日本語ＤＢ検索システムＱｕｅｓｔにおける意味解析

加藤恒昭，中川　優
（日本電信電話公社　横須賀電気通信研究所）

1. はじめに

データベース（ＤＢ）技術の発展、普及に伴い、ＤＢ検索を日常的な業務としていないユーザによる検索が増加している。しかも、その検索は従来に較べ、非定型かつ複雑なものになりつつある。我々は、そのような検索のニーズに答えるため、検索言語のシンタックスやＤＢ構成などを意識することなく、容易かつ自然にＤＢ検索を行なえるような日本語によるＤＢ検索システムＱｕｅｓｔ(intelligent QUEstion answering SysTem) の開発を進めている。

自然言語によるＤＢ検索システムとしては、既に国内外に幾つかの提案がなされているが、その多くは、関係ＤＢ（ＲＤＢ）と自然言語との対応を直接的に行なっており、自然な検索という面からはまだ不充分な点が多い[1][2][3] 。Ｑｕｅｓｔでは、この対応づけにＤＢ世界表現モデルと名付けた表現能力の高いモデルを利用し、より自然な言い回しを理解できるようになっている。

また、Ｑｕｅｓｔ開発において特に重点を置いた点はシステムのportability である。ここで、portability とは以下のような条件を満すことを言う。

・対象となるＤＢに関する情報がシステム内で局在化しており、その変更、作成が容易であること。
・対象とするＤＢＭＳに依存している部分が局在化していること。
・対象分野に固有な言い回しがある場合、その解析処理を容易に組み込めること。

Ｑｕｅｓｔではこれらの要求を以下の手法によって解決している。

・対象ＤＢに依存する部分を基本的にはＤＢ世界表現モデルに集約した。
・中間表現として、アルファ式[4]に近い論理式を採用し、対象検索言語を意識する部分を限定した。
・意味解析処理がルール集合の形をしており特殊な処理はそれに応じたルールとして追加を可能とした。

本資料では主にＱｕｅｓｔの核であるＤＢ世界表現モデルとそれを用いた意味解析手法について述べる。

2. システム構成

Ｑｕｅｓｔの構成を図１に示す。本システムはDEC-2060上のDEC10-Prologを用いて開発されている。本システムは大さく７つにモジュール分割できる。

・構文解析部

ローマ字ベタ書きで入力された質問文を単語辞書と文法を利用して解析し、文節の並びに展開する。このレベルでは文節間の係り受け関係を同定することは行なわない。このため、文法は形態素解析レベルの処理を行なうための汎用文法である。ただし、解析結果のalternative を減ずるため、文節間の接続関係を調べ、文法的に適切でない解析結果が生じないよう配慮している。

単語辞書は、自立語の場合には後に述べる意味世界表現モデルの要素との対応が記述されている。付属語の場合には、その文法的役割（主格を表わす格助詞等）が記述されている。

本モジュールの核部にはProlog上のBottom up parser BUP [5]を利用している。

構文解析部の出力例を図２に示す。

・ＤＢ世界表現モデル

対象とするＤＢの構成やその実世界との対応が記述されている。このモデルは意味解析部に利用される意味世界表現モデル(SWORD Semantic WORld Description model)と検索言語生成部に利用されるアクセスモデル(AM Access Model) に分割格納されている。AMは実ＤＢ中の表から意味の基本構成要素とな

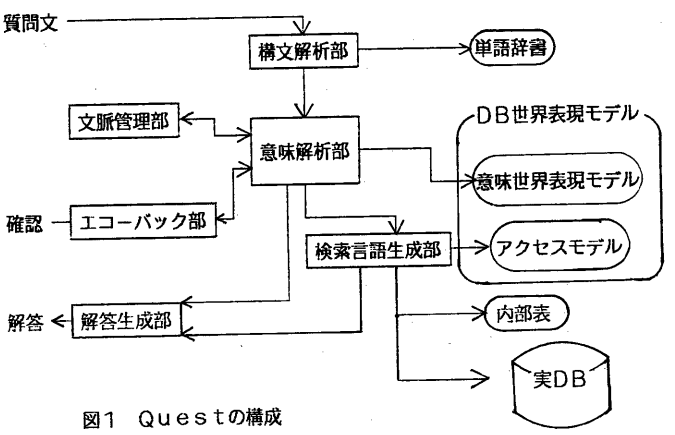
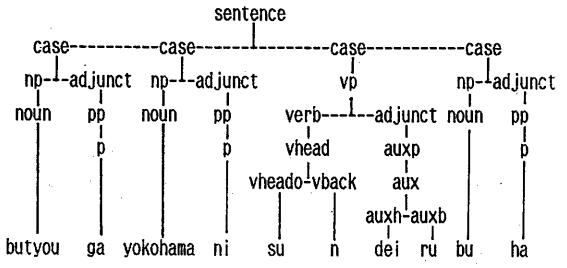


