漢文訓読のアルゴリズムについて

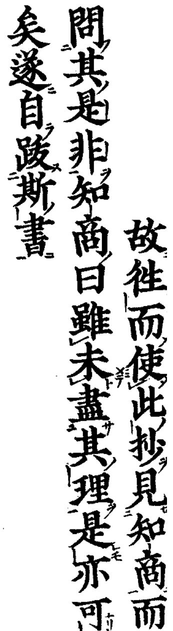
島野達雄

（近畿和算ゼミナール）

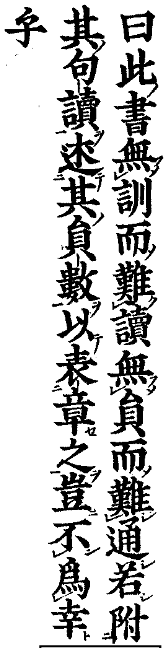
和算書の漢文を訓読するうちに、漢文から日本語への、返り点による語順変換のアルゴリズムが数学的に形式化できることを思いついたので報告する。この形式化を使うと、異なるｎ字からなる漢文の漢字列から生成される、日本語の漢字列の総数 a_n の漸化式を導くことができ、漢文から日本語への翻訳の可能性を計量的に考えることができる。

1．竪点の役割

漢文で書かれた和算書の序文や本文には、それぞれの漢字の左右や下に短い縦線を引いた例がよく登場する。貝原益軒はこの縦線を「竪点（たててん）」とよんでいる[1]。大別すると竪点には、（ｉ）漢字の音読み・訓読みの区別を指示する音符型、（ⅱ）つながれた２字以上の漢字を熟語としてあつかう熟語型、の二種類がある。


故徃而使此抄見知商而
問其是非知商曰雖未盡其理是亦可
矣遂自跋斯書

例文１


曰此書無訓而難讀無員而難通若附
其句讀述其員數以表章之豈不為幸
乎

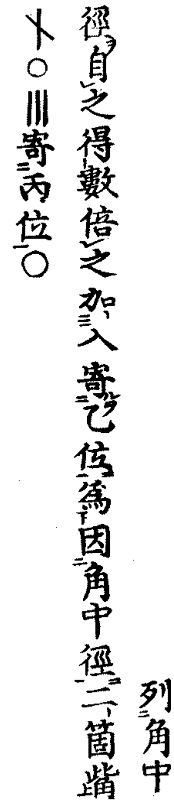
例文２

例文１（『竪亥録仮名抄』安藤有益序）の「是」「非」の右側にある竪点は、（ｉ）音符型の例。これらは「ぜ」「ひ」と音読みすることを示している。また、「徃而」「此抄」「其理」「是亦」「可矣」「斯書」の左側に添えられた竪点は、原則として、上にある漢字を訓読みすることを指示している。

（ⅱ）熟語型の例では、同じ例文１の「知商」にある竪点は、熟語（人名）として「知商」をあつかうことを示している。

例文２（同じ『竪亥録仮名抄』安藤有益序）では、「句讀」「員數」「表章」の中央に竪点があり、これらが熟語であることをあらわしている。

このうち「表章」は二点と竪点が複合している特別な例で、一点をもつ「之」から返って、「表章」と熟語で読む。


列角中
得自之得數倍之加入寄乙位為因角中徑二箇觜
以〇三寄丙位〇

例文３

例文３（『括要算法』巻３）の「加入」は三点と竪点が複合しており、一点

[1] 『点例』元禄 16 年序。現代では次頁の返読熟語型の竪点を漢字連結記号、ハイフンなどとよんでいる。

をもつ「(乙) 位」、さらに二点をもつ「寄」から返って、「加入」と読むようになっている。

ここで、一二点、上下点、天地点、東西点、甲乙点などを総称して順序点とよぶことにし、算用数字（1、2、3…）であらわすことにすると、「表章」や「加入」は順序点の２点以上と複合している、と言いかえることができる。

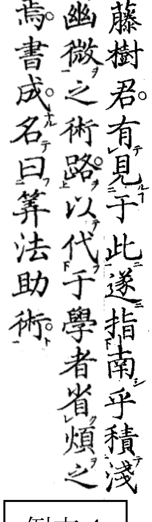
このように、順序点の２点以上と複合し、返読時に熟語としてまとめて返って読む働きをする竪点を返読熟語型とよび、単に熟語を指示するだけの働きをする（順序点の１点との複合を含む）竪点を非返読熟語型とよぶ。順序点の概念を導入した理由は、返読熟語型の竪点は一二点のほか上下点や天地点などの２点以上と複合することもあるからである。

