

## 領域知識内での構造写像を用いた動的概念の説明手法

細見 格 萩野 浩司 上原 邦昭 前川 禎男  
神戸大学工学部情報知能工学科

本論文では、自然言語によって計算機の利用を支援する質疑応答システム Assist 3 について述べる。質疑応答システムを構築する上では、対象領域の知識表現モデル、ユーザに対する応答内容の検索手法、さらに応答内容の選択手法が必要となる。知識表現モデルについては、種々の目的に対する手続きや要素間の因果関係といった動的な概念の記述を可能にした ACT-NET モデルを提案する。また、ユーザからの問い合わせに対し、類推の一手法である構造写像エンジンを適用することにより、質問のタイプに関わらず、応答内容の抽出を統一的に処理することが可能となることを示す。さらに、応答内容の候補選択をアブダクションの仮説選択問題とみなし、注釈という意味的ヒューリスティクスによって、単一の候補を選択する手法を提案する。

## A Technique for Explaining Concepts in a UNIX Domain Based on Analogical Reasoning

Itaru Hosomi Hiroshi Ogino Kuniaki Uehara Sadao Maekawa  
Department of Computer Science and Systems Engineering  
Kobe University  
Nada, Kobe, 657 Japan

In this paper, we will describe a consultation system called Assist 3 which helps naive users in natural language. In consultation systems, there exist several problems concerning with knowledge representation model, retrieving an appropriate plan for satisfying the user's goal, and eliminating the ambiguity of the user's plan. First, we propose the ACT-NET model to represent "state changes" which involves both procedures to achieve various goals and causal relations between events. Second, we show that plans for satisfying the user's goal is retrieved by using single computational mechanism called structure mapping engine, which performs an inference process of analogical reasoning. Moreover, when the inferred user's plan is ambiguous, plan selection module in Assist 3 eliminates the ambiguity by using "annotation" which is the semantic heuristics of abduction. We also give a detailed explanation of this heuristics.

## 1 序論

自然言語を用いた対話型質疑応答システムを開発する場合、汎用的な知識表現モデルを設計することは重要な問題である。従来のシステムが持つ知識には、文を解析・生成するための辞書、対象領域に関する種々の概念を記述したシステム知識、ユーザの知識や対話の履歴を保持するユーザモデルなどがあり、それぞれ異なる表現形式が用いられている場合が多い。さらに、システム知識は、概念の定義や性質などからなる“静的な知識”と、概念間の因果関係や処理の流れなどからなる“動的な知識”に分類されている場合が多い。本論文では、静的な知識と動的な知識を有機的に結び付けたネットワーク型の知識モデルを提案する。

一方、従来のシステムでは、適切な応答内容を得るために、ユーザからの質問をタイプ別に分類し、各タイプ毎に異なるヒューリスティックスを用いている場合が多い。これに対して、本研究では、全ての質問のタイプに対する応答内容の抽出を“類推”という統一的手法によって取り扱っている。

本論文では、上記のネットワーク型知識表現モデルを用いて、システム知識およびユーザモデルの双方を記述した対話型質疑応答システムについて述べる。また、類推の手法である構造写像理論を用いることにより、ユーザに不足している知識やユーザの誤解を推定する手法について述べる。さらに、ヒューリスティックスのモジュール集合であるプラン選択モジュールを利用することにより、曖昧な入力内容からユーザの発話意図を推定し、適切な応答を与える手法について述べる。