図1　Ｑｕｅｓｔの構成

入力文　部長が横浜に住んでいる部は

構文木



構文情報

```
[case(np,[adj],[attr($buchou)],[]),
 case(np,[att,atp],[entity(yokohama,#juusho)],[]),
 case(pred,34,[class(@sumu)],[contin]),
 case(np,[adj,rep],[class(*bu)],[])]
```

図2　構文解析の例

る表（仮想表と呼ぶ）を定義している。SWORD はそのような仮想表と実世界つまり日本語表現との対応を表現している。詳細については 3. で述べる。

・意味解析部

構文解析結果を基にSWORD を参照することにより、質問文の意味表現である中間表現を生成する。そのアルゴリズムについては 4. で述べる。

中間表現は仮想表に対する検索を表現するアルファ式に近い論理式となっている。アルファ式は関係論理における集合の内包的定義に相当する表現であるが、中間表現では、これを述語論理的に表現している。図３に中間表現の例を示す。ここからわかるように基本的にはPrologシンタックスを採用しており、大文字で始まるアトムは変項を表現し、',' と ';' は各々and とorを表現している。また、各述語は仮想表に対応している。

中間表現では

{[変項の並び]|条件式}

に相当する集合の内包的定義を

result([変項の並び]), 条件式

で表現している。ただし、result述語が先頭に来る必要はない。条件式中に現れ、変項の並びに現れない全ての変項には存在量記号が仮定されている。また、全称量記号に代わる表現として

all([変項の並び],述語1,述語2)

を用意した。これは述語１を真にするすべての変項並びにおいて述語２が真である時に真となる高階述語である。また、ＤＢ検索に固有なmax,min などの統計関数や出力順序の指定などについては厳密さを欠くが各々max(X,Y),min(X,Y),inc(X)と述語的に表現し中間表現に含めることとした。

・文脈管理部

意味解析部から逐次、質問文や文節の意味である中間表現を受けとって、現在とひとつ前の質問の話題を記憶しており、意味解析部が代名詞の同定や省略補完のためにそれらの情報を必要とする場合に提供するモジュールである。

意味解析部の要求している情報が保持されていないときはシステム内に定義されたデフォルト値やユーザへの問い合わせによって解決を図る。

・エコーバック部

意味解析部によって得られた質問文の解釈を入力とし、それをユーザに提示し、その解釈が適切かどうかの確認を行なう部分である。ユーザが否と答えた場合は意味解析部に対してバックトラックをかけ、解釈のalternative を捜すこととなる。

本モジュールは将来的には中間表現から疑似日本語を生成し、提示することを考えているが、現在の所、中間表現を整形したものを提示している。

・検索言語生成部

中間言語から検索言語を生成するモジュールである。Ｑｕｅｓｔは、検索の対象として外部のＤＢ実表だけでなく、システム内部に内部表と呼ばれるものを持っている。内部表はPrologプログラムとして保持されており、より自然な検索のために必要となる整理された知識やルールを実ＤＢに影響を与えることなく取り込むためのものである。

本モジュールは、まず、仮想表に対する検索表現として与えられる中間表現をAMを参照することで、実表と内部表に対する検索表現に展開する。この後、検索のスケジューリングを行ない適切な順序で、内部表と実表に対して検索コマンドを発行する。内部表への検索コマンドは中間表現に類似した形式の述

