

オブジェクト指向方式による対話理解システム

大澤一郎 米澤明恵 (東京工業大学理学部情報科学科)

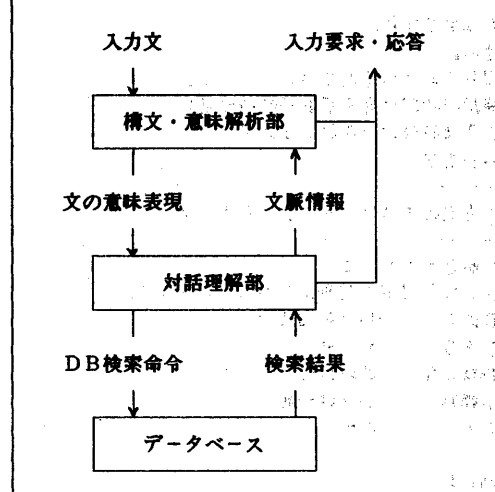
1. はじめに

日本語の対話を理解するシステムとして、オブジェクト指向の概念に基づいて構文・意味解析および対話理解を行なうODDS (Object Driven Dialog System) を開発した。ODDSでは、日本語文解析のための記述として、言語の深層格で文法を記述する格文法 [Fillmore 68] を採用し、この格文法に基づいた意味情報主導型の構文解析を行なう。さらに、各単語ごとに定義される格に制約 (constraint) および好み (preference) を設けることが可能で、制約、好みとして意味的な情報を埋め込み、それを構文解析時にチェックすることで構文解析と同時に意味解析を行なう。その際に問題となるのは、いかに簡潔にこの格文法を記述することができるかということである。格文法に基づいた他のシステムとしては、科学技術庁機械翻訳プロジェクト [長尾83] のGRADE [中村83] による日本語構文解析システム [辻井83] などがあるが、その文法記述は制御に関して強力である反面、全体としてやや複雑になっている。ODDSでは、その解決策として、一つ一つの単語をオブジェクト指向 [米澤84] [Brad84] におけるオブジェクト (単語オブジェクト) と考える。そして、そのクラスの性質の継承 (inheritance) のメカニズムを用いて、文法記述の階層化を行なう。この文法記述におけるオブジェクト指向の有効性は [Winograd83] にも触れられている。一方、解析時に各単語の格スロットに値を埋める操作手続きはその単語の意味やカテゴリーによって異なり、その手続きを一つの大域的なものとして記述することは、制御部分の修正・拡張能力をかなり低下させる。ODDSでは各単語をオブジェクトとしたことにより、この知識をいくつかの単語カテゴリーに付随する問題解決方法 (メソッド) として小さくまとめた単位に分散させ、柔軟な構文・意味解析の制御を可能としている。

対話理解を行なう過程は構文・意味解析のプロセスとは独立している [図1]。このことは人間のモデルとしては不自然であり、システム全体としての能力を弱めると考えられるが、二つのモジュール間で必要最小限の情報交換を行なうことで能力の低下を抑え、文脈に応じた文の理解などを行なう。対話理解の過程は文の構文・意味解析によって得られた意味表現に対するプロダクション・ルールによって制御する。対話では全く同じ文であっても、それに対応する行動は状況によって異なる。簡単な例では、「いいです。」の場合、状況によって賛同の「いい」とも否定の「いい」ともとれる。したがって対話理解をプロダクション・ルールで行なうとすると、状況に応じてプロダクション・ルールを大幅に変更する必要がある。そのためのメカニズムとして、ODDSでは状況をオブジェクト指向におけるオブジェクト (状況オブジェクト) としてとらえ、各状況において有効なプロダクション・ルールをそのオブジェクトの知識としてもたせる。対話の状況の変化に対しては、このオブジェクトを切り替えることで効率良く対応できる。このこと

により対話の状況に応じた適切な反応を可能にしている。

【図1】ODDSの構成



2. ODDSの適用例 - 新幹線予約システム

この章では、ODDSを用いて作成した「新幹線予約システム」の実行例をいくつか示す。使用者が入力した部分は、'》' から始まる下線の引かれたところである。また、枠で囲われた文は説明のために加えたものである。