返読熟語型の竪点は、語順変換のアルゴリズム（手順）に、きわめて重要な役割を果たしている。「表章之」を「之表章（之を表章す)」へ語順変換するには、レ点や一二点だけでは不十分で、「表章」に竪点を使うほかない。「加入寄乙位」から「乙位寄加入（乙位に寄るを加入す)」への語順変換は、レ点、一二三点などでは表現できず、「加入」に竪点を使わざるをえない。

逆に、１点から２点以上の順序点に返るときは、単漢字か返読熟語型の竪点でむすばれた２字以上の熟語しか返って読まれない。この「返って読まれる漢字列は、単漢字か竪点でむすばれた２字以上の熟語」という規則は、漢文から日本語への語順変換を規定する返り点・竪点システムを特徴づけている。

残念ながら、このように重要な役割をもつ竪点は、近現代ではほとんど無視されてきた。古くは服部宇之吉編「漢文体系」の『論語』、昭和 35 年の新釈漢文大系の『論語』、昭和 48 年の諸橋轍次『論語の講義』、和算界では例文３の『括要算法』を翻刻している『関孝和全集』などがいっさい竪点を使っていない。

2. 並置構造と入れ子構造


山本藤樹君。有見于此遂指南乎積淺近釋幽微之術路以代于學者省煩之暗記焉書成。名曰筭法助術

例文４

例文４（『算法助術』北村政房序）は、二つの文、つまり「山本藤樹君、有㆑見㆓于此㆒、遂指ー㆘南乎積㆓淺近㆒釋㆓幽微㆒之術路㆖、以代㆘于學者省㆑煩之暗記㆖焉」と「書成、名曰㆓筭法助術㆒」を並置した構造になっている。

前段の文は、「山本藤樹君」「有㆑見㆓于此㆒」「遂指ー㆘南乎積㆓淺近㆒釋㆓幽微㆒之術路㆖」「以代㆘于學者省㆑煩之暗記㆖焉」を並置している。さらに「山本藤樹君」は五つの単漢字「山」「本」「藤」「樹」「君」を並置していると言える。

一方、「指ー㆘南」と「路㆖」のあいだには「乎 (黙字)」「積㆓淺近㆒」「釋㆓幽微㆒」および単漢字「之」「術」の並置が入れ子になっている。同様に、「代㆘」と「記㆖」のあいだには、「于」「學」「者」「省㆑煩」「之」「暗」の並置が入れ子になっている。

このように、原則として、単漢字やレ点前後の２字の並置は、一二点をもつ漢字列の入れ子になり、レ点や一二点をもつ漢字列の並置は上下点をも

つ漢字列の入れ子になる。同じように、上下点は天地点の、天地点は東西点の入れ子になる。このような構造を入れ子構造とよぶ。

前節で、一二点、上下点、天地点、東西点などを順序点と総称し、算用数字であらわす、と述べたが、二つの順序点のあいだには、以上のように、どちらが入れ子になるか、上位、下位の順位が決まっている。

ちなみに、東西点を入れ子にもつ上位の返り点には甲乙丙丁…の甲乙点（十干点とも）があり、さらに上位には子丑寅卯…の十二支点、角亢氐房…の二十八宿点などがあるとのことである。

本稿では、有限個の漢字からなる漢字列の返り点・竪点システムをあつかっているので、並置や入れ子の重なり方は有限である。よって、基本となる「返読の構造」を明らかにし、それを有限の回数だけ並置したり入れ子にすればよいことになる。

3. 空の漢字列λとカッコつき表示

この節では、返読の構造を明らかにする。そのため、漢字がまったくあらわれない空の漢字列[2]（null string）の概念を導入する。便宜上、空の漢字列をλであらわす。

『発微算法』関孝和序の「引而不$_{\text{レ}}$發（引きて発せず）」（出典は『孟子』盡心上）のレ点は、λを利用すると、「引而不$_{\text{二}}\lambda$發$_{\text{一}}$」と一二点で表現できる。レ点だけをもつ漢字列は一二点の入れ子になることもあるので、この場合は、一二点よりも下位の、準一二点とよんだほうがよいかもしれない。

