一般順序点と訓読木を導入した訓読アルゴリズムの設計と実装

鵜久森 将隆† 島野 達雄‡ 望月 久稔†
†大阪教育大学 ‡関西学院大学

1. はじめに

日本では、論語・史記・周髀算経・黄帝内経・白氏文集などの中国語を読むための訓読の技法が古くから発達してきた。

訓読とは「漢文のもとの形を保ったまま，そこに返り点や送り仮名を付け加えて，本来は外国語である漢文を日本語として読む一種の翻訳方法」[3] である．しかし訓読は原文の語順を返り点に添って読むため，漢字を読む順が前後して読みづらい．

現在インターネット上には多くの漢籍が公開されており，それらは原文の語順であることが多い．そこで本論文では，一般順序点と訓読木を導入した訓読アルゴリズムを提案し，返り点がついた漢文から日本語の語順への自動的な並べ替えを図る．

2. 一般順序点

漢文の教科書 [3] では返り点を「レ点は連続する二字について前後を入れかえて読む」「一二点は二字以上離れた下の字から上の字に帰って読む」と説明している．しかしレ点は「一レ点」や「レ点の後ろが順序点を持つ漢字」のときなどにより振る舞いが変わる．ここで、順序点とは「一二点」「上下点」「甲乙点」などを総称したものであり、順序点に「レ点」を付け加えた返り点全体を一般順序点と呼ぶ [1]．一般順序点の導入により「レ点」を順序点と同様に扱い，処理の単純化と効率化を図る．

本論文では一般順序点を算用数字で表し，「一二点」などと同様に 1 点の後に 2 点，2 点の後に 3 点の漢字を読むと定義する．

2.1 置換方法

現代の返り点のつけ方には「順序点とレ点の連続は順序点が 1 点でのみ発生する」，「レ点の後ろに 1 点をもつ漢字は来ない」という規則が一般的に存在する [1]．規則を考慮すると置換方法は次の 4 つのパターンに分けられる．

順序点 一二点なら 1 点 2 点，上中下点なら 1 点 2 点 3 点のように，読む順番に対応する一般順序点を置く（表 1a).

レ点の連続 レ点が n 個続く漢字列の末尾の漢字に一般順序点 1 を置き，その前の漢字の順序点は順に $n+1$, $n\cdots$, 2 を置く（表 1b).

レ点の後ろに順序点をもつ漢字 レ点の後ろが順序点をもつ漢字のとき，その順序点が一般順序点 i ならレ点の位置には $i+1$ を置く（表 1c).

1 レ点 1 点とレ点が重なる 1 レ点のとき，1 レ点に続く漢字 A の一般順序点が i であれば 1 レ点をもつ漢字 Q には $i+1$, Q の順序点と同種の順序点 k をもつ P には $i+k$ を置く（表 1d).

表 1 一般順序点の置換方法

	置換前	置換後
(a) 順序点	$A_{下}BC_{二}DE_{一}FG_{上}$	$A_{2'}BC_2DE_1FG_{1'}$
(b) レ点の連続	$A_{レ}B_{レ}\cdots{}_{レ}C$	$A_{n+1}B_{n}\cdots{}_2C_1$
(c) レ点の後ろに順序点を持つ漢字	$A_{レ}B_i\ldots D_1$	$A_{i+1}B_i\ldots D_1$
(d)1 レ点	$P_k\ldots Q_{一レ}A\ldots$	$P_{i+k}\ldots Q_{i+1}A_i\ldots$

2.2 置換アルゴリズム

一レ点，レ点の後ろに順序点をもつ漢字があるパターンを解決するため，置換アルゴリズムは一般順序点を挿入する位置を記憶するレ点スタックと順序点スタックをそれぞれ別に用意する．一二点，上下点，甲乙点などの入れ子となる順序点に対応するために順序点スタックを複数作成し，作成した順にレベル 0, 1, ... と定義する．例えば上下点が出てくる場合はレベル 0〜1 のスタック，天地点が出てくる場合はレベル 0〜3 のスタックを利用する．

