自 然 言 語 処 理 78－1
（1990．7．19）

制約と選好による構造的多義性の解消

長尾 確

日本アイ・ビー・エム株式会社 東京基礎研究所

文法的知識と意味的知識を用いて，自然言語文の構造を決定する問題は，自然言語処理の基本的問題の一つであり，いまだ一般的な手法が開発されていないボトルネックでもある。これに関して，文が与えられたときにその文の構造が満たすべき条件を宣言的に記述することによって文法を構成するという提案がなされている。しかし，一般に，文法的知識だけでは構造を一意に決定できない。すなわち構造的多義性が発生する。そこで，意味的知識を用いる必要があるが，意味的知識は文法ほど体系化されていないために，構造を限定する条件(すなわち，制約)として用いることは困難である。しかし，ある種の，構造を別の構造と比較する基準(すなわち，選好)として用いることができる。本論文では，文法的知識を制約として表現し，また意味的知識を選好として用いることにより，構造的多義性を含む文を解析する時点で，動的にそれらを組み合わせて多義性を解消する手法について述べる。

Structural Disambiguation with Constraints and Preferences

Katashi Nagao

IBM Research, Tokyo Research Laboratory
5-19 Sanbancho, Chiyoda-ku, Tokyo 102, Japan
E-mail: nagao@jpntscvm.bitnet

One of the fundamental problems of natural language processing is to construct a unique structural interpretation of a sentence by using grammatical and semantic knowledge. This is still the bottleneck, and a general solution has not been developed. One recent idea is to represent grammatical knowledge as constraints over the finite domain. According to this idea, grammar rules are declaratively described as conditions that are satisfied by sentential structures. In this sense, syntactic analysis is considered as a constraint satisfaction problem. In general, sentential structures cannot be decided merely by applying grammatical knowledge, since this cannot resolve structural ambiguity. There is a need for semantic knowledge, but this kind of knowledge can only be used to provide measures for comparing structures (i.e. preferences), not conditions for defining structures (i.e. constraints). This paper presents a mechanism that can be used in structural disambiguation to integrate knowledge of constraints and preferences.

1. はじめに

文の構造を一意にする問題は，自然言語を処理する上で，最も基本的であり，依然として一般的な解決策が存在しない問題である。文法的知識を用いて，文を解析しても，一般に，内在する多義性のために構造を一意に決定することはできない。また，意味的知識は十分に体系化するのが困難であり，文法的知識ほどには構造を限定するのに役に立たない。そこで，意味的知識を，文法的知識を補う形で用いることが多い。そこで問題になってくるのは，2つの知識をどのように組み合わせるかである。

これらの知識を統合して文解析をするシステムの多くは，意味的知識が文法的知識(この場合は，統語解析規則)の中に埋め込まれており，知識が静的に統合されている。タイプの異なる知識をこのような形で持つことは，表現的に効率が悪い。我々の枠組みでは，2つのタイプの知識を異なる形で保存し，文解析時に動的に統合して，構造的多義性の解消に用いる。また，これらの知識のインタラクションは重要である。つまり，文法的知識が常に意味的知識を用いて構造選択の決定を下すか，あるいは意味的知識が常に文法的知識の規則の適用を決定するという一方向の情報の流れでは，情報の効率的なフィードバックを扱うことができない。我々の枠組みでは，文法的知識は意味的知識の適用あるいは不適用を決定し，意味的知識は文法的知識の適用順序を制御することができ，2つの知識は協調的に作用することができる。

最近になって，文法的知識を，文構造が満たすべき条件の集合，すなわち制約として記述する枠組み[4]が提案されている。その枠組みによると，構文解析は可能な構造をすべて含んでいる集合(厳密には，可能な構造をパックした一つの構造)に動的に制約(としての文法)を当てはめてみて，制約に合わないものを取り除いていく，というプロセスとして実現される。しかし，文法的知識だけでは構造を一意にすることができないために，やはり制約を満たす複数の構造が残ってしまう。そこで意味的知識を用いて，さらに構造の絞り込みをしたい。しかし，意味的知識は一般にあまり厳密ではなく，文法ほどには強力ではないため，制約として用いるわけにはいかない。実際には，意味的知識は複数の構造に何らかの優先関係を設定する一つの基準，すなわち選好として用いる場合が多い。

我々は，領域依存の知識を用いて，文内の係り受けの多義性の候補に優先度をつける手法[6,7]を提案しており，それを一種の意味的知識として考えることができる。そこで，これらの枠組みをベースにして，制約としての文法的知識と選好としての意味的知識を統合して，文構造を一意にする，すなわち構造的多義性を解消する手法を提案する。

