# 増進的曖昧性解消モデルに基づいた日本語解析

秋葉 友良　伊藤 克亘　奥村 学　田中 穂積

本稿では，文に遍在する曖昧性を増進的に解消するモデルに基づき，日本語の係り受け解析を行なう手法を提案する．我々は，意味の曖昧性を増進的に解消するモデルとして一般化弁別ネットワーク (GDN) を提案している．増進的係り受け解析は，係り受けの曖昧性を増進的に解消し，GDN による早期意味解析を可能にする．

本稿の手法は，増進的曖昧性解消モデルの構文解析レベルと意味解析レベルを実現するものである．早期に起動される意味解析は，形態素解析における語の区切りの曖昧性を増進的に解消し，音声認識における音韻レベルの候補を刈り込むことができる．また，文脈解析において，意味解析結果を早期に利用することができる．さらに，本稿の手法は，次に入力される語を予測することから，省略の解析に有効である．

## 1 はじめに

自然言語の解析において，文 (文章) に遍在する曖昧性をどのように解消していくかということは，重要な問題である．自然言語に曖昧性が遍在するのは，文を構成する各部分の情報がどれも部分的で[4]，文の他の部分の情報を用いないと曖昧性を解消できないことに起因する．我々は，意味の曖昧性を増進的に解消するモデル[10]を提案している[7][9]．増進的曖昧性解消モデルは，文を左から右へ読み進む過程で，バックトラックせずに，情報を得られた時点で処理しながら，段階的に曖昧性を解消するモデルである．

自然言語解析は，形態素・構文・意味・文脈などの解析レベルに分割して行なわれることが多い．一般に，ある解析レベルの処理だけでは曖昧性を十分に解消できないが，他のレベルの解析を行なうことによって解消できる場合がある．例えば，構文解析で解消できない曖昧性は，意味解析を行なうことによって部分的に解消できる．したがって，文の部分的な入力の曖昧性を解消するための望ましい方法は，各解析レベルの処理を同時に行ない，各レベルが相互に他のレベルの解析結果を利用して曖昧性を解消することである．そのために，各解析レベルをどのように統合するかが問題になる[3]．各解析レベルは，他の解析レベルとの相互作用を前提に設計されなければならない．我々の目指す増進的解析は，このような統合的解析への試みに他ならない．

本稿では，日本語の係り受け解析を増進的に行なう手法を提案する．この増進的係り受け解析では，係り受けの曖昧性を増進的に解消し，早期に一般化弁別ネットワーク (GDN)[9] による意味解析を起動する．増進的係り受け解析を他の解析レベルの中に埋め込むことによって，他の解析レベルとの相互作用が実現できる．本稿の対象とする構文・意味の解析レベルより下の解析レベルに対して，形態素解析で解消できない語の区切りの曖昧性を解消し，音声認識における音韻レベルの候補を刈り込むことができる．また，早期に得られる部分的な意味

---

A Japanese Analysis based on Incremental Disambiguation Model.

Tomoyosi Akiba, Katunobu Itou, Manabu Okumura, Hozumi Tanaka, 東京工業大学, Tokyo Institute of Technology.

Manabu Okumura. 北陸先端科学技術大学院大学, Japan Advanced Institute of Science and Technology, Hokuriku.

コンピュータソフトウェア, Vol.10, No.1(1993), pp.29–40.
[論文] 1991 年 4 月 3 日受付.

解釈は，より上の解析レベルである文脈解析・語用論解析などで，早期に利用することができる．文脈解析の結果はフィードバックされ，本稿の解析レベルでの曖昧性を解消することが期待できる．さらに，次に入力される語を予測することから，日本語の発話に顕著である省略や「うなぎ文」[12] の解析に有効である．

2節では，まずGDNについて概説し，日本語解析への貢献を明らかにする．3節では，増進的係り受け解析について説明する．4節では，解析例を示す．5節では，本稿の手法の有効性と問題点について述べ，本手法を評価する．

## 2 一般化弁別ネットワーク

弁別ネットワークは決定木の一般化であり，問題解決に広く用いられている．特に，自然言語処理においては，複数の単語の意味 (語義) を圧縮して表現するのに用いられる．弁別ネットワークは有向非循環グラフであり，一つの根ノードと複数の葉ノードをもつ．葉ノードには解が位置する．枝にはラベルとして制約が付与される．問題解決プロセスは，与えられた入力にしたがってネットワークを根ノードから葉ノードに向けて一段ずつたどる過程に対応する．

根ノードから葉ノードに向けてネットワークをたどることで到達可能な葉ノードの数が減少することから，弁別ネットワークを曖昧性解消の手法として用いることができる．弁別ネットワークを用いた意味的曖昧性解消手法には以下の利点がある．

- 弁別ネットワークは，単語の複数の意味を互いに無関係であるとして独立に扱うのではなく，単語の複数の意味間に存在する関係を考慮した表現形式である．
- 単語の複数の意味を互いに無関係であるとして，候補となる単語の意味をリスト形式で保持すると，曖昧性が増大し候補数が多くなる場合には，リストの線形探索の効率が非常に悪くなる．それに対し，弁別ネットワークを用いた手法は，葉ノードのそれぞれの単語の意味に向けて根ノードからネットワークを下向きにたどる操作になるので，線形探索に比べ効率がよい．

しかし，弁別ネットワークには，前もって決定された順序で制約が入力されないとたどれないという問題がある．制約は根ノードに近いものから順に入力されなければならない．決められた順序で入力されない場合，正しい入力が得られるまで評価が遅延されることになる．また，入力されるはずの制約が得られない場合には，ネットワークをそれ以上たどることができずデッドロックに陥ってしまう．

我々は[9]で，任意の順序で制約が入力された場合でも弁別ネットワークを増進的にたどれる手法を提案している．制約の非決定的な順序に対処できる我々の弁別ネットワークを一般化弁別ネットワーク (GDN) と呼ぶ．

### 2.1 GDNを用いた日本語の意味解析

日本語は語順が比較的自由であるため，あらかじめ制約の順序を定めることは難しい．さらに，日本語では，文脈から容易に推測可能な句が省略されることが多い．したがって，日本語解析でGDNの果たす役割は特に重要である．ここでは，GDNを用いた日本語の意味解析について説明する．

文を左から右に読み進み，動詞の意味を増進的に解消することを考える．例えば,「川に橋をかける」という文を考えよう．文の解析前は，文の (入力されるはずの) 動詞がどのような語義にもなり得るという意味で曖昧である.「川に」が入力されると動詞の「に格」に「川」をとり得るという制約が得られることから，曖昧性が解消され動詞の語義となり得る候補が減る．次の「橋を」が入力されることでさらに制約が得られ，曖昧性が増進的に解消される．

