ICOT Technical Memorandom: TM-1072

TM-1072

情報検索における柔軟な対話制御方式

大平　栄二、安部　正博
小松　昭男、市川　熹（日立）

July, 1991

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

情報検索における柔軟な対話制御方式

正員　大平　栄二　　非会員　阿部　正博　　正員　小松　昭男　　正員　市川　熹

Flexible Dialog Control on Information Retrieval

Eiji OHIRA ,Member, Masahiro ABE ,Nonmember, Akio KOMATSU ,Member
and Akira ICHIKAWA,Member

あらまし　本論文では，自然言語によるデータベース検索を主なタスクとするシステムを対象として，制限の少ない柔軟なユーザインタフェースを実現するための対話制御の枠組みについて述べている．ここでは，（1）まず対話管理の有効な方式の1つである対話対に基づいた対話管理法を，対話対間の結合関係をも統一的に扱うことが可能な枠組みに拡張することにより，利用者とシステム間の対話の主導権を柔軟に制御することを可能とした．（2）さらに，ATMSをベースとした非単調推論を適用することにより，利用者からの入力の解釈を誤った場合にも，その修復が可能な対話制御の枠組みを示した．これにより，効率的で，対話性の優れた検索システムを実現可能である．

1．まえがき

パーソナルコンピュータなどの普及に伴い，誰もが容易に計算機を利用可能なユーザフレンドリーなインタフェースが重要となってくる．このようなインタフェースを実現するための要件の１つとして，まず柔軟な理解能力が挙げられる．すなわち，利用者の要求の一を聞いて十を知る理解能力（利用者主導型の対話処理能力）である．これを実現するためには，入力文中の省略などを文脈情報や常識を用いて補足するための理解能力が必要となる．対話においては，特に文脈情報の管理が重要であり，自然言語を用いた各種の対話の研究[1]～[4]が進められている．もう１つの要件として，柔軟なガイダンス能力がある．タスクの内容をよく把握しているシステムが，主導的にガイドし対話を進めることにより，利用者の要求を満たす解を効率良く求めることができる[7]．例えば，旅行代理人のエキスパートシステムは，旅行プランなどの効率のよいガイダンスが可能であろう．このようなシステムでは，例えば，利用者の応答の内容に応じて，システムからの質問内容を変えることにより，効率の良い検索を実現する．しかし，利用者主導の対話を柔軟に制御するための汎用的な枠組みは持っていない．

このように，利用者主導型およびシステム主導型の対話の各々についての研究は進んでいるが，実際の対話では，この両者の混在した対話が生じる．例えば，利用者はシステムの質問に答えている途中で，その質問に答えるために，逆にシステムに質問するようなことがよくある．すなわち，入れ子構造のネストした対話が生じる．このため，この両者の混在した対話の下で，柔軟な理解能力とガイダンス能力を実現可能な柔軟な対話制御が必要である．

次に，上述したような入力文の省略の補足を行う場合，その解釈は常に一意に決まるのではなく，複数の解釈が可能な場合がある．このような場合，一々複数の解釈を利用者に提示して，意図した解釈を選んでもらうのでは会話性が損なわれてしまう．このため，できるだけ一つの解釈に絞り込み，対話を進めていく必要があるが，常に正しい解釈ができるとは限らない．したがって，対話の進行に伴い，システムが解釈の間違いを自動的に見いだし，その修正を行って，停止することなく対話を進めることのできる機能が必要となる．

本論文では，このようなユーザフレンドリーなインタフェースを実現する上で不可欠である対話管理機能の枠組みを提案する．具体的には，第２章において，利用者主導とシステム主導の両者が混在した対話の制御を可能とするため，対話対に基づいた対話管理方式を提案する．さらに，第３章において，これに仮定に基づいた真理保全機構（ATMS; Assumption based Truth Maintenans System)[11]をベースとした非単調推論機構を導入することにより，解釈を間違ったときにも停止することなく，対話を進めることを可能とする推論方法について述べる．なお，今回タスクとしては，行楽地の案内を行うエキスパートシステムを想定している．

2．対話対による対話制御方式

2．1　システム構成

入力の理解や応答する内容の決定に際しては，文脈情報や常識や因果関係による知識

[5, 6]が必要である．そこで，これらの各種の知識を統合しやすい黒板モデル[9]をベースとする構成を採ることにした．本システムの構成を第1図に示す．

さて，本システムは，大きくシステム主導で行楽地の案内を行うエキスパートシステムの基本部（以降ここを単にシステムと呼ぶ）と，その基本部と利用者との対話を管理するインタフェース部から構成される．