同じく『発微算法』関孝和序の「未$_{\text{レ}}$覩$_{\text{二}}$答書$_{\text{一}}$（いまだ答書を覩(み)ず）」は「未$_{\text{三}}\lambda$覩$_{\text{二}}$答書$_{\text{一}}$」と一二三点で表現できる。この場合は、レ点が一二点に吸収されたかたちになっている。

このようにλを使うと、レ点は一二点や上下点などの順序点と同等にあつかうことができる。レ点を含めた順序点を一般順序点とよび、あらためて算用数字であらわす。

一般順序点をもつ漢字列は、一般的に次のような形式をしている。

$$A\,B-C_{3}\,D\,E-F_{2}\,G\,H_{1} \qquad \cdots(3.1)$$

この漢字列の訓読は、はじめに一般順序点をもたないA、D、Gを読み、そのあと一般順序点にしたがって、H、E－F、B－Cを読む。つまり、

$$A\,B-C_{3}\,D\,E-F_{2}\,G\,H_{1}=A\,D\,G\,H\,E-F\,B-C=A\,D\,G\,H\,E\,F\,B\,C \quad \cdots(3.2)$$

と語順変換がおこなわれる。

ここで、Pを漢字１字または竪点でむすばれた２字以上の熟語、mを返り点や竪点をもった漢字列とするとき、二項演算P〈m〉をPとmの順序を入れ替えて並置する、と定義する。

$$P\langle m\rangle = m\,P$$

すると、漢字列ＡＢＣＤＥＦＧＨに対して、(3.1)の竪点はそのままにして、一般順序点の

[2] 水谷静夫『言語と数学』（森北出版、昭和 45 年）。null string には空連糸の訳語をあてている。

３点と２点の位置にカッコ開始記号〈　を挿入し、１点の位置に対応するカッコ終了記号〉をつけくわえた、

ＡＢ－Ｃ〈ＤＥ－Ｆ〈ＧＨ〉〉　　…(3.3)

は、Ｐ〈ｍ〉＝ｍＰの定義にもとづき、次のようにカッコと竪点をはずすことができる[3]。

ＡＢ－Ｃ〈ＤＥ－Ｆ〈ＧＨ〉〉
＝ＡＢ－Ｃ〈ＤＧＨＥ－Ｆ〉
＝ＡＤＧＨＥ－ＦＢ－Ｃ
＝ＡＤＧＨＥＦＢＣ

これは(3.2)の結果と等しい。つまり、(3.1)の一般順序点と竪点をもつ漢字列は、(3.3)のように表現しても同じ語順変換の結果をえる。(3.3)を(3.1)のカッコつき表示とよび、(3.2)の右辺を(3.3)のカッコをはずした結果とよぶ。

最後に、これまでにかかげた二つの例文の、返り点・竪点つきの漢字列、一般順序点・竪点つきの漢字列、およびそのカッコつき表示とカッコをはずした結果を示しておこう。

以表－$_{\text{二}}$章之$_{\text{一}}$豈不$_{\text{レ}}$為$_{\text{レ}}$幸乎　（『竪亥録仮名抄』安藤有益序）
＝以表－章$_{2}$之$_{1}$豈不$_{3}$為$_{2}$幸$_{1}$乎
＝以表－章〈之〉豈不〈為〈幸〉〉乎
＝以之表－章豈幸為不乎
＝以之表章豈幸為不乎（以て之を表章せば豈幸いと為さざらん）

山本藤樹君、有$_{\text{レ}}$見$_{\text{二}}$于此$_{\text{一}}$、遂指－$_{\text{下}}$南乎積$_{\text{二}}$淺近$_{\text{一}}$釋$_{\text{二}}$幽微$_{\text{一}}$之術路$_{\text{上}}$、以代$_{\text{下}}$于學者省$_{\text{レ}}$煩之暗記$_{\text{上}}$焉　（『算法助術』北村政房序）
＝山本藤樹君、有$_{3}$見$_{2}$于此$_{1}$、遂指－南$_{2}$乎積$_{2'}$淺近$_{1'}$釋$_{2'}$幽微$_{1'}$之術路$_{1}$、以代$_{2}$于學者省$_{2'}$煩$_{1'}$之暗記$_{1}$焉　（ダッシュ記号「´」は下位の一般順序点を示す）
＝山本藤樹君、有〈見〈于此〉〉、遂指－南〈乎積〈淺近〉釋〈幽微〉之術路〉、以代〈于學者省〈煩〉之暗記〉焉
＝山本藤樹君、于此見有、遂乎淺近積幽微釋之術路指南、以于學者煩省之暗記代焉（山本藤樹君、此に見ること有りて、遂に淺近を積みて幽微を釋くの術路を指南し、以て學者煩を省くの暗記に代う）

