TR-I-0232

HMM-LR 連続音声認識における音素コンテキスト依存型文脈自由文法の生成アルゴリズム

An algorithm of generating a phoneme-context-dependent CFG for HMM-LR continuous speech recognition.

†菊池英明、永井明人、嵯峨山茂樹、北研二
KIKUCHI hideaki, NAGAI akito, SAGAYAMA shigeki, KITA kenji

概要

一般に、コンテキスト依存の音素モデルを駆動するために文法の音素ラベルを書き換える際、単語の両端の音素のコンテキストは不確定であり、音素コンテキストによってどのように非終端記号間の接続を行なえばよいのかという問題を単語間接続 (Word Juncture) 問題と呼ぶ。本稿では、文脈自由文法の枠組における単語間接続問題に対する一つの解答を与えるものとして、音素コンテキストに依存する文脈自由文法を生成するアルゴリズムについて説明し、本アルゴリズムにより生成された文法を用いて、HMM-LR 法による文節認識実験を行なった結果について報告する。本報告は、学外実習生 菊池英明 (早稲田大学) が行なった実習の報告書である。

†早稲田大学理工学部
Department of Electrical Engineering ,Waseda University
ATR 自動翻訳電話研究所
ATR Interpreting Telephony Research Labs.

1 はじめに

本報告は、文脈自由文法の枠組における単語間接続 (Word Juncture) 問題に対する一つの解答を与えるものとして、音素コンテキストに依存する文脈自由文法を生成するアルゴリズムについて説明し、本アルゴリズムにより生成された文法を用いて、文節認識実験を行なった結果について述べる。

例えば、/aka/(赤) と /iki/(行き) の /k/ は、同じ音素であるが、音響的には大きく異なったパターンである。このような現象は、調音結合と呼ばれ、このような音響的に異なる同一音素は「異音 (allophone)」と呼ばれている。もっと一般化すれば、前後音素のみならず、発話速度やピッチなども含めた広い意味での「音素環境」によって音素パターンの変形、変動が生じる。このような音声パターンの変動は、音声認識が困難な大きな理由である。特に、連続音声においては、この種の音素変形が著しい。

そこで、音素パターンの変動を積極的に利用して、認識の単位として音素環境依存の音素モデルを用いることが、認識性能の高い音声認識のためには有効である。そのため、最近になって、音素コンテキストから予測される音素変形を考慮に入れたモデルを用いて、認識性能を向上させた報告 [1][2] がされている。ATR でも異音 (allophone) のレベルで連続音声を認識することの有効性が示されつつある [4][5]。また、どのような異音の組を持てばよいかを自動的に決定する手法として、蟯峨山の PEC(音素環境クラスタリング) がある [7][8][9]。

本研究では、音素変形の要因として特に重要な音素コンテキスト (前後の音素) を考慮して、音素コンテキスト依存の音素 (異音) モデルを活用することを考える。

一方、われわれは、連続音声認識の手法として HMM-LR 法 [6] を用いている。これは、音声認識と言語処理を同時進行させる方式であり、音声認識と言語処理の間に音素ラティス等の中間的なデータを介する必要がないため、高精度かつ効率的な認識処理系である。現在 ATR では、この HMM-LR 法の音声認識部に、音素コンテキスト依存の HMM 音素モデルを適用することを検討しており、更に高精度な認識が可能になると期待している。

従来の LR パーザは音素コンテキストに依存しない構文解析を行なうためのものであった。これに、コンテキスト依存音素モデルを組み合わせるには、音素コンテキストに依存する異音 (allophone) モデルを駆動できる LR パーザが必要である。

そのために我々は、音素コンテキスト依存型音素モデルを HMM-LR 法で駆動するために、音素コンテキストに依存した構文解析を行なう LR パーザのアルゴリズムを検討し、次の 2 方式を既に実現した [12]。

(1) LR テーブルを生成する段階で環境依存 LR テーブルを実現する方式 [10]。

(2) LR パーザの段階で音素コンテキストを動的に予測する方式 [11]。

上記の 2 方式により、音素コンテキスト依存型 HMM-LR は実現可能となった。これら両者のアルゴリズムは、音素コンテキスト要因として、先行音素、中心音素、後続音素の 3 要因を考慮したものである。今後さらに精密な音素モデルを駆動するために、先々行音素、後々続音素を含めた 5 要因を考慮したアルゴリズムの拡張が考えられる。

しかし、前記の 2 種類の実現方式では、5 要因音素コンテキストを利用するためにはアルゴリズムが非常に繁雑になり、また、組み合わせ爆発の問題も生じることで 5 要因への拡張は極めて困難である。