インタフェース部は，利用者の自然言語による入力を受け付け，自然言語処理を行う入力部と，システムからの検索結果や質問を利用者に伝える応答部およびシステムと利用者との対話の管理を行う対話管理部から構成される．そして，これらの各処理部がワーキングメモリと呼ばれる共有メモリを介して通信しあう構成をとっている．なお，入力部は，形態素，構文および意味解析までの自然言語処理を行い，文脈情報を用いた入力文の省略の補足などの文脈処理は対話制御部において行う．

2．2　対話管理法

発話は，多くの場合，相手の発話や過去の自分の発話を受けて行われる．また，なんの脈絡もなく発話が行われることは，対話においてはほとんどないが，表面的には過去のどの発話とも関連なしに生じる発話もある．このため，文脈処理や対話の制御を行うためには，入力された発話が過去のどの発話とどのような関連でなされたものかを決めることが重要である．これにより，ネストした対話に対しても正しく応答でき，省略の補足などを行なう場合の文脈情報を得ることが可能となる．この発話間の関連を捕える有効な方式として，対話を発話とその応答の対（対話対）により捕える管理法[1, 4]がある．対話対とは例えば，あいさつ－あいさつ，質問－応答のような発話の対であり，システムと利用者間のネストした対話（図2のu1～s2）の制御を容易に実現できる．しかし，図2の対話例における発話u4のように，表面的には対をなす発話が存在しない場合がある．また，u1とs2も一見対をなすように思われるが，u1のみからs2の答えを求めることはできないため，正確には対を成すとは言えない．このように，対話対が成立しない対話ではこの方式は適用できない．これに対して，論文（2）の利用者主導型の対話管理方式は，前者の発話u4ような対話に対しては対処可能である．しかし，後者の発話u1～s2ような2者間のネストした対話には対処できない．

本論文では，このような対話にも対処可能な対話管理法について述べる．本方式では，対話対による管理法を拡張し，上記のような対話をも論文（2）の枠組みで扱えることができるように変換する．具体的には，対話対が成立した発話の対が，新たな（対話対における）発話を生成すると考える．すなわち，発話の結合は，発話間のみでなく，新たな発話を介して対話対との間にも生じると考えることにより，ネストした対話を制御でき，かつ適切に文脈情報を利用可能な枠組みを提供する．ここで，本論文の主題は対話の管理法にあるため，発話の理解や発話間の結束性の評価方法などの理解処理に関しては，人間と同程度に行えると仮定して論議を進める．尚，発話間の結合関係に関しては，論文（2，4，8）などにおいて提案されている．

本方式では，図3に示すように，利用者からの質問を管理するためのスタックと，システムからの質問を管理するためのスタックを個別に設けている．対話の制御は次の4つの独立した対話制御モジュールにより実行される．ここで，各対話制御モジュールは，

ワーキングメモリ（WM）を介して通信しあう．

（1）対話制御 1

起動条件：システムからの問い合わせがWMに入力

処理：システム用スタックに問い合わせをプッシュする．

（2）対話制御 2

起動条件：利用者の入力文がWMに入力

処理：

(a) 入力文と対話対の連結が成立する発話，すなわち，入力文が応答となる質問をシステム用スタックから検索する．

(b) 対応する質問があれば，システム用スタックをその質問までポップし，さらに，その対話対から新たに利用者の問い合わせを生成し（後述），WMに出力する．

(c) 対応する質問がなければ，利用者からの文入力をそのまま新たな利用者の問い合わせとしてWMに出力する．この処理により初めて利用者の要求（質問）が確定する．すなわち，検索部が起動できる状態となる．

（3）対話制御 3

起動条件：利用者の問い合わせがWMに入力

処理：利用者用スタックに問い合わせをプッシュする．ただし，プッシュする問い合わせを包含する問い合わせがすでに利用者用スタックにあれば，その問い合わせをスタックから削除する．

（4）対話制御 4

起動条件：システム応答がWMに入力

処理：

(a) システム応答と対話対の連結が成立する質問まで，利用者用スタックをポップする．

(b) 連結の成立した対話対に基づいて，新たにシステム問い合わせを生成して，システム用スタックに問い合わせをプッシュする．

ここで，対話制御 2 の(b)で生成する新たな利用者の問い合わせとは，例えば，「どこの沿線（の行楽地）がいいですか？」というシステムの質問に，利用者が「XX線がいい」と答えたときの対話対の結果としては，「XX線の沿線の行楽地を知りたい」という利用者の問い合わせが生成される．すなわち，検索主題である行楽地の制約条件の変更[(2)]を行う質問であると考える．このように，成立した対話対の結果として，対話対における発話を生成すると考えることにより，対話対間の結合関係も対話対と同じ枠組みで処理可能となる．