図1のように，動詞全体の語義の格フレームに対してGDNを構成して動詞の意味の増進的曖昧性解消を行なう (ここでは単純化のため，8個の動詞で作成したネットワークを考える．また，動詞の語義は異なる表記に対応しているものとする)．ネットワークの枝には，動詞のとり得る表層格とその格を満たし得る名詞句に関する選択制限の組が記述される．ネットワークのノードは，枝に記述された制約により弁別 (選択) された語義を表す．葉ノードには曖昧性のない動詞の1つの語義が対応する．葉ノード以外のノードは，下向きにさらに枝をたどることで，複数のノードに到達し得ることから，到達可能な葉ノードの語義を全て含んだ曖昧な意味を表現している

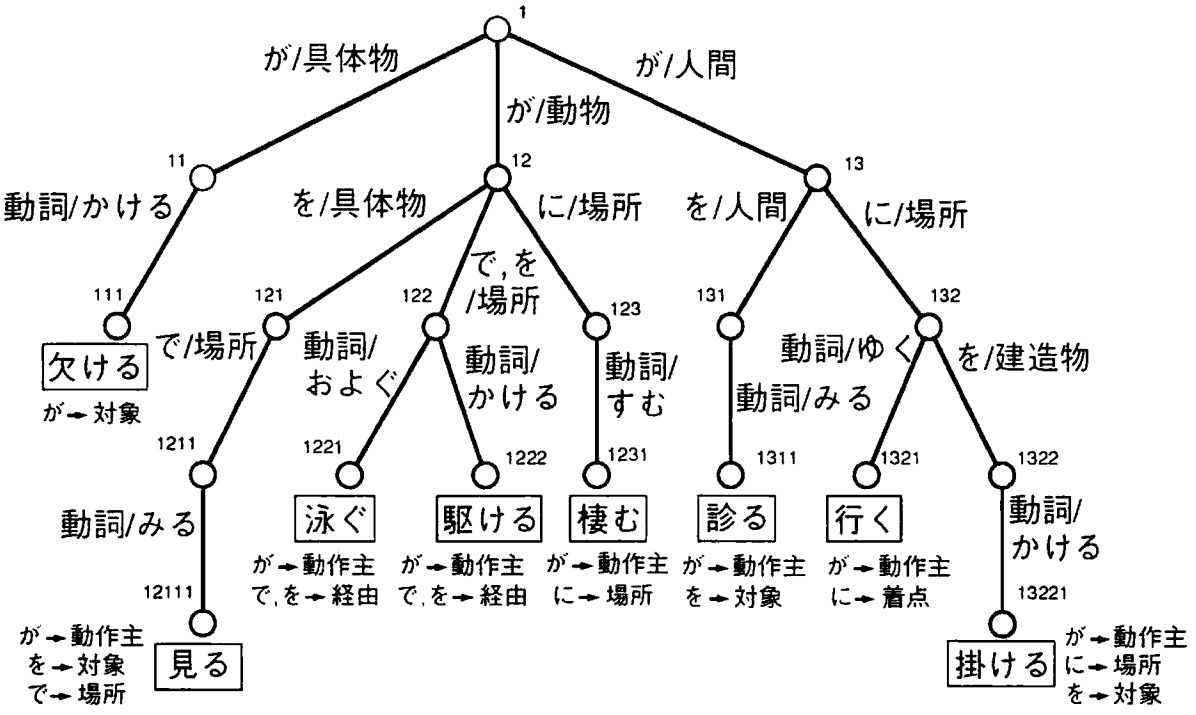


図1 一般化弁別ネットワーク

と考えられる．このGDNを用いて，根ノードから出発し，文節が入力されるごとにネットワークを下向きにたどることで，増進的曖昧性解消を行なう．この動詞の曖昧性解消は，動詞の解析の前に行なわれるので，動詞の語義の予測を行なうことに相当する．

本稿のGDNを用いた解析には，次のような利点がある．ひとつは，動詞が入力される前，すなわち後置詞句が入力された時点で意味解析プロセスが開始できることである．意味解析は文節が得られるごとに進められ，語義の増進的な曖昧性解消を行なう．早期の意味解析は，他の解析レベルとの相互作用に役立つ．例えば，形態素解析や，音声認識における音韻レベルの解析に対する意味制約として働く．また，早期に得られる部分的な意味解析結果は，文脈解析などの上のレベルで早期に利用することができる．もうひとつの利点は，文の解析の途中で次に入力される後置詞句や動詞を予測することである．文節の予測は，省略の推定に役立つ．特に，動詞の予測は，述語が省略された断片的な文や「うなぎ文」[12]の解析を可能とする．

前の例「川に橋をかける」を考えてみる．最初の文節「川に」が入力された時点で意味解析プロセスが動き，ネットワークの枝の制約のうち'に/場所'が満足されるので，ネットワークをノード123あるいは132までたどる．根ノードから，これらのノードまでのパス中で，枝に記述された制約のそれぞれ「が/動物」「が/人間」は，まだ満足されていない．この状態を{(123, 010), (132, 010)}と記述する(これをGDNの状態と呼ぶ)．括弧中の第一項はノードの識別子を表す．第二項は，根ノードからそのノードまでのパス中の制約で，未充足であるものの位置を表すビットベクタである．左側よりの桁が根ノードに近い方を示し，1が未充足0が既充足を表す．ただし，ビットベクタの最も左の桁は，識別子と桁を合わせるためのダミーであり，常に0である．上記のGDNの状態は，葉ノードへの到達可能性から，動詞の語義が「棲む」「行く」「掛ける」のいずれかであることを表している．次の「橋を」の入力では，{(1322, 0100)}までたどる．ビットベクタから，ノード1322までのパス中で制約「が/人間」がまだ満足されていないことが分かり，次の入力としてこの制約を満たす後置詞句が予想される．また葉ノードへの到達可能性から，この時点で語義は「掛ける」に決定される．ここで入力が終了した場合(「川に橋を」)や「うなぎ文」の場合(「川に橋をだ」)でも，意味解析結果が得られる．

意味ネットワーク形式の意味解析結果を得るためには，

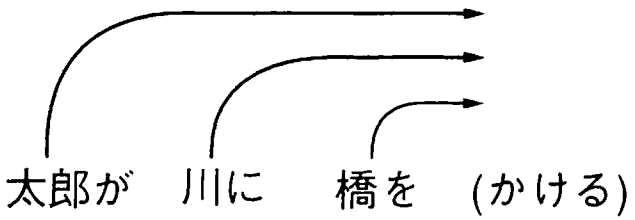


図2 文節間の係り受け

この他に動詞と名詞の間の関係，すなわち深層格を得なければならない．葉ノードに表層格と深層格の対応表を記述しておくことで，深層格も増進的に得られることになる．例えば上の例では，「川に」が入力された時点では「川」の深層格が「場所」と「着点」で曖昧であるが，「橋を」が入力されると(葉ノードが1つに決まることから)「場所」に決まる．