そこで、本報告では、このアルゴリズム拡張性の問題も含めて、単語間の接続における音素コンテキスト制約問題の定式化を、音素コンテキストに依存する文脈自由文法を文法間の変換によってのみ生成させるという、より一般的な枠組の中で再構築するアプローチを考える。この枠組においては、音素コンテキスト依存の文法生成問題は、純粋な単語間接続 (Word Juncture) 問題に帰着される。最近になって、音声言語処理、特にサーチアルゴリズムの分野でこの問題が注目を集めている [3] が、未だ完全に解決されていない。

今回試みた、文法レベルにおける音素コンテキスト依存型 LR パーザの実現アルゴリズムは、単語間接続 (Word Juncture) 問題に対する一つの解答を与えるものである。本稿では、第一段階の問題設定として、3 要因の音素コンテキストに依存する文脈自由文法を生成するアルゴリズムについて説明し、本アルゴリズムにより生成された文法を用いて、HMM-LR 法による文節認識実験を行ない、その結果について述べる。

2 音素環境依存型 HMM-LR 法の実現方式

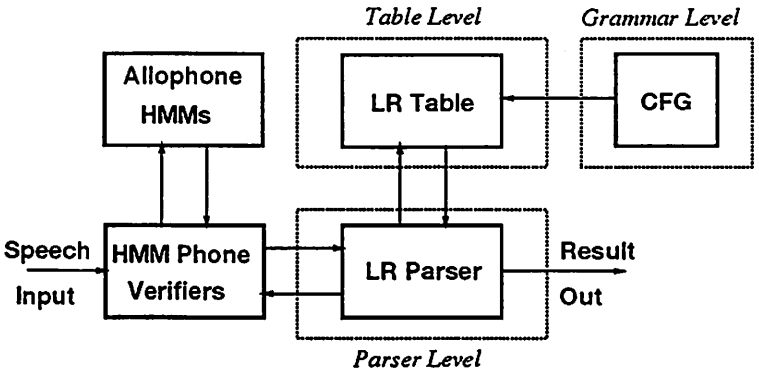

図 1: 音素環境依存型 HMM-LR 法の実現レベル

従来の HMM-LR 法を音素環境に依存する方法に変更するには、以下の 3 通りの段階での実現方法 (図 1参照) が考えられる。

(1) 文法レベルでの実現

音素コンテキストの環境クラスタに応じて、文法規則の非終端記号を複数用意し、文法規則を書き換える方法である。

(2) テーブルレベルでの実現

従来の音素コンテキストに依存しない LR テーブルを変換して、音素コンテキスト依存の構文解析動作を行なう環境依存の LR テーブルを生成する方法である。

(3) パーザレベルでの実現

従来の音素コンテキストに依存しない LR テーブルを用いて、パージング中に予測音素の先行音素、後続音素をダイナミックに参照し、音素コンテキストを予測する方法である。

以上の 3 つの方法のうち、(2)、(3) については、既に [12] において実現アルゴリズムが報告されている。

本報告では、(1) を実現するためのアルゴリズムを提案する。

3 音素コンテキストに依存した文法

ここで扱う文法は、文脈自由文法である。一般的な文法の規則は以下のようなものである。ただし、$< right[i] >$ は終端記号と非終端記号の両方をとり得る。

$$(< Left > \Longleftrightarrow (< right[1] > \ ... \ < right[i] > \ ... \ < right[length] >))$$

3.1 終端記号における環境クラスタの決定

まず、従来の音素コンテキストに依存しない文法に、付加できる音素環境情報として、終端記号を音素環境クラスタリングより得られる環境クラスタ名に変換する。

$$(< noun > \Longleftrightarrow (... \ k \ a \ i \ ...)) \tag{1}$$

音素コンテキストに依存した音素モデルを考えた場合、上の規則において、音素 a の環境は既に決定しているといえる。なぜならば、その先行音素と後続音素が一通りに決定しているからである。従って、ある終端記号の前後の記号が、終端記号である場合、その記号を環境クラスタ名に変換することにより、その記号は音素環境情報を持ったといえる。

終端記号の場合
a47

先行音素 k
中心音素 a
後続音素 i

3.2 環境クラスタが不明な終端記号

しかし、文法ルールの終端記号を環境クラスタ名に変える際、先行音素と後続音素が1ルール中で一通りであれば環境は決定するが、その記号がルールの先頭や最後の場合、あるいは非終端記号が接続する場合には、環境を一通りに決定することができない。そこで、あらかじめ全てのルールを調べて、非終端記号毎に先頭の記号や最後の記号のリストを作成する。そして、環境を決定する際に必要ならば、これを参照することにした。

$$(< np > \Longleftrightarrow (k\ i\ ...)) \tag{2}$$

例えば、上の規則において、音素kの環境を決定する時、まず、音素kの先行音素が不明であるから、$< np >$の先行音素を調べる。$< np >$の先行音素の集合が{a, o}であるとき、これらを先行音素とし、iを後続音素とする音素kの環境がそれぞれ、k1,k2と定められると、上の規則は以下の2つの規則に変換できる。

$$(< np > \Longleftrightarrow (k1\ i\ ...)) \tag{3}$$

$$(< np > \Longleftrightarrow (k2\ i\ ...)) \tag{4}$$

3.3 非終端記号に付加する音素環境情報

次の段階として、非終端記号にどのように音素環境情報を付加するかという問題がある。ここでは、音素コンテキストに依存した文法を実現するために、非終端記号に先行、先頭、最後、後続の4つの音素を付加することが、必要になった。その理由を以下に説明する。

