3D4-3

文節結合による回文の自動生成

A Search Framework for Automatic Generation of Japanese Palindromes

鈴木 啓輔　　佐藤 理史
Keisuke Suzuki　　Satoshi Sato

名古屋大学大学院工学研究科電子情報システム専攻
Department of Electrical Engineering and Computer Science, Guraduate school of Engineering, Nagoya University

This paper proposes an automatic generator of Japanese palindromes. Because palindromes have to be reversible, creation of palindromes is not an easy task for us. The proposed method is a search framework. In this framework, a state corresponds to a bunsetsu sequence, which can be a sub-sequence of a palindrome. The application of an operator to a state produces a new state that corresponds to a longer bunsetsu sequence, by attaching a connectable bunsetsu to the original state. In an experiment, from a 3.5M bunsetsu list, our effective search algorithm using a lookahead function and macro operators has produced all possible 2/3/4-bunsetsu palindrome candidates that satisfy the reversible condition and syntactic requirements. Among these candidates, there are both known and unknown palindromes.

1. はじめに

回文は、平安時代から存在する伝統的ことばあそびであり、古くから人々に親しまれてきた。『図説 ことばあそび遊辞苑』[1] では、回文を、「頭 (かしら) から読んでも尻から読んでも同じ音で、どちらも無理なく意味が通じる語句や文章 (p301)」と説明している。

回文は、音の並びに対する厳しい制約を満たす必要があるため、不自然な文になりやすく、日本語を母国語とする我々でも作りにくい文である。我々は、このような作成難度の高い回文を自動生成することに挑戦する。

回文自動生成の先行研究は、いくつか存在する。日本語の回文自動生成の研究には、小野ら [2] の遺伝的アルゴリズムに基づく回文自動生成がある。英語の回文自動生成には、名詞のみの回文を作成する Hoey[3] の試みや、パズルのように事前に用意した単語ペアを組み合わせる O'Connor[4] の試みがある。これらの手法により生成される回文は、人間が作る回文ほどに意味が通っているとは言い難い。

我々は、人間が作る回文と同程度に意味の通った回文の生成を最終目標とする。本論文では、その第一段階として、特定の文節数の回文候補を、現実的な時間で大量に生成する手法を提案する。さらに、この手法が生成する候補の中には、目標とする回文が含まれることを実験的に示す。

2. 回文生成法

2.1 回文の条件

先に述べたように、回文とは「頭から読んでも尻から読んでも同じ音で、どちらも無理なく意味が通じる語句や文章」である。これを次の二つの条件に分ける。

1. **回文条件**　頭から読んでも尻から読んでも同じ音
2. **通意条件**　無理なく意味が通じる

このうち、通意条件は、「日本語の表現として文法的に適格である」という文法的な条件を内在する。これを**文法的適格性**として独立させ、通意条件を以下の 2 つに分割する。

2a. **文法的適格性**　日本語の表現として文法的に適格である
2b. **意味的適格性**　全体として無理なく意味が通じる

本研究では、1, 2a, 2b の 3 つの条件をすべて満たす日本語表現を回文と認定する立場をとる。

ある表現が回文条件を満たすか否かは、漢字の読みさえ分かれば容易に判定できる。文法的適格性は、ある範囲で文法を定めれば、それに合致するかどうか判定できる。これに対して、意味的適格性を自動的に判定することは、現在の技術では容易ではない[*1]。そこで本論文では、意味的適格性を除いた、1 と 2a の条件を満たす回文候補を自動生成の対象とする。