## 3 増進的係り受け解析

前節では，一般化弁別ネットワークを用いてどのように日本語の動詞の意味の曖昧性を解消できるかについて述べた．しかし，実際の文は構造を持っており，文全体の意味解析を行なうには構造解析が必要である．本節では，日本語の文節間の係り受け関係を増進的に解析する手法を提案する．本節の係り受け解析過程とGDNによる意味曖昧性解消過程を相互作用させることで，意味と構造の曖昧性が増進的に解消される．

日本語の係り受け解析を，文を左から右へ読み進み，段階的に決定することを考えよう．例えば，図2は文が途中まで入力された時点での係り受け関係を表すものである．ここでは，係り先となる文節がまだ入力されていないので，文節間の係り受け関係は決定できない．係り先が未決定である文節は，それぞれが(まだ入力されていない残りの文の)どの文節にも係り得るという意味で曖昧である．しかし，構文や意味の制約を考えることで，後ろの入力を待つことなしに，これらの文節が，お互いに同じ文節を係り先とすることが可能か，可能であれば係り先となるのはどのような種類の文節であるかが，ある程度推測可能である．文解析中における，係り先未決定文節の係り先に関するこのような曖昧性を，本稿ではこの時点での「係り受けの曖昧性」と呼ぶ．

従来の係り受け解析では，この係り受けの曖昧性に関しては，その後の係り先文節の入力まで何も行なわない．図2において，「太郎が」「川に」「橋を」が同じ動詞を係り先とすることは，動詞「かける」を解析するまで決定されず，同時に意味曖昧性も解消されない．本稿の係り受け解析では，2.1節で述べたGDNを用いた早期意味解析を行なうために，この係り受けの曖昧性を解消する．そのために，この曖昧性を明示的に展開して処理する．すなわち，複数の文節が同じ文節を係り先とする，あるいは互いに異なる文節を係り先とするという仮定を，被修飾文節を解析する前に生成する．したがって，従来の係り受け解析に比べて解析対象が増大するという問題点があるが，一方，以下に述べる構文・意味制約，ヒューリスティクスを早期に適用できるため，早期に係り受けや意味の曖昧性を解消できるという利点が得られる．

まず，3.1節で，増進的係り受け解析のアルゴリズムを示す．次に，このアルゴリズム中への制約の導入を示す．3.2では構文的制約について述べる．3.3ではGDNによる意味的制約について述べる．3.4では，係りの深さに関するヒューリスティクスについて述べる．3.5では，アルゴリズムのインプリメント法について述べる．[†1]

### 3.1 アルゴリズム

入力文を，長さ$n$の文節列$c_1, c_2, \cdots, c_n$とし，$C_k = \{c_i | 1 \leq i \leq k\}$とする．増進的係り受け解析では，文節を$c_1$から1つずつ入力して段階的に係り受け関係を決定していく．すなわち，文節$c_i$を解析する段階で，文節$c_i$に係る文節の集合$M_i$を決定する．係り受け解析の目的は，全ての$i(i = 1, 2, \cdots, n)$について$M_i$を求めることである．

文節$c_i$を解析している段階でまだ係り先の決定していない文節をスタック$S_i = \{s_{i,0}, s_{i,1}, \cdots, s_{i,m_i}\}$($m_i$はスタックの要素数)で管理する．スタックの要素は文節の集合である．ここで，$s_{i,0}$はスタックのボトムで仮に$Z$を入れておく．$s_{i,m_i}$がスタックトップである．

文節を，0と1の2種類のタイプに分類する．文節のタイプは，同じ文節に係る文節どうしでその間の関係を扱うもの(タイプ1)と扱わないもの(タイプ0)の違いである．本稿では，GDNによる意味解析の対象となる文節をタイプ1とする．すなわち，動詞を係り先とする

†1 以降，アルゴリズムの入力を文節内解析の結果として仮定する．しかし，我々は[1]で文節内解析から統合的に解析する手法を示している．

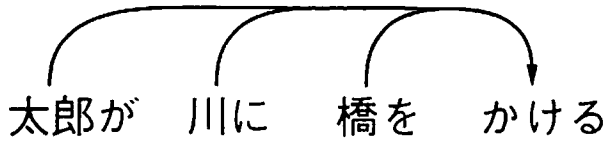


図3 アークの結合

後置詞句 (3.2節の分類でカテゴリ (n,v)) の一部 (直観的には必須格を持つ後置詞句．このような文節の決め方に関しては，5.1節で論じる．) がこれにあたる．その他の文節はタイプ0とする．タイプ1の文節に対しては，図3のようにアークを結合して解析を行なう．一方，タイプ0の文節に対しては，アークを結合しないで係り受け解析を行なう．type($c$) を文節 $c$ からタイプへの関数とする．

同様に，スタック要素も0と1のタイプに分類する．typeSE($s$) をスタック要素 $s$ からタイプへの関数とする．タイプ0のスタック要素には，ただ1つのタイプ0の文節が入る ($\forall_{i,j}\{\text{typeSE}(s_{i,j})=0 \Rightarrow \exists_{c\in C_i}\{s_{i,j}=\{c\} \wedge \text{type}(c)=0\}\}$) とする．タイプ1のスタックの要素には，文節の集合が入り，そのうちの少なくとも1つの文節はタイプ1である ($\forall_{i,j(j\neq 0)}[\text{typeSE}(s_{i,j})=1 \Rightarrow \{s_{i,j}\in 2^{C_i} \wedge \exists_{c\in s_{i,j}}\text{type}(c)=1\}]$)．便宜的に $\text{typeSE}(s_{i,0}(=Z))=0$ とする．あるタイプ1のスタック要素 $s_{i,j}$ に含まれる文節は，全て同じ文節に係る ($\forall_{i,j}\{\text{typeSE}(s_{i,j})=1 \Rightarrow \exists_k \ s_{i,j}\subset M_k\}$) とする．このタイプ1の文節の扱いは，図3のように，同じ文節に係る文節からでるアークどうしを結合していることに相当する．

