最適解探索に基づく日本語意味係り受け解析

平 川 秀 樹[†,☆]

自然言語処理では，形態素，構文，意味といった各種処理レベルで組合せ的な曖昧性が存在する．さらに，それぞれのレベルにおける言語的・意味的な知識の大半は優先的にのみしか適用できず，また，異なったレベル間の知識の相互干渉が存在する．自然言語のこうした性質のため，組合せ爆発を避けるために導入される制約的な知識の適用は，自然言語処理システムの能力を制限してしまう．本稿では，新しい言語処理へのアプローチとして，日本語の構文/意味レベルの曖昧性を統合的に，かつ，選好的に扱う文解析方式を提案する．構文的・意味的な曖昧性を統合・パックして表現する手法として独自の意味係り受けグラフを用い，そこから係り受け非交差条件・多重格占有禁止条件などの制約を満足しながら，総合選好スコアが最大である意味解析構造を抽出する．最適解の探索には，分枝限定法を採用したアルゴリズムを提案し，その試作実験評価についても述べる．

Semantic Dependency Analysis Method for Japanese Based on Optimum Tree Search Algorithm

Hideki Hirakawa[†,☆]

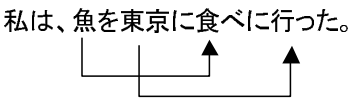
There are combinatorial order of ambiguities in each level of natural language analysis, such as morphology, syntax, semantic level. Moreover, much of linguistic knowledge in each level is preference knowledge and has mutual interference. Deterministic processing is usually introduced to avoid the combinatorial explosion. However, this will restrict the ability of natural language processing system because of the above-mentioned features of linguistic knowledge. This paper describes a sentence analysis method which uniformly evaluates syntactic and semantic preference knowledge, and shows an algorithm (based on Branch-and-bounding method) to search the optimum semantic tree from a semantic dependency graph which holds syntactic and semantic ambiguities in a Japanese sentence.

1. はじめに

自然言語文の解析における最大の問題点は，形態素/構文/意味/文脈などの各種レベルにおける曖昧性の中から，いかにして正しい解釈を認識するかという点である．形態素・構文といった各レベルにおいて組合せ的な数の解釈が存在するばかりでなく，一般に，各種レベルの知識は次の性質を持っており，組合せ爆発とともにその取扱いが本稿の主要課題である．

a. ある解釈の優先性（preference）に関する場合が多い（その知識を制約的（restrictive）に適用できない）．
b. 異なったレベルにおける知識の干渉がある．

文の解析過程において，単語の品詞，係り受け先など，種々の可能性が存在するが，それらの可能性は何らかの形で処理プロセス中で生成される．ここで，知識の制約的適用とは，この仮定を棄却するような適用の仕方であり，優先的適用とは，生成された仮説に優先順位をつけるような適用の仕方である．一般に，日本語の単語接続関係などは，制約的に適用される知識であり，意味的な知識は，優先的に適用されるべき知識である（Semantic Preference）[1)]．一般的に制約的知識と認知されている知識でも例外が存在する場合は多い．例えば，日本語の「係り受けの非交差条件」は，制約知識と考えられているが，次の文のような例外もごくまれに存在する．



知識の制約的な適用は，候補の限定を行うため，仮説の組合せ的爆発を軽減し，処理を効率化するという利点を持つが，その過度の適用は上記 a の知識の性質

† 株式会社東芝研究開発センター
Toshiba R&D Center
☆ 現在，知識メディアラボラトリ
Presently with Knowledge Media Laboratory

を無視することになり，システムの解析能力を限定してしまうため，避けることが望ましい．

次に，b の例であるが，意味知識だけでなく構文的な知識（ヒューリスティックスを含む）にも，優先的に適用すべき規則が多々あり，これらが統合的に適用されて適切な解釈が得られると考えられる．たとえば，次の文を考える．

S1. X は Y が大きい．
S2. 手は太郎が大きい．
S3. 太郎が大きい手．

「大きい」が，主題と対象という（深層）格を持つとすると，S1 では，助詞に関する知識（「は」は主題を導きやすい）を用い，S2 では，意味的な知識（「太郎」と「手」は，全体–部分の関係にある）を用いて解釈を行うのが自然である．S3 では，意味的な知識と言語上の知識（主題の要素を修飾する埋め込み文はない）との衝突により適切な判断が困難である．このように，文の解釈を決定する際には，構文的知識と意味的知識とが，その文の解釈に関係し，通常，それらは，優先的に適用され，総合的に評価される必要がある．

上記 a，b の性質をうまくシステムで扱うためには，
(1) 各種レベルの多数の解釈を効率良く保持し，
(2) それらに対し各種知識による優先度を設定し，
(3) その中から最適な解釈を探索する，

という 3 つの処理が必要である[2]．(1) に関しては，たとえば構文木を効率良く保存するために，それらをパックして保存する方法[3),4)]が提案されている．また，6 章で触れるが，制約依存文法（Constraint Dependency Grammar，CDG と略記）による解析では，制約マトリックスを用いて，係り受けの可能な解釈を内包的に保持することにより，個々の解釈に展開することなしに制約伝搬による解の絞り込みを行うワク組みを提供している[5)]．文献 2) では，日本語の構文・意味解析に関して，上記 (1)〜(3) の処理を実現するため，意味係り受けネットワークというデータ構造を提案した（本論文では，都合上，意味係り受けグラフと呼ぶ）．(3) に関しては，組合せ的な計算が最終的に要求され，それをいかに効率良く実現するか，また，実際の計算機で取り扱うことが可能かといった点が問題となる．本論文では，分枝限定法（branch-and-bound method）[6)]を，(3) の処理，すなわち意味係り受けグラフからの最適解探索に適用し，それを用いた解析実験について報告する[7)]．2 章では，係り受けグラフとその構成法，3 章では，最適解探索アルゴリズム，4 章では，その実験結果，5 章では，アルゴリズムの改良などについて述べ，6 章で関連研究について述べる．

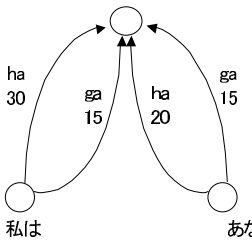


図 1 「私はあなたが好きです」の意味係り受けグラフ
Fig. 1 Semantic dependency graph for “私はあなたが好きです”.