3.3.1 先頭音素

$$(< p > \Longleftrightarrow (...\ a\ < np >)) \tag{5}$$

上の規則において、$< np >$の先頭音素の集合がk, chであるとき、音素aは後続音素として、kとchの2つが考えられることになる。kとchのそれぞれを後続音素とする音素aの環境クラスタ名をa1、a2とすると、上の規則は

$$(< p > \Longleftrightarrow (...\ a1\ < np >)) \tag{6}$$

$$(< p > \Longleftrightarrow (...\ a2\ < np >)) \tag{7}$$

のように2つに分けられる。ここで、上の2つの規則における$< np >$の先頭の音素はそれぞれkとchであり、異なることに注意して、これを区別するべきである。そこで、元の規則を次のように記述する。

$$(< k\_p > \Longleftrightarrow (...a1 < k\_np >)) \tag{8}$$

$$(< ch\_p > \Longleftrightarrow (...a2 < ch\_np >)) \tag{9}$$

3.3.2 最後の音素

それでは、3.3.1に従って、非終端記号を先頭の音素によって区別したとする。

$$(< k\_p > \Longleftrightarrow (< k\_np >\ g\ ...\ u)) \tag{10}$$

上の規則において、音素gの先行音素として、非終端記号$< np >$の最後の音素の集合がi,oである時、それぞれを先行とするgの環境名をg1,g2とすると、前項と同様に

$$(< k\_p > \Longleftrightarrow (< k\_np >\ g1\ ...\ u)) \tag{11}$$

$$(< k\_p > \Longleftrightarrow (< k\_np >\ g2\ ...\ u)) \tag{12}$$

のように書き換えられる。ただし、上の2つの規則における$< np >$の最後の記号はそれぞれ、iとoであり、3.3.1の場合と同様に$< np >$を区別するべきである。そこで、元の1つの規則を次のように記述する。

$$(< k\_p\_u > \Longleftrightarrow (< k\_np\_i >\ g1\ ...\ u)) \tag{13}$$

$$(< k\_p\_u > \Longleftrightarrow (< k\_np\_o >\ g2\ ...\ u)) \tag{14}$$

3.3.3 後続の音素

$$(< p > \Longleftrightarrow (< np > \ g \ ... \ u)) \tag{15}$$

という規則を上の 3.3.1 と 3.3.2 に従って変換したところ、以下のルールが生じたとする。

$$(< k\_p\_u > \Longleftrightarrow (< k\_np\_i > \ g1 \ ... \ u)) \tag{16}$$

また、このルールの右辺にある $< k\_np\_i >$ を左辺にもつ規則に以下のようなものがあるとする。

$$(< k\_np\_i > \Longleftrightarrow (k \ o \ i)) \tag{17}$$

この規則の右辺の終端記号の音素環境を決定する際、音素 o については既に決定している。音素 i について、後続音素が、g を含む集合 g,zh であるとき、その環境名がそれぞれ i1 と i2 であったとする。そのとき、(3) の規則は以下のようになる。

$$(< k\_np\_i > \Longleftrightarrow (k1 \ o1 \ i1)) \tag{18}$$

$$(< k\_np\_i > \Longleftrightarrow (k1 \ o1 \ i2)) \tag{19}$$

ここで、規則 (3) の右辺の $< k\_np\_i >$ には上の規則の両方が該当するわけではない。なぜなら、上の方は後続に g をとるものであるが、下の方は後続に zh をとり、g をとらないからである。従って、これらもまた区別する必要があり、それを以下に示す。

$$(< k\_np\_i\_g > \Longleftrightarrow (k1 \ o1 \ i1)) \tag{20}$$

$$(< k\_np\_i\_zh > \Longleftrightarrow (k1 \ o1 \ i2)) \tag{21}$$

3.3.4 先行音素

（3）と同様な理由で、非終端記号は先行音素によっても区別しなければならない。

$$(< p > \Longleftrightarrow (k \ ... \ e \ < np > \ g \ ... \ u)) \tag{22}$$

という規則を上の 3.3.1 と 3.3.2 と 3.3.3 に従って変換したところ、以下のルールが生じたとする。

$$(< k\_p\_u\_ch > \Longleftrightarrow (k \ ... \ e < t\_np\_i\_g > \ g \ ... \ u)) \tag{23}$$