解析は，初期状態 $m_0=0, S_0=\{s_{0,0}\}$ から始めて，順次 $M_i, m_i, S_i$ を求めていく．

**[増進的係り受け解析]**

入力: 文節数 $n$ の文 $c_1, c_2, \cdots, c_n$

出力: 文節 $c_i$ に係る文節の集合 $M_i(i=1,\cdots,n)$

1. 初期設定: $m_0 \leftarrow 0$, $s_{0,0} \leftarrow Z$, $k \leftarrow 1$
2. 文節 $c_k$ について [1文節の係り受け解析手続き] を行なう．
3. $k=n$ ならば，5へ．
4. $k \leftarrow k+1$，2へ．
5. 終了条件: $m_n=1$ かつ $s_{n,1}=\{c_n\}$ であれば，成功．$M_i(i=1,\cdots,n)$ が求めるものである．そうでなければ，失敗．

文節 $c_{i-1}$ まで解析した時点で，文節 $c_i$ が入力された時の手続きは次のようになる．

**[1文節の係り受け解析手続き]**

入力: 文節 $c_i$，スタックの要素数 $m_{i-1}$，スタック $S_{i-1}=\{s_{i-1,0},\cdots,s_{i-1,m_{i-1}}\}$

出力: 文節 $c_i$ に係る文節の集合 $M_i$，スタックの要素数 $m_i$，スタック $S_i=\{s_{i,0},\cdots,s_{i,m_i}\}$

1. 初期設定:
   $m \leftarrow m_{i-1}, s'_j \leftarrow s_{i-1,j}(for\ j=1,\cdots,m)$,
   $M_i \leftarrow \phi$
2. $\text{typeSE}(s'_m)=1$ ならば，4へ．
3. IN 操作0: 次の (a)(b) のうちのどちらかを選択
   (a) (pop) $c\in s'_m$ が $c_i$ を修飾可能であれば (制約1)，
   $M_i \leftarrow M_i \cup s'_m$，$m \leftarrow m-1$ として，2へ．
   さもなければ，失敗．
   (b) (pass) 5へ．
4. IN 操作1: 次の (a)(b) のうちのどちらかを選択
   (a) (pop) もし，$s'_m$ の全ての要素が $c_i$ を修飾可能であれば (制約1)，
   $M_i \leftarrow M_i \cup s_m$，$m \leftarrow m-1$ として，2へ．
   さもなければ，失敗．
   (b) (pass) 5へ．
5. $\text{type}(c_i)=1$ ならば，7へ．
6. OUT 操作0: (push) $m \leftarrow m+1$，$s'_m \leftarrow \{c_i\}$，$\text{typeSE}(s'_m)=0$，として8へ．
7. OUT 操作1: 次のうちどちらかを選択
   (a) (merge) もし，$\text{typeSE}(s'_l)=1$ となる最大の $l$ に対して，$s'_l$ に $c_i$ を付け加えられるならば (制約2)，
   $$s'_l \leftarrow \bigcup_{j=l,\cdots,m} s'_j \cup \{c_i\},\ m \leftarrow l$$
   8へ．さもなければ，失敗．
   (b) (push) $m \leftarrow m+1, s'_m \leftarrow \{c_i\}$, $\text{typeSE}(s'_m)=1$，8へ．
8. $m_i \leftarrow m$，$s_{i,j} \leftarrow s'_j(\text{for}\ j=1,\cdots,m)$

手続き中の IN 操作と OUT 操作は，それぞれ文節 $c_i$ に入るアークの操作，文節 $c_i$ から出るアークの操作に対応する (図4)．

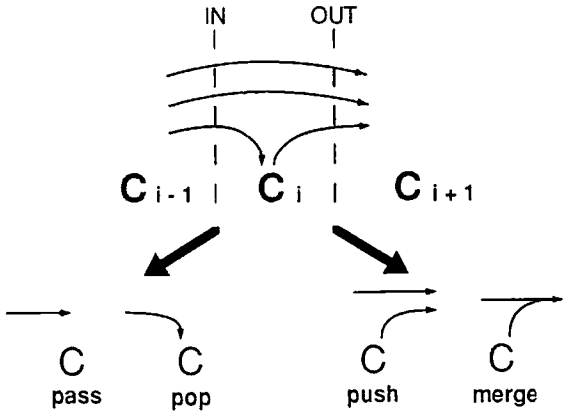


図4 係り受け操作の分類

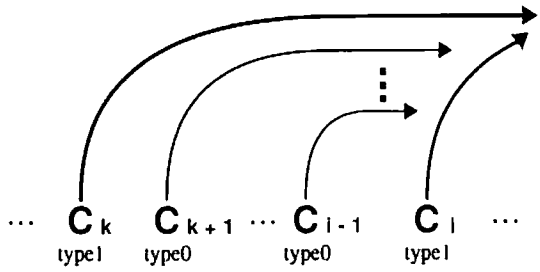


図5 merge操作

[1文節の係り受け解析手続き] 中のステップ7(a)は，タイプ1のスタック要素と文節の間にタイプ0のスタック要素を挟んでmergeが行なわれる場合を考慮している(図5)．そのような場合，間に挟まれたタイプ0のスタック要素も一緒にmergeされ，タイプ1のスタック要素に含まれることになる．例えば，「太郎が月曜日に橋をかける.」という文を考える．必須格の後置詞句「太郎が」「橋を」がタイプ1，任意格の後置詞句「月曜日に」がタイプ0である．$c_2$ =「月曜日に」まで解析した時点でのスタック$S_2$は，$s_{2,1}$ ={「太郎が」}, $s_{2,2}$ ={「月曜日に」}, (typeSE($s_{2,1}$) = 1, typeSE($s_{2,2}$) = 0) である．次の入力$c_3$ =「橋を」では，$s_{2,1}$(=「太郎が」)と$c_3$がタイプ1どうしでmergeが可能である．「太郎が」と「橋を」が同じ動詞を係り先とするならば，係り受け非交差の原則より，「月曜日に」も同じ動詞を係り先とすることがわかる．したがって，タイプ0である$s_{2,2}$も同時にmergeされ，$S_3$は，$s_{3,1}$ ={「太郎が」「月曜日に」「橋を」}となる．

表1 係り受け関係によるカテゴリの分類

| (*Mod*, *Type*) | 例 | (修飾先) |
|---|---|---|
| ( n , n ) | 日本語の | (論理) |
| ( n , v ) | 日本語を | (話す) |
| ( v , n ) | 読む | (本) |
| ( v , v ) | 歩いて | (帰る) |

### 3.2 構文的制約の導入

上で述べた手続き中の「制約1」「制約2」に，構文的な制約と意味的な制約(GDNを用いた制約)を導入することで，係り受けの曖昧性解消を行なう．まず，構文的な制約について述べる．

