

## discrimination network 上での増進的曖昧性解消について

奥村 学, スラバン・メークナウィン, 田中 穂積  
東京工業大学 工学部

### [概要]

自然言語解析では、文中の意味的曖昧性をどのように解消するかということが問題になる。我々は意味的曖昧性を増進的に解消する手法を提案している。一方、意味的曖昧性解消過程を discrimination network を下向きにたどる過程として実現する研究がある。しかし、解析過程では discrimination network を上から下に順々にたどれる保証はない。本研究では、解析過程で増進的に得られる制約の順序の非決定性を考慮して、discrimination network をたどる手法を提案する。

## Towards Incremental Disambiguation with a Generalized Discrimination Network

OKUMURA Manabu, MEKNAVIN Surapant and TANAKA Hozumi  
Department of Computer Science, Tokyo Institute of Technology  
(2-12-1 Oookayama Meguro-Ku Tokyo 152 Japan)

### Abstract

Semantic disambiguation is a difficult problem in natural language analysis. A strategy for semantic disambiguation is to accumulate information obtained during the analysis process of a sentence and disambiguate the meaning incrementally by using the information. We have proposed such a computational model of natural language analysis, called 'incremental disambiguation model'. Semantic disambiguation process can be equated with the traversing of discrimination network. However, discrimination network has a problem that it cannot be traversed unless constraints are entered in a priori fixed order. In general, constraints cannot be obtained in that order, so it is not always possible to traverse the network downward during the analysis process. In this paper, we propose the method which can traverse discrimination network with regard to the order of constraints obtained incrementally during the analysis process which is independent of a priori fixed order of the network.

## 1 はじめに

自然言語解析では、文中の意味的曖昧性をどのように解消するかということが問題になる。我々は意味的曖昧性を増進的に解消する手法 [10] を提案している [1]。意味的曖昧性が自然言語解析に遍在するのは、解析過程で得られる情報が部分的で [2]、得られた時点では十分な決定ができないことに起因する。意味的曖昧性を解消する手法として、得られた情報 (制約) を即座には解析に用いず、たとえば、文末までその処理を遅延する手法をとると、曖昧性の数 (探索空間の大きさ) は組合せ的に爆発してしまう。しかし逆に、十分な制約が得られていない時点で無闇に決定を行なうと、非決定性の数 (後戻りの数) が爆発的に増加してしまう。したがって、曖昧性を解消するための望ましい方法は、文を解析する過程で得られる情報 (制約) を増進的に蓄積し、できるかぎり早期の段階で曖昧性を段階的に解消していくことである (曖昧性の増進的解消)。増進的曖昧性解消過程は、曖昧性を含んだ (未決定の部分が残った) 意味解析結果を、新たに得られた制約により増進的に洗練し決定していく過程であると言える。したがって、文に含まれる曖昧性がその文中では十分解消できず、後続する文の情報を考慮しなければ解消できない場合にも、増進的曖昧性解消手法は自然に対処できると考えられる。

一方、意味的曖昧性解消過程を、discrimination network (弁別ネットワーク) を下向きにたどる過程として実現する研究がある。2.1 節で詳述するように、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ような利点がある：

- 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していると、曖昧性が増大し候補数が多くなると、リストの線形探索は非常に効率が悪くなる。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。

また、ネットワーク中の葉ノード以外のノードはすべて曖昧性を含んだ意味表現と解釈できるので、未決定の部分が残った意味解析結果をネットワーク中のノードで容易に表現できる。後続する入力から得られる付加的情報により新たな決定を行なうことは、現在到達しているノードから新たに得られた情報に基づいてさらにネットワー

クを下向きにたどることに自然に対応する。したがって、増進的曖昧性解消手法との整合性が大きい。

しかし、2.2 節で述べるように、弁別ネットワークは本来、前もって決定された順序で性質 (制約) が入力されないとたどることができない。ところが一般に、解析過程で制約の得られる順序があらかじめ定められた通りであるとは限らないので、解析過程で弁別ネットワークを上から下に順々にたどれる保証はない。弁別ネットワークを用いた従来の研究における、このような定められた順序からはずれた場合についての対処法の問題点については 2.2 節で述べる。