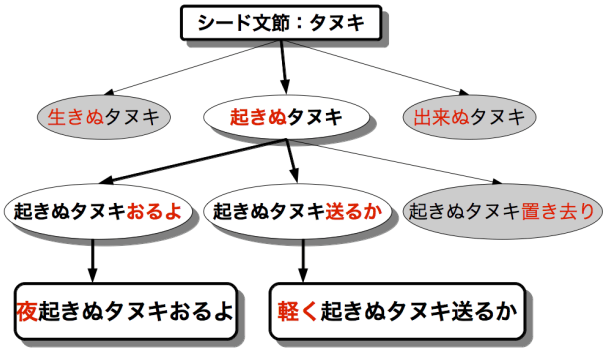

図 1: 回文候補生成法

2.2 文節結合法

本論文では、種となる文節 (**シード文節**) の前後に、別の文節を結合していくことで、回文候補を生成する (図 1 参照)。これを**文節結合法**と名付ける。文節結合法を探索問題として定式化するために、状態とオペレータを定義する。

連絡先: 鈴木啓輔, 名古屋大学大学院工学研究科電子情報システム専攻, 愛知県名古屋市千種区不老町 IB 電子情報館南棟 1 階, (052)789-4435, suzuki@sslab.nuee.nagoya-u.ac.jp

*1　意味的適格性の自動判定技術を研究することが、回文自動生成研究の究極の目的である。

2.2.1 状態の定義

文節結合法の状態は、連結された文節列である。これをひらがなで表記しておくと都合がよい[*2]。この他に、どこで折り返すかという折り返し点の情報が必要である。たとえば、図 1 の「おきぬたぬき (起きぬタヌキ)」では、折り返し点は「た」である。

折り返し点を定めると、**不足文字列**が定まる。不足文字列とは、現在の状態を回文条件を満たす状態にするために最低限必要な文字列のことである。たとえば、上記の例では、現在の状態の末尾に「お」という文字が連結されなければ、回文とはなりえない。不足文字列は、存在するならば、先頭 (左) または末尾 (右) のいずれか一方に存在する。不足文字列が存在しない状態は、回文条件を満たす状態である。

以上に基づき、状態を、連結された文節列のひらがな表記 H、先頭 (左) に存在する不足文字列 L、末尾 (右) に存在する不足文字列 R、の 3 つ組で定義する。

$$\langle L, H, R \rangle \tag{1}$$

例えば、折り返し点が「た」である「おきぬたぬき (起きぬタヌキ)」は、次のように表現される。

$$\langle \epsilon, おきたぬき, \overset{お}{\square} \rangle$$

ここでは、分かりやすさのため、不足文字列をルビ付きの空欄で示した。

2.2.2 オペレータの定義

文節結合法のオペレータは、状態 $\langle L, H, R \rangle$ にある文節 w を結合し、新しい状態 $\langle L', H', R' \rangle$ をつくり出す操作に対応する。ここでは、文節 w を左または右に結合する 2 種類の**基本オペレータ**を定義する。なお、基本オペレータが適用可能かどうかの条件は、L, R, w によって定まり、H は関与しない点に注意されたい。

左延長オペレータ ExtendLeft(w)

if $L \neq \epsilon$ and $\mathit{suffix}(L, |w|) = w$ **then**
$\langle \mathit{prefix}(L, |L| - |w|), wH, \epsilon \rangle$
elsif $L \neq \epsilon$ and $\mathit{siffix}(w, |L|) = L$ **then**
$\langle \epsilon, wH, \mathit{reverse}(\mathit{prefix}(w, |w| - |L|)) \rangle$

右延長オペレータ ExtendRight(w)

if $R \neq \epsilon$ and $\mathit{prefix}(R, |w|) = w$ **then**
$\langle \epsilon, Hw, \mathit{suffix}(R, |R| - |w|) \rangle$
elsif $R \neq \epsilon$ and $\mathit{prefix}(w, |R|) = R$ **then**
$\langle \mathit{reverse}(\mathit{suffix}(w, |w| - |R|)), Hw, \epsilon \rangle$

上記のオペレータを定義するために、以下の関数および記号を使用した。

1. $\mathit{prefix}(p, i)$: 文字列 p の長さ i の先頭部分列 (prefix) を取り出す関数
2. $\mathit{suffix}(p, i)$: 文字列 p の長さ i の末尾部分列 (suffix) を取り出す関数
3. $\mathit{reverse}(p)$: 文字列 p の文字の並びを逆順に並び変えた文字列を返す関数
4. $|p|$: 文字列 p の文字数