以下に置換アルゴリズムを示す．同種の順序点で最大のものを「最大点」，一点上点など最小のものを「開始点」，それ以外を「途中点」とよぶ．特に一レ点，上レ点などを「開始レ点」とよぶ．

(手順 1) 初期設定 スタックレベルを-1 に設定する．

(手順 2) 置換処理 終端文字まで手順 2 を繰り返す．漢字と返り点を読み，漢字を出力する．返り点により条件分岐する．

[条件イ] 最大点または途中点 最大点であればスタックレベルを 1 つ上げる．レ点スタックが空でなければ，レ点スタックの内容をすべて現レベルの順序点スタックに移動する．挿入位置を順序点スタックに積む．

[条件ロ] 開始点 1 を出力し，順序点スタックが空になるまで挿入位置を取り出して一般順序点を順に挿入する．スタックレベルを 1 つ下げる．

[条件ハ] レ点 挿入位置をレ点スタックに積む．

[条件ニ] 開始レ点 現レベルの順序点スタックの内容をすべてレ点スタックに移動する．挿入位置をレ点スタックに積む．スタックレベルを 1 つ下げる．

[条件ホ] 返り点無し レ点スタックが空でなければ 1 を出力し，レ点スタックが空になるまで挿入位置を取り出して一般順序点を順に挿入する．

文 (1) に置換アルゴリズムを適用した例を表 2 に示す．'#' は終端文字を表す．漢字の位置番号を先頭から 0, 1, ... とし，スタック内のデータは一般順序点を挿入する位置の番号を示す．表中のステップ 1 において，レ点をもつ漢字 A では条件ハにより位置番号 0 を積む．ステップ 2 において，下点をもつ漢字 B は条件イに分岐する．レ点スタックが空でないためレ点スタックの内容を順序点スタック 0 に移し，位置番号 1 を積む．ステップ 4 において，漢字 E は返り点を持たないので条件ホに分岐するが，レ点スタックが空なので漢字 E を出力するのみの処理となる．続いて一レ点をもつ漢字 F は条件ニにより順序点スタック 1 の内容をレ点スタックに移し，位置番号 5 を積む．ステップ 5 において，漢字 G は条件ホに分岐する．レ点スタックが空でないため挿入位

Proposal design and implement of Algorithm of Kanbun Reading which innovate General Order Mark and Reading Tree

† Masataka UGUMORI, Hisatoshi MOCHIZUKI
Osaka Kyoiku University
‡ Tatsuo SHIMANO
Kwansei Gakuin University

表 2　置換アルゴリズムの例

ステップ	出力	入力	レ点スタック	順スタック 0	順スタック 1	動作
0		$A_{レ}B_{下}CD_{二}EF_{一レ}GH_{上}I\#$				
1	$A_{□}$	$B_{下}CD_{二}EF_{一レ}GH_{上}I\#$	0			条件ハ
2	$A_{□}B_{□}$	$CD_{二}EF_{一レ}GH_{上}I\#$		0 1		条件イ
3	$A_{□}B_{□}CD_{□}$	$EF_{一レ}GH_{上}I\#$		0 1	3	条件ホ，条件イ
4	$A_{□}B_{□}CD_{□}EF_{□}$	$GH_{上}I\#$	3 5	0 1		条件ホ，条件ニ
5	$A_{□}B_{□}CD_3EF_2G_1$	$H_{上}I\#$		0 1		条件ホ
6	$A_{3'}B_{2'}CD_3EF_2G_1H_{1'}I$	#				条件ロ，条件ホ

置を取り出し，一般順序点を順に挿入する．

$$A_{レ}B_{下}CD_{二}EF_{一レ}GH_{上}I \quad (1)$$

3．訓読木

訓読木 [1] は漢文の構造を二分木で表したものであり，内部節点は「連節」「返節」[1]，外部節点は漢字の情報をもつ．訓読木を以下に示すルールにより走査することで，訓読漢字列を得る．

