連載講座

自然言語処理入門—IV

文を生成してみよう†

岡田直之††　中村順一††

言語を生成する場合，“何を言うか (What to say)” と “どのように表現するか (How to say)” の二つを考える必要がある．今回は，これらを考慮しつつ，文を生成する課題に取り組もう．

5. 言語生成

まず，“何を言うか” については，具体的な対象，たとえばデータベースの内容の説明とか機械の取扱いの説明とかがなければ十分に検討することができない．機械翻訳であれば，“何を言うか” は本来入力の文章に表現されていることであるので，考慮する必要がない．もし心の中で思ったこととすると，非言語的知識や心理活動が十分解明されていない現状においては大変難しいものになる．次に，“どのように表現するか” については，一般的に考えると文章作成技術の問題となり，人間でも容易でない問題となる．もし **4.** で述べた格構造や解析木が出発点であれば，こまかな表現上のことを別とすれば，比較的単純なものになり，やや興味が薄れる．

このため，言語生成の研究は，解析のそれに比べると，現在発展途上にあると言え，読者の皆さんに体系立てて説明することは困難である．そこで本章では，あえて，“心の中で思ったことを言語化する” という問題に挑戦し，この分野への読者の積極的な取組みの糸口としたい．なお，言語生成の研究全般については参考文献1)，2)を参照してほしい．

5.1 では，まず筆者らの提案している**心のモデル**を紹介する．このモデルでは，心の中の単位的な処理やデータをフレーム構造をもった “モジュール” で表現し，それらの**連鎖的な活性化**として心理活動をとらえる．これが “何を言うか” を決定する部分となる．次に，“どのように表現するか” の決定方法として，**5.2** で，寓話の世界を素材とし，一連の連鎖的活性化から文の意味的な構造（深層構造）の列を生成する方法を述べ，言語が心のどこから発生するか，に対する一つの考え方を示す．そして **5.3** では，深層構造から文の構文的構造（表層構造）を生成する方法について解説する．

5.1 心のモデルとダイナミックス

人間の心は，ソフトウェア的に表現すると，森羅万象に関する膨大なデータを背景にして，柔軟で適応性の高いプログラムが効率良く処理を進める，情報処理システムと言えよう．この巨大で複雑なシステムを解明するのは，容易なことではない．しかし巨大で複雑とは言え，そこにメカニズムが存在する以上は，長い時間をかけることにより，少しずつそれが明らかになるであろう．

5.1.1 心のモデル

まず，筆者らが従来から提案している，心のモデル化について述べよう[1]．

心の領域

心の中を “処理内容” に注目して9つの領域に分割してみよう．**図-1** に，それを示している．それらのうちいくつかは，いわゆる**知的処理** (intelligent processing) に関係している．ある外界刺激が知覚器で受理されると，その内容が認識（・理解）領域で意味理解される．外界の刺激としては，単に情景や物体のようなものだけでなく，音声などで表現された言語刺激もある．そのような刺激を意味理解の結果，それに反応するためのプランが企画（・創造）領域で練られる．続いてそのプランを実行するための物理的行動が行動（・表現）領域で組み立てられ，それが効果器に

† Introduction to Natural Language Processing: Generation of Sentence by Naoyuki OKADA and Jun-ichi NAKAMURA (Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology).

†† 九州工業大学情報工学部知能情報工学教室

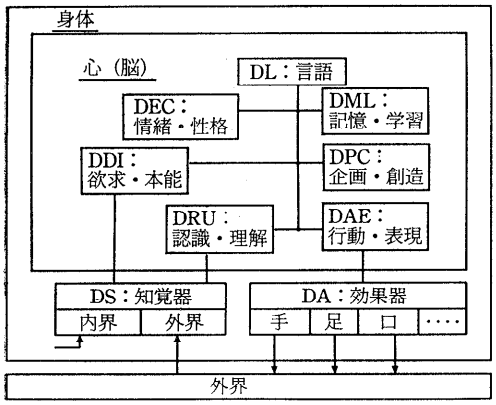


図-1 心の領域

送られて，実際に手足による行動となる．ときとして反応は言語表現のこともあろう．表現すべき内容が企画領域でプランニングされると，その内容は言語領域で言語に変換され，文章化される．得られた文章を発話するための動作が行動領域で組み立てられると，それに沿って効果器すなわち口や喉が音声を発生する．

一方，領域のあるものは，欲求や情緒の処理に関係している．図-1の知覚器には，内界からの刺激を受理する，もう一つの入力チャンネルがある．ここで“乾き”とか“空腹”といった内界刺激が知覚されると，欲求（・本能）領域で，“何か飲みたい”とか“何か食べたい”という欲求ないしは目標が生起する．これを実現するため企画領域でプランが練られ，引き続き行動領域，効果器の順で作業が進められる．当然その過程には，前述の知的処理過程も組み込まれるであろう．このように，知覚器における外部刺激が受動的にシステムを起動するのに対し，内部刺激は，能動的に起動する．また情緒（・性格）領域は，欲求領域と類似の機能をもち，プランニングや行動に大きな影響を与える．なお，記憶（・学習）領域は，自身をも含めたすべての領域からデータを受け取り，すべての領域にデータを供給する．もちろん個々の領域にもある程度のメモリが準備されてはいるが，記憶領域は大規模な，記憶専門の領域である．

[問1] 言語に関する刺激や反応には，音声すなわち話し言葉のほかに，文字すなわち書き言葉もある．書き言葉の理解と生成の過程を，上に述べた話し言葉のそれと比較しながら考察しなさい．具体的には，ある事柄Cについて，手紙を読んで返事を書く場面と電話で対話する場面とを想定し，比較しなさい．

解答例

手紙を読むときは，知覚器および認識領域でまず文字認識が行われる．文字は，音声と異なった物理的性質をもつので，この過程は対話における音声認識と大きく異なる．単語レベルまで認識が進むと，その後の構文および意味解析は，話し／書き言葉特有の表層構造は別として，かなり共通部分が出てくる．文脈解析になると，単方向の手紙と双方向の対話とでは，深層構造がかなり異なり，おのずと理解過程が異なるであろう．反応のためのプランニング過程も，文脈解析に影響を受けよう．しかし結果的には，ほぼ同じ事柄Cが理解される．

次に，反応すべき内容が言語領域で文章化される．大筋同じ処理でも，表層構造によるある程度の相違がでよう．最後に行動領域ならびに効果器での表現は，文字の場合手や腕の動作なので，口や喉とはまったく異なった動作になる．

