論　文　　Technical Paper

対話領域の独立性を指向した日本語対話理解システム

A Japanese Dialog System Applicable to Multiple Discourse Domains

渡部　卓雄*　　大澤　一郎*　　米澤　明憲*
Takuo WATANABE　Ichirou OHSAWA　Akinori YONEZAWA

*　東京工業大学理学部情報科学科
Dept. of Information Science, Faculty of Science, Tokyo Institute of Technology, Tokyo 152, Japan.

1986年11月25日　受理

Keywords: natural language processing, object-oriented programming.

Summary

A Japanese discourse understanding system is presented, which is designed to be adaptable for multiple discourse domains with small modification efforts. The system is constructed in an "object-oriented" manner so that domain-dependent/independent components are easily factored out.

Dictionaries, grammatical rules, words and discourse situations are represented as objects. Two discourse domains, a railway ticket reservation domain and an electric-mail service domain, are currently used as a test bed for the system.

1. はじめに

できるだけ少ないプログラムの変更で，複数の対話領域（対話の話題・テーマ）に対処できることをねらった日本語対話理解システムMODUS (Multi-domain Object-oriented Dialog Understanding System) を作成した[7]．システムの設計にあたっては，いわゆる心理学および認知科学上の考察は一切行っていない．この研究の目的は，人間の対話における認知モデルを実験することではなく，エキスパートシステム，データベース，オペレーティングシステムのシェルなど，各種アプリケーション・プログラムの自然言語インタフェースとしての対話システムの開発を容易にすることにある．

既に作成した対話システムにおける対話領域と比較的類似性の高い領域での対話システムを新たに作成する場合，できる限り以前のプログラムを再利用できるのが望ましい．そこでMODUSでは対話領域の実現に必要な知識を表す部分（辞書，文法，単語，対話状況）を領域に依存した部分とそうでない部分に分割した．辞書，文法，単語，対話状況はそれぞれオブジェクト指向概念のオブジェクトになっており，オブジェクトの持つ知識の局所性と継承による階層性とによって知識の分割と共有をはかっている．各種のアプリケーションへの対応は，これらのうちの領域に依存した部分を取り替えることによって行うことができる．

オブジェクト指向概念を用いた対話理解システムにODDS[3]がある．MODUSはODDSの考え方を基に，より多くの対話領域に対応させることを目指した対話システムである．

以下では，MODUSの構成を全体像，構文・意味解析部，対話処理部の順に述べ，複数の対話領域において対話システムを作成した結果と問題点について考察する．

なお，この論文の最後に対話システムの例として新幹線の切符の予約を行うシステムとUnixの電子メールシステムの日本語インタフェースの例を挙げる．システムはCommon Lispで記述されているため移植性は高く，現在Symbolics Common LispとUnix上のKCL (Kyoto Common Lisp) の上で稼動している．対話動作は，いずれにおいても実時間で行われる．