【例1】

これは東海道新幹線の下り列車の切符を
対話方式によって予約するものです。

いらっしゃいませ。TITIS旅行社によろこ。

以上前置き。後の例では省く。

何の御用ですか？

》しんよこはまからきょうとまでのきつおを2まいください。

入力はローマ字かな変換で入れる。

いつ行きたいのですか？

》らいげつですが。

8月の何日ですか？

今月が7月なので「来月」を8月と解釈する。
足りない情報を補うように会話を進める。

》13にちです。

何時ごろ京都駅に着きたいのでしょうか？

》いつでもいいが、ひかりでいきたい。

候補の制約として、時間ではなくて新幹線の種類でも良い。

大人 2名ですね。

》はい。

しばらくお待ちください。

新横浜07時57分発京都駅10時44分着の

ひかり 531号はいかがですか？

》いいです。

肯定の「いいです。」と判断する。

ありがとうございました。

それでは、以下の列車を予約しておきます。

日にち 1984年 8月13日

ひかり 531 号

新横浜駅 07時57分発

京都駅 10時44分着

大人 2 枚

【例2】

何の御用ですか？

》5かにきょうとまで3にんででかけたいのですが。

何時ごろ京都駅に着きたいのでしょうか？

》ごこの2じから3じごろまででおねがいします。

「午後」が「2時」と「3時」の両方に掛かる。

大人 3名ですね。

》いいえ、子どもが1人います。

3人の中に子供が1人いると解釈する。

しばらくお待ちください。

東京駅12時00分発京都駅14時53分着の

ひかり25号はいかがですか？

》もうすこしはやいのはないでしょうか。

他の候補を尋ねることもできる。

それでは、東京駅11時48分発京都駅14時41分着の

ひかり 171号はどうですか？

》もうすこし。

「はやいのは・・・」を省略しても理解できる。

それでは、東京駅11時24分発京都駅14時17分着の

ひかり 151号はどうですか？

》ほかにはありませんか。

東京駅11時00分発京都駅13時53分着の

ひかり 7号はいかがですか？

》それでは、さいしょから2ばんめのおねがいします。

順番を指定するような前方照応が可能。

ありがとうございました。

それでは、以下の列車を予約しておきます。

日にち 1984年 8月 5日

ひかり 171 号

東京駅 11時48分発

京都駅 14時41分着

大人 2 枚

小人 1 枚

【例3】

何の御用ですか？

》あすとうきょうをでたいのだが。

それで、どこに行きたいのですか？

入力文に合わせて会話を進める。

》なごやです。

何時ごろ名古屋駅に着きたいのでしょうか？

》なんじですか。

何がですか？

何が何時なのか分からないので質問をする。

》いまだです。

今は15時18分です。

この様に突然時刻を聞くことができる。

何時ごろ名古屋駅に着きたいのでしょうか？

》3じです。

大人 1名ですか？

》いいえ。

それでは何名ですか？

》おとな1りに子どもが2りです。

並立句の処理が可能。

しばらくお待ちください。

東京駅15時00分発名古屋駅17時01分着の

ひかり11号はいかがですか？

「3時」とだけ言わたが、常識を働かせて午後3時と解釈している。

》ごゼんの3じにでたいのです。

出発時刻の訂正が行なえる。

しばらくお待ちください。

その時間にはひかり号はありません。

どうしますか？
》いいです。

否定の「いいです。」と解釈する。

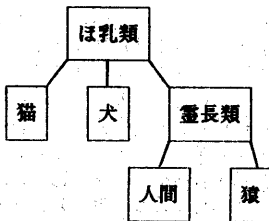
お役にたてず、もうしわけありませんでした。

実行が終了する。

3. 構文・意味解析について

日本語の文を理解するためには、CFGなどによる構文情報よりも語句間の関係に着目した格文法が適していることが多い[長尾84]。しかし、この格文法に基づく構文解析システムを作成する場合には、個々の格を辞書中にどのように記述すればよいのかが問題になる。ODDSでは各単語をオブジェクト指向におけるオブジェクト(単語オブジェクト)としてとらえ、それぞれの単語オブジェクトのクラスの上位クラス(スーパークラス)をその単語の意味的なカテゴリーとすることで、単語オブジェクトを is-a 階層[図2]に組み込んでいる。