なお、レ点と竪点で返読を示す「以命$_{\text{レ}}$刊－刻（以て刊刻を命ず）」（『拾璣算法』凡例）、三点とレ点が複合した「黄帝使$_{\text{三レ}}$隷首作$_{\text{二}}$算数$_{\text{一}}$（黄帝隷首をして算数を作ら使$^{\text{し}}$む）」（『古今算法記』福本道閑序）のような、現代の返り点とは異なるつけ方をした文例も数多いが、このような文でも、返り点や竪点の指示にしたがえば、一定の語順がえられることは明らかであろう。言いかえれば、返り点・竪点システムは歴史的産物であるが、語順変換に関して一意性がある。

むろん、同じ「有朋自遠方来」という漢文（中国語）に対して、「有$_{\text{レ}}$朋自$_{\text{二}}$遠方$_{\text{一}}$来（朋

---

[3] ここではカッコを内側からはずしているが、外側からはずしても同じ結果になる。

有り遠方より来る)」と「有$_{下}$朋自$_{二}$遠方$_{一}$来$_{上}$(朋の遠方より来る有り)」と返り点を二通りにつけることができる。これらは翻訳者の考え方の違いを示しているものであり、返り点がみちびく漢字の語順はそれぞれ一定になっている。

4. 訓読アルゴリズムの形式化

ここからは、数学の言葉をつかって漢文訓読のアルゴリズムを形式的に述べる。

(1) 連接、返接、連結記号の定義

すべての「漢字」の集合をΣとする。ここでの「漢字」は無定義用語とする。

$$\Sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_n\}$$

Σの元によるすべての順列をΣ^*とし、「漢文言語」とよぶ。漢文言語Σ^*の元を「漢字列」とよぶ。

$$\Sigma^* = \{\lambda, \sigma_1, \cdots, \sigma_n, \sigma_1\sigma_1, \sigma_1\sigma_2, \cdots, \sigma_1\sigma_n, \sigma_2\sigma_1, \cdots\}$$

ここで、λは空順列を指し、「空の漢字列」とよぶ。

Σ^*の元u, vに対して、u, vを並べる二項演算、

$$u \cdot v = uv$$

を連接とよぶ。

同様に、Σ^*の元u, vに対して、u, vの順序を入れ替える二項演算、

$$u\langle v\rangle = vu$$

を返接とよぶ。

この返接演算において、uを構成する「漢字」が、たとえば

$$u = \sigma_3\sigma_1\sigma_3\sigma_2$$

であるとき、連結記号ーを用いて$u\langle v\rangle$次のように表記する。

$$u\langle v\rangle = \sigma_3 - \sigma_1 - \sigma_3 - \sigma_2\langle v\rangle$$

この例では、$u\langle v\rangle = v\sigma_3\sigma_1\sigma_3\sigma_2$であるので、$\sigma_3 - \sigma_1 - \sigma_3 - \sigma_2$を漢字1字のように扱ってもよいことがわかる。つまり、連結記号で結ばれた2字以上の漢字は漢字1字と見なせる。

以下、連接演算の・記号を省略し、連接と返接が続くときは返接演算が優先するとする。

(2) カッコつき表示の再帰的定義

漢文言語Σ^*において、「カッコつき表示」を次のように再帰的に定義する。

(ⅰ)漢字1字はカッコつき表示である。

(ⅱ)カッコつき表示とカッコつき表示の連接はカッコつき表示である。

(ⅲ)Pを漢字1字または連結記号で結ばれた2字以上の漢字、mをカッコつき表示とするとき、Pとmの返接　$P\langle m\rangle = mP$　はカッコつき表示である。

たとえば、$A, B, C \cdots$を漢字1字とするとき、

$$ABC$$

$$A\langle B\rangle A\langle C\rangle$$

$$A\langle BC\rangle DEA\langle B\langle C\rangle\rangle$$

$$A - B\langle C\rangle D - E\langle FG\langle C - D\langle E\rangle\rangle\rangle$$