連節節点　左のリンクをたどり，その後右のリンクをたどる．
返節節点　右のリンクをたどり，その後左のリンクをたどる．
漢字節点　漢字を読む．

また，外部節点を左から順に並べることで元の漢字列を得る．例として文 (1) の訓読木を図 1 に示す．連節節点は「●」，返節節点は「○」で示している．

3.1　構築アルゴリズム

一般順序点に置換した漢字列から漢字を 1 字ずつ参照し，連節節点，返節節点，漢字節点を作成し訓読木を構築する．各節点は，漢字，連節，返節の情報をもつ *Info*，右リンク *Right*，左リンク *Left* をもち，リンクのつなげ方は漢字のもつ順序点によって異なる．構築アルゴリズムを以下に示す．現在参照している節点を *cur* とし，処理の簡略化のため根の節点をリンクにもつダミー節点を設ける．

(手順 1)　初期設定　ダミー節点を作成し，*cur* とする．
(手順 2)　節点とリンク情報の作成　終端文字まで手順 2 を繰り返す．漢字と順序点を読み込む．このとき順序点の種類により条件分岐する．
[条件イ] 最大点　連節節点を作成し，*cur* の *Right* からリンクする．作成した連節節点の位置をスタックに積む．返節節点を作成し，作成した連節節点の *Left* からリンクする．*cur* を作成した返節節点に更新する．読み込んだ漢字の節点を作成し，*cur* の *Left* からリンクする．
[条件ロ] 途中点　返節節点を作成し，*cur* の *Right* からリンクする．*cur* を作成した返節節点に更新する．読み込んだ漢字の節点を作成し，*cur* の *Left* からリンクする．
[条件ハ] 開始点　読み込んだ漢字の節点を作成し，*cur* の *Right* からリンクする．*cur* をスタックから取り出した節点に更新する．
[条件ニ] 返り点無し　連節節点を作成し，*cur* の *Right* からリンクする．*cur* を作成した連節節点に更新する．読み込んだ漢字の節点を作成し，*cur* の *Left* からリンクする．
(手順 3) 余分な節点の削除　*cur* へのリンクを *cur* の *Left* リンク先の節点に付け替え，*cur* を削除する．

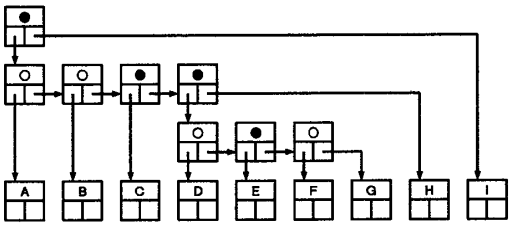

図 1　訓読木

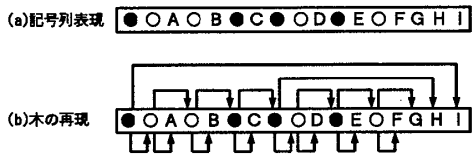

図 2　記号列表現と木の再現

文 (1) を一般順序点に置換した文 “$A_{3'}B_{2'}CD_3EF_2G_1H_{1'}I$”から，構築アルゴリズムに従い，訓読木を構築すると図 1 になり，走査することで訓読文 “CEGFDHBAI”が得られる．

3.2　訓読木の記号列表現

記号列表現は訓読木の節点を前順序で並べたものである．例えば図 1 の木は，図 2a のように表現できる．

記号列を左から順に参照し，連節もしくは返節を通りすぎたときに一つ手前からリンクすることで *Left* リンクを再現できる．また，漢字を通りすぎたときに最後に現れた *Right* リンクを持たない連節もしくは返節からリンクすることで，*Right* リンクを再現できる．再現した木構造を図 2b に示す．

記号列表現は，木構造の保存と，訓読漢字列や元の漢字列を再度得るのが容易である．

4．おわりに

本論文では一般順序点と訓読木を導入した訓読アルゴリズムを実装し，返り点がついた漢文から日本語の語順への自動的な並べ替えを実現した．今後の課題として送り仮名や再読文字への対応が挙げられる．

参考文献

[1] 島野達雄，湯城吉信．漢文訓読の数学的構造について．大阪府立工業高等専門学校研究紀要，第 41 巻．pp.25-28，2007.
[2] 島野達雄．漢文訓読のアルゴリズムについて．日本数学史学会，第 14 回数学史研究発表会．2007.
[3] 木村博．国語総合．pp.287-326，筑摩書房，2007.