本研究では、解析過程で増進的に得られる制約の順序の非決定性を考慮して、弁別ネットワークをたどる手法を提案する。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と呼ぶ。この手法は、3 節で詳述するように、制約プログラミングの考え方に基づいており、拡張ユニフィケーションにより実現されている。

2 節では、弁別ネットワークを意味的曖昧性解消に用いることの利点および、弁別ネットワークを用いた従来の研究の問題点について述べる。3 節では、一般化弁別ネットワークの原理について述べる。最後に、4 節では、一般化弁別ネットワークの利点と今後の研究課題について述べる。

## 2 弁別ネットワークを用いた意味的曖昧性解消の利点と問題点

図 1 に動詞 'take' の語義を表現した弁別ネットワークの一部を示す。弁別ネットワークの枝には、動詞の取りうる表層格とその格を満たしうる名詞句に関する選択制限の組が記述される。図 1 では、'take' が S (Subject), O (Object), with, to などの格を取りうるものが記述されている。弁別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) された語義の複数の選択肢を表す。ネットワークの葉ノードは曖昧性のない 1 つの語義を表す。葉ノードに付与されたラベル (下線を付したものはその語義を表すものとする。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 (語義) に到達しうることから、到達しうる語義をすべて含んだ曖昧性のある意味を表現していると考えられる。根ノードは、すべての語義を含んだ表現であり、意味が未知であることを意味する。

弁別ネットワークを用いた意味的曖昧性解消は、得られた制約を満足するアークに沿って弁別ネットワークを根ノードから葉ノード (曖昧性のない語義) に向けて下向きに一段ずつたどる過程であると考えられる。この過程により、文中の回りの単語の情報 (動詞が取る格要素の情

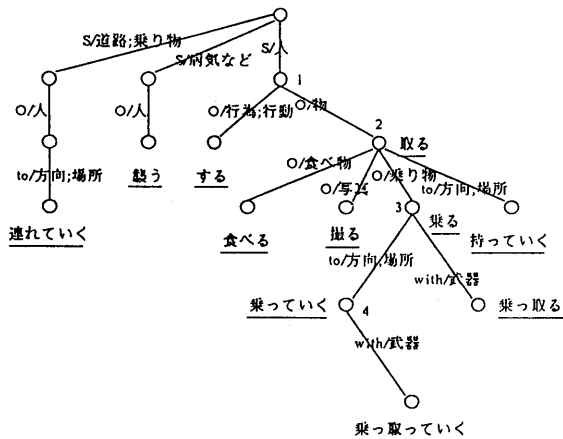


図 1: 動詞 'take' の語義弁別ネットワーク (一部)

報) から、異常な選択枝は排除され妥当な動詞の語義が弁別 (選択) される。葉ノードに到達することが曖昧性が完全に解消されたことを意味する。1文を解析し終えた時点で葉ノードに到達できていない場合、その文は意味的に曖昧であり、その時点で到達しているノードがその文の動詞の曖昧性を含んだ語義 (意味解析結果) を表現する。

図 1 の弁別ネットワークを用いて文 'John took a plane to London.' を解析することを考えてみよう。例文からは、'take' が取る格要素の情報として、集合 [S/John, O/plane, to/London] が得られる。これらの制約に基づき、図 1 の弁別ネットワークをたどると、まず表層格 S を満たす名詞句 'John' は、選択制限「人」を満足するので、根ノードからノード 1 に到達する。次に、表層格 O の名詞句 'plane' は、選択制限「物」、「乗り物」を満足するので、ノード 2 を経てノード 3 に到達する。最後に、表層格 to の名詞句 'London' が選択制限「方向;場所」<sup>1</sup> を満足するので、この文の動詞 'take' の意味は「乗っていく」(ノード 4 参照) であることがわかる。

