

結び目の作図とその簡素化

東工大・理	本間 龍雄
“	森川 治
阪大・理	落合 豊行

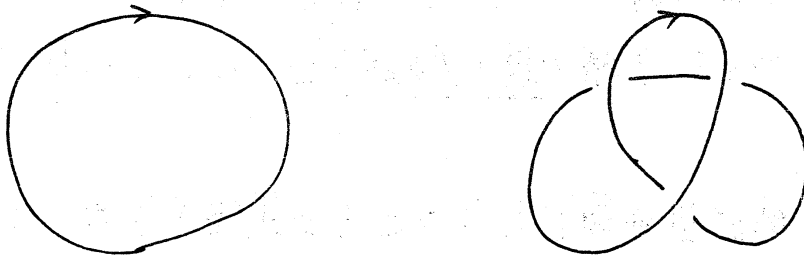
0. はじめに.

結び目理論において、結び目の判別のし方はいろいろあり、又それぞれに不変量が提案されている。しかし、完全に結び目を他の結び目と区別できる不変量は、まだ見つかっていない。結び目のうち特に図1のようないわゆる“ほどけた結び目”すなわち *trivial* な結び目と、図2のような“ほどけない結び目”との区別については、ある程度知られている。すなわち、ある与えられた結び目が、*trivial* であるか否かは、判定できるわけであるが、その判定のアルゴリズムは、あまり知られていない。これに対し、Haken [3], DeGruse [4]らは、*normal surface* という概念を導入し、*trivial* な結び目か否かを判定するアルゴリズムを提案した。しかし、この方法は完全ではあるが、実際の結び目を判定することは、天文学的な場合数に対しすべて調べる必要があり、全く

実用にならない。又、結び目群 (knot group) という不変量は、次の定理が示す通り、この判定に関する完全な不変量となっているが、今度は、群が可換か非可換かを判定することが問題となり、やはりこの判定アルゴリズムも知られていないので、実用にならない。

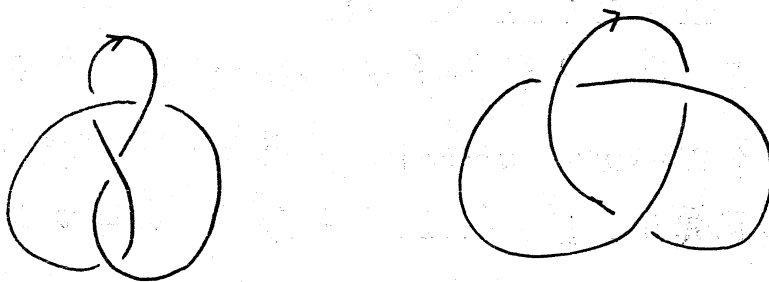
定理 結び目 K が、*trivial* である必要十分条件は、結び目群 πK がアーベル群であることである。

そこで、私達はもっと実用的なアルゴリズムを研究中である。ここで述べるのは、その中間結果である。



trivial な結び目

図 1.



trivial でない結び目

図 2.

1. 結び目の定義.

結び目とは、元来ひもを結び合わせた部分をさすものだが、ここでは、始点と終点を一致させたいわば“結び輪”を結び目という。数学的に定義すると、空間としてユークリッド空間 \mathbb{R}^3 または、 \mathbb{R}^3 を 1 点コンパクト化した 3 次元球面 S^3 内に、輪 S^1 と同相な図形を結び目ということになる。ここで、空間 S^3 、輪 S^1 共に向きが付けられているものとする。すなわち、

定義 S^1, S^3 は共に向き付けられている。 g を S^1 から S^3 への埋め込み写像。すなわち、 S^1 と $g(S^1)$ が同相となる写像のとき、 $g(S^1)$ を 結び目 (knot) といい、この $g(S^1)$ を K と表わす。

次に、結び目が同じということ定義するが、これはわかりやすく言えば、結び目が連続的に変形して互いに移り合えることをいうものである。

定義 結び目 K, K' が同値 とは、 $K = g(S^1), K' = g'(S^1)$ としたとき、 g と g' が *ambient isotopic* であることである。

g と g' が *ambient isotopic* であるとは、次の 3 条件を満たす連続写像 $F: S^1 \times [0, 1] \rightarrow S^3$ が存在することである。