2. 意味係り受けグラフ

日本語解析における曖昧性には，語の役割に関する曖昧性，語の係り先に関する曖昧性，語の係り受け意味関係に関する曖昧性，さらに文脈要素を考慮すれば語の指示，同一性に関する曖昧性がある．ここでは，これらのうち，語の係り先およびその意味関係に関する曖昧性に焦点をあてることとする．

2.1 意味係り受けグラフの構成

語の係り先およびその意味関係を表現する手段として，意味係り受けグラフというデータ表現を用いる．このグラフにおいて，ノードは，1 つの単語（文節）における 1 つの役割に相当する．2 つのノードを結合するアークは，その 2 つのノード間に係り受け関係が存在することを示し，そのアークの名前は，その係り受けの意味関係を示すものとする．また，各アークには，重みと呼ばれる優先得点が付加されている．**図 1** は，「私はあなたが好きです.」という文に対応する意味係り受けグラフである．

図中の ha，ga は，それぞれ “行為者”，“対象” に相当する意味関係である．意味係り受けグラフは，日本語の係り受け関係が，文の先頭方向から文末方向へ向かうという性質上，DAG（Directed Acyclic Graph）となる．日本語の文の意味構造は，この係り受けグラフ上の well-formed な木である．図 1 の例では，4 つの木が含まれているが，このうち「私− ha → 好き← ga −あなた（I love you）」と「あなた− ha →好き← ga −私（You love me）」の 2 つの木が，well-formed な木となる．ここで well-formed とは，次の 2 つの条件を満足する木である．

a. どの 2 つのアークをとっても，それらは交差しない（非交差条件）．
b. どの 2 つのアークをとっても，それらは用言の同じ格を埋めない（多重格禁止条件）．

図 1 の例では，b の条件により，well-formed な木は 2 つに限定されるわけである．以下では，well-formed

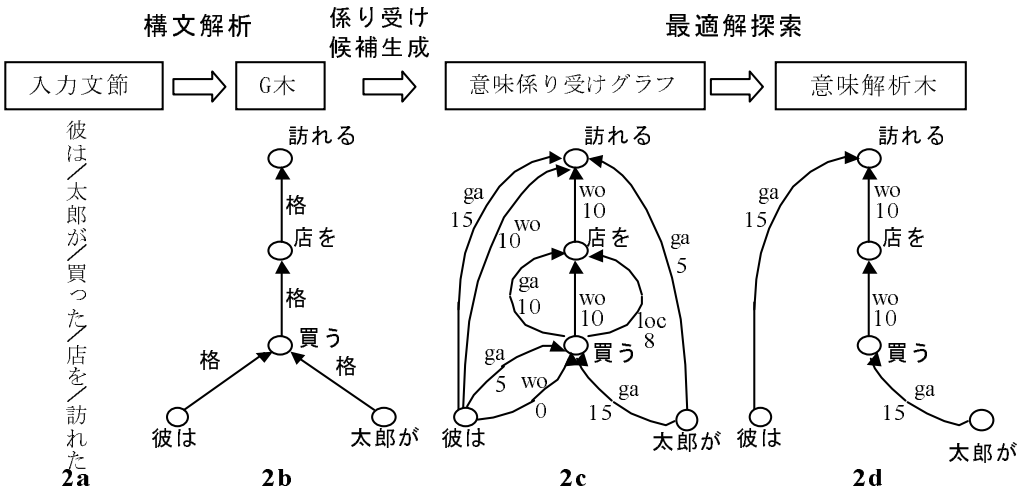


図 2 解析過程
Fig. 2 Analysis process.

な木の条件，すなわち，任意の 2 つのアーク間の非交差条件および多重格禁止条件をあわせて共起条件と呼ぶ．

2.2 意味係り受けグラフの構成

ここで対象とする処理は，構文/意味解析処理である．このため，入力は，文節の列である．意味係り受けグラフは，構文解析，係り受け候補生成の 2 つの過程より生成される．**図 2** に，意味係り受けグラフの生成を含む解析処理の過程を示す．

2.2.1 構 文 解 析

構文解析は，文節の列を入力し，曖昧性内蔵型の中間的な木構造（ここでは，G 木と呼ぶ）を 1 つ出力する．解析には，文脈自由文法ベースのパーサを用いている．この木構造の例を図 2-2b に示す．この木構造は，次の性質を持っている．

a. ノードは入力文中の文節または単語に対応する．
b. アークは文節間構文的係り受け関係を表現する．
c. 各ノードの可能な係り先ノード（文節または単語）は，その祖先ノードのいずれかである．

この構造は，c の性質により，基本的には CDG と同様に，可能な係り受け関係を内包させることができる．実際，最右文節を根とし，最左文節を葉とする直線状の G 木は，各要素はその右側の要素に係るという制約のみが適用された状態の木である．

G 木の特徴は，次のようである．

(1) 構文的に不可能な係り受けの可能性が排除されている．
(2) 木構造自体が構文的な情報を表現している．

構文解析のフェイズでは，文脈自由文法の知識による解の絞り込みが行われているといえる．

2.3 係り受け候補生成

2.2 節で得られた G 木は，明示的にはすべての解釈を表現していない．また，その表現する構造は，文の構文的な情報を表現している．係り受け候補生成過程では，G 木を入力として，それに対する意味係り受けグラフを生成する．各アークは，G 木の中の，係り受け可能な 2 つのノードを取り出し，それらの間の意味係り受け関係を検索することにより，設定される．たとえば，2.1 節であげた「私はあなたが好きです.」において,「私は」と「好きです」の間には，ha アーク（行為者格）と ga アーク（対象格）の 2 つのアークが設定される．重みは，2 つのノードの係り受けの確信度を計算する知識により決定される．この知識は，構文的な情報，意味的な情報，ヒューリスティックな情報などを含ませることが可能である．たとえば，次のような知識が含まれている．

・「は」が導く文節は，文末の述語に係りやすい．
・ 係り文節と述語格スロットの意味マーカとが一致したらその係り受けの可能性が高い．
・ 読点があると最も近い係り要素に係りにくい．

係り受けに関して制約的に適用できる知識は，この係り受け候補生成規則でアークの接続を禁止することにより実現できる．以上により，図 2-2c に示すような意味係り受けグラフが得られる．

3. 最適解探索

意味係り受けグラフから重みが最大である well-formed な木を見つけ出す問題は，任意の重み付き DAG:G=(N,A)（N はノードの集合，A はアークの集合）において，2 項制約（アークの共起制約）の集

合を満足する最大木を探索する問題であり，NP 問題である．ここでは，すでに述べたように，探索の方法として分枝限定法を採用した．

3.1 分枝限定法

分枝限定法は，NP 完全問題のような難しい問題を解く場合に用いられる計算原理である．基本的には，与えられた問題（P）をいくつかの小規模な問題に分解し，それらを解くことにより元の問題を解くものである．この計算途中で出現する部分問題について，a. 何らかの方法で部分問題の最適解が求まる，あるいは，b. 何らかの方法で部分問題が元の問題の最適解を与えないことが分かれば，その部分問題をさらに分解する必要はない．この操作を限定操作（bounding operation）と呼び，これにより探索における枝刈りが行われるわけである．限定操作には，代表的には，上界値による限定操作と優越関係による限定操作がある☆．これらを簡単に説明すると，次のようになる．