【図2】 isa 階層の簡単な例



格の定義は必要に応じて適当なクラスの定義の中に記述し、その格定義は、同じ格に関する再定義がない限り、下位のクラスに自動的に継承(inheritance)される。このメカニズムにより格の定義を簡潔に行なうことが可能となる。更に、格の定義として、意味に関する「制約」(constraint)や、制約より制限が弱く融通の利く「好み」(preference)を付加することができる。ここに意味的な情報を埋め込み、これを構文解析時に利用することで構文解析と同時に意味解析を行なう。また、解析能力を高めるために、格スロットに値が埋まった時に起動させる手続き(when-filled method)や、単語オブジェクトが他の単語オブジェクトから格の候補として参照された時に起動させる手続き(when-referred method)を定義することもできる。

構文・意味解析は、自立語(体言、用言など)や句点(。)の単語オブジェクトの場合は、格スロットに適当な単語オブジェクトをつないでいく手続きにより行なわれるが、この操作は単語オブジェクト自らが行なう。すなわち、オブジェクトが自立的かつ能動的に活動する。この操作は各オブジェクトのクラスに定義されているメソッド'parse'に基づいて行なわれる。格助詞や助動詞などの付属語に属する単語オブジェクトや、並立の「と」

オブジェクト、読点(,)に対応するオブジェクトなどもそれらのクラスの中に定義されている'parse'というメソッドが起動されて独自の構文・意味解析を行なう。詳しい動作は後の節で説明する。

3.1 単語クラスの記述

簡単な例(「聖子」クラスとその上位クラスの記述)[図3]を示して、それを基に説明する。

【図3】「聖子」クラスとその上位クラス

```
[CLASS object]
(METHOD new is $new)
(METHOD parse is $parse)
(METHOD meaning is $meaning)
```

```
[CLASS meishi is_a object]
```

```
[CLASS buttai is_a meishi]
(SLOT
[RANGE NO
(constraint (@and (class-spec junjo)
(check adapt-suushi))])
[ATTRIBUTE nil
(constraint (@and (keiyou)
(katsuyou rentai))])
[OWNER NO
(constraint (class-spec buttai))
(preference 0 (has-a))]])
```

```
[CLASS doubutsu is_a buttai]
(HAS-A ATAMA KAO kao_part ASHI ...)
```

```
[CLASS ningen is_a doubutsu]
(HAS-A buttai ...)
```

```
[CLASS kashu is_a ningen]
(SLOT
[RANGE cut])
(HAS-A UTA ...)
```

```
[TANGO SEIKO is_a kashu]
```

一つ一つの枠(フレーム)で囲まれたものがそれぞれ一つの概念を表わすクラスの定義である。「聖子」クラスの記述は一番下のフレームで、その中には、「SEIKO」というのは単語オブジェクトのクラス(宣言が「TANGO」で始まる)で、その上位のクラス(スーパークラス)は

'kashu' (「歌手」という概念のクラス)であることが書かれている。

一番上のフレームは 'object' というクラスに関する定義で、このクラスは他のすべてのクラスの最も上位の概念クラスである。ここでは、各単語オブジェクトが標準で使うメソッドである、'new' (クラスの実例 (インスタンス) を一つ作る)、'parse' (構文・意味解析を行なう)、'meaning' (意味構造に基づいて意味表現を作る) を定義している。

三番目のフレームは「物体」という概念を表わすクラスの定義 (宣言が 'CLASS' で始まる) で、「物体」には例えば、RANGE (順序の指定) と、ATTRIBUTE (修飾)、OWNER (所有者) の三つの格があることを表わしている。一番始めの RANGE という格は、「三人目の女性」という場合に使われる格である。この格の定義には、格になり得るものとして、「の (NO)」という格助詞を伴う単語オブジェクトで、「順序 (junjo)」という概念クラスに属し、序数詞の形容の仕方が意味的に妥当であることを制約 (constraint) として課している。これにより、「三人目の猫」というような意味的におかしい語句をはねている。また、「三人目の聖子」という表現もおかしいので、「聖子」クラスの上位クラス「歌手」の定義の中で RANGE という格をなくしてしまっている。

