論　文

Technical Papers

Discrimination Network 上での増進的曖昧性解消について

Towards Incremental Disambiguation with a Generalized Discrimination Network

奥村　学*,†　田中　穂積*
Manabu Okumura　Hozumi Tanaka

* 東京工業大学工学部情報工学科
Dept. of Comput. Sci., Faculty of Eng., Tokyo Institute of Technology, Tokyo 152, Japan.

1990年4月24日　受理

Keywords: natural language analysis, word sense ambiguity, incremental disambiguation, discrimination network, constraint programming.

Summary

Semantic disambiguation is a difficult problem in natural language analysis. A better strategy for semantic disambiguation is to accumulate constraints obtained during the analytical process of a sentence, and disambiguate as early as possible the meaning incrementally using the constraints. We propose such a computational model of natural language analysis, and call it the 'incremental disambiguation model.' The semantic disambiguation process can be equated with the downward traversal of a discrimination network. However, the discrimination network has a problem in that it cannot be traversed unless constraints are entered in an a priori-fixed order. In general, the order in which constraints are obtained cannot be a priori fixed, so it is not always possible to traverse the network downward during the analytical process. In this paper, we propose a method which can traverse the discrimination network according to the order in which constraints are obtained incrementally during the analytical process. This order is independent of the 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[1]手法を提案している[2]．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[3]，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，例えば，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．逆に，十分な制約が得られていない時点でむやみに決定を行うと，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制約を増進的に蓄積し，できる限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 net-

† 現在，北陸先端科学技術大学院大学情報科学研究科情報処理学専攻．

work（弁別ネットワーク）[4]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，意味間に存在する関係を考慮した表現形式である．また，（複数の意味が存在する）曖昧性を表現する際に，選択肢として複数の候補を持つのではなく，（それらをすべて下位ノードとして包含する）一つの曖昧な表現（ノード）として持つことができる．
- 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良い．

弁別ネットワークを下向きにたどる過程は，曖昧な単語の意味がしだいに洗練され，候補数が徐々に少なくなる過程に対応する．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はあらかじめ定められたとお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，制約の得られる順序が定められたとおりでない場合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，任意の順序で制約が得られても弁別ネットワークをたどることができる手法を提案する．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章で詳述するように，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章では，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章では，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章では，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

Fig. 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．Fig. 1では，'take' がS (Subject)，O(Object)，with，toなどの格を取り得ることが記述されている．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）された語義を表す．ネットワークの葉ノードは曖昧性のない一つの語義を表す．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）に到達し得ることから，到達し得る語義をすべて含んだ曖昧な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，得られた情報が満足する制約の枝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

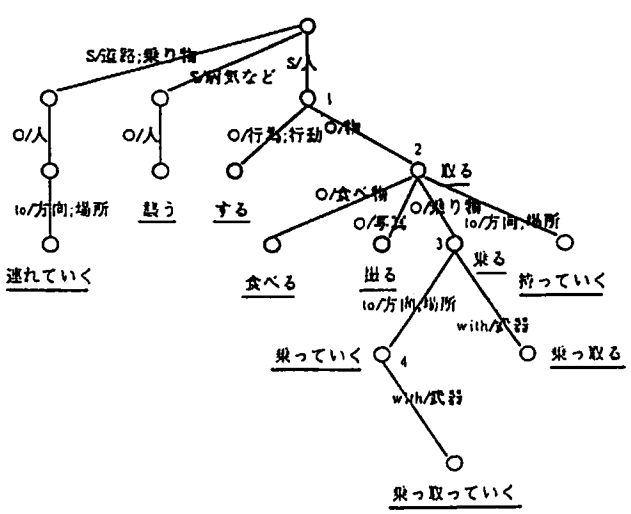

Fig. 1 A portion of the discrimination network of the word senses of the verb 'to take'.