上界値による限定操作：

部分問題 P の条件を緩和して，より簡単な問題 P? を，次の条件を満足するように作成する．

(1) g(P?) ≧ f(P) g(P?) は P?の最適値，f(P) は P の最適値．
(2) g(P?)=f(P) ならば，P?の最適解は P の最適解．
(3) P?が許容解 (制約条件を満たす解) を持たなければ，P も許容解を持たない．
(4) すでに値 z の (許容) 解が他の部分問題から得られているとき，g(P?) ≦ z ならば，P から生成される部分問題は，必ず z 以下の値を持つ．

ここで，(2)〜(4) の場合には，部分問題 P を終端（terminate）してよい．

優越関係による限定操作：

ある問題 P を部分問題に分割したときに，その部分問題 Pi，Pj の間に，f(Pi) ≦ f(Pj) という関係が成立するときに，Pi は Pj に優越する（dominate）といい，このとき，Pj を終端してよい．

ここで，**図 3** に 1 つの最適解を求める分枝限定法の一般的アルゴリズムを引用する☆☆．

3.2 分枝限定法の意味係り受けグラフ探索への適用

図 3 は，分枝限定法のスケルトンを示したものであり，実問題への適用にはアルゴリズム中の各種操作を問題にあわせて実現する必要がある．以下では，本探索アルゴリズムにおいて，図 3 の l，g，s，D ならびに分枝操作をどのように構成するか具体的に説明し，意味係り受けグラフ G=(N,A)（N はノードの集合，A はアークの集合）から最大重みを持つ well-formed な木を探索するアルゴリズムを示す．

N： すでに生成されている部分問題の集合
A： 探索図において未終端な部分問題の集合
l ： l(P)が部分問題 P の下界値をあたえる関数
g ： g(P)が部分問題 P の上界値をあたえる関数
s ：s(A) は，A から 1 つの部分問題を選択する関数
G： g(P)が許容解をもたないあるいは g(P)=f(P) である部分問題の集合
f ： f(P)が P の最適解値を現す
D： Pi D Pj ならば，Pi と Pj の優越関係が成り立つという関係

S1(初期値設定) ：A:={P0},N:={P0},z:=-∞, O:=φ
S2(探索) ：A=φ ならS9へ，A≠φならPi=s(A) としS3へ進む
S3(暫定値改良) ：l(Pi)>z ならz:=l(Pi), O:={x} (x は f(x)≧l(x)を満たすPi の許容解)，S4 へ進む
S4(G テスト) ：Pi∈G ならばS8へ，さもなければS6へ進む
S5(上界値テスト) ：g(Pi) ≦z ならばS8へ，さもなければS6へ進む
S6(優越テスト) ：Pk D Pi を満たすPk (≠Pi) N が存在すればS8へ，さもなければS7へ進む
S7(分岐操作) ：Pi の子問題 Pi1,Pi2,..Pik を生成し，A := AU{Pi1, Pi2,.. Pik} -{Pi}, N:=NU{Pi1,Pi2,..Pik}とする．S2へ戻る．
S8(Pi の終端) ：A:=A-{Pi}としてS2へ戻る
S9(停止) ：計算終了．終了時にz は，最適値 f(P0)に等しく，z>-∞の時に O に記憶されている x は P0 の最適解である．z=-∞ならば，P0 は許容解を持たない．

図 3 分枝限定法アルゴリズム（P0 の最適解を 1 つ求める）
Fig. 3 Branch and bounding algorithm for the optimum solution for P0.

許容解・下界値を求めるアルゴリズム l(P)：

図 3 では l(P) は部分問題 P の下界値を与える関数としているが，本アルゴリズムでは，下界値は許容解（well-formed な木）の重みとし，l(P) で許容解と下界値を計算することとする．l(P) のアルゴリズムを**図 4** に示す．このアルゴリズムは，基本的にはアークの重さの高い順に，共起条件を満たすように深さ優先に意味係り受けグラフを探索し許容解を 1 つ求める．このアルゴリズムは，step5 において，共起条件を満足しないアークが見つかった時点でそれを解消する最も近いチョイスポイントまでバックトラックするというオプティマイゼーションを施した深さ優先探索であり，このアルゴリズムが解を持たない場合は，問題 P が解を持たない．また，このアルゴリズムの解は，問題 P の許容解（すなわち共起制約を満たす木）であることは明白であろう．step1 において，重さの大きい順にアークグループを並べ換える操作は，本質的には不必要ではあるが，できるだけ重さの大きいアークから選択していくことにより，できるだけ良い（重い）木を求めるためのものである．この暫定解が良ければ，限定操作が有効に働き，余分な部分問題の生成を減少させることになる．このアルゴリズムの計算時間は，

☆ 意味係り受けグラフにおける最適解は，重さが最大である木であるので，最大値を持つように解を最適化するという条件で説明する．
☆☆ なお，最適解をすべて求めるアルゴリズムもほぼ同じフローにより実現できる[6]．

EP：許容解を構成するアークを保存する領域
BP：バックトラックポイントを保存する領域
N(S)：アーク集合Sの要素数
wg(EP)：EP 中のアークの重みの総和

step1（アークのグループ化とソート）：
グラフ G=(N,A)(ここで|N|=n) のアークをその始点別にグループ化して，アークの集合 $S_1,S_2,...,S_{n-1}$ を作る．各 S_i の要素を重さの大きい順に並び換える．さらに，$S_1,S_2,...,S_{n-1}$ をその中のアークの最大重さの大きい順に並び換える．これを新たに，$S_1,S_2,...,S_{n-1}$ とする．

step2（初期化）：EP:=[],BP:=[],l:=1,j:=1,k:=1,l:=0,w:= -∞

step3（終了チェック1）：i=n であれば許容解が求まっており、許容解 EP ならびにその重み wg(EP)を値として終了する．i≠n なら step4 へ行く。

step4（終了チェック2）：N(Si)≧j であれば，step5 へ，さもなければ EP=[] として終了する．(解が存在しない)

step5（制約条件チェック）：もし j>N(Si)（S_i の全アークが共起条件違反）であれば，step6 へ行く．S_i の j 番目の要素 a(i,j)と，EP の要素 $e_1,e_2,...,e_{i-1}$ に関して共起条件を EP の終りから順にチェックし，EP の要素 e_k ($1 \leq k \leq i-1$) と a(i,j)が条件を満足しない場合，l:=max(l,k), j:=j+1 として step5 へ行く．a(i,j)が EP の全ての要素に対して共起条件を満足していれば，step7 へ進む．

step6（バックトラック）：EP から $e_{l+1},...,e_{i-1}$ を除き，j:=BP[l]+1, i:=l として step4 へ行く．

step7（次ノード）：EP の最後に a(i,j)を追加し BP[i]:=j, i:=i+1, j=1 とし，step3 へ戻る．

図 4 許容解・下界値を求めるアルゴリズム l(Pi)
Fig. 4 Algorithm for obtaining l(Pi).