それでは言語は，心の中でどのような役割を果たすのであろうか？ 言語の機能には，次に示すように，いくつかの側面がある．

（1） 心の中の任意のデータや処理を観察する．

（2） 心の中の任意のデータや処理を記述する．

（3） 高次の思考や推論をまとめる．

（4） 他の人と通信する．

これらの，いずれが本質的な役割を担っているのであろうか？ 筆者らは，言語の本質的役割を(2)の機能と考える．そうすると，他の三つの機能が次のように説明できる．(1)は，記述するためには知らなければならない，ということにより，(2)の前提条件といえよう．(3)について，一つの小さな出来事は文で記述され，一連の出来事は段落で，さらに大きな出来事は節，章といったように，より大きな言語単位で記述される．これは，取りも直さず，思考や推論のまとまりの単位でもある．また(4)は，記述の直接的結果と言える．

モジュール

心の中のデータや処理の単位を，ここでは**モジュール** (module) と呼ぼう．モジュールは，M.

```
事象 ([出る,
  上位 (移動する),
  下位 (あふれる_1, 降りる_1, ---),
  格フレーム (出る ([agt, Agt], [org, Org])),
  事象自身 ([
    存在 (Agt, [t0, t1]),
    存在 (Org, [t0, t1])
    内部 (Agt, Org, t0),
    移動 (Agt, [t0, t1]),
    外部 (Agt, Org, t1)])])
```

図-2　事象—出る

ミンスキーが“心の社会”という心理モデルにおいて提唱した,“それ自身は心をもたないミニプロセッサ, AGENT”とほぼ同じものである（参考文献3）参照）. 心の中には膨大な数のモジュールがある, と仮定しよう. おのおのは単純な機能しか有しないが, いくつか集まってある機能を発揮し, さらにそれらが集まってより大きな機能をもつ, ということを繰り返し, 全体として先に述べた9つの領域を構成する.

モジュールは, 原則として初回の**1.2.2**で述べたフレーム構造で表現することにしよう. 2, 3の具体例を, **図-2～図-4**に示そう. 次項での説明の都合上, 寓話の主人公の知識や情緒に注目している.

主人公は, 自然界を動き回る. 図-2は, “(Agtが時刻t0からt1にかけてOrgから）出る”という事象概念を表している. 特に“事象自身”のスロットでは, 出るという行為をいくつかの特徴で記述している. 主人公は, 自ら出るを実行したり自然界で生じた出るを認識したりするとき, このモジュールを使用する. たとえば認識のときは, 存在 (Agt, [t0, t1]) など5つの（概念的）特徴が認識領域における特徴抽出プログラムと結びついて, 調べている出来事が出るかどうかを知る. フレーム構造は, このように動的なプログラムと結びついて, しばしば動的に振る舞う.

また主人公は, 日々の生活において各種のプランを立てる必要に迫られる. 図-3は, **p水を探して飲む**というプランニングのときに参照するモジュールである.“格フレーム”スロットについては, **5.2**で述べよう.“目標”スロットは, 乾きの度合いをある値以下にすることを表している.“具体化”のスロットは, プランニングの具体的手法を述べている. この例では, 目標達成の緊急度ならびにプランの複雑さに応じて, 3段階に分けてプランが立てられる. ちなみにレベル_1は, とりあえず周囲に水が見い出されるならそれを飲む, という単純条件チェックの方式である. レベル_2は, 日頃なじんだ常套手段である. レベル_3は, 以上がうまくいかなかったときの, 深い推論による方式である. レベル_2の表現の中で記号“$”を用いているが, たとえば“$水_1”の場合ある特定の水を指示している.

```
プラン (〔p 水を探して飲む,
  格フレーム (
    立案する ([agt, Agt], [obj, [水を探して飲む]]),
    得る ([agt, Agt], [obj, Kekka])),
  目標 (程度 (Kawaki), 以下 (Sikiiti)),
  具体化 ([
    レベル_1 (もし周りに水があれば, それを飲む),
    レベル_2 ([
      [もし$住処にいるなら, p$水瓶_1の$水_1を
       飲む],
      [もし$野原にいるなら,
        [[p$住処に帰って, p$水瓶_1の$水_1を飲む],
         [p$橋の下に行って, p$小川の$水_5
          を飲む]]],
      [もし$山にいるなら, p$清水に行って, それを
       飲む]]),
    レベル_3 (
      i検索 ([容器, 水]),
      i検索 ([場所, 水]))])]
```

図-3　プラン—p水を探して飲む

主人公は, ときおり感情的になる. **図-4**に“悔しい”の概念を示そう. 悔しいは, かなり複雑な情緒であるが, それが生起する過程, 生起したときの心理状況, さらにはそれに基づく反応が記述されている. ただし, この例において欲求発生と目標設定は特に指定されておらず,“_”はどのような欲求もしくは目標でもかまわないことを意味している.

以上示したモジュールは, 種々の関係を通じて他のモジュールと結びついている. 事象概念“出る”は, 先に触れたように, 認識のための特徴抽

```
情緒 (悔しい,
  格フレーム (悔しい ([agt, Agt], [cas, CAUSE])),
  欲求発生 (_),
  目標設定 (_)
  プラン (成功の可能性大),
  実行 (多大の努力),
  結果 (後少しのところで失敗),
  状態 ([
    [驚き, 高い可能性にもかかわらず失敗],
    [怒り, 多大の努力にもかかわらず実現しなかった],
    [悲しみ, 欲求が満たされなかった]])
  反応 ([
    [不満を解消する]; [負け惜しみを言う];
    [八つ当たりする]; [悔し涙を流す]]))
```

図-4　情緒—悔しい

出プログラムや実行のための手足の動作プログラムと関係している．またプラン**p水を探して飲む**のレベル_2とレベル_3は，いくつかのサブモジュールから構成されている．ここには示されていないが，各サブモジュールはさらに小さなサブモジュールから構成されており，階層を成している．また悔しいについては，それを構成する要素的な情緒との関係が“状態”スロットに示されている．もちろん各要素的情緒は，それ自身一つのモジュールを構成している．

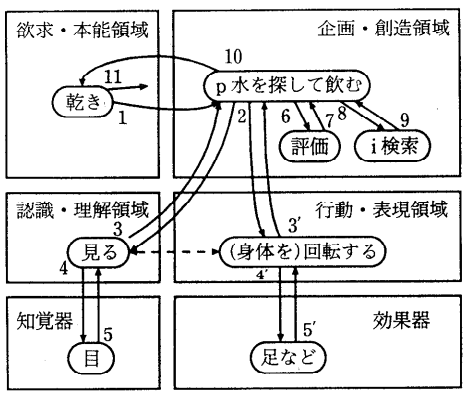
5.1.2 心のダイナミックス

一つの刺激があるモジュールに与えられたとする．そのモジュールは，刺激に対してなんらかの反応をするが，これを**活性化** (activation) と呼ぼう．活性化したモジュールは，直接関連する他のモジュールを活性化する．引き続きそれらに関連するモジュールの活性化が起きる．このようにして一連の活性化が，領域の内外に波及する．これを**心のダイナミックス** (mental dynamics) あるいはモジュールの**連鎖的活性化** (chain activation) と呼ぼう．