例えば、状態 $\langle \overset{き}{\square}\overset{ぬ}{\square}, たぬき, \epsilon \rangle$ に対しては、

1. ExtendLeft(おきぬ) は、$\mathit{suffix}(おきぬ, 2) =$ ”きぬ” なので、適用可能であり、新たな状態 $\langle \epsilon, おきぬたぬき, \overset{お}{\square} \rangle$ が生成される。
2. ExtendLeft(はしる) は、適用できない。

以下に、シード文節「タヌキ」から、回文候補「夜起きぬタヌキおるよ」が生成される過程を示す。

1. 初期状態 $\langle \overset{き}{\square}\overset{ぬ}{\square}, たぬき, \epsilon \rangle$ が与えられる。
2. この状態に ExtendLeft(おきぬ) を適用し、
   状態 $\langle \epsilon, おきぬたぬき, \overset{お}{\square} \rangle$ を得る。
3. この状態に ExtendRight(おるよ) を適用し、
   状態 $\langle \overset{よ}{\square}\overset{る}{\square}, おきぬたぬきおるよ, \epsilon \rangle$ を得る。
4. この状態に ExtendLeft(よる) を適用し、
   状態 $\langle \epsilon, よるおきぬたぬきおるよ, \epsilon \rangle$ を得る。
5. この状態は回文条件を満たすので、回文候補として出力する。

2.3 初期状態の生成

文節結合法では、入力としてシード文節が与えられると仮定する。ひとつの文節に対して、可能な折り返し点は複数存在する (少なくとも 4 つ存在する)。初期状態の生成では、与えられた文節の可能な折り返し場所をすべて求め、そのそれぞれに対して状態を生成する。

2.4 文法的適格性のチェック

上記で定義した状態とオペレータにより、回文条件を満たす回文候補が生成できる。回文条件に加えて、文法的適格性も満たすために、次の 2ヶ所に、文法的適格性のチェックを導入する。

1. オペレータ適用時のチェック

   オペレータを適用する際に、適用対象となる状態 (文節列) と結合する文節が文法的に連結可能かどうかをチェックする。具体的には、17 種類の文節タイプを導入し、文節タイプ間に連接可能かどうかを定義する。この定義に従って、新たに結合する文節が直前または直後の文節と連接可能かどうかをチェックし、連接不能の場合は、オペレータを適用しないこととする。

2. 最終状態のチェック

   最終状態が見つかった後、その状態 (文節列) に対して、可能な文節係り受け構造が存在するかどうかをチェックする。具体的には、前述の 17 種類の文節タイプ間に係り受けが可能かどうかを定義する。この定義を満たし、かつ、非交差条件を満たす係り受け構造が存在した場合のみ、その状態を出力する。

3. 探索の効率化

理論的には、前章の方法と、オペレータで結合する文節の候補集合 (**文節データベース** D と呼ぶ) があれば、回文候補の自動生成が可能である。しかし、実際には、文節データベース D のサイズが大きくなると、探索空間は非常に大きくなり、回文候補生成に莫大な時間がかかってしまう。

そこで、生成する回文候補の文節数 (**目標文節数**) を事前に定め、**先読み**と**マクロオペレータ**を導入して、探索の効率化を図る。

先読みとマクロオペレータを用いた探索では、オペレータの適用対象となる状態において、

*2 実際には、出力時に必要な、漢字仮名まじり表記も保持しておく必要がある。

1. 不足文字列が 3 文字以上なら、先読みと基本オペレータを用いる。
2. 不足文字列が 2 文字以下なら、マクロオペレータを用いる。

3.1 先読み

先読みでは、ある状態があと n 回の基本オペレータ適用で最終状態に到達可能かどうかを調べる。ここで n を "[目標文節数] – [現在の文節数]" とすれば、目標文節数で回文候補となる見込みのない状態を、早い段階で探索の対象から外すことができる。