ノード数 n に対して，step1 に対してソーティング $O(n\log(n))$，および step2〜7 に対して指数関数オーダとなるが，計算途中に共起制約によるバックトラックがない場合には，step2〜7 に対して $O(n^2)$ となる．このバックトラックの回数は，対象となるグラフの性質により，すなわちどのような重み付けがされているかにより変化するものと考えられる．実際の文において，どの程度のパフォーマンスとなるかについては，後の実験により検証する．

上界値を求めるアルゴリズム g(P)：

このアルゴリズムは，意味係り受けグラフの最大木を求めるものであり，基本的にアークの重さの高い順に，深さ優先に探索し最大木を求める．このアルゴリズムは，許容解を求めるアルゴリズムの step5 における共起条件チェックをなくせばそのまま実現できる．ここで対象となっているグラフは DAG であるため，実際，各ノードから出ているアークのうち最大重みのものを集めれば，それが最大木となっていることは明白である．このため，step2〜7 に対して $O(n)$ で計算できる．また，最大木の重さ g(Pi) は，明らかに最適木の重さ f(Pi) に対して，g(Pi) ≧ f(Pi) である．また，g(Pi) が解を持たないときは，f(Pi) も解を持たない．このため，計算途中で暫定値 z が，z ≧ g(Pi) を満たした場合は，その部分問題 Pi は，終端することができる．

優越テスト アルゴリズム D：

本アルゴリズムでは，優越テストは用いない．

分枝操作：

分枝操作は，部分問題 Pi（グラフ G(N,Ei)）の最大木を構成するアークのうちで，共起条件を満足しないアーク e1，e2 を求め，新たなグラフ，Gi1=(N,Ei-e1)，Gi2=(N,Ei-e2) を作成し，それに対して部分問題 Pi1，Pi2 を作成するという処理とする．Pi の解は，共起条件により e1 と e2 の 2 つのアークを含まないため，この操作により得られた Pi1，Pi2 のいずれかが Pi の最適解を持つことになる．なお，最大木よりアーク e1，e2 を求めるには，許容解を求めるアルゴリズムの step5 において，最初に見付かった共起条件を満足しないアークのペアを記憶しておけばよい．これにより，最も重さの大きい共起条件を満足しないアークのペア e1，e2 を得ることができる．

探索 s(A)：

活性部分問題の集合 A より，次に展開すべき問題を選択する．探索 s(A) については，深さ優先探索，幅優先探索，最良上界探索などがあるが，ここでは，最良上界探索を採用している．すなわち，活性部分問題集合のうち最大の上界値を持つ部分問題を s(A) は選択する．この探索法は，優越テストを用いない場合において，最終的に解が得られるまでに分解される部分問題数が最小であることが知られている．以上の設定による，意味係り受けグラフの最適解探索アルゴリズムは，**図 5** のようになる．

3.3 最適解探索例

本節では，例をあげて，3.2 節で述べたアルゴリズムの動作例を示す．例文は,「私も彼が机を買った店に売った.」である．この文に対応する意味係り受けグラフは，**図 6** であるとする．図において a〜l のアルファベットはアークのアイデンティファイアであり，ga, wo などは意味係り受けラベルであり，数値はアークの重みである．また，ここでは，多重格条件に関連する格フレームとして次を想定している．

売る ga[行為者] wo[対象] ni[相手]
買う ga[行為者] wo[対象] de[場所]

図において,「私」からは，f，g，h，i の 4 本のアークが出ているが，これは，副助詞「も」は,「売る」「買う」のそれぞれに対して，ga（行為者），wo（対象）の 2 つの格要素として係りうる可能性があることを示

S1(初期値設定) : A:={P0},z=-∞, O:={}
S2(探索) : A={} なら S8 へ，A≠{}なら Pi=s(A) としS3へ進む
S3(暫定値改良) : l(Pi)>z ならz:=l(Pi), O:={x}(xはPiの許容解)，S4へ進む．
S4(Gテスト) : g(Pi)=-∞または，g(Pi)=l(Pi) ならばS7へ，さもなければS5へ進む
S5(上界値テスト) : g(Pi)≦z ならばS7へ，さもなければS6へ進む
S6(分岐操作) : Pi の子問題 Pi1,Pi2 を生成し，A:=AU{Pi1,Pi2}-{Pi}とする．S2へ戻る．
S7(Piの終端) : A:=A-{Pi} としてS2へ戻る
S8(停止) : 計算終了．計算終了時に z は，最適値f(P0) に等しく，z> -∞の時にOに記憶されているxはP0の最適解である．z=-∞ならば， P0は許容解を持たない．

図 5 意味係り受けグラフの最適解探索アルゴリズム（P0の最適解を1つ求める）
Fig. 5 Algorithm for obtaining optimum solution P0 for semantic dependency graph.

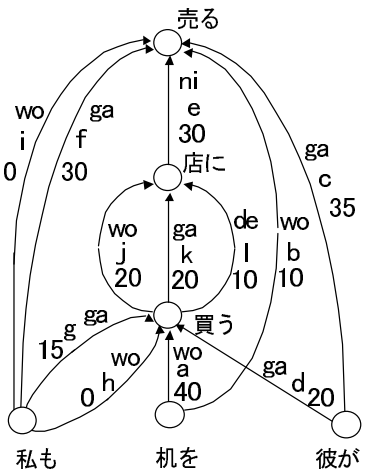


図 6 例文に対する意味係り受けグラフ
Fig. 6 Semantic dependency graph for the example sentence.