イソップ物語の“きつねとぶどう”に沿って，**図-5**を参照しながら，主人公（きつね）のダイナミックスを紹介しよう．簡単のため，演劇のシナリオふうに記述する．

初期状態　8月下旬の正午過ぎ，よく晴れて，暑い．主人公は，交差点の南 300 m の点にいる．とても喉が乾いているが，疲れはそれほどでもない．

欲求過程1　欲求領域において，乾きの生理状態が“しきい値（限界となる値）”を越えている．これを監視していた欲求モジュール“乾き”が，企画領域に喉の乾きを潤すプランニングを依頼する．条件は，“30 分以内で，危険をともなわない”，である．



注 1. 丸印はモジュールを示す．
2. 矢印は連鎖を示し，数字は順番を表している．

図-5 モジュールの連鎖的活性化（一部）

企画過程1　企画領域では，**p水を探して飲む**プランが起動される．**p水を探して飲む**は，取りあえず，レベル_1のプランの前提条件をチェックすることを，認識および行動領域に依頼する．

認識行動過程1　認識および行動領域は，それぞれ知覚器および効果器と協力して周囲を見渡すが，水は見当たらない．

企画過程2　そこで**p水を探して飲む**は，レベル_2のサブプランに焦点を当てる．現在野原にいるという条件のもとで，二つのサブプランが候補に上る．そこでこれらを評価モジュールに依頼して選択しようとするが，評価モジュールからは，いずれも 30 分以内では実行不可能の返事を得る．ただし“評価モジュール”は，時間性，存在性など5種類の尺度でプランを評価するモジュールであるが，ここでは深く立ち入らない．

次に**p水を探して飲む**は，レベル_3のi検索を起動し，記憶領域においてまず水を蓄えた容器を検索する．たとえば［$水瓶_1，$住処］が見つかり，そこへ行くまでの時間が評価されるが，条件を満たさない．検索されたすべての容器についても同様である．次に水を蓄えた場所を検索するが同じ結果である．

結局**p水を探して飲む**は，“条件を満たすプランが立てられない”，と欲求領域に回答する．

欲求過程2　乾きモジュールは，さらに生理状態が厳しい状況になっているので，条件を“多少危険であっても，急いで”に変更し，再度立案を依頼する．

……

企画領域では，最終的に“p$ 館の庭へ行く，p池を探す，pその水を飲む”という複合プランC1を立案する．C1は，認識および行動領域に転送され，実行の準備がなされる．その際，i検索で候補に上ったいくつかの対象，たとえば容器や水たまりや果物が参考データとしてC1に付け加えられる．

情緒過程1　情緒モジュール“悔しい”が起動し，C1の成否を見守る．

認識・行動過程1　認識領域では，与えられた経路(評価の過程で経路が計算されている)と時間から歩行の速度を算出する．また認識領域では，経路に沿って見えてくるであろう要所の場面を，記憶領域を検索して取り出す．

次に行動領域は歩行を開始する．認識領域は，入力画面が次第に変化するのに気づく．

……

主人公は，館に到着し中に入る．池を探すが，見つからない．そこで館の本館へ行って水瓶を探そうと，計画を変更する．途中庭の片隅で，偶然，ぶどう棚を発見する．何度も飛びつくが，あと少しでとどかない．情緒モジュール“意地”がさらに飛びつかせるが，欲求モジュール“疲れ”が中断させてしまう．最後に悔しいが，負け惜しみを言わせる．

5.2 深層構造の生成

心の中である連鎖的活性化が生じたとき，これに基づいて言語を生成することを考えよう．生成には，連鎖的活性化から深層構造を生成する段階と，深層構造から表層構造を生成する段階とがある．本節では，前段階の要点を述べるが，詳しくは文献 2)，3) を参照して欲しい．

連鎖的活性化が言語に変換される上で，特に次の三つが重要である．

(1)　話し手／書き手の意図と，聞き手／読み手の知識

(2)　文の**命題情報** (propositional information) の抽出と**法情報** (modal information) の付加

(3)　文章としての構造化

(1)は，自分の思いと相手の知識次第で何をどのように言語化するのかという，いわば枠組みを与えるものである．(2)に関して，一般に文のもたらす情報は“話し手／書き手が述べようとする内容”すなわち命題情報と，“その内容を述べる話し手／書き手の態度”すなわち法情報からなる．たとえば，“水が飲みたい”においては，“私が水を飲む”が命題情報であり，“たい”という願望が法情報である．非言語情報を言語情報へ変換するとき，命題情報は核であり，法情報がそれを修飾する．(3)は，一連のモジュールの活性化がどのように文脈構造に変換されるか，という問題である．以下では，(2)と(3)を中心に議論し，(1)については，主人公が自己の現状に関して独り言を述べる，という程度の，ゆるい枠組みにしておこう．

5.2.1 命題および法情報

命題部分の表現には，しばしば深層格フレームが用いられる．心の中のデータや処理の単位はモジュールで，それはおおむねフレーム構造で表現された．それらのモジュールが連鎖的に活性化したとき，何をどのように命題としてとらえ，深層格フレームとして表現すればよいのであろうか？

初めに，モジュールを大きく“動的”と“静的”の2種類に分けよう．プランや行動モジュールは，それが活性化したときある動きもしくは振舞いをともなうので，前者に属する．それに対して物や場所モジュールはそれ自身自立して振る舞うことはなく，動的モジュールに付属して活性化するので，後者に属する．そこで動的モジュールに注目し，その活性化を記述しよう．活性化の記述にはいくつかの観点があり得るが，特にモジュールの“振舞い”と，それによって生じる“結果”に焦点を当てよう．たとえば，図-3 で示したプランモジュール**p水を探して飲む**の場合，次のような記述が考えられる．

振舞い：立案する ([agt, Agt], [obj, [水を探して飲む]])

結果：得る ([agt, Agt], [obj, Kekka])