などは、カッコつき表示になる。

カッコつき表示に対して、返接演算をおこない、連結記号を取り除いて漢字だけを並べた列を「カッコつき表示の展開」「カッコをはずした結果」などとよぶ。

5. 生成されるカッコつき表示の総数

次に、異なる n 字の漢字からなる漢字列が与えられたとき、そのカッコつき表示の総数を数えあげてみたい。

（1） n 個の漢字の連接とカッコつき表示

n 個の異なる漢字の連接を K_n と書く。

K_n に対して、漢字の順番を変えることなく、連結記号ー、カッコ開始記号$\langle$　、カッコ終了記号$\rangle$　の記号を挿入したカッコつき表示全体の集合を $[K_n]$ と書き、「K_n から生成されるカッコつき表示」とよぶ。

$[K_n],[K_m]$ に対して、集合の連接、返接を次のように定義する。

$$[K_n][K_m]=\{uv \mid u\in[K_n], v\in[K_m]\}$$

$$[K_n]\langle [K_m]\rangle=\{u\langle v\rangle \mid u\in[K_n], v\in[K_m]\}$$

$[K_n]$ の元の総数を $f([K_n])$ であらわす。（のちほど $f([K_n])$ を a_n と書きかえる）

すると、集合の連接、返接の定義より、

$$f([K_n][K_m])=f([K_n])f([K_m])$$

$$f([K_n]\langle [K_m]\rangle)=f([K_n])f([K_m])$$

が成り立つ。

なお、K_0 は空の漢字列 λ とし、便宜上 $[K_0]=\{\lambda\}$、 $f([K_0])=1$ と定義する。

（2）漢字１字、２字、３字から生成されるカッコつき表示の数

漢字１字つまり K_1 から生成されるカッコつき表示は、元の漢字１字のままなので、

$$f([K_1])=1$$

になる。

また、$K_2=AB$ から生成されるカッコつき表示は AB , $A\langle B\rangle$ の２通りしかない。

$$f([K_2])=2$$

$K_3=ABC$ から生成されるカッコつき表示 $[K_3]$ は、

(1) ABC

(2) $AB\langle C\rangle=ACB$

(3) $A\langle B\rangle C=BAC$

(4) $A\langle BC\rangle=BCA$

(5) $A\langle B\langle C\rangle\rangle=CBA$

(6) $A-B\langle C\rangle=CAB$

の６通りで、 $f([K_3])=6$ となり、 ABC の並べ替えの総数と一致する。

ここで $K_1=A, K_2=BC$ などと考えると、 $f([K_1])=1$，$f([K_2])=2$ であるので、

(1)(2) は $[K_1][K_2]$ $\quad f([K_1][K_2])=f([K_1])f([K_2])=1\times2=2$

(3) は $[K_1]\langle[K_1]\rangle[K_1]$ $\quad f([K_1]\langle[K_1]\rangle[K_1])=f([K_1])f([K_1])f([K_1])=1\times1\times1=1$

(4)(5) は $[K_1]\langle[K_2]\rangle$ $\quad f([K_1]\langle[K_2]\rangle)=f([K_1])f([K_2])=1\times2=2$

(6) は $[K_1]\langle[K_1]\rangle$ $\quad f([K_1]\langle[K_1]\rangle)=f([K_1])f([K_1])=1\times1=1$

から、次のように計算できる。

$$f([K_3])=f([K_1][K_2])+f([K_1]\langle[K_1]\rangle[K_1])+f([K_1]\langle[K_2]\rangle)+f([K_1]\langle[K_1]\rangle)$$
$$=2+1+2+1=6$$

（３）漢字４字の場合

$K_4=ABCD$ のすべての並べ替え（$4!=24$ 通り）のうち $BDAC, CADB, DACB, DBAC$ の４つは、 $ABCD$ にどのように返接と連結記号を組み合わせても導くことができない。

つまり、 K_4 から生成されるカッコつき表示の総数は、 $24-4=20$ 通りしかない。

その内訳を書き出してみよう。

（ｉ）A の直後にカッコ開始記号〈　や連結記号ーがないもの。

$K_1=A, K_3=B\ C$ と考えると、場合の数は、 $f([K_1][K_3])=1\times6=6$ 通り。

$ABCD$

$ABC\langle D\rangle=ABDC$

$AB\langle C\rangle D=ACBD$

$AB\langle CD\rangle=ACDB$

$AB\langle C\langle D\rangle\rangle=ADCB$

$AB-C\langle D\rangle=ADBC$

（ⅱ）A の直後にカッコ開始記号〈　があるもの。10通り。

(1) $[K_1]\langle[K_1]\rangle[K_2]$ の形式。 $f([K_1]\langle[K_1]\rangle[K_2])=1\times1\times2=2$