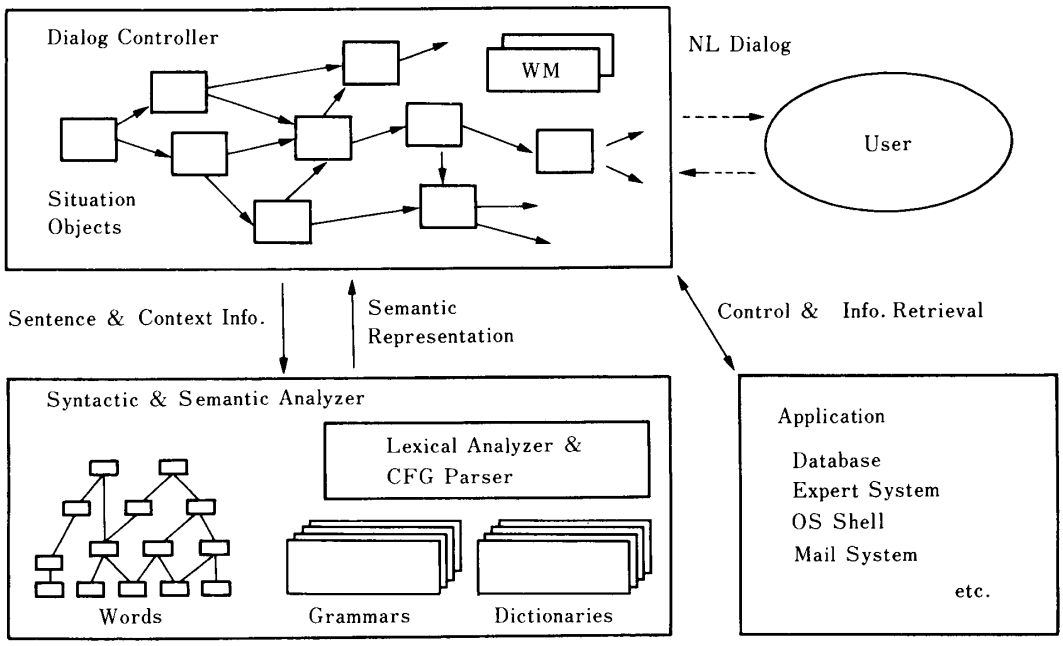


Fig. 1 MODUS system organization.

2. システムの構成

システムの構成を Fig. 1 に示す. MODUS は構文・意味解析部と対話処理部から構成されている. 実際に応用する対話システムを構成する場合には, これらにアプリケーション・プログラムが加わる. 対話処理部は人間の発話の入力, 発話生成, 知識の管理, そしてアプリケーション・プログラムとのインタラクションといった対話に関する実質的な制御をすべて行う. 対話処理部において人間の入力した文を解釈する必要が生じたときに, 構文・意味解析部が起動される. 構文・意味解析部は日本語の文を意味表現に変換する.

3. 構文・意味解析部

構文・意味解析部は, 拡張 CFG パーザ, 辞書・文法オブジェクト群, 単語オブジェクト群, そして名詞句スタックから構成されている.

3・1 拡張 CFG パーザ

パーザは, 日本語文の形態素解析を行う部分と, その結果を使って構文解析を行う部分から構成されている. これらは両方とも Lisp のプログラムである. 形態素解析と構文解析は完全に分離しており, 構文解析は形態素解析が完全に終了した後に行われる. 形態素解析は, まず文の先頭から辞書を引いて切り出し可能な単語を長い順に並べ, それぞれについて文の残りを同様に切り出していく. 結果は明らかに木構造になるが, 最後まで切り出すことのできた枝のみを残し, また同じ形の部分木は共有するようにして Fig. 2 のような結果を得る. これを深さ優先でたどれば最長一致優先の単語列を得ることができる. 実際には structure sharing で行って無駄な計算を避け, 高速に形態素解析を行うことができる. 構文解析はトップダウン予測付きのボトムアップ・アルゴリズム (left corner parsing ; BUP[2] と同様のアルゴリズム) によって行う.

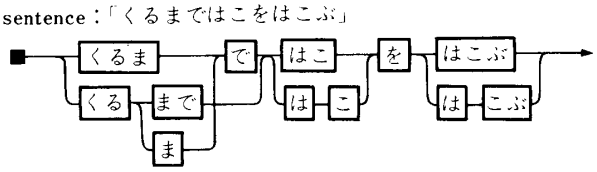


Fig. 2 An example of morphological analysis.

各構文カテゴリは有限個の属性 (attribute) を持ち, 意味計算および構文規則適用時の意味的制約に使用する. これは属性文法における合成属性に相当する. 属性値の計算は, ボトムアップの構文解析中に構文木のある部分木が完成したときに行われる. このとき完成した部分木の根に位置する構文カテゴリの属性値が, その直接の子孫の構文カテゴリの属性値から計算される. 意味的制約のチェックは, 完成した部分木が (サブ) ゴールにつながるときに行われる.

属性値の計算と意味的制約のチェックのためのコードは, 任意の Lisp の式として文法規則の中に書くことができる (この例は Fig. 4 で示される). 構文規則

の適用が失敗したときと意味的制約を満たさなかったときにはバックトラックを行うが，属性値の計算では副作用を伴うこと（たとえば後述する単語オブジェクトのスロットへの値の代入など）があり，これに対しては undo を行う必要がある．したがってパーザは undo のための機構を持っている．

3・2 辞書・文法オブジェクト

辞書オブジェクトは，文字列からその文字列が表し得る（複数の）単語の構文カテゴリへの対応表である．文法オブジェクトには拡張 CFG の構文規則が登録されているが，left corner parsing を行うのに便利なように，構文規則の右辺の最初の構文カテゴリから（複数の）構文規則への対応表となっている．