最後の OWNER という格は「私の猫」というような所有関係を有する表現を理解するために用いられる格で、格になり得るものとして、格助詞「の」を伴う単語オブジェクトで、制約として「物体」という概念クラスに属し、可能であれば (preference)、has-a 関係を有するものとしている。(この has-a 関係のチェックは各クラス定義に記述してある 'HAS-A' という宣言による。) HAS-A 関係の制約を「可能であれば」という弱いものにしたことにより、「私の聖子のレコード」というような語句に対しては「私の (聖子のレコード)」と正しく解釈し、さらに HAS-A 関係を常識的には有さない「犬の洋服」というような表現でも、他に解釈のしようがなければ好み (preference) による制約を無視して、HAS-A 関係があるかのごとく理解するという柔軟な処理が可能となっている。

3. 2 単語オブジェクトによる構文・意味解析

形態素解析は最長一致優先で、構文・意味解析の結果がフィード・バックされる構文・意味制約利用法である。この形態素解析によって文の先頭から順に適当な単語が定まると、それに対応した単語オブジェクトをその単語クラスのインスタンスとして前から順に作り出す。この単語オブジェクトは、そのクラスあるいは上位のクラスで定義された格に対応して、いくつかの格スロット (格となり得るものをつなぐところ) を有している。構文・意味解析部はこの生成された単語オブジェクトに対して 'parse' というメッセージを送る。

'parse' というメッセージを受けた単語オブジェクトはそのメッセージに対応するメソッドを実行する。例えば自立語の単語オブジェクトの場合、そのオブジェクトよりも前に位置する単語オブジェクト (文の中で前にある単語のオブジェクト) で、他の単語オブジェクトの格

になっていないものを自分に近いところから見ていき、その単語オブジェクトが自分のなんらかの格とみなせる時には、自分の格スロットにその単語オブジェクトをつなぐ。該当する格がない場合には 'parse' というメソッドの処理が終了する。但し、格スロットにつなぐオブジェクトの制約として記されている、preference によって調べた単語オブジェクトが格として却下された場合には、自分とその単語オブジェクトを組にして保留リスト (HOLD LIST) に置いておく。

付属語の単語オブジェクトの場合には、格スロットは有しておらず、直前にある単語オブジェクトの制約がそのクラスに定義されている。'parse' というメッセージを受けた単語オブジェクトは、直前の単語オブジェクトがその制約を満たしていることを確かめると、自分の正体と意味を直前の単語オブジェクトに伝えて消滅する。簡単な例として、「行きたい」という語句では、「行く」と「たい」の二つの単語オブジェクトが生成されるが、「たい」は直前の単語オブジェクトが動詞の連用形であることを確かめた上で、自分のクラスが「たい」であることと、意味が「希望」であることを「行く」に伝えて消滅する。もし直前の単語オブジェクトが制約を満たしていなければ、これより後ろの解析を行なっても正しい構文・意味解析は不可能なので、形態素解析に戻って別の単語の切り出し方を行なう。この際には、今までの結果ができるかぎり利用できるように後戻りを行なう。

文の最後で句点オブジェクトが生成されて、このオブジェクトが、'parse' というメッセージを受けて、他の単語オブジェクトの格にならずに前に残っている単語オブジェクトをすべて句点オブジェクトの「文」という格としてつなぐことができれば構文・意味解析が成功して終了する。例えば、「大阪に行き、買い物をする。」という文の場合は、「行く」と「する」という二つの単語オブジェクトが句点オブジェクトの格としてつながる。成功しなかった場合には保留リストにある単語オブジェクトを優先度に応じて順に、再び 'parse' というメッセージを送ることで動かす。この時は preference による制約は無視して格スロットへのリンクを行なうようにする。この結果うまく解析できればその結果を解析結果とし、これでもうまいかなかった文は形態素解析に戻って、新たな単語の切り出しを行なう。それでもうまいかない時は理解不能とする。

3. 3 文末の省略表現の理解

句点オブジェクトは入力された文をそのまま解析してうまくいかなかった場合に、文末に「です」という単語オブジェクトを補うことを試みる。例えば「私は真吾。」という文の場合、これをそのまま解析したのでは文として理解できない。そのため句点オブジェクトは自分の前に「です」を補って、「私は真吾です。」という文として解析を行なう。

また、前もって文末に省略される可能性のある語句を句点オブジェクトに伝えておく、句点オブジェクトは最初にその語句を補うことを試みる。例えば、省略されそうな語句として「(を) 買いたい。」を指定しておく、句点オブジェクトは「切符3枚。」のような入力に