2.1, 2.2 節ではそれぞれ、弁別ネットワークを意味的曖昧性解消に用いることの利点、問題点について述べる。

**2.1 弁別ネットワークを用いた意味的曖昧性解消の利点**  
意味的曖昧性解消過程を、弁別ネットワークを下向きにたどる過程として実現する研究として、[8, 11, 9, 3] がある。これらの研究では、弁別ネットワークを意味的曖昧性解消に用いることの利点として次のようなものがあると主張されている。

- 単語の意味の曖昧性に関しては、Hirst のポラロイド語 [7] のように、従来単語の複数の意味を互いに無関

<sup>1</sup> : は選言を表す。

係であるとしてそれぞれ独立に扱う傾向が強かった。しかし、この手法では、単語の複数の意味の間に本来あるはずの何らかの共通性をとらえられない。これに対し、弁別ネットワークは、語義的に互いに共通性の少ない意味同士をネットワーク上で離れた位置に、共通性の多い意味同士を近くに表現することが可能である。そして、弁別ネットワークを下向きにたどる過程は、漠然としていた単語の意味が次第に洗練され、より具体的な意味として明らかになる過程と対応する [8]。

- 複数の意味を互いに無関係であるとして扱う手法では、候補となる単語の意味をリストのような形式で保持することが多い。曖昧性が増大しリスト上の候補数が多くなった場合、このような表現形式から候補の単語の意味を線形探索するのは非常に効率が悪い。これに対し、弁別ネットワークを用いた場合、葉ノードに位置する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。この場合、枝に記述されている制約を索引として探索するので、探索空間は徐々に小さくなり、線形探索に比べて効率が良い [9]。

また、一般に、1文を解析し終えた時点ではその文に含まれる曖昧性を十分解消できず、後続する文の情報を考慮しなければならないことがある。複数の連続していない文よりはじめて 1 つの事実が導出されることもある。このような場合に対処するには、1文に対する意味解析結果として曖昧性を含んだ表現を許す必要があり、またその曖昧性を含んだ意味解析結果は、後続する入力から得られる付加的情報により更新できなければならない。これは増進的曖昧性解消手法の目標でもある。

弁別ネットワーク中の葉ノード以外のノードはすべて曖昧性を含んだ意味表現と解釈できるので、未決定の部分が残った意味解析結果を、葉ノード以外のノードにより容易に表現できる。また、後続する入力から得られる付加的情報により新たな決定を行なうことは、現在到達しているノードから新たに得られた情報に基づいてさらにネットワークを下向きにたどることに自然に対応する。したがって、弁別ネットワーク表現は増進的曖昧性解消手法との整合性が大きい [11]。

次節では、弁別ネットワークを用いて意味的曖昧性解消を実現している従来の研究の問題点について述べる。

## 2.2 弁別ネットワークを用いた意味的曖昧性解消の問題点

前節では、弁別ネットワークを用いて意味的曖昧性解消を実現することの利点について述べた。弁別ネットワークは

増進的に意味的曖昧性解消を行なうのに都合のよい性質を持っていることを説明した。

しかし、弁別ネットワークは本来、前もって決定された順序で性質(制約)が入力されないと下向きにたどることができない。弁別ネットワークを根ノードから下向きにたどる際には、たどる枝に付与された制約を満足しなければならない。したがって、制約は、根ノードに近いものから順々に入力されなければならない。この順序は弁別ネットワークの構造に依存して決まる。文‘John took a plane to London.’について図1の弁別ネットワークをたどり動詞‘take’の意味を弁別する例を上を示したが、この例の場合、制約はS/John, O/plane, to/Londonの順に入力されないと、弁別ネットワークを正常にたどることはできない。ところが、解析過程で制約が得られる順序はあらかじめ定められた通りであるとは限らないので、解析過程で弁別ネットワークを上から下に順々にたどれる保証は一般にないと言える。構文解析過程で部分木が構築されるのに沿って意味解析を行なう手法を用いると、上の例文の場合には、制約の得られる順序はO/plane, to/London, S/Johnとなり、この順序では図1の弁別ネットワークを正常にたどることはできない。さらに、受身文で主語が省略されてしまう場合のように、入力から必要な制約が得られないと、弁別ネットワークは全くたどることができず、デッドロック状態となる。