また、このルールの右辺にある $< t\_np\_i\_g >$ を左辺にもつ規則に以下のようなものがあるとする。

$$(< t\_np\_i\_g > \Longleftrightarrow (t \ o \ i)) \tag{24}$$

この規則の右辺の終端記号の音素環境を決定する際、音素 o については既に決定している。音素 t について、先行音素が、e を含む集合 e,o であるとき、その環境名がそれぞれ t1 と t2 であったとする。そのとき、上の規則は以下のようになる。

$$(< t\_np\_i\_g > \Longleftrightarrow (t1 \ o1 \ i1)) \tag{25}$$

$$(< t\_np\_i\_g > \Longleftrightarrow (t2 \ o1 \ i1)) \tag{26}$$

ここで、規則 (3) の右辺の $< t\_np\_i\_g >$ には上の規則の両方が該当するわけではない。なぜなら、上の方は後続に e をとるものであるが、下の方は後続に o をとり、e をとらないからである。従って、これらもまた区別する必要があり、それを以下に示す。

$$(< e\_k\_np\_i\_g > \Longleftrightarrow (k1 \ o1 \ i1)) \tag{27}$$

$$(< o\_k\_np\_i\_zh > \Longleftrightarrow (k2 \ o1 \ i1)) \tag{28}$$

音素環境情報をもつ非終端記号

以上の理由で、非終端記号においては先行、先頭、最後、後続の音素により、区別するべきである。

$< a\_b\_symbol - name\_c\_d >$

a:< $symbol-name$ > の先行の音素 (precede)
b:< $symbol-name$ > の先頭の音素 (first)
c:< $symbol-name$ > の最後の音素 (end)
d:< $symbol-name$ > の後続の音素 (follow)

以上が、文法規則中の各記号の、音素コンテキストに依存した形であり、次項に、この生成アルゴリズムを説明する。

4 音素コンテキスト依存型文法の生成アルゴリズム

以下に、各1つの文法規則を、音素環境情報が付加された文法規則に変換するアルゴリズムを説明する。文法を前項で述べたような音素環境に依存したものにする際、規則数の爆発的な増加が予想される。なぜなら、非終端記号を、先行、先頭、最後、後続の音素により区別するためである。将来的には、ここに何らかの制約を設けて、規則数を減らすことが考えられるが、本研究では何も制約を設けないで文法を生成するアルゴリズムを説明する。

4.1 全文法規則について行なう前処理

STEP 1. 先行、先頭、最後、後続音素リストの作成

全ての文法規則を読みこみ、各非終端記号毎に先行音素リスト、先頭音素リスト、最後音素リスト、後続音素リストを作成する。

まず、以降のSTEPで必要となる、各非終端記号毎の先行、先頭、最後、後続に現れ得る音素のリストをあらかじめ作成しておく。例えば、図2の文法について、図3のような各リストが作成される。

$(< start > \Longleftrightarrow (< \_start >))$
$(< \_start > \Longleftrightarrow (q1\ < p >\ q2))$
$(< p > \Longleftrightarrow (< np >))$
$(< np > \Longleftrightarrow (< noun >))$
$(< np > \Longleftrightarrow (< noun > < suffix >))$
$(< np > \Longleftrightarrow (< head > < noun > < suffix >))$
$(< head > \Longleftrightarrow (g\ o))$
$(< noun > \Longleftrightarrow (h\ a\ q\ p\ j\ o\ o))$
$(< noun > \Longleftrightarrow (sh\ i\ ts\ u\ m\ o\ =))$
$(< suffix > \Longleftrightarrow (o))$
$(< suffix > \Longleftrightarrow (w\ a))$

図 2: 文脈自由文法 (例)

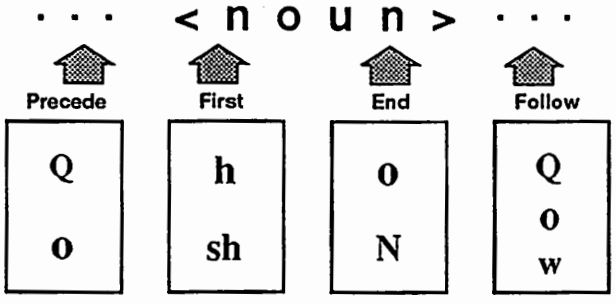

図 3: 先行、先頭、最後、後続音素リストの作成

4.2 各文法規則について行なう処理

次に、1つの文法規則を、前項で示した音素コンテキストに依存した文法規則に変換する 作業を行なう。そのために、各規則毎に、STEP2 から STEP4 を繰り返す。

