	ベクトルが再起停止要請をしている。

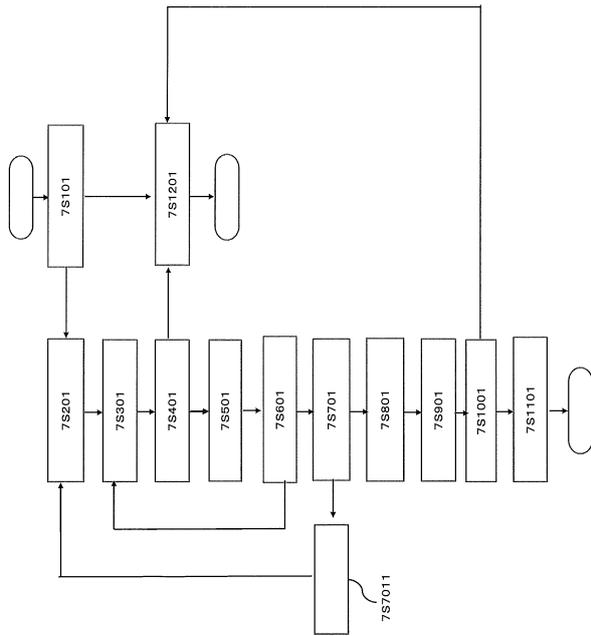
【図5】



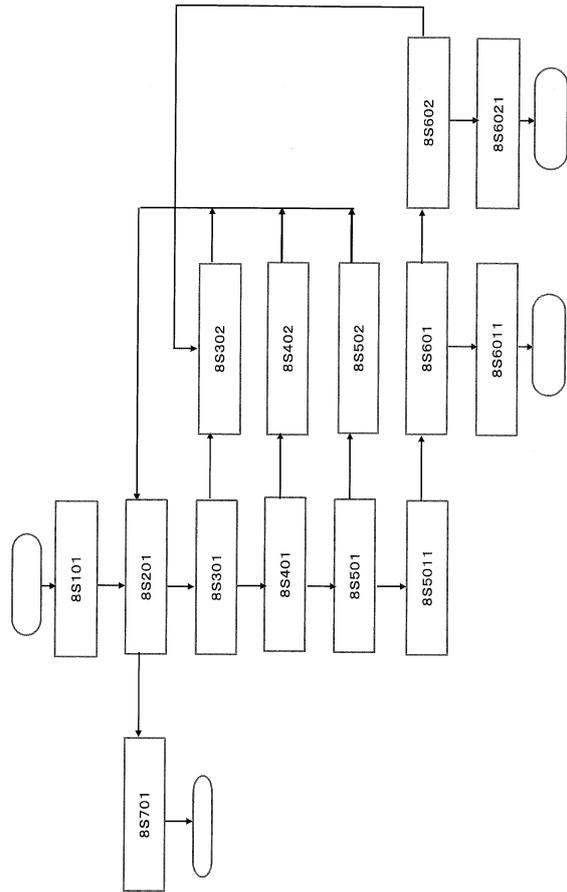
【図6】

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

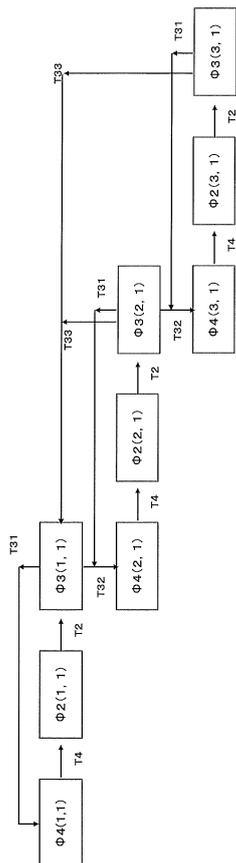
【 図 7 】



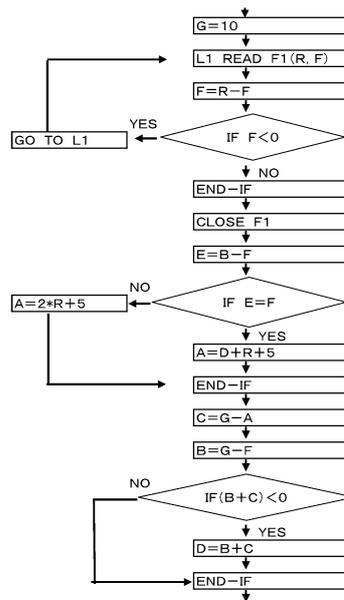
【 図 8 】



【 図 9 】



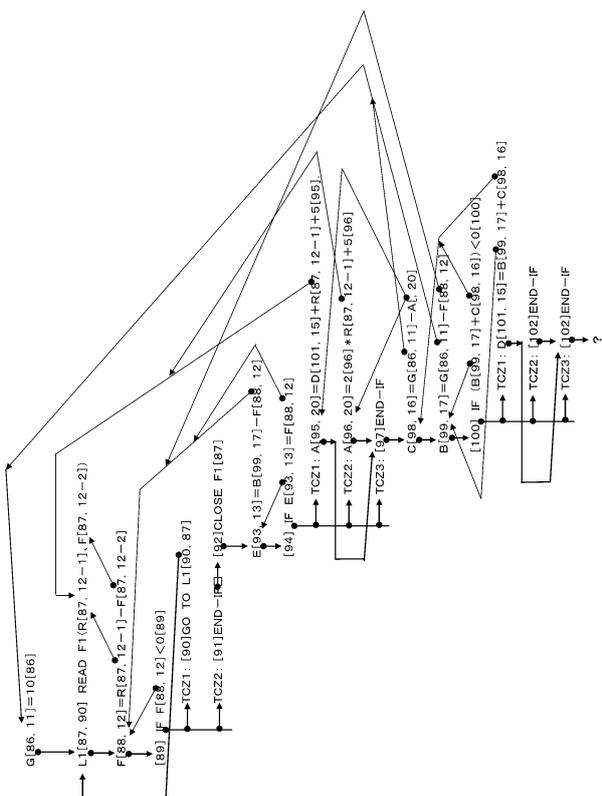
【 図 10 】



【図 1 1 A】

プログラム	文種	ベクトル種別	主語	運動文	行No	TCX	TCY	TCZ1	TCZ2	TCZ3	TCZ4
G=10	定値	L2	G		86	86	87				
L1 READ F1(R, F)	入力	R2	F1(R, F)		87	87	88				
F[86, 12]=R[87, 12]-1	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	90	87			
END-IF	閉鎖	L4	F1	●	91	91	92				
CLOSE F1	閉鎖	L4	E	●	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=*R+5	代入	L4	A		96	96	97				
END-IF	閉鎖	L4	C	●	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	102	?
D=B+C	代入	L4	D		101	101	102				
END-IF	閉鎖	L4	D	●	102	102	?				

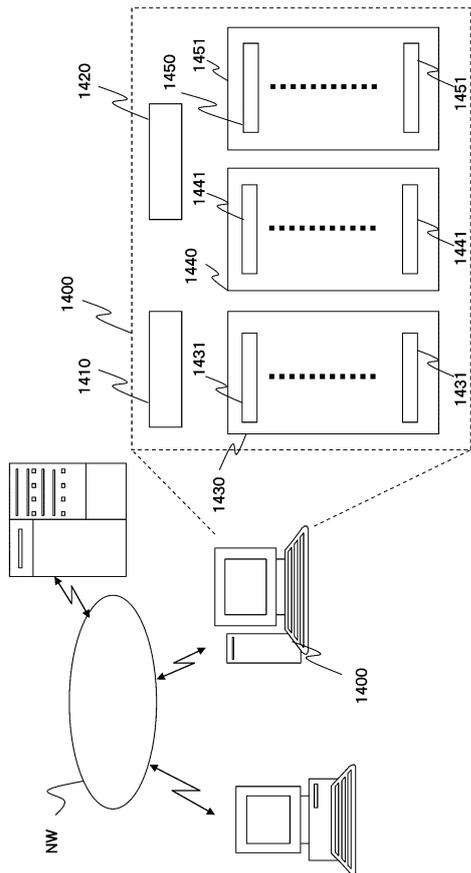
【図 1 2】



【図 1 1 B】

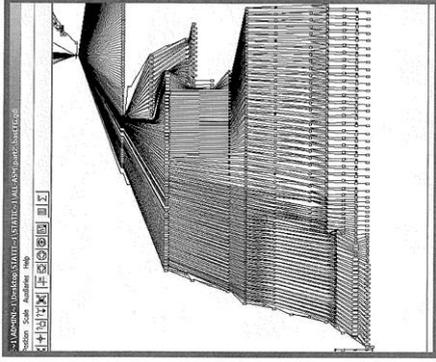
TCZ4	超言語化された单元文	主語	超言語座標 要素主語	指標	定値
G[86, 11]=10[86]	G[86, 11]				
L1[87, 90] READ F1(R[87, 12]-1, F[87, 12-2])	F1(R[87, 12-1], F[87, 12-2])			L1[87, 90]	