これに対し、格要素(制約)の集合全体が得られた後、それを用いて弁別ネットワークをたどる手法も考えられる。しかし、このように、弁別ネットワークをたどり意味的曖昧性解消を行なうのを、文末まで解析が終了した後に遅延すると、曖昧性の数は組合せ的に爆発してしまう。これは、[1]で述べたように、動詞の意味的曖昧性解消と、その格要素となる名詞句の意味的曖昧性解消は相互的なものであり、動詞の意味的曖昧性解消により(間接的に)解消されるはずの名詞句の意味的曖昧性まで解消されないまま文末まで放置されることになるからである。また、この手法は増進的曖昧性解消と相反する。

また、図1のように英語を対象言語とするのなら、英語の文型を考慮して制約の得られる順序として最も頻繁と考えられるO(Object, 目的語), p(preposition, 前置詞), S(Subject, 主語)のような順序で弁別を行なうネットワークを作り、関係節のようにその順序から逸脱する場合には、デモンのような特別な手続きを用意することにより対処するという手法も用いられている[3]。格要素に関する弁別の制約が解析過程で得られる順序は、入力文中の語順に依存する。したがって、あらかじめ定められた順序からの逸脱の頻度、すなわち制約が得られる順序の非決定性は、解析する言語の語順の自由度に依存する。したがって、英語のように語順の自由度のあまり大きくない言語

についてはデモンを用いた手法でも対処できると思われるが、語順の自由度がより大きい日本語などについては、デモンによる場合分けの数が組合せ的に爆発してしまい、仮に記述可能であるとしても手続き部分が巨大化し、可読性が失われ保守が困難となり問題である。

taxonomic lattice[13]は、弁別ネットワークのこのような問題点を指摘し、その解決策として提案されたものであり、制約の順序に依存しない弁別が可能である。taxonomic latticeは、その名の通り、制約が任意の順序で入力されても対処できるように、ネットワークをラティス化したものである。しかし、ネットワークをラティス化したことにより、同じ情報量を表現するのに非常に冗長な表現形式となり、計算機上で実現した場合に非常に大量の記憶容量が必要になる。

次節では、弁別ネットワークをあらかじめ定められた順序で制約が得られなくても弁別ネットワークを下向きにたどることが可能な手法を提案する。任意の順序で入力される制約に沿ってたどることの可能な我々の弁別ネットワークを一般化弁別ネットワークと呼ぶ。この手法により、弁別ネットワークを用いた増進的曖昧性解消を実現することができる。本手法は、ネットワークをラティス化した場合と同等の能力をもつが、ネットワーク形式をそのまま用いることから、ラティス化したときの問題点である記憶容量の増大を引き起こさないという利点がある。

### 3 一般化弁別ネットワークの原理

図2のような弁別ネットワークを考える。図2で枝に付いているラベルは制約を表す。まず、それぞれのノードにユニークな識別子として数字列を割り当てる。最上位ノードには1を割り当てる。1レベル下位の子ノードにはそれぞれ2桁の識別子 $1i$ (ここで、 $i$ は1から $n$ までの整数、 $n$ は下位ノードの個数)を割り当てる。以下順に、ノード $1i_1i_2\dots i_m$ に対して、その子ノードには $1i_1i_2\dots i_m i$ (ここで、 $i$ は1から $n$ までの整数、 $n$ は下位ノードの個数)を割り当てる。たとえば、図2のノードには図3のように識別子が付けられる。