先読みの効果を上げるためには、ある状態があと n 回の基本オペレータ適用で最終状態に到達可能かどうかを、高速に調べる方法が必要である。そこで、以下の手順を用いて、探索中に出現する可能性があるすべての状態に対し、この情報を事前に準備しておく。

1. 探索中に出現する可能性がある不足文字列 s の集合 S を、文節データベース D から求める[*3]。
2. 不足文字列が s である状態 x (つまり、$L = s$ or $R = s$ の状態) が、あと 1 回の基本オペレータ適用で最終状態に到達するかどうかを調べる。これは、D の中に、読みが s の文節があるかどうかを調べればよい。これをすべての $s(\in S)$ に対して調べる。
3. 不足文字列が $L = s$ である状態 x に対し、あと 2 回の基本オペレータ適用で最終状態に到達可能かどうかを調べる。これは、x に基本オペレータを 1 回適用し、その結果得られる状態 (不足文字列) が、あと 1 回の基本オペレータ適用で最終状態に到達可能であるかを調べればよい。ここで、後者は、すでにステップ 2 で求まっているので、前者 (x に対する基本オペレータの適用) のすべての可能性を調べればよい。同様のことを、不足文字列が $R = s$ である状態に対しても調査する。以上の調査を、すべての $s(\in S)$ に対して行なう。
4. 以上のことを再帰的に行なえば、不足文字列が s である状態が、あと n 回の基本オペレータ適用で最終状態に到達可能かどうかを、あらかじめ調べておくができる。

3.2 マクロオペレータ

一般に、不足文字列の長さが短いほど、その不足文字列を持つ状態に結合可能な文節の数は多くなる。そこで、2 文字以下の不足文字列を持つ状態に関しては、マクロオペレータを定義しておき、高速に探索を終了させる。

ここで定義するマクロオペレータは、状態の左右に n 個の文節を一度に結合することで、最終状態を生成するオペレータである。これは、左延長オペレータと右延長オペレータを組み合わせたオペレータと考えることができる。たとえば、$L =$ "きぬ" に対しては、左に「よるおきぬ (夜起きぬ)」、右に「おるよ」を結合する、次のようなマクロオペレータが定義されていれば、その適用によって一気に最終状態に到達できる。

ExtendBoth(よる. お**きぬ**, おるよ)

以下では、左に結合する文節列を w_l、右に結合する文節列を w_r で表記する。

このようなマクロオペレータを、以下の方法で作成する。

*3 文法的適格性のチェックを行なわないのであれば、同一の不足文字列を持つ状態に対して、かならず、同一の文節を結合できる (なぜならば、適用可能な条件には H は関与しない)。それゆえ、ここでは、探索中に出現する可能性のある不足文字列を全て求めれば十分である。

文節数	生成時間	生成数
2	5 h 20 m	44,673
3	22 h 33 m	2,887,472
4	142 d 11 h 51 m	156,851,145

表 1: 網羅的回文候補生成結果

1. 探索中に出現する可能性のある 2 文字以下の全ての不足文字列の集合 $\dot{S}$ を、D から求める。
2. $\dot{S}$ の各要素 $\dot{s}$ に対し、以下の方法で、w_l と w_r の文節数の和が n 以下となる (w_l, w_r) をすべて求める。
   (a) 状態 $\langle \dot{s}, |, \epsilon \rangle$ と状態 $\langle \epsilon, |, \dot{s} \rangle$ を作る。ここで、| は単なる区切り記号であり、H は空文字列とみなす。
   (b) それぞれの状態に、基本オペレータを n 回適用して、生成可能な最終状態をすべて生成する。
   (c) 得られた最終状態のそれぞれから、H を取り出す。この H を、最初に挿入しておいた区切り記号 | で分割して、左側を w_r、右側を w_l とする。

上記の手続きは、2 文字以下の不足文字列に対して、実際に、文節結合法による回文候補生成を行なっていることに等しく、探索にかなりの時間を要する。このため、特に高速化が必要な短い不足文字列 (2 文字以下) に対してのみ、マクロオペレータを準備する。