## 2 日本語質疑応答システム Assist3

Assist3は、UNIXシステムの利用に関するユーザからの質問に日本語文で答える対話型の質疑応答システムである。Assist3の構成を図1に示す。

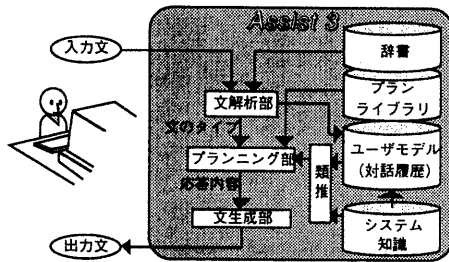


図 2.1 Assist 3 システム構成図

Assist3の処理の流れは、次の通りである。

- (2.1) 入力文を解析し、文のタイプ (what 型質問, why 型質問, etc.) を決定する。
- (2.2) 解析された入力内容をシステムの内部表現に変換して、ユーザモデルに格納する。
- (2.3) ユーザモデルおよびシステム知識から、構造写像を用いて入力内容に関する情報を取得する。
- (2.4) 得られた情報から、プラン選択モジュールを用いて適切な応答内容を選択する。
- (2.5) 入力文のタイプに合わせて応答文の構成を決定する。
- (2.6) 応答内容を日本語文に変換して出力する [1]。
- (2.7) 応答に利用されたシステム知識内の概念を、対話の履歴としてユーザモデルに格納する。

## 3 知識のモデル

質疑応答システムは、ユーザからの様々な問い合わせに対応しなければならない。そのため、システムには一般に次のような機能が要求される。

- (3.1) 各種用語についての説明。
- (3.2) 種々の目標を達成するための手段の提示。
- (3.3) トラブルの原因の推定と対処法の提示。
- (3.4) ユーザの誤解の認識と訂正。
- (3.5) ユーザの行動からの目標推定と、トラブルを未然に防止するための忠告。

以上の要求を満足するためにシステムが持つべき知識としては、次のようなものが挙げられる。

- (3.6) 各コマンドの用途や書式。
- (3.7) 各コマンドの使用時における条件。
- (3.8) 関連したコマンド間の類似点や相違点。
- (3.9) 種々の目的に応じた最適なコマンドの選択方法。
- (3.10) 各コマンドを実行した際の処理の流れと結果、およびシステムへの影響。

以上のうち、(3.6)～(3.8)に関する知識については、通常の意味ネットワークを用いれば表現可能である。しかし、(3.9)のように状況に依存した知識や(3.10)のような動的な概念についての知識を記述するためには、条件分岐や動作間の因果関係などの表現が可能な、別の表現形式が必要になる。このため、Assist3では、(3.9)に関する知識をプラン選択モジュールとしてシステム知識とは別に用意している。プラン選択モジュールとは、ユーザの発話意図を推定し、適切な応答内容を選択するための種々のヒューリスティックスを記述したモジュール群である。また、(3.10)に関する知識については、後述するACT-NETモデルを採用することにより、動的な概念の表現を可能にしている。

動的な概念を記述するために、意味ネットワークを拡張したモデルを採用している研究として Paris & McKeown [2] がある。文献 [2] では、概念の階層を表すアークと状態の遷移を表すアーク、ノード間の機能的な関係を表すアークなどが用意されている。しかしながら、このネットワークでは、ある“状態”が他の“状態”にどのように影響するかについてはノードとノードを結ぶアークによって記述できるが、ある“状態”や“処理の結果”が他の“動作”や“状態遷移”にどのような影響をおよぼすかについては記述することができない。また、状態遷移を表すアークは、一般的な処理の流れを表すアーク列 (Main Path)、状況によって他の処理系列に分岐するアーク (Side Link)、分岐の選択肢を表すアークなどに分類されているが、そのセットが完全に閉じていないので、モデルとしての統一性に問題があると考えられる。

### 3.1 ACT-NET モデルの構成

本研究で提案するACT-NET (ACTivation Transmission NETWORK) モデルは、アークをノード間のリンクに限定せず、ノードとアーク、アークとアークを結ぶ有向リンクとして利用可能にすることにより、3種類のアークを用いて階層的概念、動的な概念および処理の前後関係や因果関係を記述可能にしている。ACT-NETモデルのノードとアークには“活性度”という重みが付加されている。各ノードやアークは、自身が持つ活性度がスレッショールドを越えると活性化され、各々に設定された役割を実行する。