本論文の構成は，次のようになっている。2章では，文法的知識を制約として表わし，制約伝播によって構造的多義性を解消する枠組みについて述べ，3章では，選好，すなわち優先性を決定するために領域依存の意味的知識を用いる手法について述べる。4章では，制約と選好としての2つのタイプの知識を統合して構造的多義性を解消するアルゴリズムと，それを用いた例題を示す。5章で，関連した研究を紹介し，それらとの比較を行う。6章で結論を述べる。

2. 制約としての文法的知識

構文解析を有限領域上の制約充足問題として捉える手法[4]が開発されている。その枠組みでは，文法は制約として表現され，構文解析は，文の構成要素(単語あるいは文節)が持つ可能な役割の候補を，制約を満たさないものを除外する(それをフィルタリングと呼ぶ)という形で実行される。しかし，多くの場合，制約を満たすものが複数存在し，それ以降はバックトラック・サーチを繰り返さなければならない。

2.1. 制約伝播による構造的多義性の解消

制約としての文法的知識に基づく構造的多義性の解消は，例えば，次のように行われる。一文の中に2つ以上の多義性が含まれている場合，多義性と多義性の間の関係を制約マトリックスにして表わし，非交差の制約(係り受け関係が交差してはいけない)などを反映させておく。

例えば，図1のような多義な依存構造から，表1のような制約マトリックスが作られる。

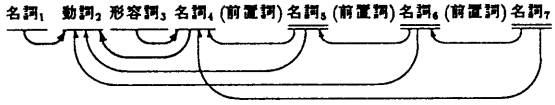

図1 多義性を含む依存構造

表1 制約マトリックス

5\6	2	5
2	1	1
4	1	1

5\7	4	6
2	0	1
4	1	1

6\7	4	6
2	0	1
5	1	1

この依存構造では，5,6,7の(位置の)単語が，それぞれ {2,4}, {2,5},{4,6} の単語に係り得るという多義性が存在する。そこで，5と6，5と7，6と7の間の係り先に関する相互依存関係をマトリックスとして表現したのが制約マトリックスで，それぞれの係り先に関して，それぞれが同時に係ることが可能ならば1，不可能ならば0をマトリックスの値とする (実際には，後で述べるようにこれよりやや複雑である)。

この制約マトリックスは，例えば，5が2に係ることと，7が4に係ることが，非交差の制約のため，同時

に起こらないことを表わしている(表1の2つめのマトリックスを参照)。ここで，もしなんらかのやり方で，7が6に係らないことがわかったとしたら，5と6がそれぞれ2に係ることは不可能であるから，自動的に5と6の係り先が決定する。また，7の係り先も4に決まり，すべての係り受けが一意になる。これが，制約伝播に基づく構造的多義性の解消である。

2.2. 統語構造と制約の表記

構造的多義性を内包した統語構造の表記法は，SeoとSimmonsによる *syntactic graph*[9]に類似している。つまりノードを[*pid* (単語の位置, position identifier), *word* (単語), *pos* (品詞, part of speech)]の3つ組で表わし，アークを [$node_1$, $node_2$, *case* (関係のラベル)]で表わした有向グラフ表現である。アークは $node_1$ が $node_2$ に *case* という関係で係ることを表わしている。つまり，依存関係を表わしている。

case には， subj (主格), obj (目的格), iobj (間接目的格), in,on,withなどの前置詞がある。多義性は，同一のノードから複数のアークが出ていることで表わされる。

また，文法的知識は次のような形式で表わされる。我々は，それを制約式と呼ぶ。

ここで， $node_i = [\ pid_i, word_i, pos_i\ ]$ $(i \in N)$とすると，

C_1: $\forall i,j,k,l\ [\ node_i, node_j, case_i\ ]$ &
$[\ node_k, node_l, case_j\ ]$ & $pid_i < pid_k \rightarrow pid_l \leq pid_j$

ここで， $pid_i < pid_k$とは，文中において $word_i$ が $word_k$より左にあることを表わしている。つまり，この制約式は2つの係り受けが交差しないことを表わしている。

C_2: $\forall i,j,k\ [\ node_i, node_j, case_i\ ]$ &
$[\ node_k, node_j, case_j\ ] \rightarrow case_i \neq case_j$