4.2.1 1つの文法規則について行なう前処理

STEP 2. 右辺の全記号

右辺の全ての記号について先頭音素リスト、最後音素リスト を登録する（図4参照）。

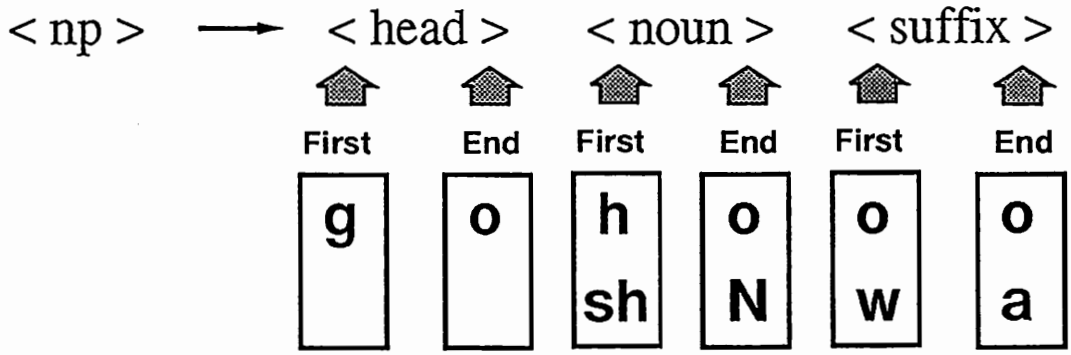

図 4: 右辺の全記号についての前処理

ここでは、STEP4 以降で用いるリストを登録する。 まず、STEP1で作成した各リストを参照して、規則の右辺の全ての記号について、先頭音素、最後音素のリストを登録する。 ただし、記号が終端記号である場合、 先頭音素、最後音素のリストにはその記号自身を登録する。 ここで、登録したリストは、後に、各記号を先行、先頭、最後、後続音素によって 区別した新しい記号に変換する際に用いられる。 記号が、終端記号である場合は、先行、後続音素によって環境を決定する際に 用いられる。

STEP 3. 左辺の非終端記号

左辺の非終端記号について先行音素リスト、後続音素リスト を登録する。（図5参照）

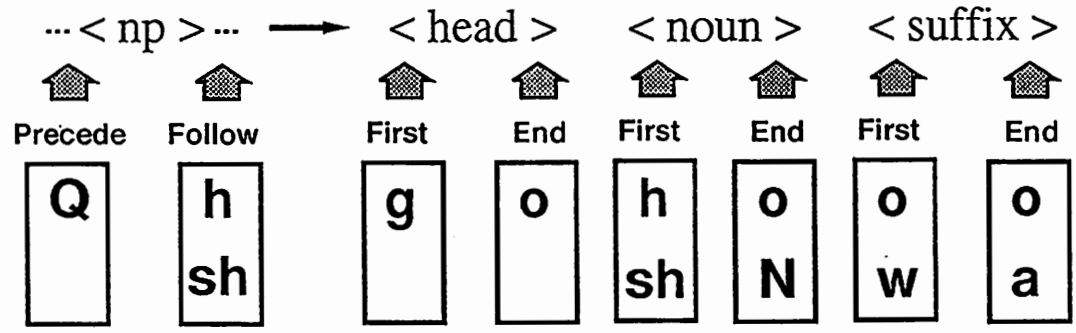

図 5: 左辺の非終端記号についての前処理

ここで登録した、左辺の記号の先行音素リストと後続音素リストは、 右辺の最初の記号の先行音素を参照する際と、 右辺の最後の記号の後続音素を参照する際に、 それぞれ必要になる。

4.2.2 1つの文法規則の各記号について行なう処理

規則中の各記号を、音素環境情報をもつ記号に変換し、 それらを組み合わせた新しい規則を生成する。 その際に、組合せに漏らしがなく、整合性が採れていなければならない。

ただし、記号が終端記号である場合、新しい記号は作らず、先行音素と 後続音素から環境名を決定する。

STEP 4. 各記号の変換 -4.1. 左辺の非終端記号

1. 右辺の最初の記号の先頭音素リストを参照して、左辺の記号の先頭音素に与える。
2. 同様に右辺の最後の記号の最後音素リストを参照して、左辺の記号の最後音素に与える。
3. そして、先行音素、先頭音素、最後音素、後続音素の全ての組合せで、記号を作る。