ここで変数 Agt には当然自分自身を表す $ 自身が代入され，また Kekka には，プランニング終了後，たとえば前記複合プラン C1 が引き渡されている．

動的なモジュールには，このような“格フレーム”のスロットを準備して，2種類の格フレームすなわち述語で，そのモジュールの活性化を表現することにしよう．そしてある動的なモジュールの活性化にともなっていくつか静的なモジュールも活性化したとき，それらは格要素として格フレームの中に埋め込むことにしよう．

一方，法情報は，動的モジュールの活性化を取り巻く状況に依存する．たとえば，欲求モジュールが欲求領域で何かを訴えたときは“願望”，プランモジュールが企画領域でプランを模索するときは“検討”，といった法情報が付加されよう．以下では，命題Pとそれに付加された法情報Mの対を“P&M 対”と呼ぼう．なお一つの P に対し，一般には複数の法情報からなるMが付加される．た

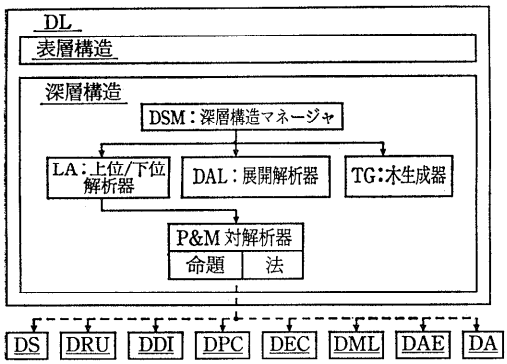


図-6 言語領域の主要モジュール

とえば,“水を飲みたいらしい”の場合，願望と推量が付加される.

5.2.2 生成過程

初めに，言語の**線形性**を議論しよう．ここで線形性とは，言語が1次元の記号列で表現されることを指すが，一般に心理的活動はいくつもの領域で並列的に起きる．たとえば，主人公が移動する際は，認識と行動の領域が互いに情報を交換しながら歩を進める．一方，込み入ったプランニングなどのときは，単一の連鎖的活性化が生じる．そこで大きく，次のように考えよう．深い推論を必要とするときは単一の連鎖が起こり，よく慣れた日常的作業の場合は並列の連鎖が可能である．そして並列の連鎖が起きたばあいは，それらを束ね，表層構造を生成する段階で必要に応じて解きほぐすものとしよう.

言語領域の深層構造部門には，**図-6** に示すように，5つの大きなモジュールがある.

P & M 対解析器 (PMA)

心は，単位時間ごとに動作するものとしよう．PMA は，すべての領域で生じた動的モジュールの活性化に注目し，単位時間ごとに，それらからP&M 対を抽出する.

上位／下位解析器 (LA)

プランモジュールからも理解されるように，一般にモジュールは，そのサブモジュール，そのまたサブモジュールというように階層をなしている．また二つのモジュールが,“日照りが1週間続く”なら“野原の水溜まりがなくなる”のように，IF-THEN の関係でつながっていることもある．ここでは，便宜上，IF-THEN のような連鎖も上位 (IF)，下位 (THEN) とみなして，階層でとらえよう.

一連の P&M 対が与えられたとき，LA は二つの対の間の上位／下位を検出し，部分木を作成する．並列の連鎖が生じているときは，それぞれにおいて部分木を作る.

展開解析器 (DAL)

DAL は，活性化の連鎖の遷移を扱う．たとえば，企画領域での一連のプランニングが失敗すると，一度欲求領域で条件が練り直され，再び企画領域でプランニングが行われる．この場合は，三つの連鎖が続く．一方，たとえば移動の際は，認識領域における路面や目標物などの認識と，行動領域における歩行や方向転換などの動作が交互ないしは同時に行われる．この場合は，二つの連鎖が関連しながら並列的に行われる．そこで DAL は，一連の P&M 対が一つのまとまりをもった部分木かどうかを判断し，まとまりのある部分木を発生順に並べる．並列連鎖の場合は，それぞれにおいて部分木を作成し，一つに束ねておく.

なお DAL の処理結果は，後に表層構造として段落などを構成するキーとなる.

木生成器 (TG)

部分木を発生順に結合し，一つの木構造にする．第3回の **4.4** で物語文法に触れたが，この木構造はそれに近い構造をもっている．生成木は通常かなり大きなものになる．そのため，要約作業に相当する“枝刈り”が必要になる．これについては，文献 4) を参照して欲しい.

深層構造マネージャ (DSM)

上記4つのモジュールを管理する．かくして，活性化したモジュールの上位／下位を反映した部分木と，それらを発生順に並べた，P&M 対の木が生成される.

5.2.3 生成例

それでは，上記過程に沿って生成した深層構造，すなわち主人公の独り言の内容を示そう．**図-7** に，深層構造としての P&M 対の一部を示している．同図においてP&M対は，原則として[P, M]の形式で表現されており，たとえば2行目の対は，

P：ある ([state, 温度], [time, 今日], [dgr, 30度])

M：ϕ(空)

であることを表している．最後に，生成木を変換して得られる文章を，参考までに示しておこう.

```
p&m 対 ([
1. [[ある ([state, 温度], [time, 今日], [dgr, 30 度]), φ],
   [いる ([agt, $ 自身], [loc, 獣みちの南 300 m の交差
    点]), φ],
2. [[乾く ([agt, $ 自身], [dgr, 程度_8]), 状態],
   [潤す ([agt, $ 自身], [obj, 喉の乾き],
         [dgr, 危険度_2, 30 分以内]]), 願望]],
3. [[探す ([agt, $ 自身], [obj, 水]),
     飲む ([agt, $ 自身], [obj, 水])], 検討],
   [[探す ([agt, $ 自身], [obj, 水], [loc, 自分の周り]),
     調査],
    [見つける ([agt, $ 自身], [obj, 水]), 否定]],
   [[いる ([agt, $ 自身], [loc, 野原], [time, 今]), φ],
    [[帰る ([agt, $ 自身], [goal, $ 住処]),
      飲む ([agt, $ 自身], [obj, $ 水瓶_1 の $ 水_1])],
      検討],
     [ある ([loc, $ 住処], [dgr, 遠い]), 評価],
     [諦める ([agt, $ 自身], [obj, 住処に帰る]),
      決定]],
    [[行く ([agt, $ 自身], [goal, $ 橋の下]),
      飲む ([agt, $ 自身], [obi, $ 小川の水_5])],
      検討],
     [ある ([loc, $ 橋], [dgr, 遠い]), 評価],
     [諦める ([agt, $ 自身], [obj, 橋の下に行く]),
      決定]],
   [考える ([agt, $ 自身], [obj, 他の方法]), 検討],
   ------])
```

注. 数字は，本文の生成文章の，段落を示す数字と対応している.