アークは、その役割により、階層アークと動作アークおよび励起アークの3種類に大別される。階層アークは、ノード間の関係を表すために用いられる。動作アークは、あるノードから他のノードやアークに何らかの働き掛けがあることを表すものである。励起アークは、その終端がノードを指している場合、指しているノードを活性化または不活性化させるために用いられる。この作用により、ある"状態"が満足されたことや否定されたことを表現することができる。励起アークの終端が他のアークを指している場合には、指しているアークの活性度を変化させることで、そのアークが表す動作や状態変化を促進または抑制することができる。さらに、この作用により、ある"状態"や"処理の結果"が他の"動作"や"状態遷移"に及ぼす影響を表現することができるようになっている。ACT-NETモデルを用いた概念の表現例を図3.1に示す。



図 3.1 ACT-NET モデルを用いた概念

図 3.1 は UNIX の rm コマンドに関する概念を表しており、以下のような意味を持っている。

rm は、ユーザがファイルを消去するために、ユーザによって実行されるコマンドである。ただし、rm が有効に働く条件としてライトパーミッションが設定されていなければならない。

上記のようなネットワークを知識として持つことにより、システムは、以下のような情報を得ることができる。

- (3.11) コマンドの仕様。
- (3.12) ユーザの目標を達成するための手段。
- (3.13) コマンドを利用する際の条件。
- (3.14) 目標を達成するための手順 (ライトパーミッションを設定してから rm コマンドを実行する…など)。

### 3.2 ラベルと活性度

ノードのラベルは、そのノードが示す概念の名前である。アークのラベルは、そのアークが動作アークであれば動作の名称となり、励起アークであれば他のアークの活性度をどれだけ変化させるかを表す値となる。階層アークのラベルはその両端のノードの関係を表している。システム知識には、各動作アークと同じラベルを持つノードが定義されている。動作アークのラベルのうち、互いに類似した意味を持つラベルや意味的包含関係にあるラベルどうしは、各ラベルに対応するノードを階層アークによって関係付けている (図 3.2 参照)。



図 3.2 動作を表す概念の階層構造

図 3.2 は、動作アークのラベル"リモートコピー"と同じラベルを持つノードが、ラベル"isa"を持つ階層アークによって、ラベル"コピー"と結合されている。これにより、"リモートコピー"という動作概念が、"コピー"の一種であることが分かる。

活性度とは、各ノードやアークの示す概念が、対話中の各時

点で、どの程度意識されているかを表す指標である。活性度の値は 0 以上 1 以下の実数値で表される。活性度が 1 に達した場合、そのノードやアークは活性化されたという。ノードが活性化されると、そのノードを始点とするアークのうち、すでに活性化されているアークがあれば、そのアークに所定の役割を実行させる。動作アークまたは階層アークが活性化されると、それらの終端ノードを完全に活性化させる (活性度を 1 にする)。励起アークが活性化された場合には、ラベルとして記された値の分だけ、その励起アークが指しているノードやアークの活性度を変化させる。ただし、システム知識において既に活性化されているアークは、その始点となるノードが活性化されるか他の励起アークに活性化されるまで待機している。

活性度は、各ノードやアークに対して以下のような役割を持っている。

- (3.11) 各ノードの示す概念が、対話中に参照されたかどうかを表す。活性度の値が正であれば、対話の中ですでに参照されていることを示している。
- (3.12) 後述する"忘却"の仕組みにより、システムが持つ概念のうち、対話中に用いられた概念の参照順序を表す。
- (3.13) 各ノードやアークがその役割を実行すべき状態にあるかどうかを表す。