対して「切符3枚を買いたい。」と省略を補って解釈する。文脈理解部はこの機構を用いて状況に応じた省略表現の理解を行なっている。

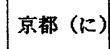
3.4 構文・意味解析の実行例

ここでは、いままでに述べた構文・意味解析が実際にどのように動作するのかを例を用いて説明する。例文として「京都に聖子と明菜が行く。」を用いる。

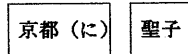
1. 「京都」オブジェクトが作られる。



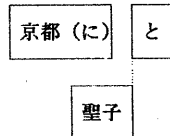
2. 「に」オブジェクトが作られ、直前の単語オブジェクトが名詞というカテゴリーに属することを確認して、「京都」オブジェクトに付く。



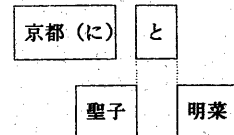
3. 「聖子」オブジェクトが作られる。



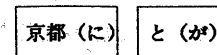
4. 「と (and)」オブジェクトが作られ、「聖子」オブジェクトを「名詞1」という格としてつなぐ。



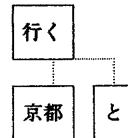
5. 「明菜」オブジェクトが作られ、「明菜」オブジェクトは「と」オブジェクトが自分の格となるかを調べようとする。ところが、「と」オブジェクトには when-referred method (parse-after-and) が定義されているので、「明菜」オブジェクトのメソッド (parse) の実行が一時中断され、そのメソッドが実行される。その結果、「と」オブジェクトは「明菜」オブジェクトを「名詞2」という格にする。そして「名詞1」と「名詞2」の格につながっている単語オブジェクトの共通概念を調べ、この場合であると「歌手」という概念を自分の仮想的なクラスとする。



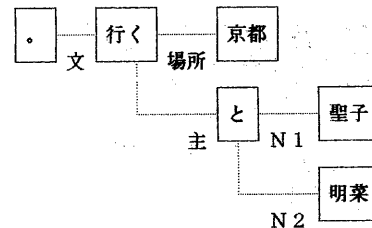
6. 「が」オブジェクトが作られ、直前にある「と」オブジェクトの〈仮想〉クラスが名詞に属することを確認して「と」オブジェクトに付く。



7. 「行く」オブジェクトが作られ、直前にある「と」オブジェクトが格助詞「が」を伴い、〈仮想〉クラスが生物というカテゴリーに属することを確かめた後、「行く」オブジェクトの主格とする。次に「京都」オブジェクトが格助詞「に」を伴い、クラスが場所というカテゴリーに属することを確認して、場所格とする。



8. 「。」オブジェクトが生成され、「行く」オブジェクトのクラスが動詞に属し、活用形が終止であることを調べて「。」オブジェクトの文格につなぐ。これで構文・意味解析は終了し、最終的な構造は以下の様になる。



3.5 意味表現の抽出

構文・意味解析を行なって、入力文の解析が成功した場合には、句点オブジェクトが結果として返される。文の意味は、この句点オブジェクトを解析木の頂点とし、単語オブジェクトの格スロットによって結ばれる、深層格の木構造で表わされる。この内部構造から文の意味表現を得たい時には、この解析木の頂点である句点オブジェクトに 'meaning' というメッセージを送れば良い。このメッセージを受けた句点オブジェクトは、自分だけでは文全体の意味は分からないので、更にこのメッセージ 'meaning' を自分の格スロットにつながっている単語オブジェクトに送る。このようにしてメッセージを送られた単語オブジェクトも同様に自分の格スロットにつながっている単語オブジェクトに 'meaning' というメッセージを送り、このメッセージは構文木の末端に位置する単語オブジェクトまで伝わる。末端の単語オブジェクトはこのメッセージを受けて自分の意味を返す。例えば「聖子」クラスの単語オブジェクトであれば、(SEIKO)

を自分の意味として返す。特殊な例としては、「今日」クラスの単語オブジェクトであると、自分の意味としてその日の日にちを調べて、例えば

(toki (NICHU 26) (TSUKI 7) (TOSHI 1984))

という意味表現を返す。末端でない単語オブジェクトは、メッセージを送った単語オブジェクトからその意味表現が返ってくるのを待ち、返って来た結果を基に、自分から下で表わされている意味を表現する。例えば、先程の

