日本語の並列係り受け解析

2S−3

赤坂 宏二
(財) 新世代コンピュータ技術開発機構

1 まえがき

本稿では、二文節間の係り受けを基礎とした日本語の構文解析を並列に行う方法について述べる。この解析法はGHC[2]のような並列論理型言語で実現可能である。

本稿で報告する手法は文脈自由文法の代表的な解析法であるCYK法同様、動的計画法を基づいており、解析の途中結果を解析表に保存し計算の重複を防ぎ効率の向上を図っている。解析表を用いた上昇型の係り受け解析法が、すでに尾関によって提案されている。本手法は次の特徴を持つ。

(1) 副作用を用いずに解析表を実現している。

(2) 全ての可能な係り受け構造をバックトラックを行なわずに生成する。

(3) 局所的な曖昧さのパッキングを行っている。

2 日本語の係り受けの制約

日本語の係り受けについては、次の三つの制約が一般に認められている[1] 。**制約1** 最後の文節以外の文節は、後続の文節のいずれか一つに係る。**制約2** 二つの文節間の係り受けは、他の二つの文節間の係り受けと交差しない。**制約3** 二つの文節間に係り受けが存在するためには、それらの文節の種類や意味が互いに一定の関係をもたねばならない。

本手法は、二つの文節間の係り受け可能性を判定する二項述語、即ち**制約3**を実現する述語が文法的知識として、与えられたときに**制約1**、**制約2**を満足する係り受け構造を生成する手法である。本稿では、この係り受け判定述語はmodifyという名前で与えられているものとする。

以下の議論では入力として、長さnの文節列ws= $w_1 w_2 ... w_n$ を固定して考える。また文節位置iからjまでの部分文節列 $w_i w_{i+1} ... w_j$ をw[i:j]と書くことにする。

3 係り受け構造

問題を定式化するため以下の定義を行なう。

[定義1] 係り受け
係り受けは二文節の順序対である。(x,y)を**文節xの文節yへの係り受け**という。

[定義2] 係り受け構造
係り受け構造は係り受けの集合である。

[定義3] 整合的な係り受け構造
係り受け構造の内、前節で述べた制約を満たすものが**整合的な係り受け構造**である。

例えば「美しい 黒い 髪の 女を 見た」という係り受け構造の一つは、{(美しい,女を),(黒い,髪の),(女を,見た)}でありこれは、図1のように図示される。この係り受け構造は明らかに整合的である。

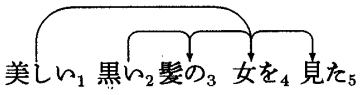

図 1: 係り受け構造の図式

文節列w[i:j]に曖昧さがある場合、w[i:j]上の整合的な係り受け構造は複数になる。w[i:j]上の**全ての整合的な係り受け構造全体**の集合をkkr(w[i:j])とする。便宜的にkkr(w[i:i])(= $kkr(w_i)$)は$\{\phi\}$と定める。kkr(w[i:j])は、当然集合の集合になる。また以下では考える文節列を固定してkkr(w[i:j])をkkr(i,j)と略記する。kkr(w[i:j])は次の三つの規則により構成される。

$$kkr(i,j) = \begin{cases} \{\phi\} & i = j \\ (\cup_{i \le k < j} chain(i,k,j)) \cup block(i,j) & i < j \end{cases}$$

$$block(i,j) =$$

$$\begin{cases} \{B | B = \{(w_i, w_j)\} \cup E, E \in kkr(i+1,j)\} & if\ modify(w_i, w_j) \\ \phi & otherwise \end{cases}$$

$$chain(i,k,j) = \{C | C = kkr(i,k) \cup block(k,j)\}$$

次節で述べる解析アルゴリズムではこれらの規則に基づき、係り受け構造を上昇的に構成していく。

上述の式の中でblock(i,j)を左辺にもつものは、図2のようにw_iがw_jに係るような整合的な係り受け構造が存在するためにはmodify(w_i,w_j)が真であり、かつw[i+1:j]の整合的な係り受け構造*kkr(i+1,j)*がなくてはならないという事実に対応している。

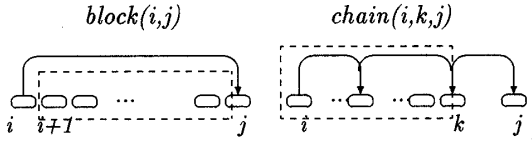

図 2: block(i,j) と chain(i,k,j)

また、*chain(i,k,j)*を左辺に持つ式はw[k:j]で図1のような形の係り受け構造が構成されるならばこれとw[i:k]

Parallel Analysis of Japanese Dependency Structure
Kouji AKASAKA
Institute for New Generation Computer Technology

の整合的な係り受け構造 *kkr(i,k)* を連結して *kkr(i,j)* の内、w_k が w_j に係るものが構成されることを示している。

本手法では曖昧さのある文節列の係り受け構造を効率よく扱うために、文節対の集合 *kkr,block,chain* をそれぞれ次のような再帰的なデータ構造を用いて表現する。