図-7 生成された深層構造—P&M 対

詳しくは，文献 3) を参照願いたい.

1. とても暑い日だ. 今，交差点から 300 m 手前の獣みちにいる.

2. とても喉が乾いている. 喉の乾きを潤したい. 早く潤したい. 危険なことをせずに潤したい.

3. 水を探して飲もうかなあ. もし周りに水があれば，それを飲もう. 水がない. 今，野原にいる. 住処に帰ろうかなあ. 住処は遠い. 住処に帰るのはやめた. 橋の下に行こうかなあ. 橋の下は遠い. 橋の下に行くのはやめた. 何か他の方法を考えよう.

4. 水のある場所を探そうかなあ. 池があった. 池を探そうかなあ. 池にはB池があった. B池はイソップワールドにあった. B池の近くには猟師小屋があった. 多分猟師小屋には猟師がいるだろう. 猟師は危険だ. B池に行くのはやめた.

5. 水分の多い食物を食べようかなあ. 果物を探して食べよう. いちごを食べようかなあ. いちごは春に実った. 今夏だ. いちごを食べる季節ではない. いちごを食べるのはやめた.

.....

注. 第3段落までが図-7 の P&M 対から生成されたものであり，段落の番号は図-7 の番号と対応している.

以上，AI の立場から，心のダイナミックスに基づいて言語の深層構造を生成する一つの考え方を示した. 最近ヒューマンインタフェースにおいて，画像や音声など“マルチメディア (multi-media)”ないしは“マルチモーダル (multi-modal)”な環境での自然言語処理が注目を浴びているが，上記の考えに従って物語からアニメーションを生成したり，情感に応じた音楽を生成したりする試みもなされている[5),6)].

5.3 表層構造の生成

深層構造の生成結果の中心は，前節で示したように，命題情報であるので，これを実際の文にする方法について考えてみよう. 文を組み立てるには，まず，“深層構造”と呼ぶ構文構造を作り，その結果に対して適切な語形変化を与え，文字の列とすればよい. これは，**3.**, **4.** で説明した形態素解析と構文解析の逆を実行することになる.

文の生成の出発点である命題情報と第2回の **2.2.2** で説明した格文法の格構造は，厳密には異なるものであるが，ここでは簡単のため，同じものとする. 第3回の **4.3.1** で説明した自然言語インタフェースを思い出してほしい. “taro drives a sports-car” という英文を解析した結果は，

[drive, [r_taro, r_sports_car]]

であった. これは，命題 (predicate) が driveで，動作主 (agent) が r_taro 対象 (Object) が r_sports_car という格構造を表している. これを入力の深層構造としよう.

この格構造から“太郎はスポーツカーを運転する”という日本語文を生成することを考える. これがうまくいけば，機械翻訳システムを作ることにもつながる. このことは一見簡単そうにみえるが，プログラムで実現するためには，次の三つのことが問題となる.

1. drive という命題には，“運転する”，“操縦する”などいくつかの動詞が対応するが，どれを選択するか. 翻訳の場合であれば，英単語の drive には，“駆動する”などの日本語も対応するので選択はより複雑になる.

2. どのような順序で単語を並べるか. その際構文的に正しいことを保証しなければならない. “スポーツカーを太郎は 運転する”は許容範囲であるが，“運転する太郎はスポーツカーを”は誤

りである.

3. 文中の位置や文脈によっては，こまかな調整が必要である．たとえば，“私は，太郎がスポーツカーを運転するのを見た”のように，“は”を“が”にしなければならない場合もあるし，“太郎は，(自分が) スポーツカーを運転する夢を見た”のように，省略するか，“自分”という語を用いる必要がある場合もある.

これらの問題は，あるものは複雑な処理メカニズムを用いることで解決されているし，あるものは研究段階で未解決のものもある．以下では，2を中心に説明しよう.

5.3.1 DCG による文生成

深層構造から文を生成する基本的な方法は，表現したい内容を述語を一つだけ含む単文の単位に分解し，次に，その中の個々の名詞や動詞を決定し，全体として調整する，ということになる．これを簡単に行う方法として，**4.** で説明した確定節文法 (DCG) を用いる方法がある．実は，DCG は文を生成するためにも用いることができるのである．これは，DCG を実行する Prolog が**計算の双方向性**と呼ばれる性質をもつことによる.

図-8 に計算の双方向性の例を示そう．(a)は二つのリストを結合するプログラム append である．このプログラムは(b)に示すように，引数の1番目と2番目で与えたリスト（入力）を繋げて，その結果を3番目の引数で指定した変数Rに渡す(出力)．これは，通常のプログラミング言語と同じ使い方である．しかし，(c)に示すように“出力”であるはずの第3引数に具体的なリストを与えると，結合した結果がそのリストになる，二つのリストのすべての組合せ（図の場合であれば X，Y について5通り）を計算してくれる．つまり，同じプログラムがリストを結合するだけでなく，分解するためにも利用できる．このように，入力と出力を逆にできることを**計算の双方向性**という．複雑な Prolog プログラムでは，このようにうまくはいかないが，簡単な DCG の文法であれば，可能である.

```
append ([], Y, Y).
append ([A|X], Y, [A|Z]) :- append(X, Y, Z).
```

(a) ふたつのリストを結合するプログラム append

```
| ?- append([taro, drives], [a, sports_car], R).
R = [taro, drives, a, sports_car] ? ;
yes
```

(b) append をリストの結合として実行した場合

```
| ?- append(Y, Y, [taro, drives, a, sports_car]).
X = [],
Y = [taro, drives, a, sports_car] ? ;

X = [taro],
Y = [drives, a, sports_car] ? ;

X = [taro, drives],
Y = [a, sports_car] ? ;

X = [taro, drives, a],
Y = [sports_car] ? ;

X = [taro, drives, a, sports_car],
Y = [] ? ;
no
```

(c) append をリストの分解のために用いた場合

図-8 Prolog における計算の双方向性

第3回の図-14 の DCG の文法は，英文を解析して [drive, [r_taro, r_sports_car]] という格構造を求める部分と，その結果をデータベースに登録する部分からできていた．データベースに登録する部分を除くと，計算の双方向性を利用することにより，このままで英文を生成することができる．しかし，ここでは英語ではなく日本語の生成を考えてみよう．第3回の**4.** で示した英語の解析システムとここで説明する日本語の生成システムを結合すれば，“おもちゃ”の機械翻訳システムとなるわけである．もちろん，実際に機械翻訳システムを開発しようとすると，もっと難しい問題に遭遇するので，新たな工夫が必要である.