1. 任意の $t \in [0, 1]$ に対し、 $F(x, t)$ を $f_t(x)$ と書くことにすると、 $f_t: S^1 \rightarrow S^3$ が homeomorphism となっている。

2. f_0 は恒等写像

3. f_1 と g との合成写像 $f_1 \circ g : S^1 \rightarrow S^3$ が g' に等しい。

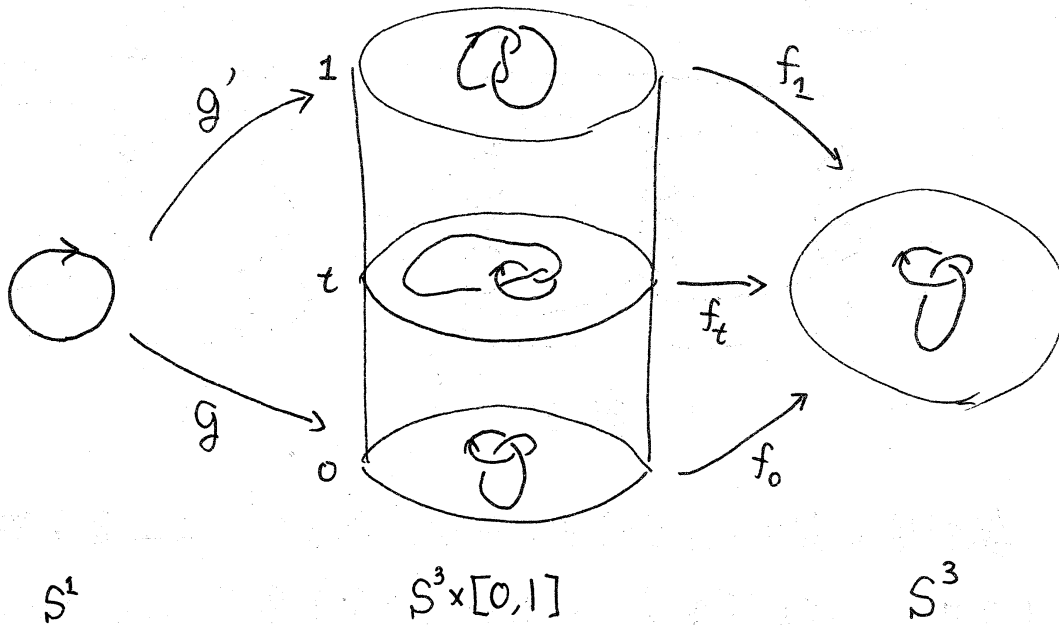


図 3.

定義は以上ですが、ここでは K は有限析れ線で表わせるものだけを対象とし、図 4 のようなものは対象としない。

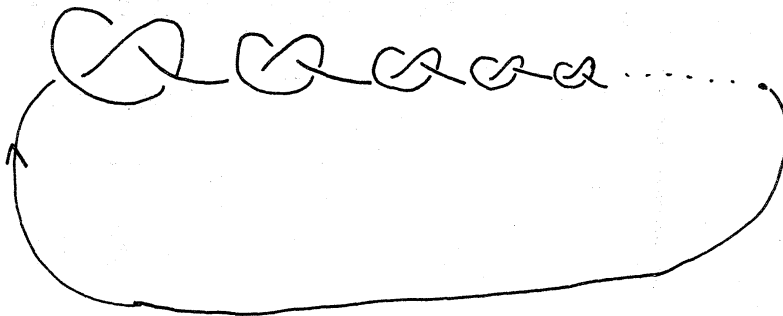


図 4.

例. 次の結び目は全て異なる.