している．同様に,「買った」が導く埋め込み文においては,「店が買った」,「店を買った」,「店で買った」の3つの解釈が存在している．各アークの重みは，2章で述べたプロセスにより，主として単語間の係り受け評価により設定されている．アルゴリズムの動作の説明の前に，計算途中で現れる部分問題の構成について説明する．部分問題は次の要素を持つ必要がある．

a. 部分問題 Pi のグラフ Gi
b. Pi に対する許容解の値 l(Pi)
c. Pi に対する上界値 g(Pi)

部分問題のグラフ Gi を各問題ごとに持つのは非効率であるので，ここでは，排除アークリストと呼ぶ，グラフから排除されたアークのリスト（rem[] で表現する）を用いて部分問題を表現する．このため，3.2節で示したアルゴリズムとは少し異なったアルゴリズムを用いることとする．すなわち，3.2節の許容解を求めるアルゴリズム l(Pi) の step1 における並べ換えは，探索の初期問題に対して1回だけ行う．図6のグラフに対しては，その結果は次のようになる．

S1：[机] a[wo, 買う，40]，b[wo, 売る，10]
S2：[彼] c[ga, 売る，35]，d[ga, 買う，20]
S3：[店] e[ni, 売る，30]
S4：[私] f[ga, 売る，30]，g[ga, 買う，15]，h[wo, 買う，0]，i[wo, 売る，0]
S5：[買う] j[wo, 店，20]，k[ga, 店，20]，l[de, 店，10]

分枝された部分問題は，上記のデータから rem[...] で表現されたアークを除いたグラフを対象として持つ．

図 7 に，上記例に対する計算過程を示す探索図（search diagram）を示す．図において，ノードは部分問題を示し，それぞれ許容解値 l，上界値 g，グラフ rem，および共起条件を満たさないアークのペア c を持つ．z は，その部分問題を計算する時点での暫定解の値を示す．また，Pi の i は問題が分枝された順序を示している．図の P0 は初期問題であり，rem[] である．この問題の許容解は，図に示すように (a,c,e,i,k) であり，その許容解値 l は 125 である．また，上界値 g は，最大木 (a,c,e,f,j) の重み 155 である．ここでは，3.2節のアルゴリズムに従い，最大木中に含まれる，共起条件を満足しないアークペア c=(c,f)（多重格禁止条件）が求められている．暫定解値 z は，その初期値 $-\infty$ であるため，許容解値 l=125 が z に設定される．c=(c,f) により，P0 は，2つの部分問題 P1，P2 に分枝される．P1 と P2 の l，g，c がそれぞれ計算され，図に示すような結果となる．ここで，次に処理されるべき問題を検索する．ここでは，P1，P2 が対象であるが，ともに g=140 であるので，いずれが選択されてもよい．P1 が選ばれたとする．以下同様に計算が進行する．P3 では，rem[c,a] となり，このグラフに対しては，許容解が存在しなくなり終端する．また，P7 では，上界値 g が 125 であり，暫定解値 z が 130 であるため，限定操作により終端される．以上の処理により，**図 8** に示す意味木が最適解として得られる．

3.4 アルゴリズムの計算量

分枝限定法の計算の効率を評価するうえで重要なパラメータは，生成される部分問題の個数 T である．一般に，T は，分枝図の高さ h に対して指数オーダで増加するため，上記のアルゴリズムの生成する部分問題

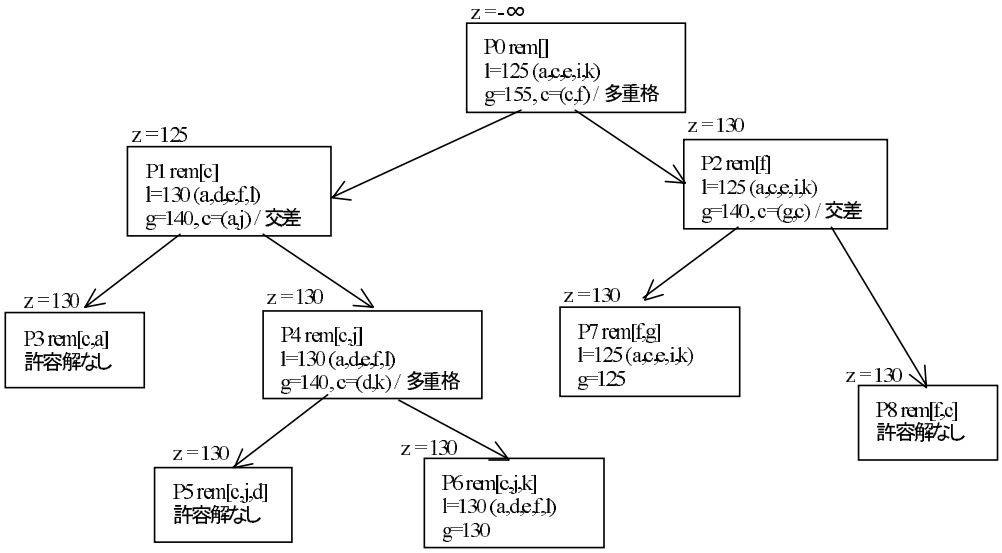


図 7 例文に対する計算過程を示す探索図
Fig. 7 Search diagram for the example sentence.

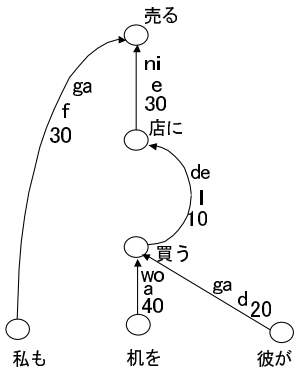


図 8 例文に対する最適解
Fig. 8 Optimum spanning tree for the example sentence.

数は，意味係り受けグラフのノードの数，すなわち入力文の長さに対して，基本的に指数オーダの部分問題を生成する．T の下界は，部分問題 Pi の上界値 g(Pi) と最適解値 f(Pi) に関して次の関係が成立することが知られている．

$$T \geqq | Pi | \ g(Pi) > f(P0) \ |$$

このため，g(Pi) の精度が良ければ，それだけ最低限生成される部分問題数が低くなる．上記アルゴリズムでは，g として最大木を使用しているが，これにより実際の文（通常，最大で 40 程度のノード数を持つ）の解析において，どの程度の部分問題数に抑えられるかが，使用上の 1 つのポイントとなる．また，計算に必要な記憶スペースには，主として，意味係り受けグラフと部分問題の 2 つがある．意味係り受けグラフは，2 つのノード間に k 個の意味係り受け関係が存在するとすれば，最大 $kn(n-1)/2$ 本のアークを有するので，$O(n^2)$ のスペースを要する．また，部分問題のための記憶スペースは，(部分問題 1 つの記憶スペース) × (計算途中の分枝図における葉の最大数) となる．1 つの部分問題には，排除アークリスト rem があるため，$O(n^2)$ の記憶スペースが必要である．また，分枝図の葉の最大数は，探索図の深さに対して指数のオーダとなる．実際の計算では，限定操作により枝刈りが行われるため，やはり上界値関数 g と許容解関数 l により葉の最大数は変化する．

以上のように，処理時間，処理スペースともに，g, l により変化する．また，意味係りグラフの構造や重み付けの状態により変化する．次章では，実際の文に上記のアルゴリズムを適用し，そのパフォーマンスを検討する．

4. 解析実験

3.2 節で述べたアルゴリズムの性質およびパフォーマンスを見るため，論文・マニュアルから 100 文を抜き出し解析実験を行った．使用した計算機はエンジニアリングワークステーション AS4260 である．また，各文に対する意味係り受けグラフの生成は，既存のソフトウェアを使用した．**図 9** に測定結果を示す．図の縦方向は，入力文に対する意味係り受けグラフのノード数による分類であり，横方向は，最適解を探索する際に作られた部分問題の個数である．この表の要素には文の出現度数と，それらの文の最適解探索に要した平均計算時間を示してある．また，図中に (x/y) の形

意味係り受けグラフのノード数	文の頻度		生成された部分問題数				
			1〜5	6〜10	11〜15	16〜25	26〜
	1〜2	9	9 3ms				
	6〜10	12	12 25ms				
	11〜15	39	37 80ms	1(4/7) 293ms	1(4/11) 436ms		
	16〜20	21	20 173ms		1(1/11) 848ms		
	21〜25	10	7 312ms	1(1/9) 1088ms	1(1/11) 1231ms		9a
	26〜30	7	6 365ms				1(4/53) 11750ms
	31〜35	1			1(12/13) 3775ms	9b	
	35〜	1	1(1/5) 1258ms				

図 9 テスト文集合に対する実験結果
Fig. 9 Experimental result for test sentences.