る．この過程で，文中の回りの単語の情報（動詞が取る格要素の情報）から，異常な選択肢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．

Fig. 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，[S/John, O/plane, to/London]が得られる．これらの制約に基づき，Fig. 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」(';'は選言を表す)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，文献(5)-(8)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，次のようなものが述べられている．

- 単語の意味の曖昧性に関しては，Hirstのポラロイド語[9]のように，従来単語の複数の意味を互いに無関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捉えられない．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どうしをネットワーク上で離れた位置に，共通性の多い意味どうしを近くに表現することが可能である．また，(複数の意味が存在する)曖昧性を表現する際に，選択肢として複数の候補を持つのではなく，(それらをすべて下位ノードとして包含する)一つの曖昧な表現（ノード）として持つことができる．そして，弁別ネットワークを下向きにたどる過程は，曖昧な単語の意味がしだいに洗練され，より具体的な意味として明らかになる過程と対応する[5]．
- 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中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．この探索は，候補数（リストの長さ）をnとすると，$O(n)$の時間を要する．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[7][10]．この探索は$O(l)(=O(\log n))$の時間で行える[11]．ここで，lは木の高さである．

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から初めて一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[7]．

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うのに都合の良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 Fig. 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に示したが，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しく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とお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 Fig. 1 の弁別ネットワークを正しく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，次に入力されるはずの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，デッドロックに陥ってしまう．

これに対し，格要素（制約）がすべて得られた後，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．これは文献(2)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，Fig. 1 のように英語を対象言語とするなら，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[4]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[8]．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度合い，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[12] 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，その名のとおり，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，計算機上で実現する場合に非常に大量の記憶容量が必要になる．

次章では，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．解析過程で増進的に，任意の順序で得ら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持つが，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

Fig. 2 の弁別ネットワークを考える．Fig. 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．根ノードには 1 を割り当てる．根ノードの 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 は子ノードの個数）を割り当てる．以下順に，ノード $1i_1i_2\cdots i_m$ に対して，その子ノードには $1i_1i_2\cdots i_mi$（ここで，i は 1 から n までの整数，n は子ノードの個数）を割り当てる．Fig. 2 のノードには Fig. 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，識別子と同じ長さで，先頭と末尾の桁が 0 でそれ以外の桁が 1 のビットベクトルを対応させる（Fig. 3 の識別子については Table 1 のようになる）．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

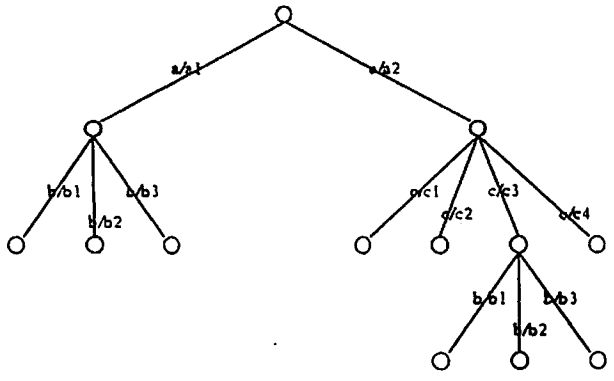

Fig. 2 A sample discrimination network.

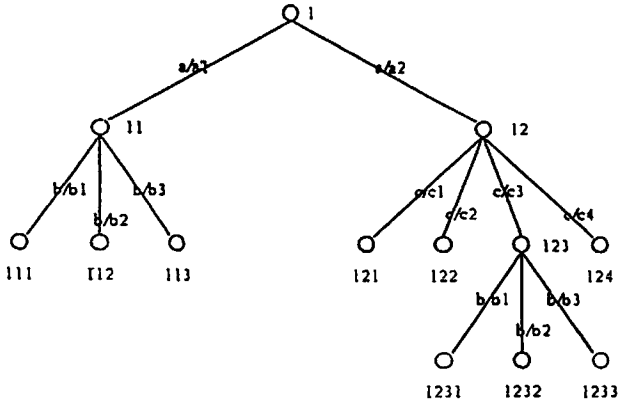

Fig. 3 Discrimination network with identifier-assigned nodes.

を表す．例えば，識別子 111，122 は 010 というビットベクトルを持つが，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（ノードの識別子と制約の対応関係は，後述するように，Table 2 から得られる）a/a_1, a/a_2 がそれぞれ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 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，その枝と直接つながった下位ノードの識別子の組を抜き出す．この際，識別子と Table 1 により対応するビットベクトルを同時に抜き出す．Fig. 3 のネットワークからは Table 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, 010), (1231, 0110)} 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，Table 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

Table 1 Correspondence between node identifier and bit vector.