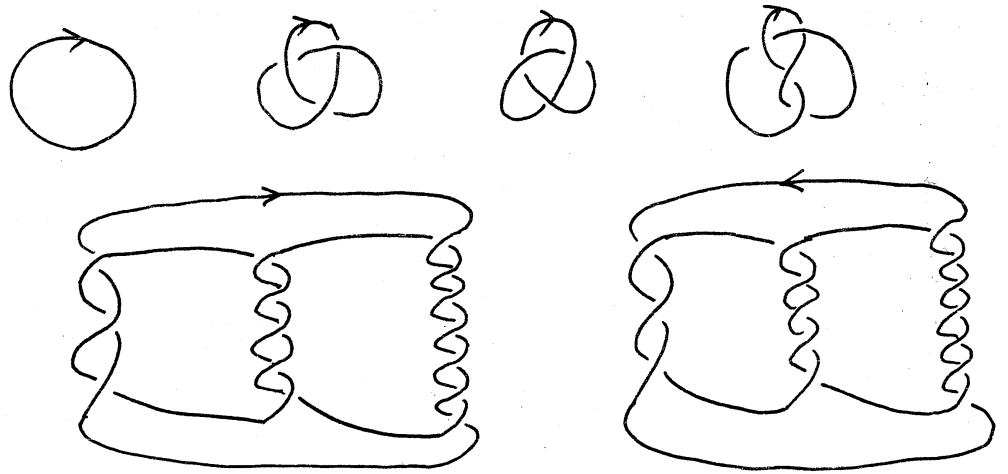


図 5

結び目は \mathbb{R}^3 (又は S^3) 内のものであるが. これを平面 \mathbb{R}^2 (又は. 2次元球面 S^2) へ射影した射影図 (knot form) により話をすすめる. このとき. 本当に射影してしまつては. 交差点の上下関係がわからなくなつてしまうので. 上にある方の部分を上道 (overpass) といい. 下の方を下道 (underpass)) といつて区別することにする. 図6の例では.

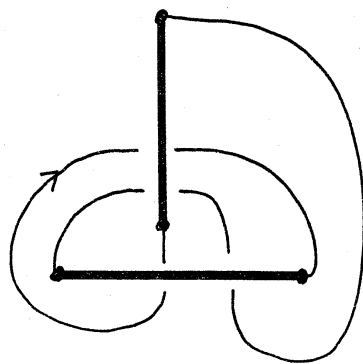


図 6

2本ずつの overpass, underpass により. 結び目が与えられているので. この射影図を. 2-bridge form という. 一般に. n 本ずつの場合. n -bridge form という. 以下. K は射影図を表わすと

し、特にことわりのない限り、結び目と射影図を同一視して述べる。

結び目の同値性に対する次の定理がある。

定理 結び目 K, K' が同値である

\iff K は次の3つの操作を有限回行って K' に変形できる。

$$T_1: \text{ (crossing) } \iff \text{ (uncrossing) }$$

$$T_2: \text{ (loop) } \iff \text{ (trivial) } \quad \text{図7}$$

$$T_3: \text{ (strand crossing) } \iff \text{ (strand crossing) }$$

$$\textcircled{\text{注}}. \text{ (loop) } \xrightarrow{T_2} \text{ (loop) } \xrightarrow{T_1} \text{ (loop) } \text{ 又 } T_0: \text{ (loop) } \iff \text{ (loop) }$$

も含まれる。

2. 同値性の問題

問題1. 結び目 K, K' が与えられたとき、これらが同値であるかを判定せよ。

問題1に対し、“ K と K' は同値である”と断定できるのは、実際に定理の T_1, T_2, T_3 の組合せで、 K から K' への変形をつくってやる以外の判定法は、ほとんど知られていない。

しかし逆に、同値でないとの断定には、種々の不変量が提案されていて、例えば図2の2つの結び目は、Alexander Polynomial という不変量がそれぞれ、 t^2-3t+1 , t^2-t+1 と異なるため結び目も同値でないとわかる。

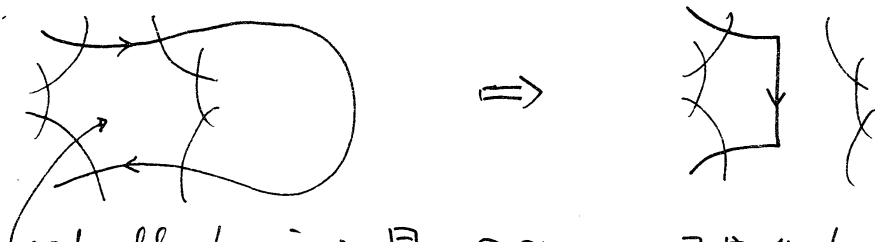
問題1は大変難しいので、部分的問題として、次のものがある。

問題2. 結び目 K が与えられたとき、 K が *trivial* な結び目であるか判定せよ。

これに対しては、はじめにも述べたが、Haken, DeGruise により解は存在するが、実用的なものは見つかっていない。部分的解として次のものがある。

定理 (本間, 落合, 高橋) [6]

結び目 K が、3-bridge form によって与えられているとき、 K が *trivial* な結び目と同値ならば、*reducible domain* による *reduction* により、 K を *trivial* な結び目 \bigcirc に変形できる。



reducible domain: 同一の *overpass* 又は *underpass* に属する2辺以上をまつ面. 図8.