次に、ノードの識別子それぞれについて、先頭と末尾の桁が0でそれ以外が1のビットベクトルを対応させる(図3の識別子については表1ようになる)。このビットベクトルは、弁別ネットワーク中の満足していない制約の位置を表す。たとえば、識別子111,122は010というビットベクトルをもつが、左から2桁目が1であることから、それぞれの識別子の左から2文字分の識別子11,12に対応する制約 $a/a_1, a/a_2$ が満足されていないことを示す。同様に、識別子1232の場合、対応するビットベクトルは0110

<sup>2</sup> ノードの識別子と制約の対応関係は、後述するように、表2から得られる。

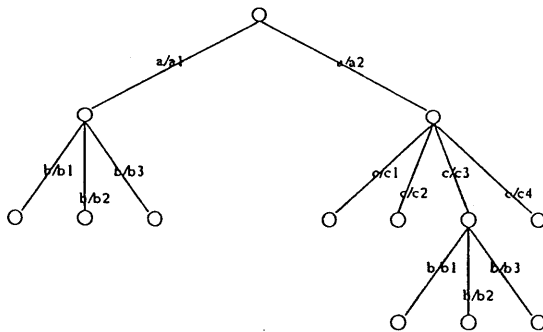


図 2: 弁別ネットワークの例

1	0
11,12	00
111,112,113,121,122,123,124	010
1231,1232,1233	0110

表 1: ノードの識別子と対応するビットベクトル

$a/a_1$	11
$a/a_2$	12
$b/b_1$	{111, 1231}
$b/b_2$	{112, 1232}
$b/b_3$	{113, 1233}
$c/c_1$	121
$c/c_2$	122
$c/c_3$	123
$c/c_4$	124

表 2: 枝のラベル (制約) と対応する識別子

であり、左から 2,3 桁目が 1 なので、左から 2 文字分、3 文字分をとった識別子 12,123 に対応する制約  $a/a_2, c/c_3$  が満足されていないことを示す。また、このビットベクトルは、後述するように、弁別ネットワーク上のあるノードに到達可能かどうかを判断する際の条件となる。

次に、ネットワークから、枝のラベル (制約) と、直接枝でつながれた下位ノードの識別子の組を抜き出す。図 3 のネットワークからは表 2 のような組が得られる。属性名  $b$  の制約については組が複数存在するため、それらをまとめた {111, 1231} のような集合表現を対応させる。この制約と識別子の組は、表 2 中の制約を満足するなら、弁別ネットワーク上で対応する識別子のノードに到達できることを意味する。たとえば、制約  $c/c_3$  が満足された場合、対応する識別子 122 のノードまで下向きに弁別ネットワークをたどれることを意味する。

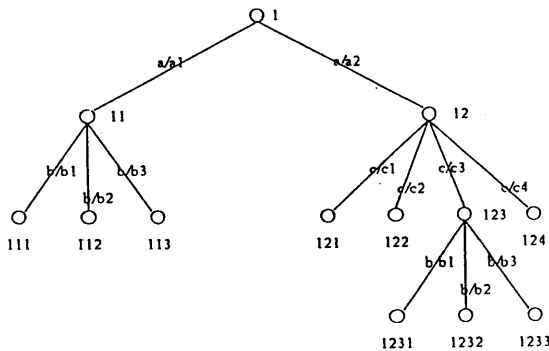


図 3: ノードに識別子を付与した弁別ネットワーク

この際、識別子に対応するビットベクトルの情報に注意する必要がある。上の例の識別子 122 の場合、対応するビットベクトル 010 から、上述したように、制約  $a/a_2$  が満足されていないことがわかるので、この場合、「制約  $a/a_2$  が満足されたら、ノード 122 までたどることができる」という「条件付き」の可到達性を表すことになる。また、属性名  $b$  の制約のように、組が複数存在する場合には、それらの複数のノードに到達可能であることを表す。

図 3 の弁別ネットワーク上を識別子 1231 のノードへ到達するための制約の順序は本来  $a/a_2, c/c_3, b/b_1$  であるが、それとは異なり、 $c/c_3, b/b_1, a/a_2$  の順に制約が得られた