この制約式は同じ単語に係る2つの係り受けは，関係のラベルが異なることを表わしている。

2.3. 制約マトリックスと制約伝播アルゴリズム

制約式から制約マトリックスが作られる。制約マトリックスは，前に示したように2つの多義性に関する2次元のマトリックスで表わされる。

例えば，ある多義性を

$A_1 = \{a = [node_1, node_2, case_1],\ b = [node_1, node_3, case_2]\}$

とし，別の多義性を

$A_2 = \{c = [node_4, node_3, case_2],\ d = [node_4, node_5, case_3]\}$

とすると，これらの多義性に関する制約マトリックスは，次のようになる。

$A_1 \backslash A_2$	c	d
a	1	1
b	0	1

このマトリックスには，制約式 C_2 が反映されている。また， $pid_1 < pid_4$ & $pid_2 < pid_3$とすると，制約式 C_1 より， a と c は同時には存在できなくなるから，先の制約マトリックスは，次のようになる。

$A_1 \backslash A_2$	c	d
a	0	1
b	0	1

このとき， A_2 は cを要素として持ち得なくなる (マトリックスの c の列の要素がすべて 0になったから)ので， $A_2 = \{d\}$ となり $node_4$に関する多義性が解消されたことになる。また， A_2 が c を要素としないことは，他の制約マトリックスに影響を与える。例えば，ある多義性 $A_3 = \{e, f, g\}$ が存在し，次のような制約マトリックスが存在するとする。

$A_2 \backslash A_3$	e	f	g
c	1	1	0
d	0	0	1

ここで， A_2 は c を持たないことを考えると，上のマトリックスは次のようになる。

$A_2 \backslash A_3$	e	f	g
c	0	0	0
d	0	0	1

このマトリックスから， A_3 は e と fを要素として持ち得ないことがわかる。このことが，同様に，他の制約マトリックスに影響する。このようなメカニズムは制約伝播と呼ばれる。

制約伝播には，MohrとHendersonによってAC-4[5]と呼ばれる効率のよいアルゴリズムが開発されている。我々は，それに若干手を加えて次のようなアルゴリズムを考案した。

ここで，多義性 A_i と A_jに関するマトリックスを $M_{i,j}$, A_i と A_j がそれぞれ要素 a と bをとるときのマトリックスの値を $M_{i,j}(a, b)$ と書くことにする。

まず，マトリックス $M_{i,j}$ と A_i の要素 a に関して，(i, a) のサポート集合 $S(i,j,a) = \{(j, b) | M_{i,j}(a, b) = 1$ (このとき， (j, b) は (i, a) をサポートするという)$\}$を構成する。このとき， $S(i,j,a) = \{\}$ であるとき，(i, a) をインアクティブ集合 INの要素とする。また，A_j の要素 bに関しても，同様に A_j の要素 b に関して (j, b) のサポート集合 $S(j,i,b)$ を構成し，$S(j,i,b) = \{\}$ のときに， (j, b) を IN の要素とする。同様なことをすべてのマトリックスとすべての要素について行なう。また， $S(i,a) = \cup_k S(i,k,a)$ とする。

IN が空になるまで，次のステップを繰り返す。

1. IN の要素 (i, a) を取り出し，それを IN から削除する。

2. $A_i = A_i - \{a\}$ とする。
3. $S(i,a)$ のすべての要素 (j,b) について，$M_{i,j}(a,b)=0$ とし，$S(j,i,b)=S(j,i,b)-\{(i,a)\}$ とする。
4. このとき，$S(j,i,b)=\{\}$ で，(j,b) が IN の要素でないならば，(j,b) を IN の要素とする。