この reduction により、knot form の交差点数は減少するので、高々、交差点の数だけ実行すれば判定ができる。

この定理の操作は、3-bridge form でなくても可能であり、拡張が考えられるが、4-bridge form ですでに、その反例が、O. J. Burgo (1977) [7] によって示されている。

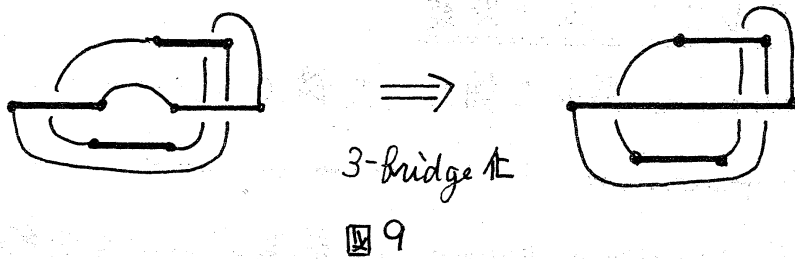


図9

しかし、この反例は容易に 3-bridge 化が行なえ、trivial な結び目であることが示せる。すなわち、定理の操作に、下記のような、bridge number 減少の操作を加えれば、この例は拡張性の否定とはならない。

◦ bridge number 減少の操作： 交互点をもたない overpass 又は underpass は前後の underpass 又は overpass の一部とみなし、bridge number を減少させる。

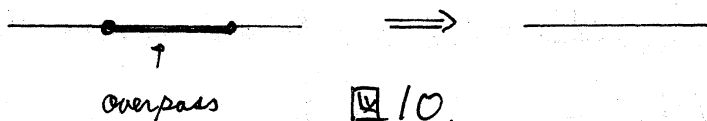


図10.

しかし、この操作を加えても、4-bridge form で反例が得られた。(森川、1980 [8])

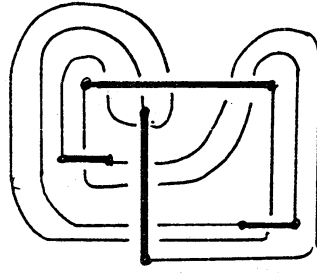
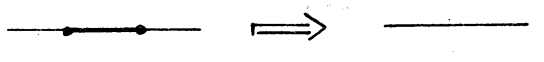


図 11 4-bridge form での反例.

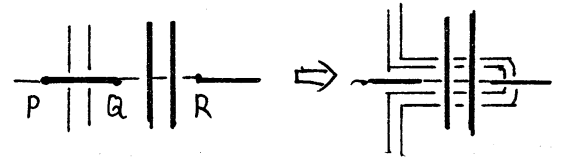
3. 新アルゴリズムの提案

bridge number 減少を軸とした操作を次のように定める.

1. Short operation : 

交差点のなりの overpass, underpass を前後の pass を短絡させることにより消去する.

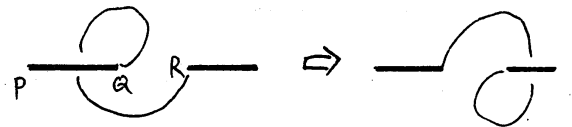
2. move operation :



PQ 間にある交差点をとる

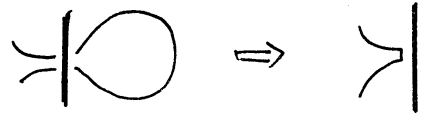
りの overpass のところまで動かしてやり, PQ を short operation により消す.

3. change operation :



underpass QR を変形し, $PQ \cap QR = \{Q\}$ とする. これにより次に move operation が行なえ, bridge number が減少する.