時に、本方式でどのように弁別が進むか、その過程について以下では述べる。ここで、処理過程を表すものとして状態を導入する。状態は、(到達できるノードの識別子、満足していない制約の位置を表現するビットベクトル)の組で表現する。初期状態(制約が何も得られていない状態)は(最上位のノードの識別子 1, 識別子 1 に対応するビットベクトル 0)の組で表される。制約  $c/c_3$  が得られた後の状態は、制約に対応する識別子 123(制約と表 2 の対応から得られる)と対応するビットベクトル 010(制約から得られた識別子と表 1 の対応から得られる)、および現在の状態(初期状態)を用いた、次のような演算によって計算される。

**識別子間の演算** 一方がもう片方を接頭部分文字列(数字列)として含む場合、その長い方の文字列を返す。

**ビットベクトル間の演算** 短い方のベクトルの末尾に 1 を付け加えて長い方のベクトルの長さに合わせた後、2 つのビットベクトルのビットごとの連言をとったベクトルを結果として返す。

識別子間の演算では、弁別ネットワーク上で一方のノードからもう片方のノードへ可到達であるかどうかを調べる。図 3 から明らかなように、あるノードから弁別ネットワーク上で可到達なノードは、その識別子が部分文字列関係になっている。たとえば、ノード 12 からは、いくつかの制約を満足することによりノード 121, 1232 へ到達することができるが、どのような制約を満足してもノード 112 へ到達することはできない。一方から他方へ到達可能な場合、より下位にあるノード(文字列として長い方)を結果として返す。これは、得られた制約を満足することにより、弁別ネットワークを下向きにたどることに相当する。

このことから、解析は、識別子に関して一方がもう片方を接頭部分文字列として含まない場合に失敗する。たとえば、図 3 でノード 11 まで到達しているときに、制約  $c/c_2$  が得られても、これ以上ネットワークをたどれず、解析に失敗する。これは、ノードの識別子 11 が、制約  $c/c_2$  に対応する識別子 122 の接頭部分文字列になっていないからである。

ビットベクトル間の演算は、制約の入力される順序が非決定的であることに対処するためのものである。弁別ネットワーク中の最上位ノードから到達するノードまでの間に存在する制約をビットベクトル形式で表現することにする。ビットが 1 であることがその位置の制約が満足されていないことを表しているので、制約が何も得られていない初期状態ではビットベクトルは一番左の 1 桁を除きすべて 1 である<sup>3</sup>。しかし、処理を開始した時点(初期

状態)では、到達するノードはわからないので、ベクトルの長さは不明である。そのため、処理が進み制約が得られるごとにそれに合わせてビットベクトルの長さを伸ばしていく方法を取り、ビットベクトルは最初 1 桁の 0 とする。ビットベクトルに関する演算中、ベクトルの末尾に 1 を付け加えてベクトルの長さを調節する部分はこれを実現したものである。

表 1 中のビットベクトルは、対応する識別子と同じ桁数で、その一番右の桁が 0 である<sup>4</sup>。これは、そのビットベクトルに対応する識別子の表 2 における対応制約が満足されたことを表している。たとえば、ビットベクトル 00 をもつ識別子 12 は制約  $a/a_2$  を、ビットベクトル 0110 をもつ識別子 1232 は制約  $b/b_2$  を、それぞれ満足する。対応する制約の位置が 0 であるこのようなビットベクトルとビットごとの連言をとることにより、制約が満足されるごとに対応するベクトルのビットは 0 に変化する。ビットベクトルがすべて 0 であることは、制約がすべて満足されたことを表しており、この場合到達するノードまで無条件にたどることができる。ビットの連言をとり 0 に変化する演算は、任意のビットから任意の順序で実行可能であるから、制約の得られる順序の任意性に対処できる<sup>5</sup>。

上で述べた演算により、制約  $c/c_3$  が得られた後の状態は (123, 010) になる。ビットベクトル 010 から識別子 12 に対応する制約  $a/a_2$  がまだ満足されていないことがわかる。したがって、状態 (123, 010) は、「制約  $a/a_2$  が満足されれば、弁別ネットワークをノード 123 までたどれる」ことを意味する。