上記の役割は、主としてユーザモデル中のノードやアークの状態を調べる際に利用される。例えば、ユーザがある目標を達成できないと入力してきた場合、ユーザの発言に矛盾が検出されなければ、ユーザの目標を達成するための手順に不足がないかチェックする。手順の中でまだ活性化されていないアークが存在すれば、そのアークを活性化させるノードおよび励起アークをユーザモデルの中から検出する。検出されたノードや励起アークのうち、まだ活性化されていないもの、またはユーザモデル中に存在しないものがあれば、そこにトラブルの原因があると推定される。すなわち、不活性のノードやアークは、対話の中でまだ触れていない概念を表しているの、ユーザがその部分を見落としている可能性があることを示している。

### 3.3 動的な概念の表現

ACT-NETモデルを用いることにより、動作間の因果関係や条件分岐のある手続きなどの動的な概念を表現することが可能となっている。以下に例を示す。

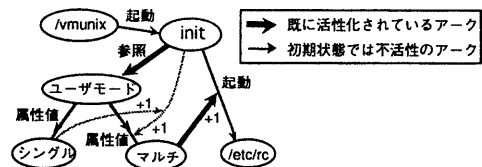


図 3.3 init の処理過程 (部分)

図 3.3 は、UNIX のカーネルである /vminux によって起動される init の処理過程の一部を表したものである。init が起動されると、まず最初に OS のユーザーモードを参照する。このとき、ユーザーモードがシングルユーザーモードであれば、init がマルチユーザーモードに切り換える。ユーザーモードがマルチユーザーモードであれば、etc/rc を起動する。このように、条件によって処理の流れが異なるような手続きを、ACT-NETモデルを用いて記述することができる。ただし、図 3.3 において、ユーザーモードの属性値を指している 2 つの属性アークが共に活性化されてはならない。

このような場合に対処するために、ある状態の名前とその状態の値とを結び属性アークについては、一つの属性アークが活性化された場合には、他の属性アークが不活性になるように、システムによって管理されている。

#### 4 構造写像による応答内容の検索

質疑応答システムは、ユーザからの様々な要求に対して柔軟に対応しなければならない。ユーザは対象領域について完全な知識をもっているわけではないので、欠落した知識を補うために質問をする。また、単に自分の知識が正しいかどうか確認したいときは yes-no 型の質問をする。このようなユーザからの質問に対して、質疑応答システムはシステム知識の中から適切な応答内容を検索しなければならない。さらに、応答内容としては、対象概念だけでなく、それに類似した概念との比較に基づいて説明すれば分かりやすくなる。あるいは、質問内容にユーザの誤解が含まれている場合には、誤解を発見、修正する必要がある。

既存のシステムでは、以上のような種々の問題に対して、異なるヒューリスティクスを用いて適切な応答内容を抽出している場合が多い。しかしながら、各モジュールが出力する全ての結果は、本質的にはユーザの要求内容とシステム知識の照合によるのみ得られるので、これらのモジュールは統合化できると考えられる。本研究では、前述のような種々の応答戦略を類推という一つの手段によって統一的に取り扱っている。すなわち、ユーザの要求内容とシステム知識を類比して、それらの類似度の高いものを応答内容として検索する。さらに、ユーザの要求とシステム知識の共通部分(共通項)や異なっている部分(差分項)を説明のための材料とする。なお、本研究では、類推の手法として Gentner の構造写像理論 [3] を採用している。構造写像理論は、単に個々の要素を比較するのではなく、要素間の構造的な側面に着目して類比を行うという点に大きな特徴がある。

冒頭で挙げた、質疑応答システムにおける種々の応答戦略は、構造写像によって次のように取り扱われる。

##### (4.1) yes-no 型の質問文に対する応答

ユーザの質問内容は ACT-NET の表現形式としてユーザモデル内に変換される。このとき、ユーザの質問内容をシステム知識の構造に写像して、矛盾がなければ肯定した応答を生成する。

例)