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

Table 2 Correspondence between constraint and node identifier.

a/a_1	(11,00)
a/a_2	(12,00)
b/b_1	{(111, 010), (1231, 0110)}
b/b_2	{(112, 010), (1232, 0110)}
b/b_3	{(113, 010), (1233, 0110)}
c/c_1	(121,010)
c/c_2	(122,010)
c/c_3	(123,010)
c/c_4	(124,010)

別子のノードに到達できることを意味する．例えば，制約 c/c_2 が満足される場合，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．

この際，同時に抜き出した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「条件つき」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．Table 2 で制約と対応する，(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）の組を以後「状態」と呼ぶことにする．

Fig. 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ときに，本手法でどのように弁別が進むか，その過程について以下では説明する．処理には，Table 2 に示した，制約と状態間の対応表を用いる．また，処理過程は状態で表現する．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，制約に対応する状態(Table 2 から得られる，識別子 123 とビットベクトル 010 の組)，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，その長いほうの文字列を返す．

ビットベクトル間の演算　短いほうのベクトルの末尾に1を付け加えて長いほうのベクトルの長さに合わせた後，二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．Fig. 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には，識別子が部分文字列関係になるように割り当てられている．例えば，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（文字列として長いほう）を結果として返す．これは，得られた制約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．例えば，Fig. 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，制約の得ら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すので，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（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）．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とり，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．

Table 2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（一番左の桁が0である理由はすでに述べた）．これは，そのビットベクトルに対応する識別子のTable 2における対応制約が満足されたことを表している．例えば，ビットベクトル00を持つ識別子12は制約 a/a_2 を，ビットベクトル0110を持つ識別子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に変化させる演算は，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（弁別ネットワーク上での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右に順々に0に変化する）．

制約に対応する状態 (123, 010) と初期状態 (1, 0) から，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 (123, 0110)) ので，それぞれについて現在の状態との演算を行う．(111, 010)のほうは，識別子どうし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．

ここで示した弁別ネットワークをたどる手法が，通常の弁別木の動作とは以下の点で異なることに注意していただきたい．我々の手法では，制約集合 $\{c/c_1, a/a_1, b/b_1\}$ が得られた場合，それらの制約をすべて用い

てネットワークをたどることができないため，どのノードにも到達できない．一方，通常の弁別木では，制約 c/c_1 を不要な情報として，たどる際に用いないことにより，(少なくとも)ノード 111 に到達することができる．通常の弁別木では，入力された制約を 'don't care' として無視する場合があるのに対して，我々の手法では，入力された制約はすべてネットワークをたどるのに用いる．

本論文で述べたような自然言語解析に弁別ネットワークを用いる場合には，我々の手法をとるのが妥当であると考える．Fig. 1 に示したように，弁別ネットワークで，動詞の格フレーム集合を表現した場合，解析する文から得られる格要素の情報が入力される制約となる．入力文から得られる情報はすべて有用な情報であり，解析結果にすべて反映されなければならない．そのため，通常の弁別木のように，任意の制約を 'don't care' として無視することを許すと，文中に余分な格要素を含む非文も受理してしまう．

必須格要素のみで弁別ネットワークを構成した場合，任意格要素は 'don't care' な制約と考えられるが，これは必須格要素と任意格要素を分離し，必須格要素についてのみネットワークをたどるための制約として用い，任意格要素については別途解析を行うことで対処できると考える（ただし，必須格と任意格の厳密な分類は難しいと考えられる）．

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，入力された二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[13][14]と考えられる．また，本手法は，制約と対応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うことにより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[15]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[16]．

4. おわりに

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，あらかじめ定められた順序とは全く無関係に，任意の順序で制約が得られても弁別ネットワークをたどることができ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で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持つが，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一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初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集合（知識ベース）のコンパイル[17][18]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．

◇参考文献◇

(1) Mellish, C. S.: *Computer Interpretation of Natural Language Descriptions*, Ellis Horwood (1985).