部長が横浜に住んでいる部は

```
result(A),bumei(B,A),
buchou(B,C),sumu(C,yokohama)
```

ボスより給与の多い社員を知りたい

```
result(A),syainmei(B,A),kyuuyo(B,C)
boss(B,D),kyuuyo(D,E),C > E
```

図3　中間表現の例

語で実現している。実ＤＢへの検索言語としてはＳＱＬを利用しており、その生成に関しては検索効率を考慮して、表結合と副照会の適切な使い分けを行なっている。

・解答生成部

Ｑｕｅｓｔは従来のＤＢ検索システムでは扱っていなかったyes-no型の疑問文をもその対象としているため、検索結果から質問文の真偽を判定する処理が必要となる。この処理と検索結果の出力を行なうのが本モジュールである。

検索結果の出力は表イメージで行なわれ、yes-no型の場合は判定の根拠となる値が、wh型の場合は、ユーザが陽に求めた検索項目とそれを一意識別するための情報が出力される。

3. ＤＢ世界表現モデル

ＤＢ世界表現モデルはＤＢもしくはそれが対象としている世界の意味を記述するためのものである。

ＤＢ技術の分野でも、ＤＢの自然な設計を目的としてこのようなデータモデルの提案が幾つかなされている。Ｑｕｅｓｔで利用しているＤＢ世界表現モデルはこれらの中で、SDM [6] 、DAPLEX [7]、E-Rモデル[8] を基礎として、日本語の表現や省略がモデル構造に対応するよう拡張整理したものである。

3.1 ＤＢ世界表現モデルの要素

ＤＢ世界表現モデルの基本構成要素は、entity, class,attribute である。

entityは、世界内に存在する「ものごと」を表現している。例えば、「加藤という名前の社員」という個人、「加藤が営業部に属す」という状態、更に、「加藤」という名前そのものなどがentityとして扱われる。これらentityはその性質によって３種類に分類される。

・もの(object)　人や建物などの具体的な物
・こと(event)　所属や存在などの出来事や状態
・名前(name)　「横浜」、「加藤」などの識別子

class はこれらentityの集合であり、共通の特徴を持ったentityが集められている。class はそこに属すentityによって以下のように分類される。

・oc(Object Class)　ものを要素とするclass
　社員、部など
・ec(Event Class)　ことを要素とするclass
　属す、住むなど
・nc(Name class)　名前を要素とするclass
　地名、社員名など

attribute はclass の要素であるentityの特徴を表わすためのもので、あるentityから同じもしくは異なるclass のentityもしくはその集合への写像である。例えば、部長attribute は部class のentityを特徴づけるもので、その値として社員class のentityをとる。

attribute が定義されているclass を定義class その値となるentityを要素とするclass を値classと呼ぶ。部長attribute の場合、定義class は部、値class は社員である。

attribute には、class におけるその役割によって特別な名前を与えられているものがある。

・pa(Primary Attributes)

このattribute の値によってclass のentityが一意識別できるようなattribute もしくはその集合

・da(Default Attribute)

日本語によって名指しできるのは、ncのentityのみであるが、一般にはocのentityはそこに定義されているdaの値によって名指しされる。例えば、「加藤」は単なる名前であるが、これによって、名前attribute の値が「加藤」であるoc社員のentityを表現している。この名前attribute をdaと呼ぶ。daの定義class は常にocで、その値class はncである。

・fa(Functional Attribute)

ecに定義されるattribute の中で、そのecを定義class とする他のattribute の値class が定義class で、値class がそのattribute の値class であるようなattribute としてふるまうものを指す。

以上がＤＢ世界表現の基本構成要素である。これらに加えて基本構成要素間の関連や基本要素から導出される要素が定義される。そのようなものとして、subclass,grouping,導出attribute がある。