$A\langle B\rangle CD=BACD$

$A\langle B\rangle C\langle D\rangle=BADC$

(2) $[K_1]\langle[K_2]\rangle[K_1]$ の形式。 $f([K_1]\langle[K_2]\rangle[K_1])=1\times2\times1=2$

$A\langle BC\rangle D=BCAD$

$A\langle B\langle C\rangle\rangle D=CBAD$

(3) $[K_1]\langle[K_3]\rangle$ の形式。 $f([K_1]\langle[K_3]\rangle)=1\times6=6$

$A\langle BCD\rangle=BCDA$

$A\langle BC\langle D\rangle\rangle=BDCA$

$A\langle B\langle C\rangle D\rangle = CBDA$

$A\langle B\langle CD\rangle\rangle = CDBA$

$A\langle B\langle C\langle D\rangle\rangle\rangle = DCBA$

$A\langle B-C\langle D\rangle\rangle = DBCA$

（iii）Aの直後に連結記号ーがあるもの。4通り。

(1) $[K_1]\langle [K_1]\rangle [K_1]$の形式。$f([K_1]\langle [K_1]\rangle [K_1])=1\times 1\times 1=1$

$A-B\langle C\rangle D = CABD$

(2) $[K_1]\langle [K_2]\rangle$の形式。$f([K_1]\langle [K_2]\rangle)=1\times 2=2$

$A-B\langle CD\rangle = CDAB$

$A-B\langle C\langle D\rangle\rangle = DCAB$

(3) $[K_1]\langle [K_1]\rangle$の形式。$f([K_1]\langle [K_1]\rangle)=1\times 1=1$

$A-B-C\langle D\rangle = DABC$

以上から、$f([K_4])$は、（ｉ）（ⅱ）（ⅲ）の合計20になる。

（４）n個の漢字から生成されるカッコつき表示の総数

$f([K_n])$も$f([K_3])$や$f([K_4])$と同様に求めることができる。

（ｉ）最初の漢字の直後にカッコ開始記号〈　や連結記号ーがないもの。

$$f([K_1][K_{n-1}])=f([K_{n-1}])$$

（ⅱ）最初の漢字の直後にカッコ開始記号〈　があるもの。

$0\le r\le n-2$である整数rに対して、一般に

$$[K_1]\langle [K_{n-r-1}]\rangle [K_r]$$

と書くことができる。よって、求める場合の数は、

$$\sum_{r=0}^{n-2} f([K_1]\langle [K_{n-r-1}]\rangle [K_r])$$

$$= f([K_1]\langle [K_{n-1}]\rangle [K_0])+\sum_{r=1}^{n-2} f([K_1]\langle [K_{n-r-1}]\rangle [K_r]) \qquad (f([K_0])=1)$$

$$= f([K_{n-1}])+\sum_{r=1}^{n-2} f([K_r])f([K_{n-r-1}])$$

（ⅲ）最初の漢字の直後に連結記号ーがあるもの。

$1\le i+j\le n-2$である整数$i\ge 1$、整数$j\ge 0$に対して、一般に

$$[K_1]\langle [K_i]\rangle [K_j]$$

と書くことができる。ここで、$[K_1]$は連結記号で結ばれた２字以上の漢字である。

よって、求める場合の数は、

$$\sum_{i+j=1}^{n-2} f\left([K_1]\langle [K_i]\rangle [K_j]\right) = \sum_{i+j=1}^{n-2} f([K_i])\, f\left([K_j]\right)$$

となる。

以上から、

$$\begin{aligned} f([K_n]) &= f([K_{n-1}]) + f([K_{n-1}]) + \sum_{r=1}^{n-2} f([K_r])\, f([K_{n-r-1}]) + \sum_{i+j=1}^{n-2} f([K_i])\, f\left([K_j]\right) \\ &= 2f([K_{n-1}]) + \sum_{r=1}^{n-2} f([K_r])\, f([K_{n-r-1}]) + \sum_{i+j=1}^{n-2} f([K_i])\, f\left([K_j]\right) \end{aligned}$$