次に，対話制御 4 の(b)で生成する新たなシステム問い合わせとは，「その話題に関する他の条件は？」という質問である．これは，「解をさらに絞り込むために，他にどのような条件があるか？」という意味の質問であるが，この質問は，ワーキングメモリに書かれることなく，すぐにシステム用スタックに格納される．すなわち，利用者には伝えられない，システムの心の中に留めておく質問である．ただし，我々人間においても，このような質問は，無意識に生成され，心の中に留めておかれると考えられる．このため，この質問は，システムが生成するのではなく対話制御部が生成する構成とした．

2．3　対話管理の例

以下に，図2の会話例を用いて，対話制御部の動作を説明する．

まず，利用者の入力u1が入力すると，対話制御2が起動する．両スタックの初期値は空の状態なので，入力u1と対話対の成立する質問はない．このため，入力u1がそのまま利用者の質問と判断され，ワーキングメモリに登録される．そして，対話制御3によって，表1のように利用者用スタックに追加される．入力u1に対してシステムの質問s1が入力されると，対話制御1が起動し，これをシステム用スタックに格納する．次に，u2が入力されると，再び対話制御2が起動し，u2がシステム用スタック内の質問s1と連結が成立するかどうかを調べる．ここでは成立するため，「明日行けて，子供が楽しめる行楽地は？」という利用者の質問u2'を生成して出力する．対話制御3は，この質問を利用者用スタックに追加するわけであるが，ここで，既に利用者用スタックに格納されている質問u1は，新たな質問u2'を包含するものであるため，表1に示すように，質問u1は利用者用スタックから削除される．ここで，質問u2'のように，文脈を全てスタックで管理することにより，システムの負担を軽減している．

この利用者の質問u2'に対するシステムの応答がs2である．すなわち，応答s2は入力u1ではなく，質問u2'との間で対話対を生成することができ，システムの内部の処理と対話処理とを一致させることができる．さて，この場合は対話制御4が起動し，利用者用スタックに格納されている質問u2'の答えであることが確認され，その対話対からシステムの質問「明日行けて，子供が楽しめる行楽地の他の条件は？」を作成し，システム用スタックに格納する．同様にして，u3が入力されると，これはシステム用スタック内の質問の答えにならないため，新たな利用者の質問となり，その答えs3により「甲州街道の他の条件は？」というシステムの質問がシステム用スタックに追加される．

次に，u4が入力されると，対話制御2が起動する．ここで，この入力は各種の推論を行えば，質問を意図した文であると判断できるであろう．しかし，単純に考えると，この文は質問文ではなく，答えの文と考えた方が自然である．本方式では，システムの質問s2'「明日行けて，子供が楽しめる行楽地の他の条件は？」に対する，条件が「甲州街道の沿道から近い」である答えとして処理可能である．そして，対話制御2は，この対話対から「明日行けて，子供が楽しめて，甲州街道の沿道から近い行楽地は？」という利用者の質問を生成する．このように本方式では，一貫した対話対による制御が可能となり，入力を対話対による連結としてとらえることができるため，入力の理解処理も単純化できると考えられる．

本方式では，利用者とシステムの質問を管理するためのスタックをそれぞれ個別に設けているが，従来通り1つのスタックで管理できないことはない．しかし，表1の制御を1つのスタック上で行うと，処理が非常に複雑になってしまう．

3．仮定に基づく対話制御法

3．1従来の非単調推論

利用者の入力の理解において，その入力に対して複数の解釈が可能な場合には，前述したように，複数の解釈を利用者に提示して利用者が意図した解釈を選んでもらうので

はなく，できるだけ一つの解釈に絞り込み，対話を進めていくことが望ましい．しかし，常に正しい解釈ができるとは限らないし，利用者は，思い込みにより会話シーケンスを狂わすような発話[2]をすることがある．このように，システムと利用者の間に解釈のずれが生じると，対話が進むにつれて話しに矛盾が生じてくる．このとき，自動的に矛盾の原因を探し，その修正を行って対話を進めることのできる機能が必要である．この問題は非単調推論の問題である．ここでは，この非単調推論を実現する効率のよい手段の１つであるATMS[11]をベースにした推論機構により上記機能を実現する方式を提案する．