4章では，このアルゴリズムを応用して，依存関係の優先度による制御に基づく多義性解消のアルゴリズムについて述べる。

3. 選好としての意味的知識

我々は，前置詞句付加などの係り受けの多義性を，既存の辞書などから獲得した知識を用いて解消する手法[6,7]を開発している。
これまでに，JensenとBinot[2]によって，オンライン辞書の定義文を用いて，前置詞句の係り受けを決定するヒューリスティックな手法が提案されているが，彼らの枠組みと我々のものは，辞書の定義文を知識源としている点は同じであるが，その利用方が異なっている。つまり，彼らの手法は，句構造によるパターン・マッチングを行い，複雑な規則によって確信度を計算しているのに対し，我々のは，依存構造木を探索し，ノードをさかのぼるという簡単なやり方に加えて，少数の単純化したプロセスで優先度を計算しているという点が異なっている。我々の枠組みでは，意味的知識として，知識ベースに登録された単語間の類義関係・階層(上位/下位)関係・依存関係を用いて，多義性の構成する各々の係り受けの候補の間に優先性を与えている。多義性の解消は，最も優先度の高い係り受けの候補を選択することによって行なわれる。単語間の関係は，実際に現れている文を解析することによって得られる。多義性の解消された依存構造は，そのまま知識ベースに登録することができる。この意味で，知識ベースは，self-extendingであるといえる(このようにして知識獲得を行うことを，我々は知識のブートストラップ (knowledge bootstrapping)[7]と呼んでいる)。また，これはいわゆる case-based (あるいは，memory-based)なアプローチ[8]である。
また，文法的知識を用いて，その多義性の解消によって得られる制約を，同じ文に含まれるその他の係り受けの多義性に伝播して，その他の多義性の係り受けの候補を限定する。このようなやり方は，局所的な優先性に基づいて決定するため，文全体として考えると，優先性が異なる場合がある。つまり，局所的には優先度が比較的低い係り受けが，文全体として見ると，最も優先度が高いという場合がありうる。つまり，全体的な優先性に基づいて決定する枠組みが必要になる。この全体的な優先性に関する議論は，次章で行う。

3.1. 局所的な優先度の計算

我々の開発した局所的な意味的優先度を計算する手法は，次のようになっている。
例えば，$<noun_1>$ $<verb_2>$ $<noun_3>$ $<prep>$ $<noun_4>$のような形式の文において，前置詞句 $<prep>$ $<noun_4>$ は，$<verb_2>$ に副詞的に係るか，または $<noun_3>$ に形容詞的に係る，という多義性が存在する(図2参照)。

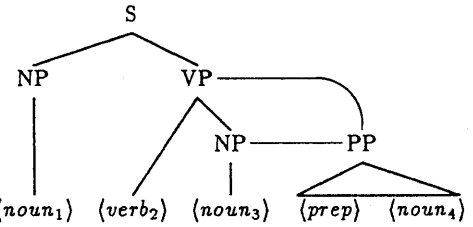

図2 多義性を含む句構造

まず，この句構造から依存構造を構成し，単語間の依存関係を明示化する(図3参照)。

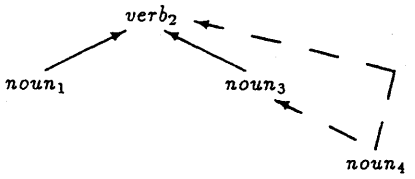

図3 多義性を含む依存構造

この依存構造には，多義性がパックされてあり，各単語は依存先の候補がわかるようになっている。次に，各多義性に関して，依存関係の候補に優先度を付ける。
それは，パス・サーチと依存距離計算というメカニズムによって行われる。
パス・サーチは，依存関係を構成する2つの単語の間に，知識ベースに登録された依存関係があるかどうかどうかを探索するメカニズムである。このとき，2つの単語間に直接の関係(同一の依存構造内に定義される関係)が存在するとは限らないので，それらの類義語あるいは上位語間の関係でも構わない。そして，類義関係，階層関係は推移することができ，依存関係も含めて，2単語間に関係の連鎖が成立することがある(図4参照)。

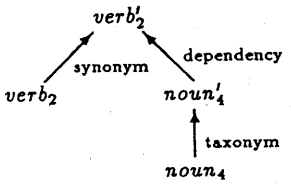

図4 $noun_4$から$verb_2$へのパス

このような関係の連鎖をパスと呼ぶ。
我々のパス・サーチはCharniakによるマーカー・パッシング[1]にかなり類似したものであるが，マーカー・パッシングが高い計算コストで多くの意味の無い関係を見いだしてしまうのに対して，我々のパスは類義・階層関係と依存関係しか含まないので，かなり探索空間を絞って，できるだけ，必要な情報のみを抽出できるようにしている。

依存距離計算は，依存関係に，依存距離と呼ばれるヒューリスティックな値を割り当てるメカニズムであり，依存距離はパスが存在しなければ最大値をとるが，存在する場合は，次の3つの条件に応じて計算される。

1. 深層格の整合性 (case consistency)
   文から予測される依存関係の深層格ラベルと，パスに含まれる依存関係の深層格ラベルとの間に整合性があること。簡単に言えば，パスの方のラベルが，文の方のラベル(の候補)の要素になっていることである。この条件を満たしている依存関係は依存距離が短くなり，その依存関係は優先度が高くなる。この整合度は，ラベルが整合していれば1，いなければ，0である。