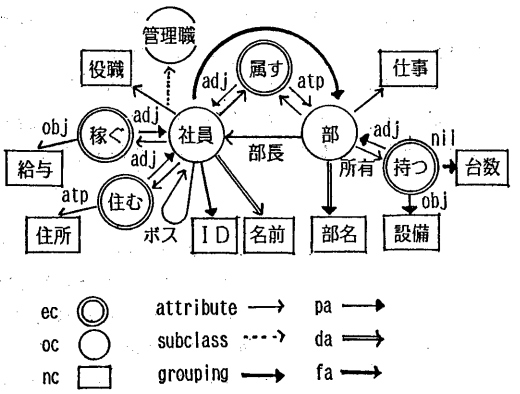


図4　SWORD ダイアグラム

subclassはあるocのentityに条件を課してそれを満すentityのみからなる部分集合である。subclassはその基となったclass のattribute を全て受けつぐ。例えば、oc社員からその役職attribute の値が課長以上という条件で、そのsubclass管理職を作ることができる。

grouping関連は他のclass のentityによってあるclass のentityがグループ分けされているような時、換言すれば、あるclass のentityが他のclass のentityの集合としての側面を持っているときに定義される。例えば、社員は彼が属す部によってグループ分けされるからoc部とoc社員の間にはgrouping関連が定義される。

導出attribute は複数のattribute を合成することによって得られるattribute である。例えば、ボスというattribute は社員が属す部の部長という形で定義できる。

3.2 ＤＢ世界表現モデルの表現

ＤＢ世界表現モデルは意味解析に利用される述語の定義を行なっていると考えることができる。この述語は先にも述べたように仮想表と呼ばれる。ＤＢ世界表現モデルの要素は述語と以下のように対応する。

・ecはそこに定義されているattribute に対応する引数を持った述語となる。
・値class,定義class が共にecでないattribute は導出attribute か否かにかかわらず、２引数の述語となる。

これらの引数の定義域は全てそのclass によって決定される。

ＤＢ世界表現モデルはこのように定義された仮想表と日本語表現とを対応づけるSWORD と実表および内部表との対応をとるAMから構成される。

SWORD に記述される情報は以上で述べたモデル構成と各要素の名前である。この名前が辞書に記述されている意味情報となる。ただし、SWORD のレベルでは、導出attribute は一般のattribute と区別されない。また、ecは日本語における動詞に対応し、

```
trans(boss(A,B),(#syain(A,-,C,-,-,-,-),#bu(C,B,-)))
                                        -は無名変項
```

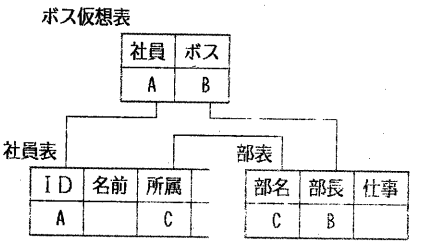


図６ アクセスモデル内部表現

そこに定義されたattribute はその格に対応するため格標識が付けられる。このダイアグラム表現を社員情報を例に図４に示す。

Ｑｕｅｓｔ内部においてはSWORD をこのダイアグラムのリンク単位にfact形式で表現している。その例を図５に示す。

仮想表は実表の写像や結合によって得ることができる一種のビューであるから、その定義をAMに記述している。AMは仮想表単位にfact形式で表現されており、これを参照することで、仮想表に対する検索表現を実表に対する検索表現に変換できる。AMの例を図６に示す。

4. 意味世界表現モデルと意味解析

質問文に現れる自立語にはその意味としてSWORD の要素が以下のように対応している。

・用言　ecのclass 名
・名詞　oc,ec,ncのclass 名
　　attribute 名
　　ncのentity

質問文中の名詞句にはSWORD のentityが対応づけられ、文には、真偽値が対応する。ただし、質問文中の文は埋め込み文となって、名詞句に係ることが多く、これによって生成された名詞句にはentityが

```
attr(type(A,*bu),type(B,*syain),$buchou,buchou(A,B),naux,no-omit,single,nil).
attr(type([A,B],@zokusu),type(A,*bu),nil,zokusu(A,B),atp,omit,single,pa).
attr(type([A,B],@zokusu),type(B,*syain),nil,zokusu(A,B),adj,omit,single,nil).
subclass(type(A,&kanrisyoku),*syain,(yakusyoku(A,B),(B=kachou;B=buchou))).
grouping(type(A,*bu),type(B,*syain),zokusu(A,B),[$sigoto]).
```

図５ SWORD 内部表現

対応する。名詞句を修飾する文は、その中にentityを表現する自由変項を含んでおり、修飾される名詞句の意味となるentityはその自由変項と同じclassに属している。