それでは，この例の格構造から日本語の文を生成する DCG 規則を，**図-9** に示そう．これを文の生成の立場から説明すると次のようになる．文 s は主語 pp_subj と目的語 pp_obj と動詞 v に分解する．主語，目的語はさらに名詞 n と格助詞 p に分解する．この場合，主語，目的語，動詞は何でもよいというわけではなく，引数を用いてど

```
s([Predicate, [Agent, Object]]) -->
        pp_subj(Agent), pp_obj(Object), v(Predicate).
pp_subj(Agent) --> n(Agent), p(subj).
pp_obj(object) --> n(Object), p(obj).

n(r_taro) --> [太郎].
n(r_sports_car) --> [スポーツカー].

p(subj) --> [が].
p(obj) --> [を].

v(drive) --> [運転する].
```

図-9 DCG による文生成のための規則

```
| ?- s([drive, [r_taro, r_sports_car]], S, []).
  1 1 Call: s([drive, [r_taro, r_sports_car]], _128, []) ?
  2 2 Call: pp_subj(r_taro, _128, _302) ?
  3 3 Call: n(r_taro, _128, _424] ?
  4 4 Call: prolog: 'C'(_128, 太郎, _424) ?
  4 4 Exit: prolog: 'C'([太郎|_424], 太郎, _424) ?
  3 3 Exit: Exit: n(r_taro, [太郎|_424], _424) ?
  5 3 Call: p(subj, _424, _302) ?
  6 4 Call: prolog: 'C'(_424, が, _302) ?
  6 4 Exit: prolog: 'C'([が|_302], が, _302) ?
  5 3 Exit: p(subj, [が|_302], _302) ?
  2 2 Exit: pp_subj(r_taro, [太郎, が|_302], _302) ?
...
  12 2 Call: v(drive, _309, []) ?
  13 3 Call: prolog: 'C'(_309, 運転する, []) ?
  13 3 Exit: prolog: 'C'([運転する], 運転する, []) ?
  12 2 Exit: v(drive, [運転する], []) ?
  1 1 Exit: s([drive, [r_taro, r_sports_car]],
        [太郎, が, スポーツカー, を, 運転する], []) ?

S = [太郎, が, スポーツカー, を, 運転する] ?

yes
```

図-10 DCG による文生成の例

のようなものかを指定し，最終的に，“太郎”や“運転する”を選択する．また，格助詞 p も主語 (subj) なら“が”，目的語 (obj) なら“を”を選択する．

この DCG 規則に先の格構造を入力して文を生成する過程は**図-10** のようになる．手近で Prolog 処理系が使えればぜひ試してほしい．まず，文 s から主語 pp_subj が呼び出される (Call)．この場合，主語としては r_taro という“指示対象としての太郎”が指定されている．主語から名詞 n が呼び出され，日本語の語“太郎”が取り出される (prolog : 'C')．最初の語“太郎”が決定できたので，名詞 n に戻る (Exit)．次に格助詞“が”を選択し，主語の部分“太郎が”の決定が終了する．最終的に動詞 v を“運転する”に決定し，文全体の生成が終了する．格構造を主語や動詞に分解しながら単語を選択し，並べていることが分かるであろう．

[問 2]第 3 回の図-10 に示した解析の状態と図-10 とを比較してみよ．

解答例

英語と日本語である点，図-10 では格構造を表す引数が多い点と，そこに始めから具体的な値が与えられている点を除けば，全体の流れは非常に似ている．DCG の処理方式である第 3 回の図-4 のトップダウン解析の流れは，見方を変えれば，文生成そのものになる．

5.3.2 単一化文法の利用

第 3 回の **4.3** でも示したが，DCG の引数だけではこまかな文法記述をするには無理がある．そこで，単一化文法 (UG) を生成に用いることを考えてみよう．図-9 を少し拡張して UG で表現すると**図-11** のようになる．拡張点は，二つある．一つは，主語，目的語といった区分を pp_subj, pp_obj のようなカテゴリの区別ではなく，1 の規則にあるように case という素性で区別している点である．もう一つは，politeness という素性により，表現の“丁寧さ”のレベルを扱えるようにしている点である．

文生成の例を**図-12** に示す．例題の格構造を素性構造 (Feature Structure) で表現すると図中の Input FS のようになる．これは，第 3 回に示したものと同じである．生成結果の一つが Output FS のようになり，これに語の列“[太郎, さん, が, スポーツカー, を, 運転される]”が対応することになる．

まず，主語，目的語の区別がどのようにして実現されているか調べてみよう．図-11 の規則 1 で，F@subj = F1 により，主語が最初に来ることが，F1@case = subj により，主語の FS F1 には，case という素性名の値として subj がなければならないことが示されている．確かに，図-12

```
1 s(F) --> pp(F1), { F@subj = F1,
                     F1@case = subj,
                     F1@politeness = F@politeness },
           pp(F2), { F@obj = F2,
                     F2@case = obj,
                     F2@politeness = F@politeness },
           v(F3), { F = F3 }.
2 pp(F) --> np(F1), { F = F1 },
            p(F2), { F = F2 }.
3 np(F) --> prpn(F1), { F = F1 },
            suffix(F2), { F = F2 }.
4 np(F) --> n(F1), { F = F1 }.
5 suffix(F) --> [さん], { F@politeness = 1 }.
6 suffix(F) --> [], { F@politeness = 0}.
7 prpn(F) --> [太郎], { F@sem@lex = r_taro }.
8 n(F) --> [スポーツカー], { F@sem@lex = r_sports_car }.
9 p(F) --> [が], { F@case = subj }.
10 p(F) --> [を], { F@case = obj }.
11 v(F) --> [運転する], { F@sem@predicate = drive,
                          F@subj@sem = F@sem@agent,
                          F@obj@sem = F@sem@object,
                          F@politeness = 0 }.
12 v(F) --> [運転される], { F@sem@rel = drive,
                          F@subj@sem = F@sem@agent,
                          F@obj@sem = F@sem@object,
                          F@politeness = 1}.
```

図-11 UG を用いた文法規則

```
*** Input FS:
F0| sem: F1| predicate: drive                  | |
  |         | agent: F2| lex: r_taro |         | |
  |         | object: F3| lex: r_sports_car | | |

*** Output FS:
F0| sem: F1| predicate: drive                  | |
  |         | agent: F2| lex: r_taro |         | |
  |         | object: F3| lex: r_sports_car | | |
  | subj: F4| sem: F2      |                    |
  |         | politeness: 1 |                   |
  |         | case: subj    |                   |
  | politeness: 1                              |
  | obj: F5| sem: F3      |                     |
  |        | case: obj    |                     |
  |        | politeness: 1 |                    |

*** Output Japanese:
[太郎, さん, が, スポーツカー, を, 運転される]

S = [太郎, さん, が, スポーツカー, を, 運転される] ? ;
```