を得る。ただし、$\sum_{i+j=1}^{n-2}$ は$i \geq 1,\ j \geq 0$の条件で総和をとる。

6. 漢文から日本語への翻訳の可能性

（1）漸化式の変形

ここからは、$f\left([K_n]\right)$をa_nであらわす。

すると、５.（４）の最後の式は、

$$\begin{aligned} & a_n \\ &= 2a_{n-1} + \sum_{i=1}^{n-2} a_i a_{n-i-1} + \sum_{\substack{i+j=1 \\ i\geq 1, j\geq 0}}^{n-2} a_i a_j \\ &= 2a_{n-1} + (a_1 a_{n-2} + \cdots + a_{n-2} a_1) + \{a_1 a_0 + (a_1 a_1 + a_2 a_0) + \cdots + (a_1 a_{n-3} + \cdots + a_{n-2} a_0)\} \\ &= 2a_{n-1} + \{a_1 a_0 + (a_1 a_1 + a_2 a_0) + \cdots + (a_1 a_{n-3} + \cdots + a_{n-2} a_0)\} \\ &\quad + (a_1 a_{n-2} + \cdots + a_{n-2} a_1 + a_{n-1} a_0) - a_{n-1} a_0 \\ &= a_{n-1} + \sum_{i+j=0}^{n-2} a_{i+1} a_j \end{aligned}$$

と変形できる。ここで、数列$\{a_n\}$の階差数列$\{b_n\}$を考える。

$$b_n = a_{n+1} - a_n = \sum_{i+j=0}^{n-1} a_{i+1} a_j$$

$$b_{n+1} = \sum_{i+j=0}^{n} a_{i+1} a_j = b_n + (a_1 a_n + \cdots + a_n a_1 + a_{n+1} a_0)$$

さらに、数列$\{b_n\}$の階差数列$\{c_n\}$を考えると、

$$c_n = b_{n+1} - b_n = a_1 a_n + \cdots + a_n a_1 + a_{n+1} a_0 = \sum_{i=0}^{n} a_{i+1} a_{n-i}$$

より、

$$a_{n+1} = a_n + b_n = a_n + (b_{n-1} + c_{n-1}) = a_n + (a_n - a_{n-1}) + \sum_{i=0}^{n-1} a_{i+1} a_{n-i}$$

であるから、

$$a_{n+1} = a_n + (a_n - a_{n-1}) + (a_1 a_{n-1} + a_2 a_{n-2} + \cdots + a_{n-2} a_2 + a_{n-1} a_1 + a_n a_0)$$
$$= 3a_n + a_{n-1} + \sum_{i=2}^{n-2} a_i a_{n-i} \qquad (n \geq 4)$$

を得る。

（２） a_n の意味するもの

a_n は、ｎ個の異なる漢字からなる漢文の漢字列を日本語の語順に並び替えるとき、高々、何通りの並び替えの可能性があるか、を示している。

言うまでもなくこの可能性の総数はｎ！（ｎの階乗）であるが、このうち「連結記号（—）と返り点（レ、一二…、上下、天地など）を使った並び替えの場合の数 a_n 」が漸化式（差分方程式）であらわされることが、これまでの結論であった。

この漸化式を実際に計算すると、次の表になる。

n	$n!$	a_n	$k_n = a_n \div n!$	（参考）	$r_n = a_{n+1} \div a_n$	$k_{n+1} \div k_n$
0	1	1	1.00000		1.00000	1.00000
1	1	1	1.00000		2.00000	1.00000
2	2	2	1.00000		3.00000	1.00000
3	6	6	1.00000		3.33333	0.83333
4	24	20	0.83333		3.50000	0.70000
5	120	70	0.58333		3.62857	0.60476
6	720	254	0.35278		3.73228	0.53318
7	5040	948	0.18810		3.81646	0.47706
8	40320	3618	0.08973		3.88557	0.43173
9	362880	14058	0.03874		3.94309	0.39431
10	3628800	55432	0.01528		3.99159	0.36287
11	39916800	221262	0.00554		4.03298	0.33608
12	479001600	892346	0.00186		4.06869	0.31298
13	6227020800	3630680	0.00058		4.09979	0.29284
14	87178291200	14885042	0.00017			

この表は、たとえば、「１４字の異なる漢字からなる漢文（中国語）が与えられたとき、日本語の語順として成り立つ可能性があるのは、並び替えの全体の 0.00017 程度しかない」ことを明らかにしている。

ｎが大きくなればなるほど、この割合は減少する。

以上、この節では、カッコつき表示という形式化により、漢文から日本語への翻訳の可能性を計量的に考えることができることを示した。

（了）