これらのオブジェクトは，他の複数のオブジェクトのエントリを継承することができる．こうすることによって，複数の対話領域に共通な部分，すなわち領域と独立な部分をいくつかのオブジェクトにまとめておいて各種の対話領域で使うことができる．また辞書や文法を対話の状況に応じて切り替えることによって，柔軟な処理が可能である．

また，これらのオブジェクトは，クラスから複数のインスタンスを作る必要がない（クラスを「型」として利用するわけではない）ため，1つのオブジェクトの記述からはただ1つのオブジェクトが作られるだけである．

3・3 単語オブジェクト

自立語の意味的な情報を表すためのオブジェクトである．動詞，名詞のオブジェクトは格や性質を表すスロットを持っている．Fig. 3 に単語オブジェクトの記述の例を示す．単語オブジェクトのスロットの種類はあらかじめ単語ごとに決まっているが，この例の Mail のようにある対話領域において特別な役割を果たす単語の場合には，その目的に応じたスロットを設けることによってアプリケーションに必要な情報の獲得をスムーズにする．

単語オブジェクトの記述は1つの単語クラスを作る．これは MODUS のほかのオブジェクトと異なり，クラス/インスタンスがある．単語オブジェクトは，同じクラスで個として異なるものが，構文・意味解析中に動的に生成されるためである．また名詞オブジェクトの場合は，それがどのようなクラスに所属するか，すなわち型情報が意味解析の際に重要な情報（実際には用言の格スロット値の制約条件など）となる．したがって，名詞オブジェクトのクラス/サブクラス関係はかなり大きなネットワークとなっている．

```
(defword (Phys-obj :mixin) (Meishi)
  (:slots
    (:attr :type (:meishi-attr))))

(defword (Seibutsu :mixin) (Phys-obj))

(defword Mail (Phys-obj)
  (:slots
    (:date :type (:mail-attr))
    (:from :type (:mail-attr))))

        (a) Example of Noun.

(defword (Seibutsu-Act :mixin) (Act)
  (:slots
    (:agt :type (:case)
          :class (Seibutsu))))

(defword Taberu (Seibutsu-Act)
  (:slots
    (:obj :type (:case)
          :cm (WO-cm)
          :class (Tabemono Seibutsu))))

        (b) Example of Verb.
```

Fig. 3 Example of word objects.

3・4 名詞句スタック

これは名詞句の指示表現の前方照応を行うときに用いる．詳細については次節で述べる．

4．意味解析と意味表現形式

構文・意味解析部において意味計算を行う方法と，意味表現形式について述べる．

4・1 意 味 計 算

Fig. 4 に示すのは文法オブジェクトの1つのエントリの例であり，拡張 CFG の構文規則の1つを表している．構文規則を表すS式の最初の要素は規則の左辺の構文カテゴリ，2番目は右辺を表し，残りの部分はこの規則によって計算される属性の名前と値を計算するためのコードの交替リストになっている．属性名は Lisp のキーワードである．右辺にはそれぞれのカ

```
;; NP --> S-1, NP.
(NP ((S-1 :katsuyou (eqs :rentai)) (NP))
    :use-attrs 2
    :facts (cons (or (sendw (attr 1 :value)
                            :fill-case (attr 2 :value))
                     (fail))
                 (attr 2 :facts))
)
```

Fig. 4 An example of grammatical rule.

テゴリの属性に関する意味制約をチェックするコードを書くことができる．ここでNP, S−1はそれぞれ名詞句と文を表すカテゴリ名である．この規則を適用するにあたっては，まず右辺のS−1は動詞の活用は連体形でなければならない．規則の適用が成功した場合，属性の計算が行われる．属性計算を行うコードの中では，(attr n name) で右辺のn番目のカテゴリのnameという属性値を参照できる．この例について説明すると，まず，:use-attrs 2で，右辺の2番目のカテゴリ，すなわちNPの属性を全てそのまま（左辺の属性として）使う．:valueという属性の値は単語オブジェクトである．右辺のS−1の:valueの値はこの文の用言オブジェクト，NPの:valueの値は名詞オブジェクトである．またNPには:factsという属性があり，このNPに関する表明を値とする．