次に、制約  $b/b_1$  には、対応する識別子、ビットベクトルの組が複数存在する ((111, 010) と (1231, 0110)) ので、それぞれについて現在の状態との演算を行なう。(111, 010) の方は、識別子同士が接頭部分文字列関係にないので解析に失敗する。したがって、演算結果は、(1231, 0110) に対するものだけを考えればよい。具体的には、識別子 123 と 1231 から 1231 が得られ、ビットベクトル 0101(長さを合わせるために末尾に 1 が付加されている)と 0110 のビットごとの連言から 0100 が得られる。ビットベクトル 0100 は、依然識別子 12 に対応する制約  $a/a_2$  が満足されていないことを示している。

最後に、制約  $a/a_2$  が得られると、(1231, 0100) と (12, 00) の間で演算が行なわれ、結果として (1231, 0000) が得られる。ビットベクトルがすべて 0 であることから、満足されていない制約がなく、したがって弁別ネットワークをノード 1231 まで無条件にたどれることがわかる。

任意の順序で入力される制約に沿って弁別ネットワーク

<sup>3</sup> 一番左のビットは、識別子と桁数を合わせるためのもので、対応する制約は存在しない。

<sup>4</sup> 一番左の桁が 0 である理由はすでに述べた。

<sup>5</sup> 弁別ネットワーク上で制約の本来の順序に制約が得られた場合、ビットベクトルは左から順々に 0 に変化する。

```

take:
する::          持っていく::
  [subj $ isa:human]      [subj $ isa:human]
  [obj  $ isa:action]      [obj  $ isa:thing]
                           [to   $ isa:direction;
                           isa:location]

撮る::          食べる::
  [subj $ isa:human]      [subj $ isa:human]
  [obj  $ isa:photograph] [obj  $ isa:food]

乗っ取っていく::  乗っ取る::
  [subj $ isa:human]      [subj $ isa:human]
  [obj  $ isa:vehicle]     [obj  $ isa:vehicle]
  [to   $ isa:direction;   [with $ isa:weapon]
  isa:location]

乗っていく::     連れていく::
  [subj $ isa:human]      [subj $ isa:road;
                           isa:vehicle]
  [obj  $ isa:vehicle]     [obj  $ isa:human]
  [to   $ isa:direction;   [to   $ isa:location;
  isa:location]           isa:direction]

襲う::
  [subj $ isa:disease]
  [obj  $ isa:human].

```

図 4: 動詞 'take' の辞書記述

をたどる手法について述べた。本手法の識別子間の演算は、入力された2つのノードに関して、「互いに到達可能であるかどうかを検査し、到達可能な場合には下位ノードを結果として返す」という意味で、上位/下位階層上に拡張したユニフィケーション [4, 12] と考えられる。また、本手法は、制約とそれを満足する識別子の組を与え、制約が得られるごとに、識別子間の拡張ユニフィケーションを行なうことにより識別子の選択肢を減らし、探索空間を段階的に小さくしていく手法であり、制約プログラミング [5] の考え方に基づいている。

一般化弁別ネットワークは、現在 Sun-3 上の Sicstus-Prolog, X-Window 環境で実現されている。その処理過程を付録に示す。付録は、制約 *to/direction; location, subj/human* が順に入力され、ノード *s1111, s1131* に条件付きで到達可能であることを表している。白抜きのノードは到達可能なノードを、点線の枝は満足された制約を表す。付録に示した弁別ネットワークは、図 4 のような動詞の辞書記述を変換することにより得られる<sup>6</sup>。

<sup>6</sup> 現在の版では、動詞の語義に当たるノードのラベルは表示していない。

#### 4 おわりに

あらかじめ定められた順序とは全く無関係に任意の順序で弁別ネットワークをたどる手法について述べた。この手法は、弁別ネットワークを用いた増進的曖昧性解消の実現と考えられる。この手法は、制約プログラミングの考え方に基づいており、拡張ユニフィケーションにより実現されている。