⟨KKR⟩ ::= "⟨CHAIN⟩ のリスト"|[[]]
⟨BLOCK⟩ ::= '('⟨ 文節 ⟩','⟨ 文節 ⟩','⟨KKR⟩')'
⟨CHAIN⟩ ::= '('⟨BLOCK⟩','⟨KKR⟩')' |[]

kkr(i,j),block(i,j),chain(i,j,k) に対応するデータをそれぞれ KKR(i,j),BLOCK(i,j),CHAIN(i,j,k) とすると BLOCK(i,j)=(w_i, w_j,KKR(i+1,j)) となる。BLOCK(i,j) は、w[i+1:j] の中の曖昧さを吸収し (w_i, w_j) の係り受けを共通に持つ係り受け構造を一つに纏めあげるための構造であり、w[i+1:j] の中の係り受け構造は第三要素 KKR(i+1,j) によって表される。

このような構造を用いることにより局所的な曖昧さの組み合わせによる計算量の増大を抑制することができる。

次にこの構造を並列処理の観点からみると BLOCK(i,j) からより大きな係り受け構造で表現する CHAIN(l,i,j)($l \leq i$) を構成する処理と BLOCK(i,j) の要素である KKR(i+1,j) を構成する処理を同時並列に実行することができて都合が良い。このような並列処理は GHC を用いて自然に実現可能である。この BLOCK 構造と CHAIN 構造を並列上昇的に構成しながら解析は進行する。

4 並列解析アルゴリズム

図3のように三角行列状に配置されたプロセス群 $P_{i,j}(1 \leq i \leq j \leq n)$ によって解析は実行される。また同様に配置された変数 $b_{i,j}$, ストリーム $s_{i,j}(1 \leq i \leq j \leq n)$ を用いてプロセス間の通信を行なう。ここではプロセスを中心に説明する都合上インデックスの付けかたが前節と異なっているが、BLOCK(i,j) が $b_{i-j+1,j}$、KKR(i,j) が $s_{i-j+1,j}$ に、CHAIN(i,k,j) が $s_{i-j+1,j}$ の要素にそれぞれ対応している。

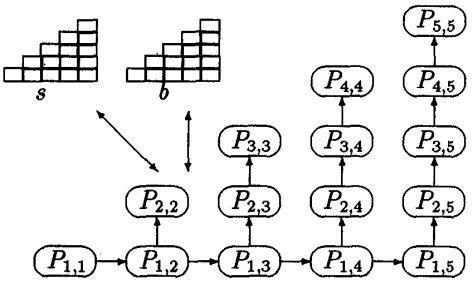

図 3: テーブル状に配置されたプロセスとストリーム

4.1 各プロセスの役割

プロセス $P_{1,k}$ $(1 \leq k \leq n)$ は次の手続きを実行する。

[手続き 0] $s_{1,k}$ = [[]] $b_{i,k}$= []

プロセス $P_{i,j}$ $(2 \leq j \leq i)$ は、文節位置 j で終わる長さ i の文節列 w[j-i+1:j] の整合的な係り受け構造を計算しストリーム $s_{i,j}$ に送る。従って解析終了時には $s_{n,n}$ に w[1:n]=ws の全ての係り受け構造が入れられる。w[j-i+1:j] から構成されるブロックとチェインを生成するためそれぞれ手続き 1 と手続き 2 を並列に実行する。

[手続き 1] modify(w_{j-i+1},w_j) が真の場合、[手続き 1.1] を実行し、偽の場合、[手続き 1.2] を実行する。

[手続き 1.1] ストリーム $s_{i-1,j}$ にデータが来るのを待ち、係り受け構造が存在しない ことを示す空リスト以外のデータが来た場合、(w_{j-i}, w_j,$s_{i+1,j}$) を $b_{i,j}$ に書き込み、$s_{j-i,j}$ に ((w_{j-i}, w_j,$s_{i+1,j}$),[]) を送る。空リストが来た場合 $b_{i,j}$ に false を書き込む。

[手続き 1.2] $b_{i,j}$ に false を書き込む。(参図 4)

[手続き 2] $2 \leq k < j$ である全ての k に対して並列に [手続き 2.1] を行なう。[手続き 2.1] の出力はマージされストリーム $s_{i,j}$ に送る。

[手続き 2.1] $b_{k,j}$ と $s_{i-k,j-k+1}$ にデータが来るのを待つ。$b_{k,j} \neq$ false かつ $s_{i-k+1,j-k+1} \neq$ [] の場合、($b_{k,j}$,$s_{i-k,j-k+1}$) を出力する。それ以外の場合、[] を出力する。(参図 5)

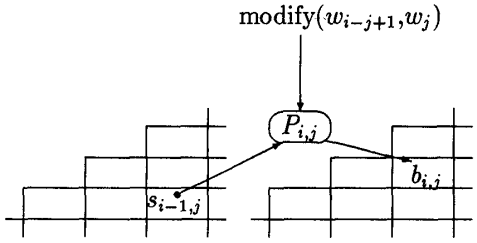

図 4: ブロックの生成

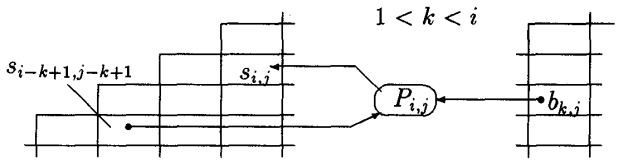

図 5: チェインの生成

(まとめ)

本アルゴリズムは現在 GHC でプロトタイプシステムがインプリメントされている。今後、評価を行ない改良や負荷分散処理方式の検討を行なう予定である。

参考文献

[1] 吉田, 二文節間の係り受けを基礎とした日本語文の構文分析, 信学論 (D),55-D,4, 昭 47

[2] Ueda Kzunori: Guarded Horn Clauses, ICOT Techinical Reprot, No.103,1985