式で記述してある部分は，それぞれ次のデータを示している．

x： 最終的に最適解となった許容解を生成した部分問題の番号

y： 最終的に全体として展開された部分問題の数

また，実験対象文に対する平均計算時間は，305.8 ms であった．図 9 より，この実験において以下のことがいえる．

・ 分枝された部分問題の数は比較的に少ない

最終的に展開される部分問題数が 5 以下である文は 100 文中 93 文であり，全体としてかなり低い数値になっている．この理由の 1 つとして，許容解を求めるアルゴリズム l(Pi) において，グリーディに求めた解が，比較的に良い結果を出し，それによる部分問題の終端が有効に働いたことがあげられる．

・ ほとんどの場合で最適解は計算の初期に得られた．

問題解決中に部分問題の分枝を行うが，その最大数は 53 であった．この文は図中の 9a で示した文である．9a では，展開された部分問題は多かったが，最終的に最適解となった解は，第 4 番目の部分問題の許容解であった．すなわち，第 5 番目以降は，結果的に第 4 番目の部分問題の許容解が最適であることの確認のために展開されたことになる．実際，この実験において，100 文中 99 文は，5 つの部分問題を計算した時点で求まっている．ただ 1 つ例外であったのは，図中の 9b で示した文である．このような結果となった理由としては，意味係り受けグラフの重み付けの結果が非交差条件や多重格禁止条件にあまり矛盾が生じない状態であったということがあげられる．たとえば，「彼は本は買う.」という文の意味係り受けグラフにおいて，意味的な理由により「彼」は「買う」の「行為者」にあたる解釈（アークの重み）が最も強く，また，「本」は「買う」の「対象」にあたる解釈が強いという重み付けがされていれば，そのグラフの最大木が最適解と一致するわけである．

今回の実験では，数 ms〜1,300 ms 程度で，ほとんどの場合，解を求めることができたが，中には図中の 9a のように 12 秒近くも時間を要する例もあった．9a では，最適解は，比較的初期（4 番目）に求まっていたが，限定操作がうまく適用できず，結局多くの部分問題を生成してしまっている．このような場合には，解の上限（現状では，最大木の重さ）をもう少し精度の高いものに変えることが有効である．また，乱暴な手段ではあるが，展開する部分問題の数を限定してしまう（もちろん近似解を得ることになる）という方法も有効ではある．しかしこの方法では，図中の 9b のような場合に対して問題となる．9a とは対称的に 9b では，部分問題の展開は，最適解を求めるために必要な数だけ行われたのみであった．この例に対して計算の収束を早めるためには，最適解の探索を早めに決定できるようにすることが必要である．

5. アルゴリズムの改良

4 章で述べた実験では，大部分の例文について数百 ms 程度で最適解を得ることができたが，なかには，10 秒以上も計算時間を要した例もあった．次に，アルゴリズムの改良について検討する．

5.1 上界値関数の改良

すでに述べたように，生成される部分問題の数を減らすことは，全体の処理効率を上げるうえで重要である．4 章の結果より，最適解は多くの問題で比較的早い時点で求まっており，その解が最適解であることを確認するための時間がかなり多いことが分かった．これを改善するために，上界値関数 g を改善することを考える．3.2 節のアルゴリズムでは，最大木を上界値としたが，新たな g として，最大木中で共起条件を満たしていないアークのペアに対しては，安全なペナルティをその全体の重みより差し引くという改良が考えられる．たとえば，アーク Si[k] と Sj[l] が共起条件を満足しない場合に，

$$\min(w(Si[k])-w(Si[k+1]),w(Sj[l])-w(Sj[l+1]))$$

（w はアークの重み）

を最大木の重みから差し引くといった改良である．ただ，この新たな g は，$O(n^2)$ の計算を必要とし，最大木の計算より時間がかかる．このため，全体としての

トレートオフを考える必要がある．

5.2 部分計算の共有化

部分問題をその子供の問題に展開する際に，共起条件を満足しない2つの問題に展開する．親の問題と子供の問題は，かなり似た問題である．子供の問題の計算の際に，親の問題での計算の部分を子供の問題の計算に利用することが考えられる．ここで提案したアルゴリズムは，許容解の計算にバックトラックベースのアルゴリズムを使用しており，効率が良くない場合がある．このため，許容解の計算を親子で共有することが有効であると思われる．子問題では，親問題から最大木中で共起条件を満足しないアークの1つが除かれている．このアークが出ているノードが，図4の許容解計算アルゴリズムl(P)におけるアーク集合Siに対応するノードであるとすると，S1からSi-1までのl(P)の計算状況は親問題と同じであるため子問題の計算で再利用することが可能である．

実際に上記の2つの改良を施したアルゴリズムを実装した結果，実験対象文に対する平均計算時間は，305.8 msから162.1 msへと改善した．また，図9の9 aの文では，生成された部分問題数は，53 → 9となり，実行時間は，11,750 ms → 1,326 msと大幅に改善された．また，9bの問題は，生成された部分問題数が，13 → 11となり，解析時間は，3,775 ms → 2,826 msへと改善された．

6. 関連研究

本稿では，句構造文法（Phrase Structure Grammar）を利用しながら☆1，日本語解析手法として古くから研究されてきた係り受け解析[8]を基本としたフレームワークを採用している．係り受け解析は，2文節間の係り受け関係（修飾関係）を基本構造とし，依存構造木を求めるものであり，語の依存関係に基づく依存文法（Dependency Grammar）の一種であると考えられる☆2．