まず文節を，体言(n)と用言(v)の2種類に分類する．そして，各文節に，自分自身の種類(*Self*)と係り先の種類(*Mod*)の組(*Self*, *Mod*)をラベル付けする．これを文節のカテゴリと呼ぶ．カテゴリは，(n,n), (n,v), (v,n), (v,v)の4種類[†2]に分類される(表1)．このカテゴリは，文節内の解析によって得られる．

構文的制約は次のように記述できる．[†3]

(制約1)

$$\forall_{c\in s'_m}, \mathrm{mod}(c) = \mathrm{self}(c_i),$$
$$\neg\exists_k\{k = \max\{x | m \le x \le m_{i-1},$$
$$\mathrm{typeSE}(s'_x) = 1\} \wedge \mathrm{cat}(s'_k) = \mathrm{cat}(s'_m)\}$$

(制約2)

$$\mathrm{cat}(c_i) = \mathrm{catSE}(s'_l),$$
$$\forall_{j=l+1,\cdots,m}\forall_{c\in s'_j}\{\mathrm{mod}(c) = \mathrm{mod}(c_i)\}$$

ただし，self($c$)とmod($c$)を，文節$c$からそのカテゴリのそれぞれ*Self*と*Mod*の値を返す関数，cat($c$)を，文節からカテゴリへの関数とする．同様に，catSE($s$)をスタック要素$s$からカテゴリへの関数とする．ただし，$s$がタイプ0の場合，catSE($s$)の値はそれに含まれるただ1つの文節のカテゴリ，$s$がタイプ1の場合，catSE($s$)の値はそれに含まれる任意のタイプ1文節のカテゴリとする．

†2 その他のカテゴリとして，副詞・連体詞などが考えられる．これらは，*Self*を省略した(-,v)(-,n)などとして，同様に扱うことができる．

†3 実際の処理では，(制約2)と$s'_m$の作り方から，タイプ1の$s'_m$中のタイプ1の文節は全て同じカテゴリに属しているため，各スタック要素に対し構文的制約の検査は一度行なうだけで済む．

(制約 1) の第 2 式は，カテゴリの同じタイプ 1 のアークが，連続して同じ文節に係れないことを表している．

### 3.3 GDN の導入

増進的係り受け解析の意味的制約として GDN を導入する．増進的係り受け解析では，ある 1 つの文節に係る (予定の) タイプ 1 の文節の集合を同一のタイプ 1 のスタック要素に入れて管理する．本稿では，GDN による増進的意味解析を行なうため，カテゴリ (n, v) の文節の一部をタイプ 1 とした．このタイプ 1 のスタック要素に含まれる文節集合は同じ動詞に係ることが仮定されているので，ある GDN の状態に対応していなければならない．そのような GDN の状態に対応できなければ，その後置詞句集合が共に係れる動詞がないことを意味している．例えば，$s_{i,j}$ = { 太郎が, 川に, 橋を } であれば，対応する GDN の状態は，{(1322, 0000)} であり，この文節集合は同じ動詞 (掛ける) に係ることができる．一方，$s_{i,j}$ = { 端が, 川を, 山に } であれば，対応する GDN の状態が存在しないため，これらの文節集合は同じ動詞に係れない．タイプ 1 のスタック要素に後置詞句を加える (merge する) ごとに，GDN をたどれるかどうかの検査を行ない，意味的制約とする．GDN をたどることは，2 節で述べたように，同時に意味曖昧性も解消している．

**(制約 2)**

catSE($s'_m$) = cat($c_i$) = $(n, v)$ の場合，次の検査を行なう．

GDN($s'_m$) から $c_i$ の制約で GDN をたどることができる．(そのときの状態を新たな GDN($s'_m$) の値とする．)

ただし，GDN($X$) $\stackrel{\text{def}}{=}$ ⟨文節集合 X に対応する GDN の状態⟩，とする．

push 操作によってスタックの要素に加える最初の文節については，GDN を根ノードからたどらなければならない．[1 文節の係り受け解析手続き] 中の 7b (push) に次の手続きを加える．

もし，cat($c_i$) = $(n, v)$ ならば，GDN の根ノード (状態 (1, 0)) から $c_i$ の制約で GDN をたどり，その状態を GDN($s'_m$) の値とする．

(制約 1) は，係り受けの適格性を判定する制約である．後置詞句の集合がある動詞に係るための条件は，後置詞句の集合に対応する GDN の状態から動詞の制約によって GDN をたどれることである．

**(制約 1)**

typeSE($s'_m$) = 1 ∧ catSE($s'_m$) = $(n, v)$ ∧ self($c_i$) = $v$の場合，次の検査を行なう．

GDN($s'_m$) に対応する GDN の状態から $c_i$ の制約で GDN をたどることができる．(そのときの状態を新たな GDN($s'_m$) の値とする．この値は以下の格要素型連体修飾に用いられる．)

格要素型の連体修飾 [13](すなわち，カテゴリ (v,n) の文節が (n,v) か (n,n) の文節に係る場合) についても，同様に GDN をたどれるかどうかで判定することができる．まず，連体修飾を行なう (v,n) 型の文節 $c_i$ に対して，上記の (制約 1) によって得られる $c_i$ を係り先とするタイプ 1(n,v) 型の文節の集合 $s'_m$ の最終的な GDN の状態 GDN($s'_m$) を割り当てる．これを，vmap($c_i$) と表す．ただし，このような $s'_m$ が存在せず $c_i$ が単独で連体修飾する場合は，GDN を根ノードから $c_i$ の制約でたどった時の状態とする．格要素型連体修飾の制約は，次のように記述できる．

(制約 1)

catSE($s'_m$) = $(v, n)$ かつ self($c_i$) = $n$ の場合，次の検査を行なう．

全ての $c \in s'_m$ に対して，vmap(c) から，格を任意とした $c_i$ の制約で GDN をたどることができる．ただし，「格を任意とした $c_i$ の制約」とは，$c_i$ の制約 ⟨case⟩ / ⟨const⟩ のうち，⟨case⟩ を任意の値とマッチング可能としたものである．

### 3.4 スタックの深さ

以上の構文制約・意味制約では，アルゴリズム中の IN 操作での (pop) と OUT 操作での (merge) を制約している．一方，IN 操作での (pass) と OUT 操作での (push) は，その可能性を否定する絶対的な制約は存在せず，常にその可能性を考慮しなければならない．しかし，(pass) や (push) が連続すると，必然的にスタックが深くなる．これに対し，以下のように，スタックの深さに関するヒューリスティクスを導入する．

スタックの深さは，英語における center embedding [8] の深さ，日本語では [6] における「自包的構造」の深

さに対応しており，文の読みやすさに影響を与えることから，実際に発話される文ではそれほど深くならないと考えられる．例えば[6]では，「太郎ガ [花子ガ [夏子ガ愛シテイル] ソノ少年ニ書イタ] 手紙ヲ読ンダ．」を理解度の低い文としている．