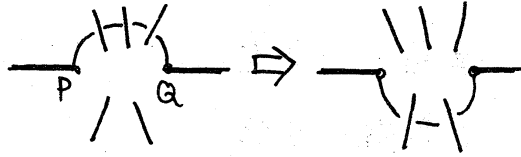
4. annihilation :



overpass のとなり合った

交差点が, 同一の underpass と交差しているとき, underpass を変形して交差点を減らす.

5. color operation :
underpass PQ を現在の



状態より交差数の少ない他の underpass に変更する。(この underpass を見つけるとき、面に色をつけたことより、この名前となった。)

以上5つの操作によって結び目を変形することを、新アルゴリズムとした。これに対し、次の3つの目標を考えた。

1. 結び目 K が与えられたとき、このアルゴリズムに従って K を変形してゆけば、trivial なるか判定できる。
2. さらに、 K が non-trivial のときも、bridge index という不変量が決定できる。

3. 2つの結び目の同値性が示せる。

以上3つの目標に対し、人力で数々のサンプルで実験したところ、うまくゆくので、コンピュータ実験を行なうことになった。

4. コンピュータのために… 結び目のデータ構造

結び目 K は、次の情報により表現できる。

1. 交差点及び、overpass と underpass の境目。
2. これらのつながり具合、交わり具合。
3. 結び目の向き。

この情報を、細胞 $cell = (OV_{next}, UD_{next}, orient)$ の3つ組によつて記憶させる。

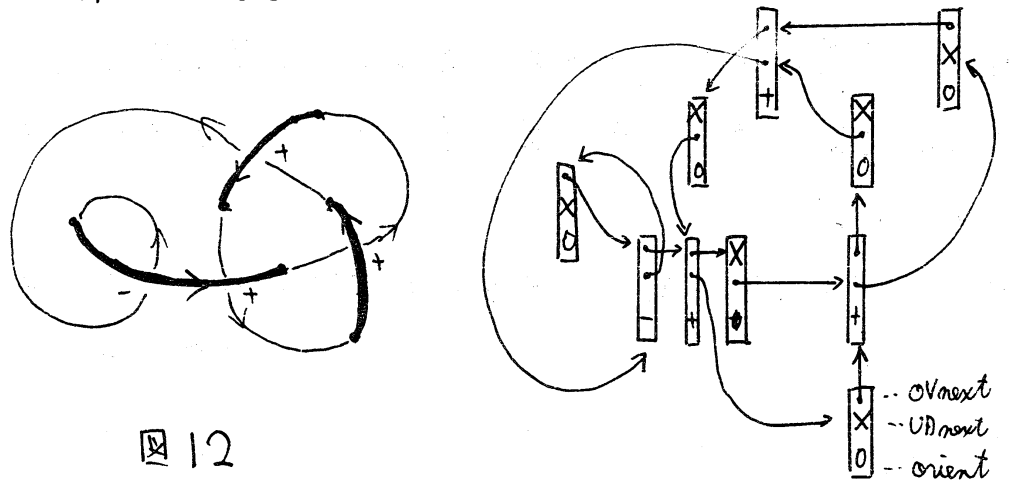


図 12

OV_{next} は overpass どのながつている点を指すポインタ型変数, UD_{next} は underpass どの点へのポインタ, $orient$ は、交わり方で、 \downarrow は +, \uparrow は -, \rightarrow または \leftarrow は 0 とした。

結び目は全てこのようなデータによつて表わせるが、任意のデータが結び目を表現しているとは限らない。そこで、次の定理がある。

定理 内部データが結び目を表現する必要十分条件は、このデータより導かれる oriented closed surface の genus が 0 であることである。

oriented closed surface は、oriented faces により構成される。oriented face の構成法は、紙面の都合上、省略する。

5. 使用言語

本実験は、操作の定義から明らかのように、各操作に *unique* 性はなく、1つの状況から、多数の操作後の状況が生ずる。この各々をチェックする必要があるので、次のような流れ図となる。