まず，ATMSについて，本論文に関係する範囲でその概要を説明する．問題解決システムにおいて，推論エンジンが，ルールの実行などを担当するのに対して，TMSは，データベースの管理や，データベース中のデータの真理の維持を担当する役割を果たす．この真理の維持は，次のようにして行われる．ATMSでは，推論に用いる導出結果（ノードと呼ばれる）を次の３つのタイプに分ける．即ち，無条件に成り立つ事実を表わす前提ノード，否定する事実が無ければ確実な根拠のないまま仮に信じる仮定ノードおよび，事実や仮定に基づいて推論された結果を示す導出ノードである．ATMS内部では，ノードは次のようなデータ構造で表現され管理される．

ノード：＜データ，ラベル，支持理由＞

データ Dは，推論エンジンから，

$[J_1,J_2,\ldots,J_k => D]$

の形態で通知され，$\{J_1,J_2,\ldots,J_k\}$をデータ Dの支持理由と呼ぶ．ATMSは，データをワーキングメモリ（データベース）に登録する際に，推論エンジンから通知される支持理由に基づいて，その環境を計算しノードに記録する．この環境とは，そのノードを支持する仮定の集合である．例えば，環境が[A,B]であるノードと環境が[B,C]であるノードを支持理由として生成されるノードの環境は[A,B,C]となる（実際には，ノードを支持する環境の集合であるラベルに追加される）．そして，矛盾が生じると，矛盾ノードの環境（その矛盾ノードが生成される根拠となった環境）を信じることを止めることにより，真理保全を行なう．このATMSを対話の制御に利用する場合，次のような問題がある．

（１）ATMSに限らず，従来の非単調推論では，導出結果は基本的には追加されるのみである．すなわち，プロダクションシステムにおける追加（make）のみが実行される．しかし，例えば対話の制御に用いる対話スタックは，対話の進行に応じて内容を変化させる必要があるため，これを実現するためにはノードをフレーム型知識表現可能な構成にすると共に，プロダクションシステムにおけるノードの修正（modify）を扱えるようにすることが望ましい．

（２）矛盾は，必ずシステム内部のデータや推論結果に基づいて生成されることが仮定されている．しかし，対話においては，システム外部の利用者から直接矛盾であることが指摘される場合もある．この時の矛盾は，システム内部のデータや推論結果に直接依存しない．すなわち，システム内部のデータの因果関係から矛盾の原因となる環境を探すことができない．

（3）ATMSでは，代替の仮説を並列に評価可能である．この機能は，人の頭の中だけでの思考錯誤の過程を高速に実現するには適している．しかし，対話においては，利用者とのインタラクションがあるので，ある1つの解釈を一応正しいと仮定して推論を進めていく，JTMS(Justification-based TMS)[10]的な推論が必要となる．すなわち，対話の制御に適したコンテクストの管理法を提供する必要がある．

3．2　非単調な対話制御法

従来のTMSでは，データ（ノード）は1つのワーキングメモリ上で管理されるが，ここで提案するTMSでは複数の格納領域からなるワーキングメモリにより管理を行なう．1つの格納領域には，同じ環境（仮定の組み合わせ）をもつデータのみを格納する．このため，すでに登録されている環境とは異なった環境のデータが推論システムから通知されたとき，本TMSは新たな格納領域を生成し，データをその格納領域に格納する．この格納領域は，次のようなデータ構造で管理される．

格納領域：＜格納領域名，環境，支持環境，仮定データ名リスト＞

仮定データ名リストは，その格納領域に格納されている全ての仮定データ名を記録するものである．環境は，ATMSのノードの環境と同じものである．支持環境は，格納領域上の各データの支持理由の集合を登録順に記録するものである．ただし，ATMSのノードの支持理由とは異なり，格納データを支持するデータ名ではなく，支持するデータが含まれている格納領域名を記録する．例えば，推論エンジンから通知される支持理由$\{J_1,J_2,J_3\}$のデータの内，データ$\{J_1,J_2\}$が格納領域{CM1}，データ$\{J_3\}$が格納領域{CM2}に含まれている場合，ここで記録される支持理由は{CM1,CM2}である．次に，各格納領域には，継承関係を持たせる．各格納領域は支持環境に記録された格納領域中の全てのデータを継承する．継承順は追加登録された順とする．上記の例では，格納領域CM1,CM2の順に継承する．次に，推論システムからのデータ検索要求に対しては，トップの格納領域から検索を開始し，もしそこに該当するデータがなければ，継承している格納領域の検索を行なう．この構成により，従来の第1点目の問題を解決できる．