4. 実験と検討

以上述べてきた文節結合法を実際に実現し、回文候補を網羅的に生成する実験を行なった。

すでに述べたように、文節結合法では、結合の候補対象とする文節データベース D が必要である。本実験では、形態素解析システム JUMAN の内容語辞書[*4]をもとに、文節データベースを作成した。まず、内容語辞書に含まれるそれぞれの見出し語に対し、JUMAN の活用定義[*5]と人手で作成した付属語の付加規則を用いて、生成可能な文節をすべて求めた。その後、ひらがな表記と文節タイプの両方が一致する文節群をグループ化して一つにまとめた[*6]。最終的に、文節データベース D のサイズは、3,472,534 件となった。

4.1 実験１：網羅的な回文候補生成

ここでは、文節データベース D を用いて生成可能な、2 文節から 4 文節のすべての回文候補を生成する実験を行なった。具体的には、D に含まれるそれぞれの文節をシード文節として、$n(2 \leq n \leq 4)$ 文節の回文候補をすべて求めた。

表 1 に、生成に要した時間と、生成された回文候補数を示す。なお、4 文節の回文候補の生成は、16 分割して並列に実行し、それに要した時間の総和を示した。

4.2 実験２：意味のある回文を含むか？

今回実装した文節結合法が生成するのは、意味的適格性を考慮していない回文候補である。そこで、実験 2 として、生成した回文候補の中に、意味的適格性を持つ回文が存在するかどうかを調査した。

*4 JUMANver5.1.2 の ContentW.dic を用いた。
*5 JUMANver5.1.2 の JUMAN.katsuyou を用いた。
*6 文節結合法において、ひらがな表記と文節タイプの両方が一致する文節は、区別されない。

表 2: Web 上に存在した回文 (22 文)

回文の表記	回文の読み
断固喘ぐエアコンだ	だんこ. あえぐ. えあこんだ
サンダルが上がる段差	さんだるが. あがる. だんさ
痛い兄貴に会いたい	いたい. あにきに. あいたい
死ぬ烏賊の飼い主	しぬ. いかの. かいぬし
私怒り理解したわ	わたし. いかり. りかいしたわ
滝浴び飽きた	たき. あび. あきた
断然安心安全だ	だんぜん. あんしん. あんぜんだ
良い滝行きたいよ	よい. たき. いきたいよ
良いタレ入れたいよ	よい. たれ. いれたいよ
諍いなど無い家裁	いさかいなど. ない. かさい
妻勇む歳末	つま. いさむ. さいまつ
農家も芋買うの	のうかも. いも. かうの
歌ウキウキ歌う	うた. うきうき. うたう
携帯浮いた池	けいたい. ういた. いけ
良き嘘こそ浮き世	よき. うそこそ. うきよ
うどん打つ運動	うどん. うつ. うんどう
洋服の羽毛の工夫よ	ようふくの. うもうの. くふうよ
私右翼を供養したわ	わたし. うよくを. くようしたわ
仲間売る馬かな	なかま. うる. うまかな
良い絵の栄誉	よい. えの. えいよ
爆薬置く役場	ばくやく. おく. やくば
形見の斧見たか	かたみの. おの. みたか

表 3: Web 上に存在しなかった回文 (27 文)