例の「京都に聖子と明葉が行く。」で、「と」オブジェクトは意味表現として

((AND (SEIKO)
(AKINA)))

を返す。そして、句点オブジェクトは

((MODE KOUTEI)
(ACT IKU)
(SUBJECT (AND (SEIKO)
(AKINA))))

((SPACE-TO (KYOUTO)))

を意味表現として返す。

3. 6 構文・意味解析の例

ここでは構文・意味解析の結果、どのような意味表現が得られるかをいくつかの例で示す。

1. 明日の3時から5時までに京都に行きたい。

((MODE KOUTEI)
(ACT IKU)
(TIME ((TIME-FROM (toki (JI 3)))
(TIME-TO (toki (JI 5)))
(TIME (toki (NICHU 26)
(GATSU 7)
(NEN 1984))))))
(SPACE-TO (KYOUTO))
(ATTRIBUTE (KIBOU)))

2. 京都までひかりで切符を買いに行く。

((MODE KOUTEI)
(ACT IKU)
(PURPOSE ((MODE KOUTEI)
(ACT KAU)
(OBJECT (KIPPU))))
(SPACE-TO (KYOUTO))
(MEANS (HIKARI)))

3. 大人3人と子供が2人います。

((MODE KOUTEI)
(IRU
(COMPONENT (AND ((NUMBER (2))
(CLASS (KODOMO)))
((NUMBER (3))
(CLASS (OTONA))))))

4. 聖子と明葉の歌。

((MODE KOUTEI)
(OBJECT (SORE))
(PREDICATE (UTA (OWNER (AND (SEIKO)
(AKINA))))))

5. 聖子の歌と明葉のうたが好きだ。

((MODE KOUTEI)
(SUKIDA
(OBJECT (AND (UTA (OWNER (SEIKO))
(UTA (OWNER (AKINA))))))

6. ばらのはなとひろこのはな。

((MODE KOUTEI)
(OBJECT (SORE))
(PREDICATE (AND (FLOWER (OWNER (BARA)))
(NOSE (OWNER (HIROKO))))))

7. ひろこのはなはきれいです。

((MODE KOUTEI)
(KIREIDA
(SUBJECT (FLOWER (OWNER (HIROKO))))))

8. きれいな、ひろこのはな。

((MODE KOUTEI)
(SUBJECT (SORE))
(PREDICATE
(FLOWER (OWNER (HIROKO))
(ATTRIBUTE ((MODE KOUTEI)
(KIREIDA))))))

9. 3人目の男性。

((MODE KOUTEI)
(OBJECT (SORE))
(PREDICATE
(OTOKO (RANGE (JUNJO (NUMBER (3))))))

10. 3人目のひかり号。

(理解不能)

4. 0 対話理解について

4. 1 状況オブジェクトによる対話理解

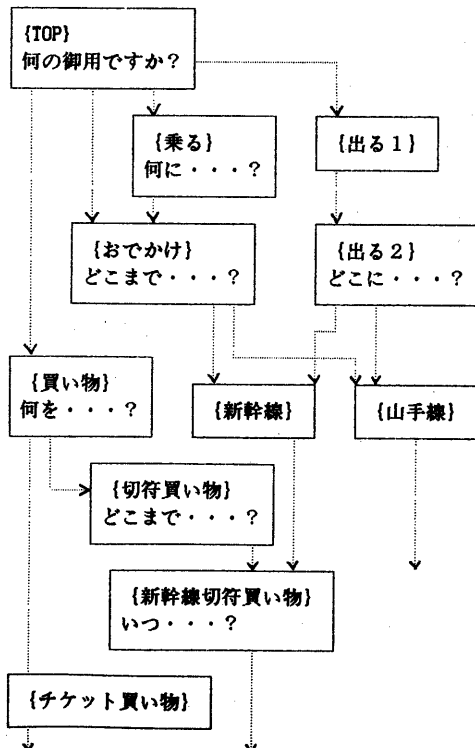
ODDSでは対話理解の制御を、入力文から得られた意味表現に基づいたプロダクション・ルールによって行なう。しかし、自然言語は曖昧性を含むので、たとえ同じ入力文であったとしても、状況に応じて異なった意味を有することが多々ある。一番簡単な例として、「いいです。」という表現は「受理します。」という肯定の場合と、「遠慮します。」という否定の場合があり得る。したがって対話を正しく理解するためには、その発話行為が行なわれた状況を正しく把握して、状況に応じて必要なプロダクション・ルールだけを有効にし、そうでないものを無効にしなければならない。これを単なるプロダクション・システムだけで実現させるのは非常に難しい。ODDSでは、個々の状況に対応する状況オブジェクトというものを考え、そのクラスに対応する状況で有効なプロダクション・ルールを記述しておくことにした。そして、対話理解時には、その時の状況に応じた状況オブジェクトを活性化することでプロダクション・ルールを状況に合ったものに切り替え、適切な反応を効率良く行なうことができる。更に、ルールの記述に関しても、ある程度共通で使われるルールは、オブジェクト指向におけるインヘリタンスの機構を利用して、上位のクラスに記述しておくだけで良いため、その記述は容易である。