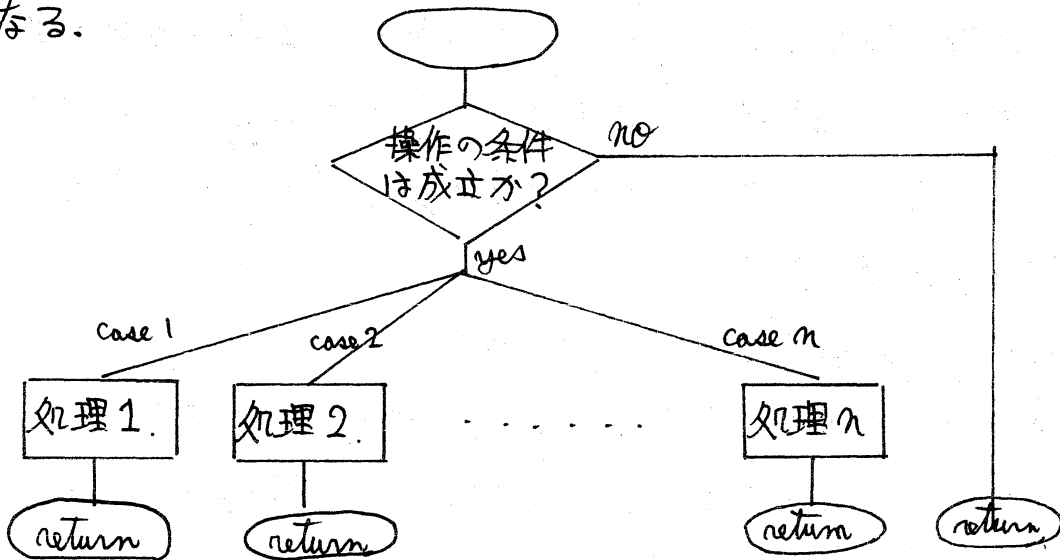


図 13

このルーチンの入口では1台だった計算機が、出口では n 台の計算機に分裂して、各々別々に後の処理を行なっていく必要がある。そこで、これを可能にする“CPU分裂命令”をもった言語を製作した。従来は、複数個の独立した仕事をさせるときでも、それらの処理順序を指定して、ともかく1列に並べなければならなかったが、この言語ではシステムが自動的に処理順序を定めて行なってくれるので、ユーザは、概念上複数台のコンピュータが同時に処理を行なっていると

思ってプログラミングができる。このため、プログラミングが大変楽になった。

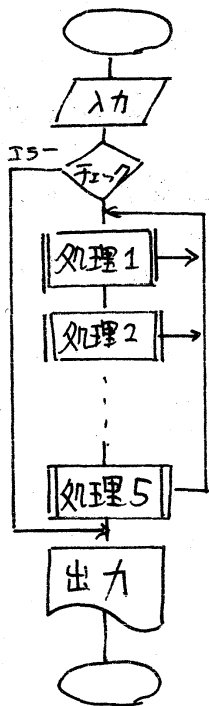
○ CPU 分裂命令の実際

計算機は実際は1台なので、分裂命令があっても、Case 1 の処理を続ける。しかし、1台目の計算機の処理が終了した時点で、Job 終了とせずに、先ほどの case 2 の処理をつづけて行なう。この2台目の処理も終わると3台目、4台目と順次以下同様に行ない。すべてのコンピュータが終わったとき、初めて Job 終了となる。ユーザにとっては、1台目、2台目かどういふ順序で処理されようと関係なく、結果としてすべてをつくしてくれればよく、この方法で問題ない。さて、ここで2台目のコンピュータが動き出すためには、何か必要かというところ、1台目のコンピュータが分裂命令を出した直前の状態を再現してやればよいことになる。この“状態”は、プログラムによって、何か必要で何か不要な情報かが異なるので、システムで行なうときは、全てを記憶する必要があり、メモリ効率がよくない。そこで、本言語では、必要なデータをユーザが指定し、ユーザプログラムで記憶させるという方式をとった。このため、コンパクトにしてから記憶させることができ、メモリの有効利用が計られた。私のプログラムの場合、実際記憶したのは、結び目の内部表現、作業ポインタ、

サブルーチンのネストの状況, 2台目のコンピュータが走り出すプログラムアドレス, さらに, デバッグのためのXメッセージである。そこで, それらを記憶するルーチン *Push*, 逆に記憶をもとにもどすルーチン *Pop* をユーザプログラム内に作成した。

○その他の命令.