第3点目の問題にあげたコンテクストの管理を行なうため，格納領域は，格納領域用のスタックにより管理する．格納領域は，生成されると格納領域用スタックにプッシュ（格納）される．前述したトップの格納領域とは，格納領域用スタックの先頭の格納領域を意味し，推論システムからはこの格納領域のみが見える．但し，格納領域用スタックにすでに代替となる格納領域が格納されている時は格納しない．ここで，代替となる格納領域は，同じ支持環境下で，異なった仮定に基づいて生成される．このため，代替となる格納領域とは，格納領域用スタックに格納されている格納領域に対して，支持環境がリストの先頭を起点とした部分リストで，ラベルの異なる格納領域である．ここで，先頭を起点とした部分リストとは，[A,B,C]に対して[A,B]は部分リストであるが，[B,C]は部分リストと呼ばない．

さて，推論システムから矛盾が通知された場合，矛盾の原因となる仮定を検索し，信じることを止めることにより矛盾を解消する．このとき，矛盾が推論システム自身が生成したものか（格納領域上のデータを根拠としていることが記録されている）否かによって次の2種類の検索を行なう．前者の場合は，格納領域用スタックをポップしながら，

代替となる格納領域（仮定）が存在する格納領域を検索し，後者の場合は，矛盾と共に指摘される事実と矛盾関係にある仮定を仮定データ名リストに持つ格納領域を検索することにより矛盾原因を見つける．この後者の処理により，従来の第２点目の問題が解決可能となる．ここでの後戻り処理はJTMSとは異なり，スタック上で格納領域単位で行われるため，計算は複雑にはならない．

３．３　対話制御例

２節の推論法について図２の会話例を用いて具体的に説明する．入力u5「もっと近いところは？」の解釈としては，会話の流れからは，表１のあと生成されるであろう「明日行けて，子供が楽しめて，甲州街道の沿道から近い行楽地の他の条件は？」というシステムの質問s4'を受けての発話であると判定できる（図４）．すなわち，「s４で提示された行楽地（X，Y）よりも，もっと甲州街道に近い行楽地は？」（au51）であると解釈できる．しかし，利用者は自分の住居に近い場所を望むことが多いため，「行楽地（X，Y）よりも，もっと自分の住居に近い行楽地は？」（au52）との解釈も可能である．すなわち，条件を「もっと甲州街道に近い」と仮定（ass51）した場合と，「もっと自分の住居に近い」と仮定（ass52）した場合に基づく２種類の解釈が成り立つ．推論システムがこの２つの推論結果au51とau52をTMSに書き込むと，TMSは，図４に示すように新たに２つの格納領域（CM51，CM52)を作成し，その一方の名前（CM51)のみを格納領域用スタックにプッシュする．この２つの格納領域のうち，一方の格納領域CM51は仮定ass51を環境に含み，他方の格納領域CM52は環境に仮定ass52を含む．そして，どちらも前に作成された格納領域（CM４とする）を支持理由として持つ．

以上の処理の結果，格納領域用スタックの先頭の格納領域（CM51）が知識源からアクセス可能となる．これにより，検索部は，au51の解釈に基づいた検索を行ない，その結果行楽地AとBを求める．そして，応答部は解が行楽地AとBの２つのみなので，s5でこれを全て提示する．また，この処理において対話制御部により変更される対話スタックは，格納領域CM４上の対話スタックとau51の解釈に基づいて生成されるため，格納領域（CM51）に作成される．このため，入力u5の解釈の生成以前の対話スタックは，格納領域CM４上に保存される．また，au51の解釈に基づいて格納領域CM51上に生成される対話スタックは，s5入力によっても変更されるが，対話を対話対を単位として扱っているため，この変更は問題ない．

続いて，問い合わせu6「他にないか？」が入力すると，この解釈は「提示された行楽地AとB以外に，行楽地（X，Y）よりも，もっと甲州街道に近い行楽地はないか？」（au6）と言うものであり，これは在りえない問い合わせである．なぜなら，u5－s5の対話で，それ以上解がないことを利用者が知っているはずだからである．このため，この要求文au6は何らかの誤解に基づいたものと判断し，この解釈を矛盾であると通知する．具体的には，要求文au6を根拠とする矛盾データを登録する．