スタックの深さ (stack_depth) を次のように定義する．

$$\mathrm{stack\_depth}(S_i) = \sum_{j=2}^{m_i} \mathrm{ind}(s_{i,j}) + 1$$

ただし，関数 ind : $\{s_{i,j} | j = 2, \cdots, m_i\} \mapsto \{0, 1\}$ を，

$$\mathrm{ind}(s_{i,j}) = \begin{cases} 1 & \textit{if } \mathrm{typeSE}(s_{i,j}) = 1 \ \vee \\ & \quad \forall_{c_1 \in s_{i,j}} \forall_{c_2 \in s_{i,j-1}} \\ & \quad \mathrm{mod}(c_1) \neq \mathrm{mod}(c_2) \\ 0 & \textit{otherwise} \end{cases}$$

とする．

関数 ind は，引数に与えたスタック要素に含まれる文節が，すぐ下のスタック要素に含まれる文節と同じ文節を修飾できる場合は0，そうでない場合は1となる．

stack_depth の定義は，タイプ0のスタック要素の係り先を最小に見積もった場合の自包的構造の深さを表している．自包的的構造の深さは，おおよそある一定数 $D$ †4を超えないと考えられる．アルゴリズム [1文節の係り受け解析手続き] の8に次の制約を加える．

[スタックの深さに関するヒューリスティクス]

stack_depth$(S_i) > D$ならば，失敗．

### 3.5 インプリメント

このアルゴリズムは，[1文節の係り受け解析手続き] のステップ3,4,7で任意の処理を選べるので，非決定的である．この非決定性に対し，我々のインプリメントでは横型探索を行なう．すなわち，スタックを順次分岐させて解析を行なう．分岐したスタックに対してプロセスを割り当てることで，並列解析が実現できる[11]．

その際，スタックを木構造化することで無駄な再計算を避けることができる[11]．木構造化スタックは，スタックどうしで共通の (スタック要素の) 先頭部分を融合したものである (図6)．スタックを木構造化することで，同じスタックトップをもつスタックの操作を1回でまとめて処理することができる．さらに手続き中の (push) は，全てのスタックに対して1度行なうだけで済む．pushするスタック要素でそのまま木構造化すればよいからである (図7)．

(n,v) 型の文節集合を持つスタック要素には，文節集合の代わりに GDN の状態を入れる．GDN の状態は文節集合をコンパクトにまとめた表現であると考えられる．

現在このアルゴリズムは，SICStus-Prolog と flat-GHC でインプリメントを行なっている．しかし，まだ大規模な実験は行なっていない．

## 4 解析例

前節で説明した増進的係り受け解析の解析例を示す．この係り受け解析への入力は文節内解析の結果である．以下の解析例では，文節内解析が結果として ⟨ ⟨GDN への入力となる制約⟩, ⟨ カテゴリ ⟩, ⟨ 文節のタイプ ⟩ ⟩ の3つ組を出力すると仮定する．カテゴリは，表1の分類に従ったものである．ここでは，カテゴリ (n,v) の文節をタイプ1，その他をタイプ0とする．「太郎が」の文節内解析の結果は，⟨ が/太郎, (n,v) 1⟩ となる．また，GDN は図1を使用する．スタック要素は ⟨ タイプ ⟩, ⟨ カテゴリ ⟩, ⟨ 文節の集合を表す GDN の状態 ⟩, ⟨ 格関係のリスト ⟩ で表現する．ここでは，意味解析結果を得る手法については，説明が繁雑になるのを避けるため簡単な処理にとどめる．

「川を海に棲む魚が泳ぐ」という文の解析例を示す．まず，文節内解析によって「川を」の解析結果 ⟨ を/川, (n,v), 1⟩ が係り受け解析に渡される．スタックの初期状態は空なので，typeSE$(Z) = 0$ より IN 操作0が選択され (アルゴリズム中の，step 2)，(pop) が失敗し (pass) が成功する．したがって，$M_1 = \phi$が得られる (step 3)．type$(c_1) = 1$ より OUT 操作1が選択され (step 5)，スタックが空なので (push) だけが成功する．(push) では $c_1$がタイプ1でカテゴリ (n,v) の文節なので，GDN を根ノードから「を/川」でたどる．その状態 $\{(122, 010)\}$ をスタックに積む (step 7b)．この GDN の状態と格関係は，到達可能な葉ノードの集合から，意味表現 [{ 泳ぐ, 駆ける }, 経由/川](ただし，{a,b,... } は，選択を表す) を内包している．

†4 $D$の実際の値について，我々はこれまでの調査結果から3程度という感触を得ているが，厳密にはより多くの調査が必要である．また，スタックの深さの計算式とその意味についても，今後の課題である．