算術命令は加減に限り, 数は16ビット精度の *integer* とした。用途がはっきりしているため, これに十分とした。変数はポインタ型と数値型の2種類。配列は1次元だけとした。文構造として, *if-then-else* 文, *for* 文, *while* 文, *repeat-until* 文を用いることにより, *goto-less* プログラムとなるようにした。しかし, 図14のように本実験では, 何らかの



処理(結び目の変形)を行なうと, 再びループの初めからチェックするアルゴリズムのため, ループから飛び出す *break* 文, ループの次回へスキップする *next* 文を採用した。この2文がなくても書くことはできるが, プログラムが煩雑になる恐れがあるので採用した。

ブロック構造は, *element* 内と外の

2種類だけである。しかし、1つの *element* 内に複数のルーチンを入れることができるようにしたため、関連の深いルーチンは、同一の *element* 内に作成し、極力変数は *local variable* として、外部からは見えられないようにし、モジュール化が計れた。

ポインタ型変数を採用したことにより、次の2種の記号 $!$ 、 $@$ を次の意味として用いた。 $!$ 名前は、名前のアドレス定数の意味で、 $A := !B$ により、ポインタ A が B を指すことになり、 $@$ 名前は、名前の指しているアドレスの内容の意味で、 $C := @A$ を実行すると、 C にポインタ A の指している変数、 C には B の内容が代入される。すなわち、 $!$ は代入文の右辺にあり、左辺にはポインタ型変数がある。また、 $@$ の直後の名前は、ポインタ型変数である必要がある。

システムで用意したカービス・ルーチンは、*standard I/O* モジュールで、*getC*, *getNo*, *putC*, *putNo*, *putNoF*, *putStr*, *putTab*, *putback* の8つである。 C は *character*, No は *number*, *str* は *string*, *Tab* は *tabration*, *NoF* は *number with format* の略である。文法については表の通りである。

なお、本言語作成に当っては、本学科(情報科学学科)・木村研の学生、久野 靖 氏の協力を得た。

文法:

エレメント = element 名前 ; [規制 ;]⁰⁰⁰ 文 { ; }⁰⁰⁰ end.

規制 = (global | external) 名前 { , }⁰⁰⁰

文 = procedure 名前 ; 文 { ; }⁰⁰⁰ end |

if 条件 then 文 [else 文] |

for 文 [step 文] [while 条件] do 文 [until 条件] |

while 条件 do 文 [until 条件] |

repeat 文 [until 条件] |

break | next | 式 |

call [a] 名前 [(式)] | return [(式)] |

(文 { ; }⁰⁰⁰) | <コード文 { ; }⁰⁰⁰>

コード文 = アセンブラ

条件 = (plus | not plus | minus | not minus | zero | not zero) [(式)] |

式 (> | >> | < | << | = | <=>) 変数

式 = [(レジスタ | 変数) :=]⁰⁰⁰ ([+|-] 数 | レジスタ

| 変数) [(+|-) (数 | レジスタ | 変数)]⁰⁰⁰

レジスタ = R0 | R1 | R2 | R3 | R4 | R5 | R6 | R7

変数 = [! | a] 名前 [(式)]

6. 実験結果

Alexander-Briggs の knot table の結び目をチェックしたところ、 8_4 までは全て同一の bridge number, 交点数の reduced knot form が得られたが、 8_5 は異なる交点数のものが見つかった。すなわち、目標3の反例が見つかったわけだが、さらに 8_{21} まで行なったところ、全て bridge number は同一であったので、目標1, 2については否定されていない。