さて，矛盾原因検索では，矛盾が格納領域上のデータau6を根拠とするシステムが生成したものであるため，代替となる仮定を持つ格納領域を格納領域用スタックをポップしながら検索する．ここでは，先に示した解釈au51を格納する格納領域（CM51）が検索される．そして，代わりに代替の共通格納領域（CM52）を格納領域用スタックにプ

ッシュすることにより，前者の解釈au51が破棄され，後者の解釈「行楽地（X，Y）よりも，もっと自分の住居に近い行楽地は？」（au52）の処理を，対話を中断することなく行なうことができる．ここで，矛盾とされた仮定ass51を含む環境（ポップされた格納領域の環境）は，ATMSと同様にno_goodのデータベースに格納しておくことにより，真理保全を実現できる．

上述した例は，矛盾を利用者に気づかれることなく，システム自身が気づいて訂正を行なうケースであったが，利用者の思い込みが原因ではあるが，利用者から矛盾を指摘される場合もある．そのような例を次に示す．

u11:日野市と八王子市のJRの駅は？
s11:日野駅，八王子駅など５駅在ります．
u12:そこに近い行楽地は？
s12:２０ケ所あります．
u13:京王線に近いのは？
s13:XとY遊園地です．
u14:ちがう．駅で！

この例は，u13で検索の主題を行楽地と解釈（仮定）して，s12の結果のうち「京王線に近い行楽地」を検索して応答したら，実は，利用者はs12の結果を絞るために，その前のs11の結果をまず絞ろうとして，「京王線に近い駅」を問い合わせた例である．実際の会話では，このような会話シーケンスを乱すような会話が生じることがある[2]．

この例のようにu14で指摘される矛盾は，システムが推論し生成するものではなく，利用者から与えられるものであるため，ワーキングメモリに登録される時，この矛盾には格納領域上のデータとの関係が記録されない．

さて，このような，矛盾が外部から与えられた場合，矛盾原因検索は，指摘された事実（検索の主題＝駅）と矛盾する解釈（仮定）を持つ格納領域がみつかるまで格納領域用スタックをポップする．ここでは，u13の文脈処理において検索の主題を行楽地と仮定したため，この主題が仮定データとしてワーキングメモリに登録されている．このため，u13における検索主題に関する仮定（検索の主題＝行楽地）を持つ格納領域以降に生成された格納領域がポップされ捨てられる．この後，指摘された事実（検索の主題＝駅）を格納領域用スタックの先頭の格納領域に格納することにより，利用者の意図した「s12の結果のうち，京王線に近い駅」の検索を行ない，応答することができる．ここで，このケースでは，先の例とは異なり，同じ格納領域上に互いに矛盾する仮定（検索の主題＝行楽地）と事実（検索の主題＝駅）が存在することになるため，両者の矛盾を通知する必要がある．さらに，仮定（検索の主題＝行楽地）に基づく推論を停止させるため，ポップされた格納領域の環境をno_goodのデータベースに格納することにより，真理保全を実現する．

真理保全処理で述べたように，間違った解釈に基づいた対話や推論結果は，格納領域用スタックからポップされ捨てられる．しかし，この捨てられた区間でも，利用者との行き違いはあるものの，利用者との間にインタラクションがあるため，副作用が生じる．このため，対話がもっと進んだ時点で，この捨てられた情報が話題になることがあるか

もしれない．この捨てられた情報をどのように管理するかは，今後の課題である．特に，上述した矛盾原因検索の前者の例は，システムが代替案を準備できる仮定を優先して処理するための制御を提供している．しかし，代替案を準備できなかった仮定が間違いであることもある．この場合は，利用者に矛盾内容を問い合わせて，後者の矛盾原因検索処理を行う必要がある．このため，前者の例でポップされ捨てられる情報は復帰可能な形で保持される必要があると考えられる．

また，本方式では，データは複数の格納領域で管理する構成としたが，実際的には，各格納領域にデータを格納するのではなく，データは１つのワーキングメモリで管理し，各格納領域ではデータ名のみを保持する構成をとることによりATMSの枠組みを崩すことなく実現可能である．このため，データ間や格納領域間の因果関係はATMSの機能により記録され，利用可能な枠組みになっていると考えられる．

４．むすび

対話のすべての発話を，発話の対により統一的に扱うことが可能な対話対に基づいた対話管理方式，並びに非単調推論をベースにした仮定に基づく対話制御方式を提案した．これにより，利用者とシステム間の効率的で柔軟な対話が実現可能となる．さらに，対話制御における非単調推論の適用方法を示すことができたと考える．また，以上のような対話制御の構成をとることにより，エキスパートシステムなどのアプリケーション処理と対話制御を含むユーザインタフェースの処理の分離を進めることができると考える．