2. 共起修飾語の整合性 (co-occurrence consistency)
   共起関係に関する条件で，文の依存関係の係り先の単語に依存しているその他の単語とパスの依存関係の同じ位置にある単語の間で，整合性があること。つまり，共起している単語が一致するか，または単語間に類義・上位/下位関係が存在し，かつ関係のラベルが整合していることである。その依存関係の依存距離が短くなり，依存関係の優先度が高くなる。この整合度は，整合している共起修飾語の数をとる。

3. 文脈との整合性 (context consistency)
   パスに含まれる依存関係が，前文までの文脈に含まれていること。これは，前文までの依存構造を文脈ベースという形で保存しておき，文脈ベースに対してパス・サーチを行ったときに，文脈のパスが存在することである。もとのパスの依存関係の依存距離が短くなり，その結果，そのパスを持つ依存関係の依存距離が短くなり，その依存関係の優先度が高くなる。この整合度は，文脈と整合しているパスの依存関係の数をとる。

依存距離は，これら3つの条件をすべて考慮して計算され，各依存関係の候補の優先度が決定される。依存距離の計算式は次のようになる。

$$Distance = \frac{|Dep| - n \times V_{Cont}}{(V_{Case} + 1) \times (V_{Cooc} + 1)}$$

ここで，$|Dep|$ は，パスに含まれる依存関係の数で，V_{Case} は，深層格の整合度，V_{Cooc} は，共起修飾語の整合度，V_{Cont} は，文脈との整合度である。
この式は，深層格と共起修飾語の整合度はパス全体の距離に影響し，文脈との整合度は，パスに含まれる各々の依存関係の距離に影響することを反映している。
また，n は，$0 < n < 1$ の実数で，それは，文脈との整合度をどの程度重要とするかに関するヒューリスティックに決められるパラメータである。
多義性は，優先度の最も高い依存関係を選択することによって，解消される。優先度は依存距離の逆数，つまり1/依存距離である。

4. 文法的知識と意味的知識の統合

コネクショニズムの手法を用いて，文法的知識と意味的知識を統合するやり方[11,12]が開発されている。そこでは，文法的知識と意味的知識が，リラクゼーション・ネットワークの形で統合され，活性伝播の後に最も活性度の高い解釈を選択する。しかし，これは活性度が有限回の伝播の後に平衡状態に達するという仮定の下で有効なのであり，一般にその保証はない。また，文法を記号的制約として用い，明示的に構造を限定するというメリットが損なわれてしまう。したがって，我々はこのような統合の手法を用いず，文法的知識と意味的知識の新たな統合法を開発する。

4.1. 優先度の制御に基づく制約伝播

これまでに述べてきた文法の制約マトリックスとしての記述に優先度としての意味を付与した表現法を導入する。そこで，2章の例 (図5参照)を再び用いると表2のようになる。

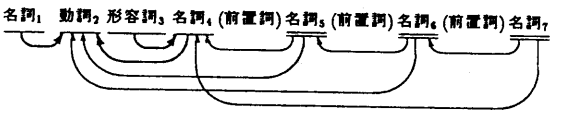

図5 多義性を含む依存構造

表2 優先度付き制約マトリックス

5\6	2	5	$P_{5,i}$
2	1	1	2
4	1	1	0
$P_{6,j}$	2	1	

5\7	4	6	$P_{5,i}$
2	0	1	2
4	1	1	0
$P_{7,j}$	4	2	

6\7	4	6	$P_{6,i}$
2	0	1	2
5	1	1	1
$P_{7,j}$	4	2	

この場合の多義性解消は，任意の2つの多義性間の制約が満たされている中で，各々の多義性がそれぞれ優先度が最も高い候補を取ることによって行なわれる。つまり，優先度の小さい順に制約伝播によるフィルタリングを行い，もし可能な組合せがなく

なったらキャンセルし，次に優先度の小さいものについて同様に行ない，残ったものの中で優先度が最も高い組合せを選択するというやり方である。上の例では，5が4に係る優先度が0で最小なので，5が4に係ることを表わす行の値を0にする。このとき，マトリックスの値がすべて0になったものがあれば，キャンセルするが，なければそのままにする。さらに，7が4に係る可能性が0になるので，7が4に係ることを表わす列を0にする。このとき，6が2と5に係る可能性はともに1なので，優先度の高い方，すなわち2に係る方を選択する。結果として，5は2に係り，6は2に，7は6に係る。ここで，7は局所的には4に係る優先度が高いが，文全体としてみると，6に係る優先度が高かったことになる。
このように，優先度はどのように制約を満たす候補を絞り込み，かつ最終的な選択をするかに関する制御情報として用いられる。