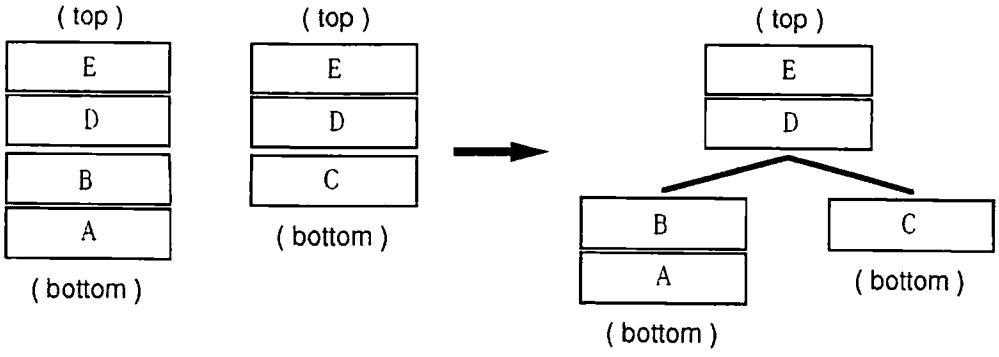


図6 木構造化スタック

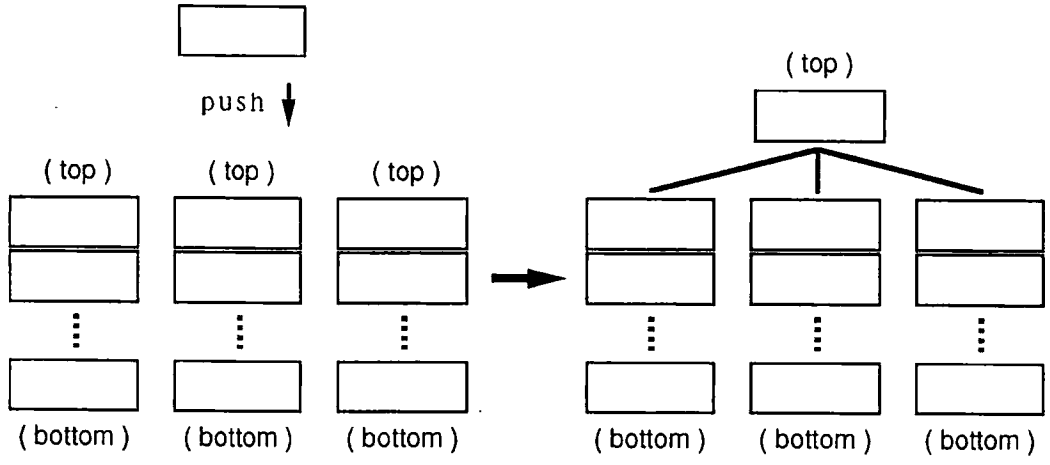


図7 PUSH

(top)

| 1, (n,v), {(122,010)}, [(を)/川] |
|---|

(bottom)

次の「海に」の文節内解析の結果は，⟨ に/海, (n,v), 1⟩ である．スタックトップはタイプ1であるのでIN操作1が選択される (step 2)．(pop) は構文制約 (スタックトップの *Mod* と現文節の *Self* が一致する) を満たさないので失敗し，(pass) によってスタックはそのままOUT操作に渡される ($M_2 = \phi$)(step 4b)．step 5では，文節のタイプが1なのでOUT操作1が選択される．step 7a(merge) では，構文制約 (スタックトップのカテゴリと現文節のカテゴリが一致する) は満たしている．しかし，スタックに積まれているGDNの状態から現文節の制約「に/海」({(123,010), (132,010)}) でGDNをたどることができないため，GDNの制約を満たさず失敗する．すなわち，文節「川を」と「海に」が同じ動詞に係らないということがこの時点で判別される．結局，step 7b(push) のみが成功し，スタックは次のようになる．スタックトップのGDNの状態と格関係が内包している意味表現は，[{ 棲む, 行く, 掛ける }, { 場所, 着点 }/海] である．

(top)

| 1, (n,v), {(123,010),(132,010)}, [(に)/海] |
|---|
| 1, (n,v), {(122,010)}, [(を)/川] |

(bottom)

「棲む」($c_3$) の文節内解析は，⟨ 動詞/棲む, (v,n). 0⟩ である．step 4a(pop) は，スタックトップの状態から制約「動詞/棲む」でGDNをたどることができるので成功する ($M_3 = \{$ 海に $\}$)．また，「棲む」の葉ノード (1231) に記述されている格の対応表によって，「に」格の深層格が「場所」であることがわかり，意味表現 [棲む, 場所:海] が得られる．一方，step 4b(pass) も成功す

る ($M_3 = \phi$) のでスタックは2つに分岐する．ここでは (pop) の方の処理について説明する．(以下，各分岐で最終的に成功する方の選択を行なう．)

$c_3$のタイプが0なので，OUT 操作0が選ばれる (step 5)．ここで，後続する解析で格要素型の連体修飾を扱うために，step 4a(pop) で計算した GDN の状態をスタックに積む (step 6)．

(top)

| 0, (v,n), {(1231,0100)}, [棲む, 場所/海] |
|---|
| 1, (n,v), {(122,010)}, [(を)/川] |

(bottom)

「魚が」($c_4$) の文節内解析は，⟨ が/魚, (n,v), 1⟩ である．step 3a(pop) で連体修飾の適格性は，スタックトップの GDN の状態から，制約「⟨ 任意の格 ⟩/魚」で GDN をたどれることで判別される (⟨ 任意の格 ⟩ はどの格ともマッチング可能であることを意味する)．スタックトップにある GDN の状態は，識別子より動詞「棲む」の格フレームを表しており，ビットベクトルから「が」格がまだ得られていないことが解る．この状態から GDN をたどれることは，未定の格 (この場合「が格」) に連体修飾される名詞が適合することを意味する．この例では (pop) に成功し，$M_4 = \{$ 棲む $\}$ が得られる．同時に「魚」が意味表現中の「が」格すなわち動作主格を埋めることが解析され，意味表現 [棲む, 場所/海, 動作主/魚] が得られる．

step 7a(merge) で，構文制約と GDN 制約共に成功し，スタックは次のようになる．内包する意味表現は，[{ 泳ぐ, 駆ける }, 経由/川, 動作主/魚] である．

(top)

| 1, (n,v),{(122,000)} [(を)/川, (が)/魚] |
|---|

(bottom)

最後の文節「泳ぐ」($c_5$)(文節内解析は ⟨ 動詞/泳ぐ, (v,n), 0⟩) では，step 4で (pop) に成功 ($M_5 = \{$ 魚が $\}$)(step 4a)．意味表現 [泳ぐ, 経由/川, 動作主/魚] が得られる．step 6で (push) が行なわれると，スタックは最終的に次のようになる．これは終了条件に適合する．

(top)

| 0, (v,n),{(1221,0000)} |
|---|

(bottom)

## 5 考　察

本節では，本稿の手法を実際の文の解析に用いる時の問題点とそれに対する対処法について述べ，増進的曖昧性解消モデルの評価を行なう．

### 5.1 任意格・主題の扱い

増進的係り受け解析では，文節を2つのタイプに分割して処理を区別した．タイプ1の文節の解析では，早期に係り受けの仮定を生成して早期意味解析を可能としている．一方，タイプ0の文節の解析では，係り受けの可能性を曖昧なまま保持している．タイプの区別は，その文節を増進的解析の対象とするかしないかの区別である．

3節では，GDN を用いた増進的意味解析を行なうため，カテゴリ (n,v) の文節の一部をタイプ1として扱ってきた．そのためには，(n,v) 型の文節を，増進的意味解析の対象として扱うものとそうでないものとに区別することが必要である．動詞の格の種類における必須格・任意格の区別がおおよそこれに当たる．任意格は全ての動詞に存在することから，任意格を持つ文節を制約としても動詞の語義は弁別されないからである．しかし，動詞によっては通常任意格と思われているものが必須格のような振舞いをするものも存在する．どのように，必須格・任意格を区別し，GDN を構成するかは，今後検討しなければならない問題であると考える．