ユーザ : ファイルを消去するには m を使えばいいのですか?  
Assist 3 : そうです。

##### (4.2) wh 型の質問文に対する応答

wh 型の質問文に対しては、ユーザの質問内容は未知の部分欠落した不完全な構造となっている。構造写像によって不完全なユーザの知識構造を完全なシステムの知識構造に写像することにより、ユーザの知らない情報を補完することができる。

例)

ユーザ : cp は何をするためのコマンドですか。  
Assist 3 : cp コマンドはファイルをコピーするためのコマンドです。

##### (4.3) 類似概念を用いた分かりやすい説明

前述のように、説明内容は、ユーザの質問内容をシステム

知識に写像することによって検索される。したがって、対象概念とともに類似した概念を検索できれば、両者の比較を基に分かりやすく説明することができる。

例)

ユーザ : rcp とは何か?

Assist 3 : rcp コマンドは cp コマンドと同様にファイルをコピーするコマンドですが、cp コマンドと違ってファイルシステム間でのコピーができます。

##### (4.4) ユーザの誤解の発見

ユーザの発話内容には、現在のユーザの知識状態が含まれている。ユーザの発話内容をシステム知識に写像することにより、ユーザの誤解を発見し、修正することができる。コマンドのオプションの選択ミスもこれに含まれる。

例)

ユーザ : m でディレクトリを消そうとしたら失敗した。

Assist 3 : m コマンドはファイルを消去するためのコマンドです。ディレクトリを消去するには rmdir コマンドを使います。

以下では、構造写像を用いてユーザの発話内容をシステム知識に写像し、応答内容を検索する手法について詳しく述べる。

#### 4.1 SME を拡張した構造写像アルゴリズム

構造写像を実現するシステムとして Falkenhainer らによる SME (Structure Mapping Engine) [4] がある。SME は二つの領域(ベースとターゲット)の記述集合が与えられると、ベースからターゲットへと構造を写像するが、木構造の写像を対象に設計されているので、ACT-NET のようなネットワーク型の構造の場合には有効に写像することができない。例えば、SME では、ネットワーク構造を写像する際に、ベースには存在してターゲットには存在しない中間ノードを発見することができない。また、SME では、ベースには存在するがターゲットには存在しない情報を候補推論として導くことができるが、逆にターゲットには存在してベースには存在しない情報を検出することはできない。

本研究では、SME を拡張することによって、一般的なネットワーク構造を有効に写像できるようにしている。Assist 3 における構造写像ルーチンでは、ベースとなる概念からターゲットとなる概念に構造を写像し、図 4.1 のような結果を出力している。

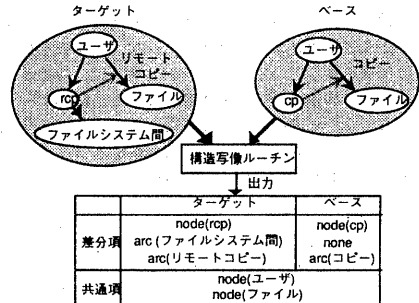


図 4.1 構造写像ルーチンの入出力

図 4.1 では、ベースを cp コマンド、ターゲットを rcp コマンドとする写像を考えている。SME を適用した結果、ベース・ターゲット間のノードやアークの全ての対応関係が明らかになっており、共通項としては node(ユーザ)、node(ファイル)が検出さ

れている。また、ターゲットには存在しベースにはない node (ファイルシステム間) や、ターゲットとベースで異なっている arc (リモートコピー)、arc (コピー) などが差分項として検出されている。

#### 4.2 応答内容の検索例

4.1 節で示した構造写像ルーチンを用いることにより、前述の4つの応答戦略が全て可能となる。例として、(4.3)の対話について応答内容を検索する様子を図4.2に示す。図4.2では、ユーザが「rcpとは何ですか」と質問すると、ユーザの要求はユーザモデル内にACT-NETの表現形式として構造化される。これをターゲットとし、システム知識をベースとして構造を写像すると、評価スコアの高い対象概念 rcp と類似概念 cp が検索でき、両者の共通項、差分項を用いて分かりやすく説明することができる。