4.2. 制約と選好による構造的多義性解消アルゴリズム

意味的知識による優先度を考慮するために2章で導入したアークの表現に若干の拡張をする。すなわち，$[node_1, node_2, case, pref$ (優先度)]として，アークに優先度を付加するのである。(多義性のないアークの優先度は nil とする。)
このとき，優先度の制御に基づく制約伝播による多義性解消のアルゴリズムは次のようになる。
ここで，2章と同様に，多義性 A_i と A_jに関するマトリックスを $M_{i,j}$，A_i と A_j がそれぞれ要素 a と bをとるときのマトリックスの値を $M_{i,j}(a, b)$ と書く。また，多義性 A_i の要素 a の優先度を $P_{i,a}$ と書く。まず，各々の多義性 A_iとその要素に関して，優先度の小さい順に左から右に並べた順列 PRF_i を構成する。つまり，PRF_i の任意の要素 a, bに関して，a が b の左ならば $P_{i,a} \leq P_{i,b}$である。ただし，a と b の優先度が等しいときは，各々の要素に含まれる2つのノードの pidの値の差が大きい方を左にする (これは，構造の局所的優先性である)。

1. PRF_i の最も左の要素の $P_{i,a}$がすべての A_i について最小である a を取り出し，それを PRF_i から削除する。
2. $a \in A_i$ ならば，次のステップへ。そうでなければ，ステップ1へ。
3. 任意の j, b に関して，$M_{i,j}(a, b) = 1$ならば，その値を0にし，(i, a, j, b) を集合 CHG に登録する。
4. このとき，制約伝播アルゴリズム[5]により，すべての多義性とその要素に関して　2章と同様に，サポート集合とインアクティブ集合 IN を構成する。
   まず，マトリックス $M_{i,j}$ と A_i の要素 a に関して，(i, a) のサポート集合 $S(i, j, a) = \{(j, b) \mid M_{i,j}(a, b) = 1\}$を構成する。
   また，$S(i, a) = \cup_k S(i, k, a)$ である。
   このとき，$S(i, j, a) = \{\}$ であるとき，(i, a) をインアクティブ集合 INの要素とする。同様のことをすべてのマトリックスとすべての要素について行なう。
5. IN が空になるまで，次のステップを繰り返す。
   a. IN の要素 (j, b) を取り出し，それを IN から削除する。
   b. $A_j = A_j - \{b\}$ とする。
   c. このとき，$A_j = \{\}$ ならば，$A_j = \{b\}$ とし，CHG のすべての要素 (k, c, l, d) に関して $M_{k,l}(c, d) = 1$ とし，$CHG = \{\}$ として，ステップ1へ。
   d. $S(j, b)$ のすべての要素 (k, c) について，$M_{j,k}(b, c) = 0$とし，$S(k, j, c) = S(k, j, c) - \{(j, b)\}$とする。
      また，(j, b, k, c) を CHG に登録する。
   e. このとき，$S(k, j, c) = \{\}$ で，(k, c) が IN の要素でないならば，それを IN の要素とする。
6. すべての多義性 A_i が singletonならば終了。そうでなければ，ステップ1へ。

4.3. 全体の処理の流れ

構造的多義性の解消の処理の流れは，図6のようになる。

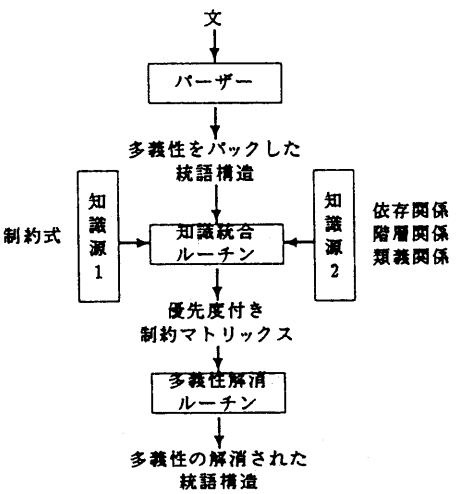

図6　多義性解消の処理の流れ

まず，文から通常のパーザーにより構造的多義性のパックされた統語構造が構成される。これは，2章で紹介した依存関係に基づくグラフ構造である。この構造に制約式を適用することによって2つのアークの間の制約として制約マトリックスを構成し，また同