これら、質問文の部分的意味を表現するために以下に示す途中表現を利用する。

type(X,CLASS):条件式

これは、条件式を満すようなclass CLASS のentity Xを意味する。条件式は必ずX を含んでおり、そこに現れるX 以外の変項のあるものには存在量記号が仮定されている。図７に日本語とSWORD と中間表現との基本的な対応を示す。

意味解析で行なわれる処理は、単語の意味から始まって、名詞句の意味、文の意味と合成してゆき、最終的に質問文全体の意味を得ることに他ならない。

質問文の各部分の意味を上記のようにentityを表現する変項と考えると、その処理は変項どうしの関係づけを行なうことに相当する。

Ｑｕｅｓｔでは、このような処理を係り受け関係の決定ととらえている。各文節はそのタイプや付属語に応じて、各々係りの役割と受けの役割を有している。これは表１のようにまとめられる。

Ｑｕｅｓｔの意味解析の基本的処理はSWORD を参照することで、係り受けのチェックを行ないながら、各単語の役割に応じた処理を行なうことである。

5. 意味解析部の構成と動作

5.1 構成

意味解析部は図８に示すような構成を持っている。本構成は一種のプロダクションシステムと見ることもできるし、turing機械と考えてもよい。

入力となる構文解析結果は文節単位のセルを持つ入力テープであり、これをp-infoと呼ぶ。w-stack

表1　係りの役割とその受けの分類

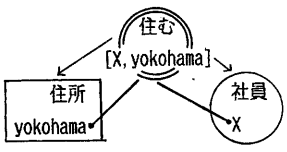
係り	役割	受け
名詞句＋格助詞	格要素	用言（文）
名詞句＋連体修飾助詞	名詞句の連体修飾	名詞句
名詞句＋並列助詞	名詞句の並列	名詞句
連体文	文の連体修飾	名詞句
連用文	文の並列	連体文

は係りの情報が入れられるスタックである。具体的には、先に述べた途中表現のtype(X,CLASS) と係りの役割が入れられている。s-infoは解析結果である中間表現が格納されるが、解析の途中では、先に述べた途中表現の条件式が基本的には","(and)で結合されて入れられている。u-infoには解析途中で得られた自由変項がtype(X,CLASS) の形式で格納されている。これは埋め込み文の解析に利用され、その内容は文の解析が終了した時点でw-stack に移される。o-infoは意味解析部が後処理として必要とする処理が何であるかを記憶している部分であり、compositionalityが保たれない幾つかの処理で利用される。SWORD 探索部は、動作部の要求に応じて、SWORD を探索し、必要とする情報を与える。

エンジン部の基本的な動作は、次に示すサイクルの反復である。

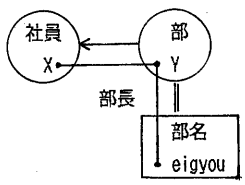
- p-infoとw-stack の状態から適当なルールを選ぶ。
- ルール適用が可能であるかをSWORD 探索部に問い合せ、可能であれば必要な情報を得る。実際には、Prologのunification を利用しているので、この２つの処理は同時に実現される。
- ルールのaction部に従って、p-info,s-info,w-stack,u-info,o-info などの変更を行なう。

動作部の停止条件はp-infoを最後まで読み取り、



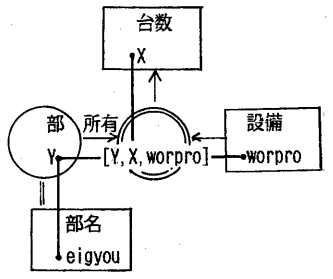
横浜に住む社員

type(X,*syain):sumu(X,yokohama)



営業部の部長

type(X,*syain):bumei(Y,eigyou),buchou(Y,X)



営業部のワープロ所有台数

type(X,#daisuu):bumei(Y,eigyou),motu(Y,X,worpro)