埋め込み文S−1の用言オブジェクト((attr 1 : value)の値）の適当な格スロットを，修飾されるNPの名詞オブジェクト((attr 2 : value)の値）で埋める．用言オブジェクトはNPの:facts属性に加えられる．スロットを埋めるのに失敗すると構文規則の適用も失敗する．スロットを埋めるのは副作用を持つ計算であるので，undoを行う必要がある．

単語オブジェクトは，メッセージ：semによって自身の意味表現形式を作成する．文の解析が成功したとき，ゴールである構文カテゴリの属性：valueの値の用言のオブジェクトにメッセージ：semを送り，文の意味表現を計算する．メッセージはスロット中の単語オブジェクトにも送られ，文の意味表現が構成される（Fig. 5）．

意味表現を作っていく際，名詞オブジェクトが現れたらそれを名詞句スタックに積む．代名詞や以前に出現した名詞句を参照するような表現（「〜の（もの）」など）が出現した場合，名詞句スタックを上からサーチして意味的制約に適合する名詞オブジェクトを見つけ，それを意味表現に用いる．

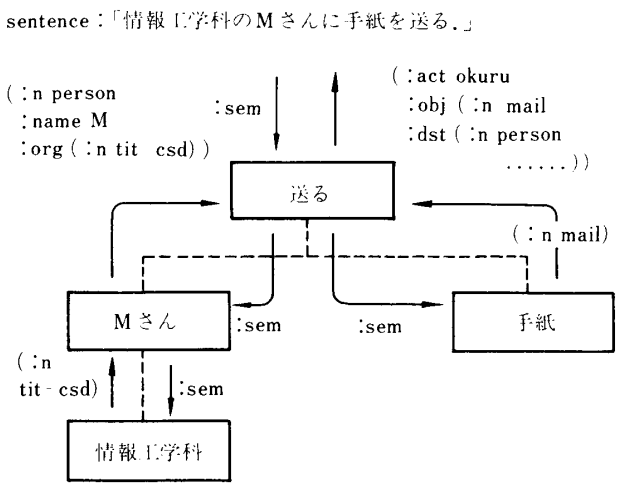


Fig. 5 An example of semantic representation retrieval.

4・2 意味表現形式

文の意味は意味素性（feature）とその値のペアの有限個の集まりで表している．素性の値としては他の文や名詞句の意味のほかに，任意のLispオブジェクトをとってもよい．例をFig. 6に示す．意味素性としては，ここでは日本語の意味表現に一般的によく用いられている格文法の考え方を採用している．たとえば用言の素性として，型，動詞の格，時制，様態などを用いている．

名詞句については，名詞の種類，性質などを意味素性としている．単語オブジェクトのスロットに領域に応じた特殊なものがある場合は，それに応じた素性を持つこともある．

```
sentence: 「先月１８日からのMさんからの手紙を見たい。」

(:act  miru
 :type :kibou
 :obj  (:n    mail
        :from (:n     person
               :name  "M"
               :org   tit-csd )
        :date (:n     date-<
               :year  1986
               :month 10
               :day   18 )
       ))
```

Fig. 6 An example of semantic representation.

5. 対話処理部

対話処理部は対話の状況を表す状況オブジェクト群と2つのワーキング・メモリから構成されている．

5・1 状況オブジェクトの構成

状況オブジェクトは1つ1つが対話の1つの状況を表しており，それぞれの状況に独自の知識を手続きとして持っている．

状況オブジェクトにおいては，一般にInitial-action, Conditional-actionそしてDefault-actionの3種類のアクションを定義できる．このうちConditional-actionはif-then規則の形式をとる．これらのアクションは他の（複数の）状況オブジェクトのものを継承できる．Conditional-actionの条件部は，4・2節で述べた意味表現形式をパターンにしたものである．アクションとしては，ワーキング・メモリへのアクセス，発話生成，人間の発話の解釈，他の状況オブジェ

```
(defsituation Command-Wait (Global-Situation)
  (:dictionary Global-Dictionary-1)
  (:grammar Global-Grammar-1)
  (:conditional-action
    (rule *input*
      ((:act Dasu              ; 「～をだす・だしたい」
        :type {koutei kibou}
        ) -->
       (sends 'Mail-send :activate))
      ({(:act Miru             ; 「～をみたい」
         :type kibou
         )
        (:act Miseru           ; 「～をみせろ」
         :type Meirei
         )} -->
       (sends 'Mail-show :activate))
      .....
      ))
  (:default-action
    (format *terminal-io*
      (command-wait-random-messages))
    (input-and-parse))
  )
```

Fig. 7 An example of situation object.