図-12 文生成の実行例1

の出力の FS 中の F4 には case: subj がある．次にこの FS は，規則2をみれば，格助詞pと単一化することになるので，規則の9と 10 が単一化の対象となるが，単一化可能なのは 9 の“が”のほうだけであるので，最終的に主語の格助詞“が”が選択される．解析の場合と同様，条件を満たすものを決定するのに単一化がうまく利用されている．

それでは，“丁寧さ”のレベルはどのようになっているのであろうか．**図-13** に別の生成結果“[太郎，が，スポーツカー，を，運転する]”を示す．図-12 と図-13 の出力の FS を比較すると，文全体，subj および obj の FS 中の politeness の値が前者はすべて1で後者はすべて0になっていることが分かるであろう．規則の 11 では，“運転する”は politeness が0のものと単一化しなければならないことが示されているので，図-13となる．

```
*** Output FS:
F0| sem: F1| predicate: drive                  | |
  |         | agent: F2| lex: r_taro |         | |
  |         | object: F3| lex: r_sports_car | | |
  | subj: F4| sem: F2      |                    |
  |         | politeness: 0 |                   |
  |         | case: subj    |                   |
  |politeness: 0                               |
  | obj: F5| sem: F3      |                     |
  |        | case: obj    |                     |
  |        | politeness: 0 |                    |

*** Output Japanese:
[太郎, が, スポーツカー, を, 運転する]

S = [太郎, が, スポーツカー, を, 運転する] ? ;

no
```

図-13 文生成の実行例2

また規則の 12 から“運転される”は，図-12 のほうとなる．“さん”については，規則の3で固有名詞 prpn には接尾語 suffix の付くことが示されている．さらに規則5で politeness が1のときは“さん”が付くが，規則6で0のときは何も付かない（[]）ことが示されている．これらにより，“太郎さんが”と“太郎が”の二つの表現が出力される．

[問3] 図-11の文法では，“太郎さんがスポーツカーを運転する”や“太郎がスポーツカーを運転される”のように，丁寧さに関して少し不自然な文は生成されない．なぜか．また，一つの格構造に対して二つの文が生成されるのはなぜか．

解答例：

素性の politeness は規則1で文全体で共通な値になるように単一化される．規則の上では3行目の Fl@politeness = F@politeness と6行目の F2@politeness = F@politeness，および7行目の F = F3 により記述されている．特に最後の F = F3 という単一化の指定により，主語や目的語の politeness だけでなく動詞の politeness も一致しなければならないことになる．このため，名詞と動詞で politeness の値が異なる“太郎さん(1)が運転する(0)”や“太郎(0)が運転される(1)”は単一化に失敗し，生成されない．これは，第3回の図-17 の英語の文法で主語と動詞の数の一致を条件としたのと同様である．このように，politeness は，文全体としては同じ値でなければならないが，入力の格構造ではどちらとも指定されていないので，条件を満たす二つの文が出力される．

最後に，生成のための規則と解析のための規則の関係について補足しておく．**4.** で示した DCG と UG による解析規則は，実はそのままで生成にも利用できる．また，本章で示した生成規則をそのままで解析に利用することもできる．うまく組み合わせれば，英日翻訳システムになるだけでなく，日英翻訳システムにもなるのである．このように，解析にも生成にも利用できる文法を**reversible 文法**と呼ぶ場合がある．実験としては面白いので，Prolog の処理系が手元にあればぜひ試してもらいたい．もし，大規模で実用的な reversible 文法が作成できれば，二つの文法規則を開発する必要がなくなるので実用上も有効であ

ろう．しかし，工学的立場から筆者らは，(1)解析の場合はより広い範囲の文，場合によっては，文法的に少しおかしい文でも解析できなければならないが，生成の場合は文法的に正しく自然で分かりやすい文のみを生成したいこと，(2)次項で述べるような自然な文の生成は解析とは異なる技術が必要であること，などの点から必ずしもreversible であることにこだわる必要はないと考えている．

5.3.3 自然な文の生成

実際に複雑な文を生成しようとすると，まだまだいろいろな工夫が必要になる．この節の最初にもふれたが，具体的にどの語を用いるかは簡単には決められない．異なった語の間には微妙な意味や用法の差があるが，その微妙な差をプログラムで判断できるようにすることは容易でない．また，語に限らず，たとえば“手／に／入れる”という句と“入手する”という動詞は意味的にはよく似ているが，どちらを選択すべきかは，状況や文法的な制約から決めなければならない．このように意味的に類似した表現もたくさんあり，どれを選択すべきかは簡単には決定できない．さらに，日本語の助詞の“は”と“が”の使い分けや，英語の前置詞の at と in の使い分けなども難しい問題である．

文の構文構造の点からも工夫が必要である．短い文であればあまり問題ないが，複数の命題が，複雑に入り組んでいる場合は，1文であっても，どの部分からどのような順序で文を組み立てればよいかは簡単ではない．場合によっては1文ではなく2文以上に分割したほうがよいこともあろう．これは人間が分かりやすい文章を作成する場合とまったく同様の問題である．また，文が複雑になって代名詞などの参照表現が表れた場合にどのような表現をするか，あるいは省略するかも考えなければならない．たとえば，“太郎がスポーツカーを運転しているときに（そのスポーツカーが）故障した”であれば，“故障したほうのスポーツカー”を省略したほうがよいが，“スポーツカーが故障したのは太郎が（そのスポーツカーを）運転しているときだった”であれば“運転しているほうのスポーツカー”を省略したほうがよい．

5.2 で文は命題情報だけでなく，法情報からも構成されていることを示したが，さらに文が用いられる状況や聞き手に依存して決めなければならないこともある．たとえば，日本語の敬語は，話し手と聞き手のその場での身分の関係に依存して適切に使い分ける必要がある．“太郎さんが運転された”は太郎に対しては敬意が払われているが，もし，聞き手が話し手よりも目上であれば，“運転されます”と丁寧な表現を用いなければならない[7]．また，(1)“これはスポーツカーです”，(2)“これはスポーツカーですよ”，(3)“これはスポーツカーですね”は，命題としてほぼ同じであるが，話し手と聞き手がもつ知識の差に依存して微妙な差がある．(1)は中立的な表現であるが，(2)は相手が知らないと話し手が判断した場合に，(3)は相手が知っていると話し手が判断した場合に用いられる．このような微妙な言い回しまでを考慮した自然な文を生成する方法については，まだまだ研究が必要である．