図７　日本語、中間表現、SWORD の対応

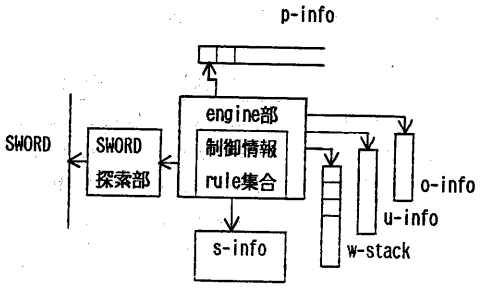


図8　意味解析部構成

かつ、w-stack が空であることであり、これ以前に適用できるルールがなくなった場合にはバックトラックを生じる。また、処理の高速化のために、ルール適用後の状態が既に生じた状態と同じであるかをcheck し、そうであれば強制的にfailして無駄な探索を減らしている。

ルールは大きく以下の３種類に分類される。

・受けのルール

w-stack の要素とp-infoの要素の間に係り受け関係が成立すれば、p-infoをその係りを受けた後の意味に変換し、w-stack の要素をpop する。更に、その関係を表現する仮想表をs-infoに加える。

・係りのルール

p-infoの先頭要素を取り除き、その意味と係りの役割をw-stack にpushする。必要があれば、u-infoを更新する。

・変型ルール

p-infoの表層的な構造の変換を行なう。例えば、「最も給与が多い」を「給与が最も多い」に変換する。このようなルールを設けることで、上記の係り受けルールの数を減すことができる。

これらのルールは、headに各作業域のルール適用前後の状態がペアで記述されており、bodyにSWORD探索部の呼び出しと必要となる情報の抽出手続きが記述されている。図９に、ルールの一例を示す。実際には、clause indexを利用するためや、デバックのための付加的な引数が加えられている。

5.2　動作例

簡単な例によって、意味解析部の動作を説明する。質問文が、「部長が横浜に住んでいる部は」であったとする。初期状態として、p-infoには図２に示した構文解析結果が入り、それ以外は全て空である。

「部長が」文節

部長attribute の値class *syainとそのentityを表現する変項B 、係りの役割が主格の格要素であることが、w-stack にpushされる。仮想表buchou(A,B) がs-infoに入れられる。ここで、A は部長attribute の定義class *bu のentityである。このA は自由変項であるから、class 情報と共にu-infoに格納される。

「横浜に」文節

同様に処理が行なわれる。

「住んでいる」文節

用言の受け処理としてsumu(X,Y) のX,Y が各々B,yokohamaにunify される。このunify と同時に、w-stack の内容がpop され、s-infoには、sumu(X,Y) が加えられる。prologのunification を利用しているため、s-infoを直接操作することなく、変項の関係づけがなされる。また、w-stack により係り受けの非交差が自然に実現できている。

係りの処理として、u-infoにあるtype(A,*bu) をw-stack にpushする。

「部は」文節

受けの処理として、文節の意味をentity(A,*bu) に変換し、w-stack の先頭をpop する。
係りのルールによって、case([adj],type(A,*bu)) がw-stack にpushされる。

終了ルールとして、w-stack の先頭要素のentityをresultの引数とunify する処理が起動されるが、このentityがncのentityでない場合には、そのdaの値class のentityがunify され、仮想表bumei(A,C) がs-infoに加えられる。最終的に得られる中間表現は図３に示したものとなる。

以上の様に意味解析の基本的動作は受けのルールによって各文節の意味であるentityとそれを制約する条件を明らかにし、係りのルールでそのentityの係りの役割を明らかにするというサイクルの反復で

```
rule([case(np,P-list,[attr(Name)|Other],[])|P-info],P-info,
     W-stack,[case(P-list,type(E2,C2))|W-stack],
     U-info,New-U-info,
     O-info,New-O-info,
     S-info,(Term,S-info))
 :-not-overlap(P-list,[or,and,para,naux]),
   check-sword(attribute,[attr(Name)|Other],
                ,O-info,New-O-info,-,type(E2,C2),Term),
   free-var(E2,(Term,S-info),U-info,New-U-info).
```