クトの起動，任意の Lisp の式の実行などが可能である．状況オブジェクトの例を Fig. 7 に示す．

発話生成は，簡単な穴埋め方式をとった．入力された文は，構文・意味解析部によって意味表現になり，ワーキング・メモリの1つに保持される．構文・意味解析部を起動する際，辞書オブジェクトと文法オブジェクトをそれぞれ陽に指定する．こうすることにより，同じ日本語の表現でも解釈を行う状況ごとに異なる辞書/文法オブジェクトを用いれば，状況によって解釈を変えることが可能である．

実際の対話システムでのこのような例としては，システムからの質問に人間がこたえるときに省略した表現が使えるよう，省略表現用の辞書/文法オブジェクトをいくつか作って状況によってそれらを切り替えて用いている．こうすることによって文脈に依存した非文に近い表現の解析を効率的に行うことができる．Fig. 7 では，:dictionaryおよび:grammar で辞書，文法を指定している．

5・2 状況オブジェクトの動作

状況オブジェクトはメッセージ:activate を受け取るとアクティブな状態になる．したがって，ある状況から他の状況へ遷移するには，遷移先の状況オブジェクトにメッセージ:activate を送ればよい．アクティブな状態になったときの状況オブジェクトの動作を次に示す．

(1) Initial-action を実行する．

(2) Conditional-action を実行する．下位のオブジェクトに書かれていたプロダクション・ルールから優先的に条件部をチェックする．また条件にマッチするワーキング・メモリの要素ではもっとも新しいものを優先する．

(3) (2)において，どのプロダクション・ルールの条件も成立しなかった場合のみ，Default-action を実行する．

ワーキング・メモリには人間の入力した文の意味表現を保持しておくものと，システムが得た知識を保持しておくものの2つがある．入力文の解釈を行う状況とそれ以外の状況では別々のワーキング・メモリを用いることによって探索空間を減らしている．

対話処理部の動作は，まず最初に決められた状況オブジェクト（初期状況）にメッセージ: activate を送り，その状況オブジェクトを活性化する．後は状況オブジェクト同志でメッセージ:activate を送り，対話状況の遷移を行う．

Fig. 7 で例示したのは電子メールシステムのコマンド待ち状態の状況オブジェクトの一部である．ここで，sends は状況オブジェクトに対するメッセージ送信を行う関数である．＊input＊は人間の入力を保持するワーキング・メモリである．ここに示した部分では入力文に「～を送る（肯定，希望）」という意味の表現があることによって Mail-send という状況に遷移し，「～をみる（希望）」，「～を見せる（命令）」と入った表現によって Mail-show という状況に遷移することが示されている（このように，どの状況オブジェクトに遷移するかは各状況オブジェクトの設計時に決められている）．

遷移先の状況では，入力文からコマンドの実行に必要な情報を確保し，必要ならばさらに人間に対して質問を行う．

6. IPS によるオブジェクト指向プログラミング

階層的な知識を用いるシステムをプログラミングするときには，継承機構を持った知識表現のための機能（たとえばフレームなど）がよく用いられている．しかし，現在プログラミング言語としてこのような特徴を持ったものが一般に普及していないので，システムの試作を行うためには，Lisp や Prolog などの一般的なプログラミング言語の上に，①汎用の知識表現機構を作成し，それでシステムを記述する．②システムごとに専用の知識表現のための機構を作成し，その上でシステムを記述する，といったアプローチがとられている．MODUS においては，辞書，文法規則，単語，対話状況をオブジェクト指向概念のオブジェクトとし

て扱うことによって継承の概念を用いており，知識の共有・分割をスムーズに行っている．これらのオブジェクトでは，上位のオブジェクトから継承するものや継承の仕方がそれぞれ異なっている．また，単語オブジェクトは単語クラスのインスタンスとして，個として異なるものが生成されているが，その他のオブジェクトはクラスを必要とせず，オブジェクトの記述からはただ1つの実体が作られている．