さて，本方式においては，異なった環境のデータを各々別個の格納領域で管理するため，記憶領域の効率が悪くなることが考えられる．例えば，ある入力に対して１０個の解釈が生成されたとき，単純には，それぞれの解釈が仮定となるので１０個の異なる環境が生成され１０個の格納領域が生じる．しかし，我々は一般に，正しいと思った解釈が間違っていることが分かった時，次善の解釈が１つのみか，又は，次善の解釈がその他の解釈に対して確信度が非常に高いときには，無条件に次善の解釈が正しかったとして処理を行なう．けれども，それ以外の時には代替の解釈を利用者に示し，利用者に意図した解釈を選んでもらうであろう．そのほうがより良い応答であると考えられる．このため，もっとも確信度が高い最善の解釈を仮定とするものと，残りの（９個の）解釈を仮定とするものの２つの格納領域のみですむ．なお，構文，意味解析のような理解処理における代替の解釈の評価処理は，prologのような効率の良い処理系で処理すべきであろうと考える．

本論文では，対話対に関しては，データベースの検索に関連する質問－応答に関する検討にとどまっている．今後は，他の対話対についてもその結合関係[4]の検討を行なっていく必要がある．また，ここで柔軟な対話管理の枠組みを提案したが，システム全体としてより柔軟な処理を可能にするためには，アプリケーション処理を宣言的に定義し，それをメタレベルで制御可能な枠組みを実現する必要があると考えられる．

謝辞　本研究は，第五世代コンピュータプロジェクトの一環として，新世代コンピュータ技術開発機構ICOTからの委託により行ったものである．ご支援頂いた生駒研究部長代理，新田第７研究室長に感謝致します．

文　献

（1）宮地泰造，伊草ひとみ，近藤省造，太細　孝，古川康一；”話題管理機構をもつ対話システムの試作”，情処学会　知識工学と人工知能38-7 (1985)

（2）加藤恒昭，中川　優；”質問応答における意図の把握と話題の管理”，情処学会　自然言語処理58-6 (1987)

（3）北橋忠宏；”発話対にもとづく対話モデル”，文字言語・音声言語の知能的処理第１５２委員会，第１６回研究会資料16-1 (1990)

（4）T.Yamamoto,Y.Ohta,Y.Yamashita,R.Mizoguchi;"Dialog management system MASCOTS in speech understanding system",Proc. of ICSLP ,Kobe (1990)

（5）長尾　真，辻井潤一，田中一敏，”意味および文脈情報を用いた日本語文の解析－文脈を考慮した処理”，情処学論，１７，１，pp.19-28 (1976.6)

（6）丸山，”グラフのマッチングを用いた意味解析”，自然言語研究会資料58-3 (1986)

（7）西山敏雄，大山芳史；”データベース検索における協調的な自然語対話処理と評価”，信学論（D-II），J73-D-II,4,pp.625-632 (1990)

（8）小暮　潔，有田　英一，野垣内　出，飯田　仁；”端末間対話における言語理解方式”，情処学会　自然言語処理62-11 (1987)

（9）L.D.Erman;"The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainly" ACM Computing Surveys, Vol.12, No.2,pp.213-254

(10) Doyle,J.; "A Truth Maintenance System", Artificial Intelligence 12 (1979),pp.231-272

(11) de Kleer,j.;"An Assumption-based TMS",Artificial Intelligence 28 (1986),pp.127-162

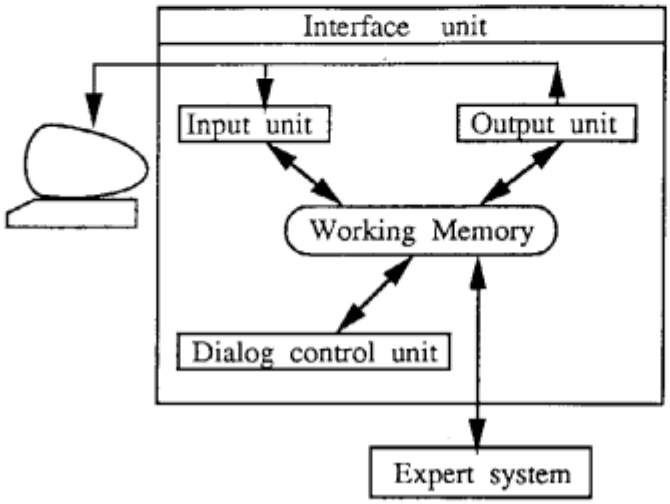

図1　対話処理システムの構成
Fig.1 Components of dialog processing system.