CDG[9),10)]は，Σ（語彙カテゴリ），R（意味役割），L（ラベル集合），C（制約集合）で定義され，入力文のすべての単語間にすべての依存関係を仮定してしまい，文解析は，その中から制約C☆3を満足する解釈に絞り込むプロセスとして実現するEliminative Parsingという考え方をとっている．通常の文解析は，Generative Parsingと呼ばれ，可能な解析構造を生成するという形で文の解析が進行する．本稿では，文の解析過程では，できる限り可能な解釈は先に生成・保持しつつ，適切な所で解釈を絞っていく（後戻りによる別解釈の生成は行わない）という考え方をとっており[2)]，基本的に文献5)の考え方と同じである．しかしながら，不用意に大きな仮説生成は，効率面で問題となる可能性が高いため，仮説生成は，入力されたデータ（単語）に基づく範囲で行い，それに対して検証を行うという方式をとっている☆4．実際，CDGの研究では，Fast filtering algorithmやLabel tableなどにより，制約生成の限定や処理の効率化が導入されている[13)]．意味係り受け解析手法とCDGの本質的な違いは，前者が種々の言語規則や知識は，選好的に適用するという考えに立脚しているのに対し，後者は，制約的に適用するという立場である点である．CDGの方式では，意味的な解釈の可能性を制約として適用すると解釈不能になったり☆5，可能な解釈が複数あった場合に1つに絞り込むことができなかったりするなどの問題が生じる．この点については，Graded constraintという一種の優先度の枠組みが近年になり提案されている[14)]．

Bealeは，Hunter-Gatherer法（HG）を言語の意味解析に適用することを提案した[15)]．HG法は，非最適な解や不可能解を探索空間から排除するHuntingと，探索空間から効率的に解を抽出するGatheringを行う手法である．HGは，知識ベースの機械翻訳システムであるPangloss[16)]において，単語語義（word sense meaning）の同定タスクに使用されている．入力文は，Panglyzer[17)]と呼ばれる構文解析モジュールにより，Text Meaning Representation（TMR）という主に単語の依存関係を表現する言語非依存構造の表現に解析される．この表現から単語語義に関する制約関係を表現する制約依存グラフ（constraint dependency graph）が得られる．これにグラフ分割を行った分割された制約依存グラフ（partitioned constraint dependency graph）がHGの入力となっている．HGでは，意味的な関係を適切に扱うため，制約関係に “tendency” と呼ばれる優先スコアを導入し，単語語義に対する最適解を求めるという処理を行っている．制約を満たす解を取り出すソリューション合成において，グラフの分割と分枝限定法を導入して効率

☆1 G木の生成は，文脈自由文法ベースのパーサを利用している．

☆2 日本語は，受けの文節は，必ず係り文節の後に位置するという制約が入っている点が特徴的．

☆3 単項制約（unary constraint），2項制約（binary constraint）の2種類がある．

☆4 ここでは触れないが，文解析における曖昧性を増進的に解消するモデルも11)，12)などで提唱されている．

☆5 たとえば，「豚は飛ばない」という意味解釈制約を入れてしまうと，「豚が飛んだ」という文に対して解釈を出すことができなくなる．

化を図っている．HGは，本稿と対象とするタスクやフレームワークは異なっているが，選好意味論と分枝限定法の導入の効果を示しているといえる．

日本語の係り受け解析において，最適解探索を行うというフレームワークでは，種々の研究が行われているが．手法的には，係り受けマトリックスにより係り受けの可能性を表現し，係り受けの可能性をスコア化しておき，動的計画法（Dynamic Programming）の手法を用いて最もスコアの良い解釈を検出するというアプローチがとられている．たとえば，黒橋らは，長い日本語文の解析を高精度に行うために，文の大局的構造である並列構造の解析にこの手法を適用している[18]．この並列構造解析手法では，並列構造の構文的・パターン的類似度や語の意味的類似度を統合して数値化し，係り受けの非交差条件を満たす最適解を検出しており，異種の知識を統合評価している例となっている．大局的な並列構造を決定した後，局所的な係り受け解析を決定的な処理や種々の言語的ヒューリスティックスなどを用いて実行する[19]．一方，尾関は，音声言語解析などへの係り受け解析の適用を想定し，与えられた文節間関係のpreference scoreと文節単体のreliability scoreを総合解釈して，非交差条件を満たす係り受け解釈をDPに基づき高速に計算するアルゴリズムを提案した[20),21]．実際，文節間距離情報や韻律情報をスコアとして利用され有効性が報告されている[22]．これに対し，本稿のフレームワークは，単語間の係り受け（依存）関係を意味係り受け関係に拡張している点，用言の格フレームを導入し多重格制約を入れている点で，従来の係り受け解析とは異なっている．多重格制約は，3つ以上の構成要素がからむため，DPの枠組みが適用できない．このため，本稿では，係り受け解析のベースとして係り受けマトリックスではなく意味係り受けグラフを導入し，制約充足（共起条件）し，かつ最大スコアを有するmaximum spanning tree探索として係り受けの最適解探索を定式化し，分枝限定法に基づくアルゴリズムを提案した．

意味係り受けグラフは，文を構成する単語列に対応する可能な依存構造を表現する枠組みであるが，これと類似の構造にSyntactic Graphが提案されている[23]．Syntactic Graphは，構文解析部と意味解析部を分離した解析方式を実現するために☆，入力文の可能な解釈に対応するすべての依存関係構造をコンパクトに保存する枠組みである．Syntactic Graphは，句構造解析文法による全解探索の構文解析（Chart Parserをボトムアップに適用）を行いながら構成され，完全にフォーマルな証明は与えられていないが，Packed Shared Parse-Forest[3)]に存在する可能な構文解釈と1対1対応がとれる依存関係構造のみを保持する．Syntactic Graphの表現では，語のDependency（依存関係，係り受け関係）がアークで示されるが，複数のDependencyを持つノードは，依存関係での曖昧性を持ち，1つの解釈に相当する依存関係構造については，a) 依存先は1つのみ，b) ルート以外は依存先を持つ，c) 非交差条件を満たすなど，意味係り受けグラフと類似している．主な違いは，意味係り受けグラフは，文を構成する1単語列に対しての可能性を表現しているのに対し，Syntactic Graphは，文献20)と同様，それぞれの単語の品詞の多義も表現しており，文の解釈全体を表現できる．この性質は，すべての名詞が動詞たりうるといわれるような英語など品詞の多義性が非常に多い言語の解析では必須であるが，日本語のように形態変化などで品詞がほとんど判定できる言語☆☆ではその実質的重要度は高くない☆☆☆．一方，Syntactic Graphでは，依存関係ラベルには，snp（文–名詞句），svp（文–動詞句），npp（名詞句–前置詞）といった句構文関係由来のものが1つだけ付けられており，意味係り受けグラフで採用している意味的な複数の依存関係は表現されていない．また，多重格の制約も当然のことながらスコープに含まれていない．