図9　ルールの例

ある。

5.3 幾つかの個別処理

(1) 省略を含んだ言い回しへの対処

質問文において、２つの要素に係り受けが可能であることはSWORD において、その２つの要素の間にリンクがあることと等価である。その最も基本的な形式は「ａのｂ」という表現において、ａがentityを意味しており、ｂがそのentityを要素とするclass を定義class とするattribute を意味しているとき、係り受けが可能で、その意味はｂの値class のentityになるというものである。例えば、「加藤が属す部の部長」という表現では「加藤が属す部」がａにあたり、「部長」がｂにあたる。

質問文に現れるのは、このような厳密な言い回しのみではないが、それらもSWORD との対応によって理解することができる。以下に例を挙げる。

・「営業部の部長」はこの形式にあてはまらない。なぜなら「営業部」は部名class のentityであって部class のentityではないからである。この場合は、daとの関係で、「営業部」の意味を名前attributeの値が「営業部」である部class のentityと解釈する。

・「加藤の部」や「加藤の住所」という表現は、上の省略補完を行なっても形式にあてはまらない。なぜなら「部」や「住所」はclass 名であってattribute を意味していないからである。この場合、そのclass 名で、ecを介した導出attribute を意味していると考える。

・「営業部のコピー機の台数」の場合は、faの名前によって省略がなされていると考える。これによってec名を補い、「営業部のコピー機の所有台数」の省略と理解する。

・「加藤の仕事」の場合は、上記の省略補完を行なっても、「加藤」は社員class のentityであり、「仕事」は部class を定義class とするattributeである。このような言い回しではgrouping関連の定義が利用される。この例では、部と社員とがgrouping関連で結ばれているから、その解釈は「加藤が属す部の仕事」となる。grouping関連に伴うもうひとつの省略は統計関数と共に生じる。「営業部の平均給与」は「営業部に属す社員の平均給与」として解釈される。

・subclassのにおけるattribute の継承も省略と同じように扱える。「営業部に属す管理職」は管理職class が社員class のsubclassであるから意味を持つ。この解釈はsubclass定義の条件を加え、「営業部に属していて、役職attribute の値が課長以上の社員」となる。subclass定義の条件が付加されるのは、この例のようにsubclass名が陽に示されたときとsubclassにのみ定義されたattribute が指定された場合である。これらがひとつのentityに重複して現れた場合にsubclass定義の条件が重複して中間表現に現れないように、subclassのentityが現れたという情報をo-info中に保存しておき、解析の後処理として条件を中間表現に加えている。

これら省略補完の処理は、そのほとんどがSWORD探索部によって行なわれ、ルールがそれを意識する必要はない。

(2) 並列の処理

質問文に現れる並列は以下の２つに分類される。

・検索項目の並列　　「加藤の住所と給与」
・条件要素の並列　　「営業部と販売部の社員」

検索項目の並列の時に問題となるのは、上の例で言えば、加藤が給与にも係ることの判定である。これを実現するため、「住所と」文節に対する係りルールでは、w-stack に

para(entity,type(X,#juusyo),[type(Y,*syain)])

という形式で、この文節に係っているentityを保存しておく。これらの情報はs-infoを探索することで得る。この後、「給与」文節で並列を受ける時点で、自由変項があれば、このw-stack 中の変項とunifyする。このような処理によって得られる名詞句の意味はtype([X,Z],[#juusho,#kyuuyo]) と表現し、result述語以外では受けられないようにする。

条件項目の並列での問題は、主にそれがand 並列であるかor並列であるかの判定である。Ｑｕｅｓｔでは、これを以下のように判定する。

・埋め込み文の格でないものはor並列
・埋め込み文の格であるものの内、その文の用言の意味となるecに定義されているattribute がpa,fa のみであればand 並列
・埋め込み文の格であるものの内、その文の用言の意味となるecにpa,fa 以外のattribute が定義されていればor並列

これによって、

「営業部と販売部の社員」　　or 並列
「ワープロとコピーを持っている部」　and 並列
「横浜と横須賀に住んでいる社員」　　or 並列

となる。この判定はand に判定されることが多少多いが、or並列は「ワープロかコピー」というように陽に指定できるため問題はないと考えている。

この判定を実現するために、並列を含んだ格は

case([p],type(para([X,Y]),CLASS),[OP,X=A,Y=B])