回文の表記	回文の読み
つい会うあいつ	つい. あう. あいつ
臭い愛裂く	くさい. あい. さく
見事な穴とゴミ	みごとな. あなと. ごみ
夜、家居るよ	よる. いえ. いるよ
糞以下の会則	くそ. いかの. かいそく
渋く行く武士	しぶく. いく. ぶし
少ない愛無くす	すくない. あい. なくす
歯科有る証	しか. ある. あかし
熊の胃の膜	くまの. いの. まく
開拓行く対価	かいたく. いく. たいか
女子の遺体の所持	じょしの. いたいの. しょじ
以下の一意の解	いかの. いちいの. かい
田舎いちいち行かない	いなか. いちいち. いかない
話題がいちいち意外だわ	わだいが. いちいち. いがいだわ
息子と従兄弟住む	むすこと. いとこ. すむ
新規の依頼の禁止	しんきの. いらいの. きんし
知事の異例の自治	ちじの. いれいの. じち
タレ色々入れた	たれ. いろいろ. いれた
タイが絵を描いた	たいが. えを. えがいた
遺跡の絵の規制	いせきの. えの. きせい
キジの絵の磁器	きじの. えの. じき
地位得た英知	ちい. えた. えいち
夜、オーナー居るよ	よる. おーなー. おるよ
噛んで美味しいおでんか	かんで. おいしい. おでんか
神経多い犬歯	しんけい. おおい. けんし
良い大物も多いよ	よい. おおものも. おおいよ
携帯置いた池	けいたい. おいた. いけ
確か弟犯した	たしか. おとうと. おかした

実験 1 で生成した 3 文節の回文候補のうち、シード文節の読みがア行で始まる回文候補 (386,198 文) に対して、次の手順で調査を行なった。まず、それぞれの回文候補を表記に展開した。この結果、1,392,238 件の漢字仮名まじり表記 (回文候補) に展開された。次に、以下のような、Web のヒット数を用いた絞り込みを行った。ここでは、調査対象の 3 文節の回文候補 P を、文節の漢字仮名まじり表記 B を用いて、$P = B_1B_2B_3$ と表記する。

1. 「"B_1B_2" + "B_3"」を検索クエリとし、Web のヒット数を取得する。
2. 「"B_1" + "B_2B_3"」を検索クエリとし、Web のヒット数を取得する。
3. いずれかのヒット数が 0 である P を削除する[*7]。

この絞り込みの結果、回文候補は 8,003 文に絞り込まれた。その 8,003 文から、意味的適格性を満たす回文を人手で抽出した。この結果、49 文の回文が見つかった。その後、それぞれの回文を検索クエリとして Web 検索を行ない、その回文が、Web 上に回文の実例として存在するかどうかを調査した。

得られた結果を表 2 と表 3 に示す。

4.3 検討

実験 1 では、350 万件弱の文節データベース D から生成可能な 4 文節の回文候補のすべて (約 1 億 5000 万件) を、のべ 142.5 日の計算時間で生成することができた。この結果は、文節結合法が現実的な時間で比較的短い回文候補を大量に生成可能であることを示している。

さらに、実験 2 の結果、文節結合法が生成した回文候補中には、Web 上で回文として掲載されているものが存在することが明らかになった。このことは、文節結合法が、人間が生成する回文と同レベルの回文を生成可能であることを意味する。

*7 ヒット数が 0 であった P は、P が持つ文節が共起する可能性が低いと考えられるため、削除する。

加えて、生成した回文候補の中に、Web 上に存在しない回文も見つかった。このことは、文節結合法が、まだ知られていない新しい回文を生成する能力を持っていることを示している。

一方、実験 2 より、意味的適格性を満たす回文は、生成した回文候補の中に僅かな割合しか存在しないことが明らかになった。より長い回文候補の網羅的生成には膨大な時間がかかると予想されるため、回文候補生成過程で意味的適格性のチェックを導入する必要があると考えられる。意味的適格性のチェック法としては、実験 2 で行った Web 検索を用いたチェック法の他に、データベースの各文節に意味を付与し、その意味を利用する方法などが考えられる。

参考文献

[1] 荻生待也 (編): 図説 ことばあそび遊辞苑, 遊子館 (2007).

[2] 小野 智司, 原田 健一郎, 中山 茂: Web サーチエンジンと進化計算法を用いた回文生成, 情報処理学会火の国情報シンポジウム 2006, B-6-3 (2006).

[3] Dan Hoey: Panama Palindromes, http://www2.vo.lu/homepages/phahn/anagrams/panama.htm (2010/4/15 確認).

[4] Bill A. O'Connor: An automatic palindrome generator, http://www.thefreelibrary.com/An+automatic+palindrome+generator-a083553460 (2010/4/15 確認).