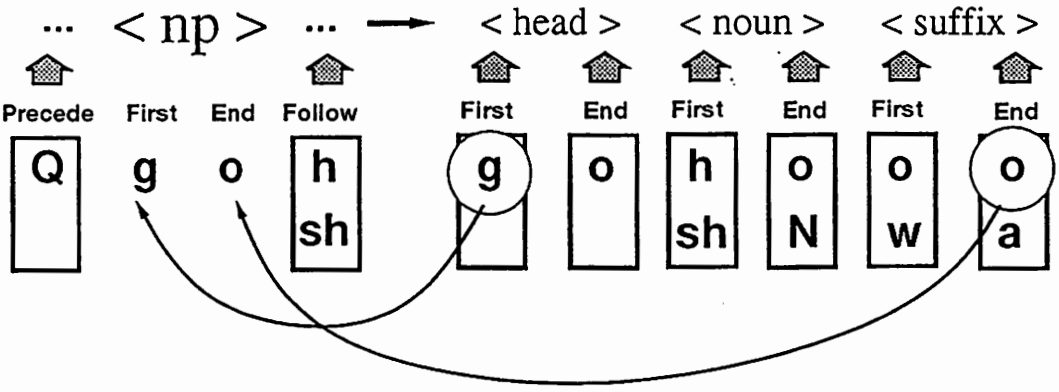

図 6: 左辺の非終端記号の変換

図 6では、右辺の最初の記号である < $head$ > の先頭音素リストから、 音素 g を、左辺の記号の先頭音素に与える。 同様に、右辺の最後の記号である < $suffix$ > の最後音素リストから音素 o を、 左辺の記号の最後音素に与える。 このようにして、外から与えられた情報と、 既に持っている、自身の先行、後続音素リストから、 新たな記号を作る。

STEP 4. 各記号の変換 -4.2. 右辺の最初の記号

1. 右辺の最初の記号について、左辺の記号の先行音素リストを参照して、 先行音素に与える。
2. 次に、右辺のその次の記号の先頭音素リストを参照して、 後続音素に与える。
3. そして、先行音素、先頭音素、最後音素、後続音素の全ての組合せで、記号を作る。

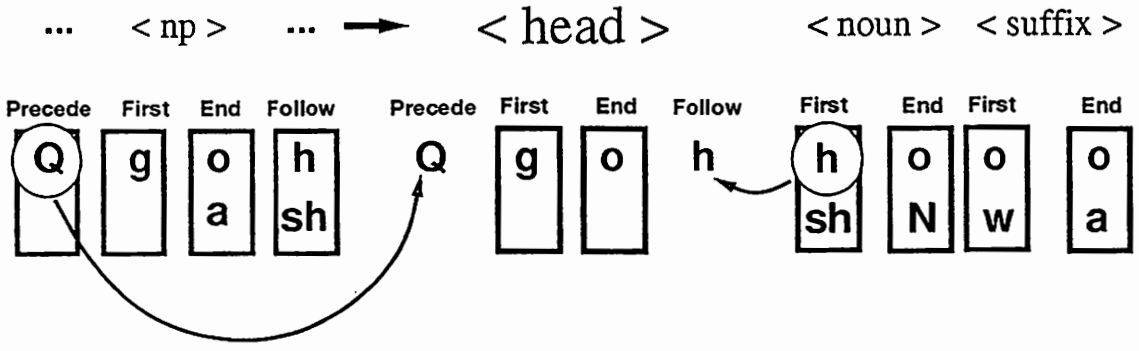

図 7: 右辺の最初の記号の変換

図 7において、左辺の記号である < np > の先行音素リストから、 音素 - を、右辺の最初の記号の先行音素に与える。 同様に、右辺の次の記号である < $noun$ > の先頭音素リストから音素 h を、 右辺の最初の記号の後続音素に与える。 このようにして、外から与えられた先行、後続音素と、 既に持っている、自身の先頭、最後音素リストから、新たな記号を作る。

STEP 4. 各記号の変換 -4.3. 右辺の記号

1. 右辺の記号について、その 1 つ前の記号の最後音素リストを参照して、 先行音素に与える。
2. 次に、 1 つ後の記号の先頭音素のリストを参照して、 後続音素に与える。
3. そして、先行音素、先頭音素、最後音素、後続音素の全ての組合せで、 新しく記号を作る。

この動作を右辺の 2 つ目の記号から最後から 2 番目までの記号について 行なう。

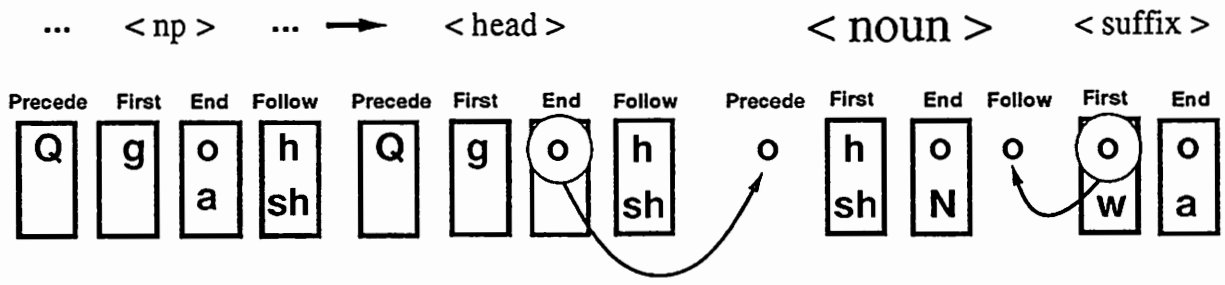

図 8: 右辺の記号の変換

図 8 において、記号 < *noun* > について、 右辺の 1 つ前の記号である < *head* > の最後音素リストから音素 o を、 先行音素に与える。
同様に、右辺の次の記号である < *suffix* > の先頭音素リストから音素 o を、 後続音素に与える。
このようにして、外から与えられた先行、後続音素と、 既に持っている、自身の先頭、最後音素リストから、 新たな記号を作る。