また，格助詞のない副助詞「は」「も」だけの文節は，GDN をたどる制約を，格を任意とした制約，すなわち ⟨ 任意の格 ⟩/⟨ 名詞句 ⟩ とすることで扱うことができる．しかし，格に対する制約が任意となる分，多くの GDN の枝とマッチングしてしまう．そのため，この場合も語義の弁別のための強い制約とはならない上，GDN の状態表現がコンパクトにならない (例えば，文節「橋は」に対して，制約「が/具体物」「を/具体物」... などの制約を持つ枝が適合してしまい，GDN の状態を表すノードのリスト ({(11,00), (121,010)...}) が大きくなる) といった問題がある．このような文節も増進的係り受け解析としてはタイプ0として扱える．しかし，主題化された文節は，意味・文脈解析において特別な働きを持つと考えられ，これをどのように扱うかは今後の課題である．

### 5.2 格の交換

助動詞の中には，動詞と名詞の格関係を交換させるものがある．使役・受身の助動詞がそれに当たる．例えば，受身の助動詞を含む文「橋が川に掛けられる」で「が」格を占める「橋」は，動詞「掛ける」の格フレームでは「を」格に入る．このような助動詞を含む動詞句に対しては，格関係が交換してしまうため，動詞の語義を葉ノードとするGDNを増進的にたどることができない．

この問題に対しては，動詞の語義と受身・使役情報の両方を葉ノードとしてGDNを構成することで対処できると考える．すなわち，語義「掛ける」について，「掛けられる」も葉ノードとする．「掛けられる」の格パターンは「が/建造物，に/場所」となる．このような格パターンをパスとして持つ葉ノードをGDNに加える．

### 5.3 有効性

増進的曖昧性解消モデルは，文の部分に対して全ての解析レベルの処理を可能な限り行ない，その部分の持つ情報をできるだけ早期に抽出し，解の増進的洗練を行なう．本モデルの応用である増進的係り受け解析には，次のような利点がある．

- 各解析レベルを同時的に実行することによって，解析レベル間の相互作用を扱う．ある解析レベルで解消できなかった曖昧性は，他の解析レベルによって解消できる．
  - 増進的係り受け解析は，構文と意味の解析を相互作用させることで曖昧性解消を行なっている．
  - 早期に意味解析を起動し，より下の解析レベルでの曖昧性を解消することができる．例えば，形態素解析で解消できない語の区切りの曖昧性を増進的に解消できる．また，音声認識における音韻レベルの候補をバックトラックしないで刈り込むことができる．さらに，GDNによる解析は次の入力の予測を行なうことができる．我々は，本モデルを音声認識システム[2]の制約として利用する予定である．
  - 早期に意味解析を行なうことで提案される部分的な意味解釈は，より上の解析レベルである文脈解析・語用論解析などで早期に利用することができる．文脈解析の結果はフィードバックされ，本稿の構文・意味解析レベルやより下の解析レベルでの曖昧性を解消することが期待できる．
- 日本語の実際の発話において顕著に現れる省略を扱うことができる．特に，これまでの解析法では例外として扱ってきた，述語部の省略や「うなぎ文」を扱うことができる．本モデルでは，動詞の解析前に後続する動詞の予測が可能となるからである．

## 6 おわりに

本稿の手法は，増進的曖昧性解消モデルの構文解析レベルと意味解析レベルを実現するものである．我々の目標は，増進的曖昧性解消モデルに基づき，より多くの言語解析レベルを統合することである．特に文脈解析をどのように導入するかが大きな課題である．本稿の対象とする解析レベルより下のレベルに関しては，[1]で文節内解析と本稿の手法を統合する手法について示している．また，[5]で未定義語を含んだ辞書引きをバックトラックなしに行なう手法を示しており，本手法に自然に組み込むことができると考える．

本稿のGDNを用いた解析は，動詞全体の語義を用いて曖昧性解消を行なっているために，動詞が得られてから意味解析を行なう従来の手法に比べて必ずしも効率が良いとは言えない．特に問題は，多数の動詞を用いて大規模な弁別ネットワークを構成した場合である．この場合，扱う語義(葉ノード)の数が増大するが，制約の枝を共有することで，ネットワーク自体は語義の数に比例しては大きくならないことが期待される．

動詞の語義の予測に関しては，格フレームの情報以外に，談話的な情報，解析する文章の対象領域の知識などが重要となる．対象領域を限定したGDNを用いれば，語義予測に関する対象領域の知識を表現できると考える．一方，本稿で扱っている格フレーム情報だけでは，任意の領域で十分な予測を行なうのは難しい．どのように談話情報を利用するかは，今後の課題である．

## 謝辞

本稿の査読者の方々には，大変有意義なコメントをいただきました．ここに感謝の意を表します．

## 参考文献


[1] 秋葉友良，伊藤克亘，奥村学，田中穂積: 増進的曖昧性解消モデルに基づいた統合的日本語解析，「自然言語処理における統合」シンポジウム論文集, 1991.

[2] 伊藤克亘，速水悟，田中穂積: 拡張 LR 構文解析法を用いた連続音声認識，信学会技術報告, SP90-74, pp. 49–56, 1990.

[3] Carter, D. M.: Control Issues in Anaphor Resolution. SRI International technical reports, 1987.

[4] 橋田浩一: AI とは何でないか — 情報の部分性について，*bit*, Vol. 20, No. 8(1988), pp. 48–60.

[5] 堀内靖雄，伊藤克亘，田中穂積: 拡張 LR 構文解析アルゴリズムによる未定義語を含む日本語文の構文解析，情報処理学会第 40 回全国大会, 1990.

[6] 久野すすむ: 日本文法研究，大修館書店.

[7] 奥村学，田中穂積: 自然言語における意味的曖昧性を増進的に解消する計算モデル，人工知能学会誌, Vol. 4, No. 6(1989).

[8] Johnson-Laird, P. N.: Mental Models — Towards a Cognitive Science of Language, Inference and Consciousness, Cambridge Univ. Press, Cambridge, 1983. (邦訳, AIUEO: メンタル・モデル — 言語・推論・意識の認知科学，産業図書, 1987)

[9] Okumura, M. and Tanaka, H.: Towards Incremental Disambiguation with a Generalized Discrimination Network. AAAI, 1990.

[10] Mellish, C. S.: Computer Interpretation of Natural Language Descriptions. Ellis Horwood, 1985. (邦訳, 田中穂積: コンピュータのための自然言語意味理解の基礎，サイエンス社, 1987)

[11] 沼崎浩明，田村直良，田中穂積: 並列論理型言語による一般化 LR 構文解析アルゴリズムの実現，情報処理学会, NL74-5, 1989.

[12] 奥津敬一郎: 「ボクハウナギダ」の文法 — ダとノ — くろしお出版, 1978.

[13] 佐藤龍一，田中穂積: 常識を用いた日本語連体修飾の解析，電子情報通信学会, NLC89-17.