更に対話らしさを実現するために、対話の特徴を考えてみると、

- 1) 対話をしている者双方が対話の流れをコントロールできる。
- 2) 質問に対して間接的な答え方ができる。
- 3) 質問に対して省略した返答が行なえる。
- 4) 前の表現を受けて省略した表現が使えらる。
- 5) 代名詞などによる前方照応が行なえる。
- 6) ある程度決まった対話のパターンがある。

などが、さしあたりあげられる。この中でも対話理解システムの構築に特に重要なのは、いかに対話の流れを扱うかであろう。GUS [Bobrow77] では、対話の流れを欠損情報の列としてとらえ、対話理解はその欠損情報を順に得るという手続き (to-filled procedure) により行なわれる。これは非常に計算機的な方法である。ODDSでは、もう少し人間に近いと思われるシステムの構築を試みた。すなわち、人間が計算機と対話を試みる時はなんらかの目的を持っている。そして、その目的を達成するためにプランをたて、それに基づき発話行為を行なう。計算機はその発話行為 (計算機への入力) から逆に人間の目的を推定し、それにしたがって問題解決を行なえば良い。ODDSでは各状況が状況オブジェクトで表わされているので、その状況において、人間からの入力より予想されるようなゴールへ到達可能な状況オブジェクトへと漸進的に移行することで対話の流れを実現している。【図4】

【図4】状況オブジェクトの移行



4.2 状況クラスの記述

個々の状況クラスには、そのオブジェクトが活性化された時に行なう動作が記述されている。入力文に対する反応は、その意味表現を条件とするプロダクション・ルールによって制御される。このルールの記述は状況クラスの定義では 'RULE' という項目で示される。プロダクション・ルールの条件部はほとんどが 'input?' というメソッドの呼び出しで、これにより、今までに入力された文の意味表現と、ここに記述されている意味表現とのパターン・マッチングが行なわれる。パターン中で '*' で始まるアトムは変数を表わし、 '?' で始まるリストはそれが任意であることを示す。行動部では、'create' というメソッドを実行して、次に活性化したい状況オブジェクトを作り出し、それに対話理解の制御を渡す。この機構により状況に応じた対話の理解を効率良く行なう。どのプロダクション・ルールも実行できなかった場合には、'WHEN-NOTHING' で指定される動作を行なう。ここでは大抵の場合、'question' というメソッドを実行して、計算機が人間に対して質問を発し、人間からの応答を待つ。'DEFAULT-ELLIPSIS' という項目は、対話理解部が構文・意味解析部に文末に予想される省略表現を必要に応じて伝えるためのデータである。これは、その時活性化している状況オブジェクトが自分のクラスに定義してある予想省略表現を構文・意味解析部に伝えることで行なわれる。他に、'WHEN-CREATED' という項目もあるが、これはその状況オブジェクトが作られた時に参照されるプロダクション・ルールである。ここには、状況オブジェクトが活性化される以前の状態に基づくルールを定義する。これは極力無駄なプロダクション・ルールを参照しないための処置である。以下に簡単な状況クラスの定義【図5】【図6】を示す。