Syntactic Graphや LFG（Lexical Functional Grammar）[24)]などは，構文解析による句構造解析（C-structure）と，その上位レベルの依存構造や機能構造（F-structure）を有しているが，日本語係り受け解析では，直接係り受け関係で解析が行える．これは，日本語では，ある単語の係り先は必ずその後ろにあるという強い制約によるものであると考えられ，より一般的なフレームワークとしては句構造解析とその上位構造の解析という2段階をとる方が自然であると考えられる．本稿でも，句構造解析と係り受け解析の併用を行っており，英語の解析にも適用されているが，依存構造解析という意味では汎用性を持っており，特に大きな問題はない．

7. おわりに

本稿では，自然言語の各種レベルの知識の性質（優

☆ Seoは，構文解析の1結果を意味解析し，意味的に成立しない場合次の結果を試みるという方式を想定しており，本稿のアプローチとは異なっている．

☆☆ 語境界に関する曖昧性は重要であり，実際のシステムでは，別の枠組みで対応している．

☆☆☆ この制約のため，本稿では，日本語意味係り受け解析と限定を入れた表現をとっている．

先性を現す場合が多い．異なったレベルの知識の干渉がある）に合致した処理方式へのアプローチとして，構文/意味レベルの曖昧性を処理する枠組みとその実験について報告した．この種の処理では，最終的には組合せ的な計算を行う必要があり，その手法として分枝限定法を用いている．今回対象としたのは，構文/意味レベルの知識であるが，同様の議論はその他のレベルにもあてはまる．たとえば，形態素解析における解釈の曖昧性が，意味や文脈レベルの知識と関係を持つような場合である．特に，英語の解析においては，多品詞語が非常に多く存在するため，品詞の解釈を他の知識と整合性良く行う必要がある．今後は，多品詞の存在を許す方式への拡張を行い，英語など他品詞が重要課題となるような言語を扱える枠組みに展開していく予定である．

参 考 文 献

1) Wilks, Y.A.: An Intelligent Analyzer and Under-stander of English, *Comm. ACM*, Vol.18, pp.264–274 (1975).
2) 平川秀樹，天野真家：構文/意味優先規則による日本語解析，人工知能学会第 3 回全国大会論文集 (1989).
3) Tomita, M.: An efficient augmented context-free parsing algorithm, *Computational Linguistics*, Vol.13 (1987).
4) Barton, G.E. and Berwick, R.C.: Parsing with Assertion Sets and Information Monotonicity, *Proc. International Joint Conference of Artificial Intelligence-85* (1985).
5) 丸山　宏：制約依存文法に基づいた日本語解析支援システム，情報処理学会自然言語処理研究会資料 69-6 (1988).
6) 茨木俊秀：組み合せ最適化，産業図書出版 (1983).
7) 平川秀樹，天野真家：日本語解析における最適解探索，情報処理学会自然言語処理研究会 74-2 (1989).
8) 吉田　将：二文節間の係り受けを基礎とした日本語文の構文解析，電子情報通信学会論文誌，55-D（4），pp.238–244 (1972).
9) Maruyama, H.: Constraint Dependency Grammar and its weak generative capacity, *Computer Software* (1990).
10) Karlsson, F.: Constraint grammar as a framework for parsing running text, *13th International Conference on Computational Linguistics*, Vol.3, pp.168–173 (1990).
11) 奥村　学，田中穂積：Discrimination network 上での増進的曖昧性解消について，人工知能学会誌，Vol.7, No.4 (1992).
12) 杉村領一：論理型文法における制約解析，情報処理学会自然言語処理研究会 67-2 (1988).
13) White, M.C.: Rapid Grammar Development and Parsing: Constraint Dependency Grammars with Abstract Role Values, Ph.D. Thesis, Purdue University (2000).
14) Heineck, J., Kunze, J., Menzel W. and Schroder I.: Eliminative parsing with graded constraints, *Proc. Joint Conference COLING-ACL*, pp.526–530, (1998).
15) Beale, S.: Hunter-Gatherer: Applying Constraint Satisfaction, Branch-and-Bound and Solution Systhesis to Computational Semantics, Ph.D. Thesis, Carnegie Mellon University (1997).
16) Frederking, R., Nirenburg, S., Farwell, D., Helmreich, S., Hovy, E., Knight, K., Beale, C., Domashnev, C., Attardo, D., Granners, D. and Brown, R.: Integration Translations from Multiple Sources within the Pangloss Mark III Machine Translation System, *Proc. 1st Conference of the Association for Machine Translation in the America*, Columbia, Maryland (1994).
17) Farwell, D., Helmreich, W., Casper, J.M., Hargrave, J., Molina, H. and Weng, F.: PANGLYZER: Spanish Language Analysis System, *Proc. 1st Conference of the Association for Machine Translation in the Americas*, Columbia, Maryland (1994).
18) 黒橋禎夫，長尾　眞：並列構造の検出に基づく長い日本語文の構文解析，自然言語処理，Vol.1, No.1, pp.35–57 (1994).
19) Kurohashi, S. and Nagao, M.: A Syntactic Analysis Method of Long Japanese Sentences based on the Detection of Conjunctive Structures, *Journal of Computational Linguistics*, Vol.20, No.4, pp.507–534 (1994).
20) 尾関和彦：係り受けの整合度に基づき最適文節列を選択する多段決定アルゴリズム，電子情報通信学会論文誌，J70-D（3），pp.601–609 (1987).
21) Ozeki, K.: Dependency Structure Analysis as Combinatorial Optimization, *Information Sciences*, 78(1-2), pp.77–99 (1994).
22) 尾関和彦，張　玉潔：最小コスト分割問題としての係り受け解析，言語処理学会第 5 回年次大会ワークショップ論文集，pp.9–14 (1999).
23) Seo, J. and Simmons, R.F.: A Syntactic Graphs: A Representation for the Union of All Ambiguous Parse Trees, *Computaional Linguistics*, Vol.15 (1989).
24) Kaplan, R.M. and Bresnan, J.: Lexical Functional Grammar: A Formal System for Grammatical Representation, *The Mental Representation of Grammartical Relations*, Bresnan,

J. (Ed.), pp.173-282, MIT Press, Cambridge, Mass. (1982).

(平成 12 年 12 月 21 日受付)
(平成 13 年 12 月 18 日採録)

平川 秀樹（正会員）

昭和 31 年生．昭和 55 年京都大学大学院工学研究科電気工学専攻修士課程修了．同年（株）東京芝浦電機入社．機械翻訳システムの研究開発に従事．昭和 57〜59 年新世代コンピュータ開発機構（ICOT）研究員，平成 6〜7 年 MIT Media Lab. 派遣研究員，平成 8 年より（株）東芝研究開発センターヒューマンインタフェースラボラトリ所属，自然言語処理，知識処理，ヒューマンインタフェースに関する研究に従事．電子情報通信学会，人工知能学会，言語処理学会，ACL 各会員．