時にパス・サーチと依存距離計算によって各多義性の要素(候補)となるアークの優先度を計算する。この結果，優先度付きの制約マトリックスが構成され，多義性解消のアルゴリズムが適用される。
構造的多義性の解消された依存構造は，文脈ベースと呼ばれるデータ・ベースに蓄えられ，次からの文の解析に利用される。

4.4. 例題

次の文を用いて，多義性解消の例を示す。
"CMS executes the program in the package that is written in REXX."
図7は，この文の句構造を表わしている (?マークは，係り先のその他の候補を表している)。また，この文から，表3のような制約マトリックスが構成される。この表の優先度は，パス・サーチと依存距離計算によって計算された。$P_{i,a}$ とは，多義性 A_i が要素 a をとるときの優先度である。優先度が0になるのは，依存関係に対して，パスが存在しなかった場合である (このとき，依存距離は最大値をとる)。

```
DECL NP     NOUN*   "CMS"
     VERB*"executes"
     NP     DET     "the"
            NOUN*   "program"
     ?      PP      PREP    "in"
                    DET     "the"
                    NOUN*   "package"
            ?       RELCL   PRON    "that"
                            VERB    "is"
                            VERB*   "written"
     ?      ?       ?       PP      PREP    "in"
                                    NOUN*   "REXX"
     PUNC   "."
```

図 7　構造的多義性を含む句構造

表 3　優先度付き制約マトリックス

$A_1\backslash A_2$	c	d	$P_{1,i}$
a	0	1	2
b	1	1	1
$P_{2,j}$	2	0	

$A_1\backslash A_3$	e	f	g	h	$P_{1,i}$
a	0	0	1	1	2
b	1	0	1	1	1
$P_{3,j}$	1	4	2	2	

$A_2\backslash A_3$	e	f	g	h	$P_{2,i}$
c	1	1	0	1	2
d	1	1	1	1	0
$P_{3,j}$	1	4	2	2	

また，制約マトリックスに現われている記号の意味は次の通りである。

- $A_1=\{a,b\}$: 前置詞句"in the package"が"executes"か "the program"のどちらかに係るという多義性
- $A_2=\{c,d\}$: 関係節"that is written"が"the program"か "the package"のどちらかに係るという多義性
- $A_3=\{e,f,g,h\}$: 前置詞句"in REXX"が"executes"か "the program"か"the package"か "written"のどれかに係るという多義性

ここで，A_1 と A_3 の制約マトリックスで，A_3 の f の列は，すべて0であるため，A_3 が f を要素として持たないことがわかる。この結果は，A_2 と A_3 の制約マトリックスにも伝播される。また，A_2 が d をとる優先度は最も小さいので，A_2 の d の行と列の値をすべて0にする。その結果，マトリックスは次のようになる。

$A_1\backslash A_2$	c	d	$P_{1,i}$
a	0	0	2
b	1	0	1
$P_{2,j}$	2	0	

$A_2\backslash A_3$	e	f	g	h	$P_{2,i}$
c	1	0	0	1	2
d	0	0	0	0	0
$P_{3,j}$	1	4	2	2	

このとき，A_1 は a を要素として持ち得ないことがわかり，$A_1=\{b\}, A_2=\{c\}$ となる。また，$A_3=\{e,h\}$ であるが，優先度の低い方を落とすと，$A_3=\{h\}$ となり，すべての多義性が解消されたことになる。

5. 関連した研究

文法的知識と意味的知識を統合して文を解析する研究は，これまでにも，いくつか行なわれている。ただし，ここで紹介するものは，2つのタイプの知識が互いに作用し合って働くものであり，文法的処理が 完全に終了してから意味処理で解釈を一意にする，いわゆるカスケード結合のものは含まない。

Lytinen[3]は，パージング時に動的に文法的・意味的知識を統合する統合パーザーを開発している。彼は，これまでの統合システムの問題として，文法と意味のルールを一つのルール・セットにコンパイルして持っているため，知識のモデュラリティが低く，効率が悪い点を挙げている。我々の枠組みも彼のと同様に2つのタイプの知識を別々に持ち，解析時に動的に組み合わせている。ただし，彼のシステムが構文規則の適用時にその適切さを意味的知識を用いて判断し，適切でなければ適用しないのに対し，我々のは，あくまで文法的知識は制約として構造を限定するのに用い，意味的知識は選好として可能な候補に優先性を与えるのみとする点が異なっている。