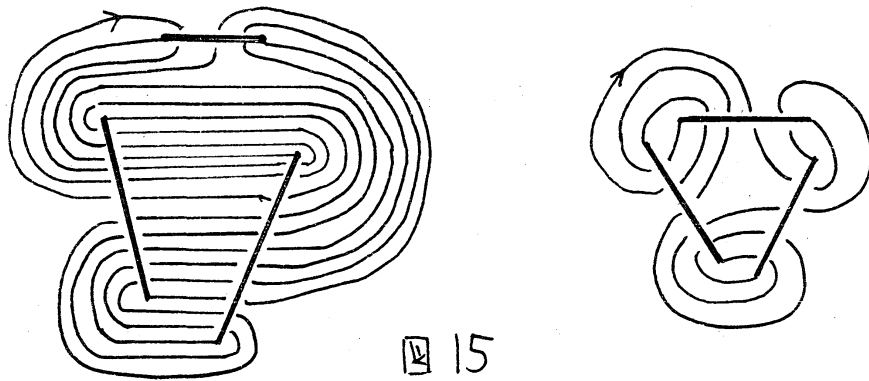


図 15

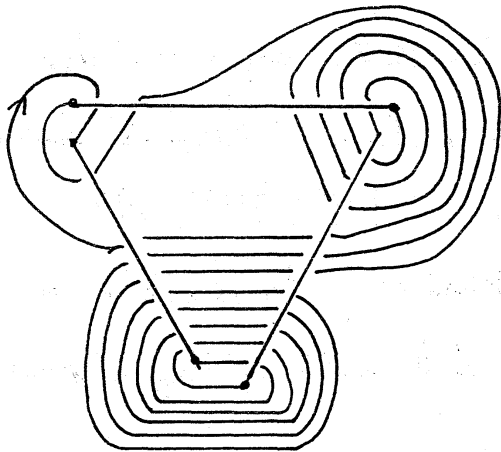
この実験のアルゴリズム4より明らかのように、次の結果も得られた。

定理 任意の non-trivial な結び目 K は次のような結び目群 πK の表示が可能である： $\pi K = \langle x_1, x_2, \dots, x_n \mid r_1, \dots, r_{n-1} \rangle$ において、 $\theta : \pi K \rightarrow \pi K / [\pi K, \pi K] = \mathbb{Z} : \text{homomorphism}$ とすると、任意の i, j に対し、 $\theta(x_i) = \theta(x_j)$ 。 V_c の word は、同一のアルファベットは並んで出現しない、又、奇数回現れるアルファベットは、必ず3回以上現れている。

この定理の条件をみたす表示は、*reduced knot form* から容易に求められる。この結果を利用して、結び目が、*trivial* でないことを示すアプローチとして、次の予想がある。

予想: 結び目 K を *trivial* でないとし、 $\pi K = \langle x_1, \dots, x_m \mid r_1, \dots, r_{m-1} \rangle$ を定理の条件をみたす群表示とする。このとき、 $G = \langle x_1, \dots, x_m \mid r_1, \dots, r_m, x_i^2 = 1 \ (i=1, 2, \dots, m) \rangle$ とおくと、 G も非可換群であろう。

この予想は、次の例では正しい。



$$\begin{aligned} \pi K &= \langle x, y, z \mid \\ &xyx^{-1}(zy^{-1})^2z^{-1}(yz^{-1})^2xy^{-1}, \\ &z(x^{-1}z)^3y^{-1}xy^{-1}x^{-1}y(z^{-1}x)^3 \rangle \\ G &= \langle x, y, z \mid (xy)^3, \\ &(yz)^5, (zx)^7 \rangle \end{aligned}$$

図 16

G が非可換なことは容易に示せる。よってこの結び目は、*trivial* でない。しかし、Alexander polynomial という不変量では、この結び目も、*trivial* な結び目も共に 1 であり区別できない。現在、この例の結び目を拡張した、ある種

の結び目の集合 (無限個) において予想が正しいことが示され、又、それらは全て、Alexander polynomial が1であるが、予想により non-trivial な結び目であることが示せることがわかっていく。

参考文献

- [1] Rolfsen: Knot and Links, Publish or Perish, Inc. 1976
- [2] R.H. Crowell and R.H. Fox (H. Terasaka, H. Noguchi): 結び目理論入門, 岩波書店, 現代科学選書
- [3] W. Haken: Theorie der Normalflächen, Acta math., 105 (1961) 245-375
- [4] G. DeGruise: _____, Dep. Math. Colorado State Univ. Sep (1975)
- [5] T. Homma, M. Ochiai: on relations of Heegaard diagrams and Knots, Math. Seminar Notes 6 (1978) 383-393
- [6] T. Homma, M. Ochiai, M. Takahashi: An algorithm for recognizing S^3 in 3-manifold with Heegaard splitting of genus two, to appear
- [7] O. A. Виро: Гипотеза Володина - Кузнецова - Фомченко о диаграммах хелпа трехмерной сферы не верна, Uspehi Mat. Nauk 32 (1977), No. 5 (197), 175-176
- [8] O. Morikawa: A counterexample to a conjecture of Whitehead, to appear