そこで，MODUS のプログラミングに使用した IPS (Inheritance Programming Schema) では，Lisp 上で継承を用いたプログラミングを容易にするための基本的機構を提供することにした．IPS では meta-object という概念を導入し，そこでオブジェクトのインプリメンテーションや継承の方法などをプログラマが指定できるようにした．IPS システムは基本的には(多重)継承のネットワーク上での各オブジェクトのバージョン管理やキャッシングを容易に行うための枠組みのみを提供しているわけであるが，他のオブジェクト指向言語と同様のこともできるように，メソッドやインスタンスを作るためのプログラムもライブラリとして用意している．

CommonLoops[1] においてもメタクラスをプログラマが記述することによって上に述べたことと同様のことができるが，手近に適当な処理系がなかったこともあって，MODUS のインプリメントでは必要最小限の機構を持った IPS を作成して用いた．また CommonLoops にも meta-object という概念があるが，IPS のものとは直接の関係はない．

7. ま と め

辞書オブジェクト，文法オブジェクト，単語クラスに関しては階層化によるプログラムの部品化に比較的成功したといえる．特に辞書，文法オブジェクトのコーディングでは，基本的な日本語の語彙，文法のほかに，時刻や日付などの頻繁に使われる表現や，領域や文脈に依存した表現に関するものを個々に効率的に開発できた．

状況オブジェクトについては，現在のところ部品化のための決め手となる表現形式および効率的な開発手法は存在していない．ただ，類似した領域での対話の一連の流れにおいては，状況オブジェクトの遷移のパターンも似ている．そこで，既存の状況オブジェクト群からそのパターンを抜き出して類似のオブジェクトを新しく作成するためのプログラミング・ツールのようなものがあると，かなりの開発の助けになるであろう．

現在の方法では，対話システム全体からみた状況オブジェクト1つ1つの役割が必ずしも明確ではなく，状況オブジェクト間の結合が強すぎる．異なった方向からの抽象化が必要である．たとえば，ここで採用しているプロダクション・システムをとってみても，現在のワーキング・メモリのような平坦な情報の構造よりは人間の記憶構造に合わせた構造化（たとえば，MOPs[5]）を行ったほうがよいであろう．文脈に依存した意味解析は，現在は名詞句スタックを使った指示表現の解析以外は領域に依存したアドホックな方法で行っている．

さらに，言い直しや訂正などで一度人間が与えた情報とは異なる情報を後で与えた場合（信念の翻意）の処理も，現在では状況オブジェクトにその動作を行うルールを書かなければならない．領域に依存した方法を採ることによって実行効率はよくなるが，プログラムの部品化はできなくなる．こうした処理を領域によらずに統一的に行うことが必要であるが，対話システムは実時間で動作が行われなければならないので，あまり複雑な推論を用いることはできない[4]．

ODDS[3] では単語オブジェクトが格情報などの意味的知識のほかに構文解析を行うための（手続き的）知識も持ち，単語オブジェクト自身が能動的に構文解析のコントロールも行うのに対し，MODUS では単語オブジェクトが持つのは意味的知識のみであり，構文上のコントロールは CFG 規則によってパーザが行っている．

この方式では，単語オブジェクトに加えて構文規則も記述しなければならず，また人間の言語認知過程のモデルとしても決して良いものとはいえない．しかし，CFG で書かれた構文規則は直感的に理解しやすいため，コーティングのやりやすさと読みやすさは ODDS と比較した場合には向上している．

構文・意味解析においては，ユニフィケーション文法[6] の考え方を導入することによってより簡明で効率的な開発が行えることが期待できる．MODUS の意味表現形式には自然な形で導入できるはずである．この場合のプログラミング言語としては，Lisp よりも Prolog のようなユニフィケーションの機構を持った言語を使うほうがよい．

最初に述べたように，この研究では認知科学的考察を全く行っていないのであるが，認知科学上の成果を取り入れることによって対話理解システムの開発効率があがるのならば，それは採用すべきである．