辻井ら[10]はKGW＋pと呼ばれるシステムにおいて，制約的知識と選好的知識を構文解析時に動的に組み合わせて用いる手法を提案している。それは，我々の枠組みと同じ目的を持っているが，メカニズムは全く異なる。彼らは，制約的規則を文の構成要素が満たす部分的な関係を表わす部分解析木を構成するために用い，選好的規則をその部分解析木の尤度を計算するのに用いているのに対し，我々の枠組みでは，制約は統語構造が文法的条件を満たしている状態を保持するために用い，選好を可能な依存関係の候補を削除する順序を制御し，最適な構造を得るために用いている。また彼らの手法はルールに基づいているため複雑で，可能な部分構造を全て生成するため効率が悪い。しかし，我々のは多義性を

パックしたデータ構造を持ち，効率の良いフィルタリング・アルゴリズムを用いているため効率的にも優れている。

Wermter[12]は，コネクショニストモデルに基づく知識の統合を行なっている。統合はリラクゼーション・ネットワーク上で行なわれ，意味的知識はバックプロパゲーションによって学習される。バックプロパゲーション学習によって獲得される知識はしばしば有用であるが，表現が非明示的なものになってしまい，知識を管理するのに適切ではない。我々の枠組みでは，意味的知識は明示的な関係の連鎖で表わされるため，知識の内容を把握するのに適切である。また，リラクゼーションによる統合は文法などの記号的制約の充足が不完全になってしまう場合があるが，我々の枠組みでは，制約マトリックス上に0と1で表わされた制約がそれを防いでいる。

6. おわりに

構造的多義性の問題に対して，制約としての文法的知識と選好としての意味的知識を協調的に利用して解消する手法について述べた。この手法は，従来のルールに基づく統合に比べて，知識源を完全に分離して，文解析時に動的に組み合わせることができるというメリットがある。制約としての文法は，構造を強く限定するが，宣言的な記述性を持ち，メンテナンスが容易である。選好としての意味は，選択制限などに見られる制約としての記述に比べて構造を限定する能力が乏しい反面，すべての構造を許容する柔軟性がある。また，意味的知識の知識源は実例からの学習によってカバレージを上げることができる。ここでの議論は，係り受けの多義性に限定しているが，語義の多義性などのその他の問題を同時に考慮することは重要である。そのために，語義の多義性などを反映できる構造と，その構造上の制約と選好の記述法を考慮する必要がある。この構造は，ある種の深層構造であるため，概念的な制約や選好が重要となる。このような知識をどのようにして獲得するかは，今後の課題である。

文献

1. Charniak, E.: A Neat Theory of Marker Passing. *Proceedings of AAAI-86* (1986) pp. 584-588.
2. Jensen, K. and Binot, J-L.: Disambiguating Prepositional Phrase Attachments by Using On-Line Dictionary Definitions. *Computational Linguistics* **13** (1987) pp. 251-260.
3. Lytinen, S.L.: Dynamically Combining Syntax and Semantics in Natural Language Processing. *Proceedings of AAAI-86* (1986) pp. 574-578.
4. Maruyama, H.: Structural Disambiguation with Constraint Propagation. *Proceedings of the 28th Annual Meeting of the ACL* (1990) pp. 31-38.
5. Mohr, R. and Henderson, T.: Arc and Path Consistency Revisited. *Artificial Intelligence* **28** (1986) pp. 225-233.
6. Nagao, K.: Dependency Analyzer: A Knowledge-Based Approach to Structural Disambiguation. *to appear in Proceedings of COLING-90* (1990).
7. Nagao, K.: Structural Disambiguation with Knowledge on Word-to-Word Dependencies. *to appear in Proceedings of InfoJapan'90* (1990).
8. Sato, S. and Nagao, M.: Toward Memory-based Translation. *to appear in Proceedings of COLING-90* (1990).
9. Seo, J. and Simmons, R.F.: Syntactic Graphs: A Representation for the Union of All Ambiguous Parse Trees. *Computational Linguistics* **15** (1989) pp. 19-32.
10. Tsujii, J., Muto, Y., Ikeda, Y., and Nagao, M.: How to Get Preferred Readings in Natural Language Analysis. *Proceedings of COLING-88* (1988) pp. 683-687.
11. Waltz, D.L. and Pollack, J.B.: Massively Parallel Parsing: A Strongly Interactive Model of Natural Language Interpretation. *Cognitive Science* **9** (1985) pp. 51-74.
12. Wermter, S.: Integration of Semantic and Syntactic Constraints for Structural Noun Phrase Disambiguation. *Proceedings of IJCAI-89* (1989) pp. 1486-1491.