F[86, 12]=R[87, 12]-1, F[87, 12-2]	F[88, 12]		R[87, 12-1], F[87, 12-2]		
IF F[86, 12]<0[89]	F[88, 12]		F[88, 12]		0[89]
GO TO L1[89, 87]				L1[90, 87]	
END-IF [91]					
CLOSE F1[92]	F1[87]				
E[93, 13]=B[89, 17]-F[88, 12]	E[93, 13]		B[89, 17], F[88, 12]		
IF E[93, 13]=F[88, 12]	E[93, 13]		E[93, 13], F[88, 12]		
TCZ1: A[95, 20]=D[101, 15]+R[87, 12-1]+5[95]	A[95, 20]		A[95, 20]		5[95]
TCZ2: A[96, 20]=2[96]*R[87, 12-1]+5[96]	A[96, 20]		A[96, 20]		2[96], 5[96]
END-IF [97]					
C[98, 16]=G[86, 11]-A[97, 20]	C[98, 16]		G[86, 11], A[97, 20]		
IF B[99, 17]=G[86, 11]-F[88, 12]	B[99, 17]		G[86, 11], F[88, 12]		
IF B[99, 17]+C[98, 16]<0[100]	B[99, 17]		B[99, 17], C[98, 16]		0[100]
END-IF [102]					

【図 1 4】

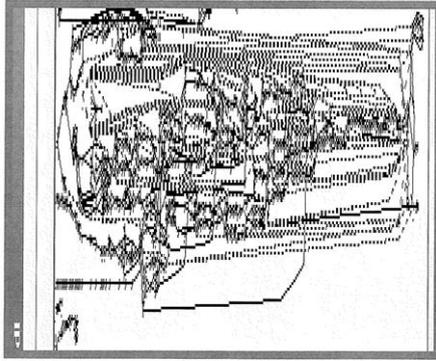


【 図 1 E 】

(データ結合型プログラム)



(論理結合型プログラム)



Programs analysis by SamCOITS (Static Analyzer of Malicious COITS) Laval University

【 図 1 3 】

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) C(1, 1)	(1, 1) (1, 1)
2			

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)
G06F21/50-21/57