制約の得られる順序の非決定性に関して、従来はネットワークをラティス化することによる解決が試みられていたが、本手法を用いれば、この問題に対しネットワークのままに対処することが可能である。本手法は、ラティス化した場合と同等の能力をもつが、ラティス化した場合の問題点である記憶容量の増大は引き起こさない。

一般化弁別ネットワークを用いた増進的曖昧性解消により、文を解析している過程で得られる格要素の情報から、文の後方にある未解析の動詞の意味をある程度推測できる可能性がある。動詞辞書から動詞全体の意味に関するただ1つの弁別ネットワークを構築できるなら、解析過程で段階的に得られる名詞句の意味と格情報を用いてその弁別ネットワークをたどることで、動詞自体を解析する以前に、動詞の意味をある程度推測することが可能であるからである。したがって、文の後方にある動詞の意味が得られてはじめて文の意味を計算するのではなく、名詞句が得られるごとに段階的に文の意味をも推測していくことが可能になる。

また、機械翻訳における動詞の意味による訳し分けなどにも一般化弁別ネットワークは応用可能であると考えられる。さらに、プロダクションルールのコンパイル [6] も一般化弁別ネットワークを用いて可能であると考えられる。以上の応用について一般化弁別ネットワークを適用し、その有効性を具体的に実証していきたい。

今後の課題としては、ネットワークを下向きにたどるのに用いた制約に誤りがあった場合に、ネットワークをたどった経路を後戻りし、正しい経路を修復する機構を用意することが挙げられる。

#### 参考文献

- [1] 奥村学, 田中穂積. 自然言語解析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, 4(6), 1989.
- [2] 橋田浩一. Ai とは何でないか-情報の部分性について. *bit*, 20(8):48-60, 1988.
- [3] G. Adriaens and S.L. Small. Word expert parsing revisited in a cognitive science perspective. In S.L. Small, G.W. Cottrell, and M.K. Tanenhaus, edi-

tors, *Lexical Ambiguity Resolution : Perspectives from Psycholinguistics, Neuropsychology, and Artificial Intelligence*, pages 13-43, Morgan Kaufmann Publishers, 1988.

- [4] K. Dahlgren and J. McDowell. Kind types in knowledge representation. In *Proc. of the 11th International Conference on Computational Linguistics*, pages 216-221, 1986.
- [5] M. Dincbas. Constraints, logic programming and deductive databases. In *Proc. of the France-Japan Artificial Intelligence and Computer Science Symposium 86*, pages 1-27, 1986.
- [6] C.L. Forgy. Rete: a fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1):17-37, 1982.
- [7] G. Hirst. *Semantic Interpretation and the Resolution of Ambiguity. Studies in Natural Language Processing*, Cambridge University Press, 1987.
- [8] P.S. Jacobs. Concretion: assumption-based understanding. In *Proc. of the 12th International Conference on Computational Linguistics*, pages 270-274, 1988.

- [9] S.L. Lytinen. Are vague words ambiguous? In S.L. Small, G.W. Cottrell, and M.K. Tanenhaus, editors, *Lexical Ambiguity Resolution : Perspectives from Psycholinguistics, Neuropsychology, and Artificial Intelligence*, pages 109-128, Morgan Kaufmann Publishers, 1988.
- [10] C.S. Mellish. *Computer Interpretation of Natural Language Descriptions*. Ellis Horwood, 1985.
- [11] G.D. Moerdler and K.R. McKeown. Beyond semantic ambiguity. In *Proc. of the 7th National Conference on Artificial Intelligence*, pages 751-755, 1988.
- [12] J.F. Sowa. *Conceptual Structures : Information Processing in Mind and Machine*. Addison-Wesley, 1984.
- [13] W.A. Woods. Taxonomic lattice structures for situation recognition. In *Theoretical Issues in Natural Language Processing 2*, pages 33-41, 1978.