の形式でw-stack にpushされる。用言の受け処理で、

このような要素が現れたら、ecを参照して、OPを",
" もしくは";" にunify し、項形式に変換してs-infoに加える。更にこのecを表現する仮想表のコピーを作り、それぞれの引数をX,Y とunify する。例えば、「ワープロとコピーのある部は」の中間表現は、

```
result(A),bumei(B,A),motu(B,C,X),
  motu(B,C,Y),(X=worpro,Y=copy)
```

となる。

(3) 否定

「ワープロを持っていない部」などの否定を含む文については、以下のような中間表現を生成することを考えている。

```
resuit(A),bumei(B,A),
 all(C,motu(B,D,C),not(C=worpro))
```

しかし、この形式では検索言語生成部の負荷が大きいため、これを意味解析後処理で、

```
result(A),bumei(B,A)
 all(E,motu(E,D,worpro),not(B=E))
```

に変換する。このような否定が現れたという情報はo-infoに格納して、後処理の起動を行なう。

(4) 最上級

「営業部で、給与が最も多い社員は」という最上級を含んだ言い回しに対しては次のような中間表現が対応する。

```
result(A),syainmei(B,A),zokusu(B,C),
bumei(C,eigyou),kyuuyo(B,D),max(E,D),
zokusu(F,G),bumei(G,eigyou),kyuuyo(F,E)
```

このような言いまわしの解析が困難である理由は、何の中で「最も」であるのかが、「最も」を解析しているタイミングでは明らかになっていない点にある。現在、「最も多い」を解析する時点では、s-infoとo-infoにmax(E,D)を加え、意味解析の後処理で、o-infoにmax(E,D)があった場合は、s-info中のD に関連する中間表現を抽出し、これをE の条件に変換して、s-infoに加えることを考えている。

(5) 一意化情報の付加

ＤＢ検索においては、質問文中に陽に示された意味だけを忠実に抽出するだけでは不充分であることがある。例えば、「加藤と中川の住所」という質問文は、住所の検索を要求しているが、それだけでは出力されるのは住所だけであり、どちらがどちらの住所であるかがユーザにわからない。このような不親切な解答になることを防ぐため、Ｑｕｅｓｔでは意味解析後処理として、一意化情報の付加を行なっている。これは中間表現がor並列や自由変項を含んでいて、解が複数になる場合には、それらの変項をresult述語の引数に加えるものである。

6. おわりに

現在、我々はここに述べた機能の実現と文脈処理部の設計を進めている。文脈処理部では、さしあたり、entityを意味する代名詞や省略を扱う予定である。基本的には、文節単位の意味が確定した時点で、その意味である変項とs-infoの内容をclass 名に対応させてstack しておき、代名詞がどのclass のentityかが明らかになった時点でその情報を引くという方式をとる予定である。また、前文の検索結果の参照などのために前文で何が話題の中心であったかも保存しておく必要があると考えている。

Ｑｕｅｓｔの適用領域は、ここで述べた事務用のＤＢや案内ＤＢであると思われるので、この２つのＤＢに対してシステムを作成し、オピニオンテストなどによって有用性を確認する予定である。

謝辞

日頃御指導いただく寺島信義知識ベース研究室室長に感謝します。

参考文献

[1] 藤崎他　データベース照会システム「ヤチマタ」と名詞句データ模型
情処論文誌vol.20 no.1 (1979)

[2] 泉田他　対象世界のモデルを利用したデータベース検索システム
情処データベースシステム研究会43-2 (1984)

[3] Martin.P et al: Transpaortability and Generality in a Natural-Language Interface System　IJCAI pp573-581

[4] 植村：データベースシステムの基礎
オーム社 (1979)

[5] 松本他　Prologに埋め込まれたBottom-up parsar:BUP
情処自然言語研究会34-6 (1982)

[6] Hammer.M:Database Description with SDM : A Semantic Database Model
ACM Trans. Database System vol.6 no.3 (1981)

[7] Shipman.D.W:The Functional Data Model and The Data Language DAPLEX
ACM Trans. Database System vol.6 no.1 (1981)

[8] Chen.P.P : The Entity-Relationship Model - Toword a Uified View of Data
ACM Trans. Database System vol.1 no.1 (1976)