STEP 4. 各記号の変換 -4.4. 右辺の最後の記号

1. 右辺の最後の記号について、右辺の 1 つ前の記号の最後音素リストを参照して、 先行音素として与える。
2. 次に、左辺の記号の後続音素リストを参照して、 最後音素として与える。
3. そして、先行音素、先頭音素、最後音素、後続音素の全ての組合せで、 新しく記号を作る。

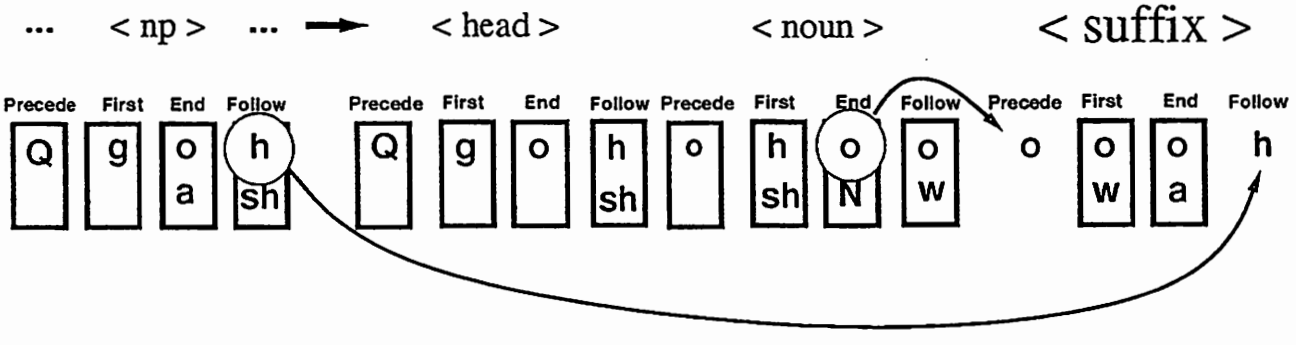

図 9: 右辺の最後の記号

図 9 において、右辺の最後の記号 < *suffix* > について、 右辺の前の記号である < *noun* > の最後音素リストから音素 o を、 先行音素に与える。 同様に、左辺の記号である < *np* > の後続音素リストから音素 h を、 最後音素に与える。 このようにして、外から与えられた先行、後続音素と、 既に持っている、自身の先頭、最後音素リストか

ら、 新たな記号を作る。

記号が終端記号の場合

終端記号について、新たな記号に変換する際には、前項で述べたように、 前後の音素から決定する、環境クラスタ名が与えられる。 図9において、右辺の終端記号である音素gの先行音素には、 STEP4.1で左辺の先行音素から、音素-が与えられる。 後続音素には、次の記号である、音素oが与えられる。 これらから、音素gの環境は1通りに決定する。

以上の動作により、図4から9で示した文法規則から、図10の音素コンテキストに依存した文法規則が生成される。

$$(< q\_g\_np\_o\_q > \Longleftrightarrow (< q\_g\_head\_o\_h > < o\_h\_n\_o\_o > < o\_o\_suffix\_o\_q >))$$
$$(< q\_g\_np\_o\_q > \Longleftrightarrow (< q\_g\_head\_o\_sh > < o\_sh\_n\_N\_o > < N\_o\_suffix\_o\_q >))$$
$$(< q\_g\_np\_a\_q > \Longleftrightarrow (< q\_g\_head\_o\_h > < o\_h\_n\_o\_w > < o\_w\_suffix\_a\_q >))$$
$$(< q\_g\_np\_a\_q > \Longleftrightarrow (< q\_g\_head\_o\_sh > < o\_sh\_n\_N\_w > < N\_w\_suffix\_a\_q >))$$

図 10: 生成された音素コンテキスト依存型文法

5 生成された文法規則

前項に述べたアルゴリズムにより、５７０の規則をもつ文法から生成された音素コンテキスト依存の文法の規則数を表1に示す。

非終端記号を区別しない場合、クラスタ数の増加に伴い、規則数が増える。それは、クラスタの細分化により、非終端記号との隣接部分の音素の環境が増えるためである。規則数が爆発せずに飽和しているのは、終端記号に非終端記号が接続する際、前後の音素の全ての組合せの数を上限としているためである。非終端記号を区別した場合、クラスタ数を変化させても、規則数が変化しないのもこのためである。