◇参考文献◇

(1) Bobrow, D. G., *et al* : CommonLoops : Merging lisp and object-oriented programming, Proc. OOPSLA, pp. 17-29 (1986).
(2) Matsumoto, Y., Tanaka, H., *et al* : BUP : A bottom-up parser embedded in prolog, New Generation Computing, Vol. 1, No. 2, pp. 145-158 (1982).
(3) 大澤一郎, 米澤明憲：オブジェクト指向方式による対話理解システム, コンピュータソフトウェア, Vol. 2, No. 1, pp. 11-28 (1985).
(4) 大澤一郎, 米澤明憲：対話システム, 信学技報, Vol. 86, No. 71, NLC 86-3 (1986).
(5) Schank, R. C. : Reminding and Memory Organization : An Introduction to MOPs, Strategies for Natural Language Processing, LEA (1982).
(6) Shieber, S. M : An Introduction to Unification Based Grammar, CSLI Stanford Press (1986).
(7) 渡部卓雄, 大澤一郎, 米澤明憲：対話領域の独立性を指向した日本語対話理解システム, 日本ソフトウェア科学会第3会大会論文集, A-1-1, p. 1 (1986).

〔担当編集委員：石崎　俊〕

Appendix：対話システムの実行例

“＞”はプロンプト．人間の入力には下線を引いた．入力は実際には仮名とアルファベットのみであるが，ここでは見やすくするため漢字仮名混じりにしている．

〔1〕 新幹線の切符予約システム

いらっしゃいませ．何の御用でしょうか？
＞京都までの切符を下さい．
いつおでかけになるのでしょう？
＞さ来週の月曜日です．
何時ごろ京都駅にお着きの予定ですか？
＞朝9時ごろ東京駅を出たいのですが．
大人1人でよろしいですか？
＞大人2人と子供2人にして下さい．
わかりました．
東京駅発9時12分，京都駅着12時5分のひかり153号はいかがですか？
＞それで結構です．
はい，わかりました．
それでは次の列車を予約しておきます．
……予約内容を表示……
ありがとうございました．

〔2〕 電子メール・システムの日本語インタフェース

(例1)

これはUnixの電子メール・システムです．
新しいメールが3通届いています．
なんの御用でしょうか？
＞新しい手紙は誰からのですか？
システムから1通，Oさん*から1通，情報工学科**のMさんから1通です．
＞では，Oさんからのを見せて下さい．

＊　実際には人の名前である．
＊＊　人やサイトの名前は，辞書にある場合には漢字が使える．ないときにはアルファベットを「」でくくって使う．

わかりました.

…… Mail の内容を表示……

ほかには何かありますか？

＞ほかのをみせて下さい.

…… Mail の内容を表示……

…… Mail の内容を表示……

後はどうしますか？

＞システムからのを捨てて，ほかをフォルダーにしまってほしい.

フォルダーの名前はなんですか？

＞Oさんからのを「sked」に，Mさんからのを「circle」にしまって下さい.

わかりました．ほかに何かありますか？

＞先週の木曜日からの「bugs」フォルダーの手紙をプリントアウトしてほしいのですが.

わかりました.

ほかに御用は？

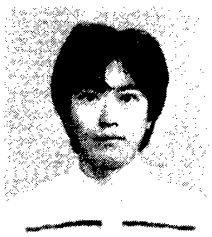
＞結構です.

さようなら.

著 者 紹 介

渡部 卓雄

昭和61年東京工業大学理学部情報科学科卒業．現在，同大学院理工学研究科修士課程情報科学専攻在学中．ソフトウェア基礎論に興味を持っている．日本ソフトウェア科学会，情報処理学会会員.



大澤 一郎

昭和59年東京工業大学理学部情報科学科卒業．昭和61年同大学院理工学研究科修士課程情報科学専攻修了．現在，同大学院理工学研究科博士課程情報科学専攻在学中．自然言語を用いた対話行為に関する知識の体系およびコンピュータ上への実現方式に興味を持つ．日本ソフトウェア科学会，情報処理学会，日本認知科学会，ACM 各会員.

米澤 明憲（正会員）

昭和45年東京大学工学部計数工学科卒業．同大学院博士課程修了．昭和53年より57年まで MIT 計算機科学科大学院に在学し Ph. D. in Computer Science 取得．現在，東京工業大学理学部助教授．ソフトウェア基礎論および人工知能に興味を持っている．編著書に「Object-Oriented Concurrent Programming」（MIT Press），「算法表現論」（岩波書店）などがある．Sigma Xi 会員.