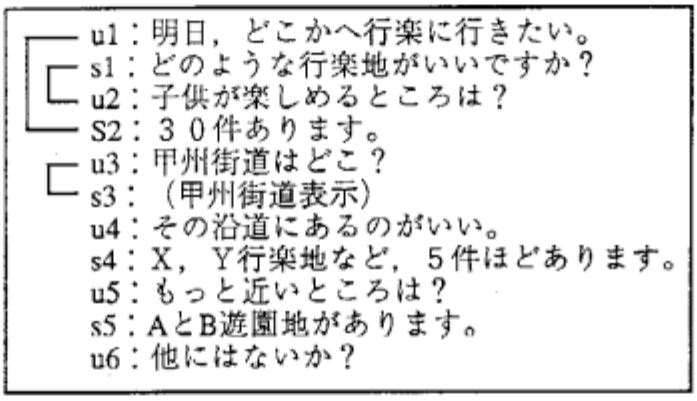

図２　対話例
Fig.2 Example of dialog

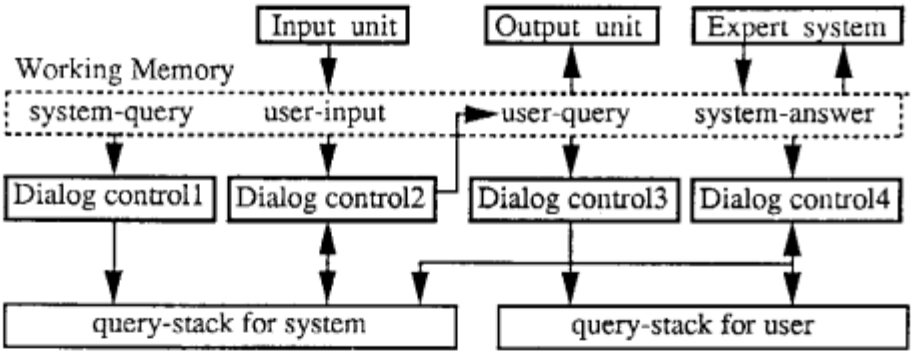

図3　対話制御部の構成

Fig3. Components of dialog control unit

表1　動作例

入力	生成質問	問合せスタック	
		利用者	システム
u 1	u1 :明日行ける行楽地は？	[]	[]
s 1	s1 :明日行ける行楽地の条件は？	[u1]	[]
u 2	u2' :明日行けて、 子供が楽しめる行楽地は？	[u1]	[s1]
s 2	s2' :明日行けて、 子供が楽しめる行楽地の条件は？	[u2']	[]
u 3	u3 :甲州街道（の位置）は？	[]	[s2']
s 3	s3' :甲州街道の条件は？	[u3]	[s2']
u 4	u4' :明日行けて、 子供が楽しめて、 甲州街道の沿道に近い行楽地は？	[]	[s3',s2']
		[u4']	[]

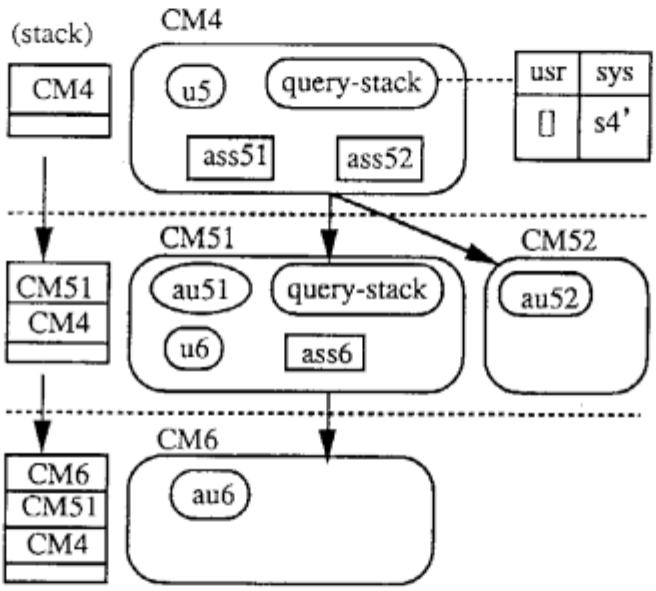

図4　ワーキングメモリとスタックの処理過程
Fig4. Management of working memory and stack