非終端記号を区別した場合、規則数の増加は著しいが、これは非終端記号を音素環境情報をもった形に変換する際、先行、先頭、最後、後続の４つの音素の組合せにより、分けたからであり、当然の結果である。ここで、生成された文法には、実際には必要のない区別も行なっているので、今後はそれを制約するようなアルゴリズムの検討が望まれる。

表 1: 生成された文法の規則数

(a) 非終端記号を区別しない場合

クラスタ数	100	200	300	400	500	600	700	800	900	1000	1024
規則数	945	1369	1872	2132	2367	2618	2743	2827	3101	3184	3193

(b) 非終端記号を区別する場合

規則数	570	1407
生成された規則数	15317	204877

6 文節認識実験

以上述べた環境依存型文法生成のアルゴリズムについて、動作の確認 と性能評価のために文節認識実験を行なった。分析条件を表2、実験条件を表3 に示す。また、ビーム幅64で認識実験を行なった結果を表4 に示す。

表 2: 分析条件

sampling 周波数	12 kHz
窓関数	ハミング窓
窓長	21.3 ms
frame 周期	9 ms

表 3: 実験条件

環境分割数	200
音素環境	3 要因 (先行、中心、後続)
HMM 学習	重要語 5240 単語 (MAU)
認識対象	文節単位発声 279 文節 (MAU)
文法	タスク向き文法 570 ルール

表 4: 文節認識率 (%)

1 位	77.8
～2 位	88.5
～3 位	93.9
～4 位	95.0
～5 位	96.4

7 おわりに

HMM-LR 連続音声認識において、 LR パーザが音素コンテキストに依存した構文解析を行なうために、文法規則を音素コンテキスト依存型に変換するアルゴリズムを提案した。
このアルゴリズムを用いて、音素コンテキスト依存型文法を生成し、実際に認識実験を行なった。その結果から、本アルゴリズムに原理的に正しく動作しているものと考えられる。
前項でも述べたが、今後は終端記号と非終端記号の隣接部分において、何らかの制約を設けることにより、生成される規則数の爆発を抑えることが望まれる。

謝辞　研究の機会を与えていただいた、ATR 自動翻訳電話研究所 樽松 明社長に感謝 いたします。親切に御指導して下さった永井明人さんに感謝いたします。 また、熱心に討論いただく ATR 研究員の皆様に感謝いたします。

参考文献

[1] Y.L.Chow, M.O.Dunham, O.A.Kimball, M.A. Krasner, G.F.Kubala, J.Makhoul, P.J.Price,S. Roucos and R.M.Schwartz: "BYBLOS: The BBN Continuous Speech Recognition System", ICASSP87, pp.89-92(1987).

[2] K.F.Lee, H.W.Hon,M.Y.Hwang, S.Mahajan and R.Reddy: "The SPHINX Speech Recognition System", ICASSP89, pp.445-448 (1989).

[3] L.C.Wood, D.J.B.Perarce and F.Novello: "Improved Vocabulary-Independent Sub-Word HMM Modeling" , ICASSP91, pp.181-184 (1991).

[4] 鷹見淳一、片岸一起、嵯峨山茂樹：" 音素環境情報を利用した 連続型 HMM による音素認識 ", 音講論集、3-P-12, pp.171-172(1991.3).

[5] 鷹見淳一、片岸一起、嵯峨山茂樹：”単一ガウス分布 HMM の音素環境木構造 に基づく平滑化による頑健な音素認識”, 信学会音声研資料、 sp91-19、 pp9-16(1991.6).

[6] 北研二、川端豪、斉藤博昭：”HMM 音韻認識と予測 LR パーザ を用いた文節認識”, 音講論集、 2-P-7, pp.259-260(1988.10).

[7] 嵯峨山茂樹：”音素環境クラスタリングの原理とアルゴリズム”, 信学会音声研資料、 SP87-86,pp.1-6(1987).

[8] 嵯峨山茂樹、本間茂：”音素認識における音素環境クラスタリング の効果”, 信学会音声研資料、 SP89-78, pp.17-24(1989).

[9] S.Sagayama: “Phoneme Environment Clustering for Speech Recognition”, ICASSP89, pp.397-400(1989).

[10] 永井明人、北研二、嵯峨山茂樹：”HMM-LR 法における音素文脈依存型 LR パーザの検討”, 音講論集、 3-8-16, pp.125-126(1990.9).

[11] 永井明人、嵯峨山茂樹、北研二：”音素コンテキスト依存型 LR テーブルの 生成アルゴリズム”, 音講論集、 3-5-2, pp.91-92(1991.3).

[12] 永井明人、嵯峨山茂樹、北研二：”HMM-LR 連続音声認識における音素環境依存型 パーザの実現アルゴリズム”, 信学会音声研資料、 SP91-23、 pp.41-48(1991.6).