(2) 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, Vol. 4, No. 6, pp. 687-694 (1989).

(3) 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, pp. 48-60 (1988).

(4) Charniak, C. K., Riesbeck, E. and McDermott, D. V.: *Artificial Intelligence Programming*, Lawrence Erlbaum Associates (1980).

(5) Jacobs, P. S.: Concretion: assumption-based understanding. In *Proc. of the 12th Int. Conf. on Computational Linguistics*, pp. 270-274 (1988).
(6) Moerdler, G. D. and McKeown, K. R.: Beyond semantic ambiguity. In *Proc. of the 7th National Conf. on Artificial Intelligence*, pp. 751-755 (1988).
(7) Lytinen, S. L.: Are vague words ambiguous? In Small, S. L., Cottrell, G. W. and Tanenhaus, M. K. eds. *Lexical Ambiguity Resolution: Perspectives from Psycholinguistics, Neuropsychology, and Artificial Intelligence*, pp. 109-128, Morgan Kaufmann Publishers (1988).
(8) Adriaens, G. and Small, S. L.: Word expert parsing revisited in a cognitive science perspective. In Small, S. L., Cottrell, G. W. and Tanenhaus, M. K. eds. *Lexical Ambiguity Resolution: Perspectives from Psycholinguitics, Neuropsychology, and Artificial Intelligence*, pp. 13-43, Morgan Kaufmann Publishers (1988).
(9) Hirst, G.: Semantic Interpretation and the Resolution of Ambiguity, *Studies in Natural Language Processing*, Cambridge University Press (1987).
(10) 奥村 学, スラパン・メークナウィン, 田中穂積：半順序関係をもつ制約と弁別ネットワーク, Technical Report, ICOT (1990) (予定).
(11) Aho, A. V., Hopcroft, J. E. and Ullman, J. D.: *Data Structures and Algorithms*, Addison-Wesley (1983).
(12) Woods, W. A.: Taxonomic lattice structures for situation recognition. In *Theoretical Issues in Natural Language Processing*, pp. 33-41 (1978).
(13) Dahlgren, K. and McDowell, J.: Kind types in knowledge representation. In *Proc. of the 11th Int. Conf. on Computational Linguistics*, pp. 216-221 (1986).
(14) Sowa, J. F.: *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley (1984).
(15) Dincbas, M.: Constraints, logic programming and deductive databases. In *Proc. of the France-Japan Artificial Intelligence and Computer Science Symposium 86*, pp. 1-27 (1986).
(16) 奥村 学, スラパン・メークナウィン, 田中穂積：Discrimination network 上での増進的曖昧性解消について, 情処学自然言語処理研報, 75 (1990).
(17) Forgy, C. L.: Rete: a fast algorithm for the many pattern/many object pattern match problem, *Artif. Intell.*, Vol. 19, No. 1, pp. 17-37 (1982).
(18) Highland, F. D. and Iwaskiw, C. T.: Knowledge base compilation. In *Proc. of the 11th Int. Joint Conf. on Artificial Intelligence*, pp. 227-232 (1989).

〔担当編集委員：白井英俊, 査読者：鈴木浩之〕

著者紹介

奥村 学（正会員）

1962年生まれ, 1984年東京工業大学工学部情報工学科卒業. 1989年同大学院博士課程修了. 同年, 東京工業大学工学部情報工学科助手. 1992年北陸先端科学技術大学院大学情報科学研究科情報処理学専攻助教授, 現在に至る. 工学博士. 自然言語理解, 知識表現に関する研究に従事. 情報処理学会, 日本ソフトウェア科学会, 日本認知科学会, AAAI, 計量国語学会各会員.

田中 穂積（正会員）

1964年東京工業大学理工学部制御工学科卒業. 1966年同大学院修士課程修了. 同年, 電気試験所（現, 電子技術総合研究所）入所. 1983年東京工業大学工学部情報工学科助教授. 1986年同大学教授. 工学博士. 人工知能, 自然言語処理の研究に従事. 情報処理学会, 電子情報通信学会, 日本認知科学会, 計量国語学会各会員.