【図5】対話開始時の状況クラス

```

[ DIALOG {TOP} is_a SUPER-DIALOG-A ]
( WHEN-NOTHING
  ( question " 何の御用ですか? " ) )
( DEFAULT-ELLIPSIS ((WO) KAU shuushi) )
( RULE [ if ( or ( input? ((MODE KOUTEI)
  ( ACT KAU)))
  ( input? ((MODE KOUTEI)
  ( ACT TSUKU))) )
  then ( create {ODEKAKE} ) ]
  [ if ( or ( input? ((MODE KOUTEI)
  ( ACT KAU)))
  ( input? ((MODE KOUTEI)
  ( ACT HOSHII))) )
  then ( create {KAU} ) ]
  [ if ( input? ((MODE KOUTEI)
  ( ACT DERU)))
  then ( create {DERU} ) ]

```

【図6】どこに行きたい時の状況クラス

```
[DIALOG {ODEKAKE} is_a SUPER-DIALOG-A]
(WHEN-NOTHING
  (question "どこまで行きたいのですか"))
(DEFAULT-ELLSIPS ((NI) IKU shuushi))
(RULE [if (input?
  ((MODE KOUTEI)
    (SPACE-TO
      (TK-S *SPACE-TO))
    ?(SPACE-FROM
      (TK-S *SPACE-FROM))
    ?(TOOL *TOOL)
    ?(COMPONENT ((NUMBER *NUMBER)
                  ?(CLASS *CLASS)))
    ?(TIME *TIME))
  then (create {SHINKANSEN})])]
```

5. インプリメント

VAX-780 の UNIX4.2bsd 上の Franz lisp を用いて作成した。開発・作成に要した期間はおおよそ4ヶ月であり、作成はすべて一人で行なった。現在、単語クラスの定義は 344個(うち、単語オブジェクトのクラスは 216個)で、状況クラスの定義は 31個である。以下に各部分の行数を示す。

構文・意味解析部	
LISPプログラム	2148行
単語クラスの記述	1680行
対話理解部	
LISPプログラム	
(汎用)	606行
(領域依存)	271行
状況クラスの記述	930行
DB検索	
LISPプログラム	482行

6. 考察・結論

システム全体をオブジェクト指向の概念に基づいて構成したために、そのプログラムは簡潔で分かりやすいものとなった。それは単語クラスの記述をも、修正、拡張性の高いものとするのに役立った。構文解析は格文法を基本として用いる、意味情報主導型である。そのため、制約(CONSTRAINT)、好み(PREFERENCE)による意味処理が付加しやすく、構文解析と同時の複雑な意味解析が可能となった。更に、文脈情報に関する処理をこの機構の中に付加すれば、文脈処理をも融合できるのではないかと考えている。

対話理解においては、状況オブジェクトを用いてプロダクション・ルールをモジュール化したため、適切で効率の良い応答が可能となった。今回は状況オブジェクトの上位クラスに汎用のプロダクション・ルールをあまり用意しなかったが、この部分のルールを増やして、ゴールの推測と、プランニングの考え方[Allen83]を取り

入れると更に柔軟なシステムができると考えられる。

また、このシステムの領域依存の部分は状況オブジェクトの記述と、それに伴ういくつかのメソッドの定義だけであり、この部分は他の部分と完全に独立している。したがって、かなり汎用性の高いものと考えている。

現在はODDSの基本となる部分を考案し、それを実現した段階である。今後はもっと複雑な文を解析し、より広範な領域に対処できるようにODDSを拡張していく予定である。

参考文献

[米澤84]

「オブジェクト指向型プログラミングについて」、コンピュータ・ソフトウェア、1, 1, 1984.

[長尾83]

「科技庁機械翻訳プロジェクトの概要」、自然言語処理研究会資料38.

[長尾84]

「機械翻訳技術の現状と展望」、コンピュータ・ソフトウェア、1, 1, 1984.

[辻井83]

「日本語構文解析」、自然言語処理研究会資料38.

[中村83]

「文法記述用ソフトウェアGRADE」、自然言語処理研究会資料38.

[Brad84]

'Message/Object Programming: An Evolutionary Change in Programming Technology', IEEE SOFTWARE, 1 (1984).

[Winograd83]

LANGUAGE AS A COGNITIVE PROCESS (I) SYNTAX, ADDISON WESLEY, 1983.

[Bobrow77]

'GAS, A Frame-Driven Dialog System', Artificial Intelligence, 8 (1977).

[Allen83]

'Recognizing intentions from natural language utterances', Computational Models of Discourse, MIT press, 1983.

[Fillmore68]

'The case for case', Universals in linguistic theory, Holt, Rinehart, and Winston, 1968.