5.3.4 その他の手法：システミック文法

文生成の他の手法としては，M. ハリデーの**システミック文法**[8]を用いるものがよく知られている．システミック文法は，句構造規則ではなく，**図-14** に示すような一種の弁別ネットワーク（場合分けの仕方をグラフで記述したもの．枝分れの部分に条件を書いておく．）の集合により文法を記述する．図の右側の Finite（定形），Indicative（叙述），Imperative（命令）と書かれた部分が“システム”と呼ばれるネットワークである．この部分は，

> （英語の）定形の文は，叙述文か命令文の形式のどちらかである．叙述文の場合は主語（subject）が必要である．

という，一種の構文的な可能性を記述している．このようなネットワークを組み合わせることにより文を組み立てる方法を表現する．

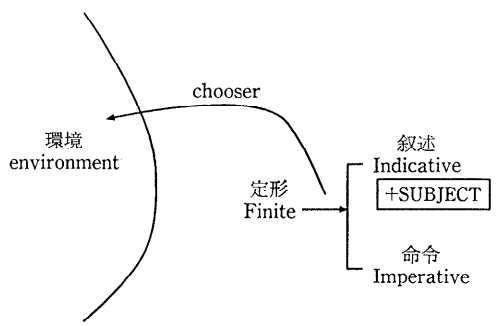


図-14 システミック文法における生成方法

実際に文を生成する場合は，個々のネットワークに付加する“chooser”と呼ばれるメカニズムを用いて，どれを選択するかを決定する．“chooser”は，構文的な選択に必要な条件を“environment”と呼ばれる“文を生成する環境”に問い合わせる．図のネットワークの場合であれば，“相手に行動を要求したいがどうか”といった問合せをすることになる．“environment”は，文を生成するために必要な情報をすべて所有しているとする部分で，このような問合せに答えるものである．たとえば，**5.1** で述べた心のモデルもその一つの表現形式と考えてよい．ネットワークが必要な問合せを終了した段階で必要なすべての情報が収拾され，文が生成される．

表層構造の生成に関する研究・開発は，機械翻訳やヒューマンインタフェースなどの作成には必要であるので，多く行われているが，体系的にまとめられているものは少ない．しかし，システミック文法を用いた大規模なシステムとして，M. マシューセンらが開発した Nigel などは，詳細が発表されており，また，英語だけでなく日本語への適用例もあり[9]，興味深い．

これまで，自然言語を計算機で扱ういろいろな手法について説明してきた．これらの手法の中心は文法やその処理メカニズムであるが，すべて個個の規則や語に関するいろいろな情報を参照することにより動作する．これらに関する情報を蓄えておく規則ベースや辞書がなければ，どのように細かな話をしても機械は動かない．最終回では，言語の機械処理のすべての部分で必要となる文法と辞書の作成方法について考えてみよう．

文　献

1) 岡田直之：情報科学からみた心のモデルとダイナミックス，脳と精神の医学，Vol. 2, No. 2, pp. 433-448 (1991).
2) 岡田直之：心と言語，BME（医用電子と生体工学），Vol. 7, No. 8, pp. 67-76 (1993).
3) Okada, N. and Endo, T.: Story Generation Based on Dynamics of the Mind, Computational Intelligence, Vol. 8, No. 1, pp. 123-160 (1992).
4) 岡田直之，滑　純子：イソップワールド—思考および行動過程に基づく物語の構造表現と要約，人工知能学会研究会資料 SIG-KBS-8805 (2/2), pp. 61-69 (1991).
5) Noma, T., Kai, K., Nakamura, J. and Okada, N.: Translating from Natural Language Story to Computer Animation, Proc. of 1st Singapore Int'l Conf. Intell. Syst., pp. 475-480 (1992).
6) Nakamura, J., Kaku, T., Noma, T. and Yoshida, S.: Automatic Background Music Generation Based on Actor's Emotion and Motions, Proc. of 1st Pacific Conf. Comput. Graph. Appli., pp. 147-161 (1993).
7) 中村順一，大庭美香，甲斐郷子，吉田　将：日本語文生成における待遇表現の取り扱いについて，情報処理学会第44回全国大会 (1992).
8) Halliday, M.: An Introduction to Functional Grammar, Edward Arnold (1985).
9) Matthiessen, C. and Bateman, J.: Text Generation and Systemic-Functional Linguistics: Experiences from English and Japanese, Printer Publishers (1991).

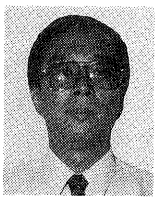
参　考　文　献

1) Zock, M. and Sabah, G.: Advances in Natural Language Generation, Vol. 1 and Vol. 2, Ablex Publishing (1988).
言語生成に関する論文集．言語生成の問題点，各種手法に関する紹介がまとめられている．
2) 徳永健伸，乾健太郎：1980年代の自然言語生成 1-3 人工知能学会誌，Vol. 6, No. 3, pp. 380-387, No. 4, pp. 510-519, No. 5, pp. 651-662 (1991).
言語生成に関して網羅的によくまとめられた数少ない日本語の文献．1980年代の言語生成の研究の全体像がよく分かる．
3) Minsky, M.: The Society of Mind, p. 339, Simon and Schuster, New York (1985).
安西祐一郎訳：心の社会，p. 574, 産業図書 (1990).
AGENT と呼ばれるミニプロセッサを単位として，心の中の種々の現象をボトムアップ的手法でとらえている．
4) Johnson-Laird, P. N.: The Computer and the Mind—An Introduction to Cognitive Science, William Collins Sons & Co., Ltd. (1988).
海保博之他訳：心のシミュレーション，p. 428, 新曜社 (1989).
認知科学への入門書で，種々の心的現象を計算機処理の立場から丁寧に解説している．

（平成5年10月18日受付）

岡田　直之（正会員）

1964年東海大学工学部卒業．1966年九州大学大学院工学研究科修士課程修了．同年同工学部助手，1976年大分大学工学部助教授，1978年同教授を経て，現在九州工業大学情報工学部教授．工学博士．人工知能の研究に従事．電子情報通信学会，人工知能学会各会員．



中村　順一（正会員）

1979年京都大学工学部卒業．1982年同大学院工学研究科博士後期課程中退．同年京都大学工学部助手，1989年九州工業大学情報工学部助教授．工学博士．自然言語処理，音楽情報処理の研究，計算機ネットワークの管理に従事．電子情報通信学会，ソフトウェア科学会，日本認知科学会，Association for Computational Linguistics 各会員．