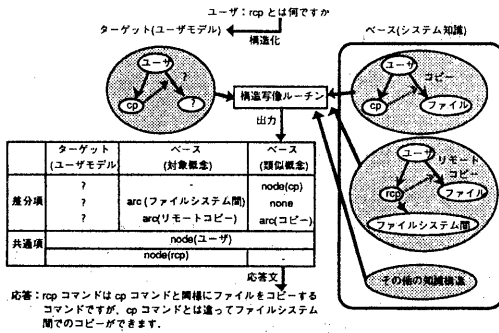


図 4.2 構造写像による応答内容の検索例

また、ユーザモデルからシステム知識への写像において、システム知識には存在して、ユーザモデルには欠落しているような差分項が得られれば、ユーザへの補足説明となる。さらに、システム知識は正しいものと仮定しているため、システム知識とユーザモデルの間で食い違った情報があれば、その差分項はユーザの誤解となり、ユーザに注意を与えることができる。

### 5 プラン選択モジュールによるユーザの意図抽出と応答内容の選択

ユーザから得られた発話内容は、しばしば省略や曖昧さを含んでおり、発話内容から直接発話意図を推定することは困難な場合が多い。例えば、「ファイルを見たい」という発話 T は、「カレントディレクトリ下にあるファイル名の一覧を見たい(A)」と「ファイル内容を閲覧したい(B)」の二通りに解釈できる。したがって、この段階では、ユーザに提供すべきプランを決定することはできない。しかしながら、さらに対話を進めてユーザの意図を一意に決定できれば、ls コマンドと cat コマンドのいずれをユーザに提供すればよいかを一意に決定できる。このように、意図抽出問題と応答内容の選択問題は区別することができる。言い換えると、ユーザの意図を抽出することは応答内容を決するための必要十分条件と言える。

ここで、前述のユーザが「ファイルを見たい」と発話した例を論理的に定式化すると、

T, A, D, B, D  
A, U, B

となるので、A と B のどちらか一方を選ぶ問題はアブダクションにおける小前提(仮説)の選択問題となる。アブダクションにおける仮説選択のためのヒューリスティックスは数多く提案されているが[5][6]、これらの選択基準の殆どは、仮説を含んだ説明構造からの数値的な統計算出のみに基づいており、意味的情報は考慮されていない。

ユーザにとっての好ましいプランを選択するためには、ユーザが意味的にどんなプランを望んでいるのかを考慮すべきである。ユーザに対して提供するプランには、潜在的に意味的な付加属性が備わっており、その付加属性の顕在性はユーザが置かれている状況に依存する。質疑応答システムにおいてユーザの意図を抽出するためには、各候補プランに備わっている意味的な付加属性のうち、どの付加属性をユーザが期待しているのかを決定すべきである。つまり、単に直接的にユーザに意図を質問するのではなく、ユーザが発話するに至った状況についての質問を行うべきである。

本論文では、ユーザの意図に曖昧性が生じた場合は、プラン選択モジュールが対話履歴から曖昧性解消のために必要な情報を抽出して、候補プランに注釈を付加することにより、応答内容を決する手法を提案する。プラン選択モジュールとは、ユーザに対して提供するプランの候補が複数存在する場合、ある基準によって単一の候補を選択するための手続きを記述したモジュールである。また、注釈とは対話履歴を利用して候補プランに付加される意味情報である。注釈を利用してプランを選択する手法としては、文献[7]があるが、注釈の種類が4つしかなく、対象領域に依存した注釈は扱っていない。また、注釈の付加方法に関しても述べられていない。

本研究では、注釈の種類に関しては、注釈を階層的に表現することにより、一般的な注釈から対象領域に依存した注釈まで幅広く取り扱うことを可能にしている。また、知識ネットワークに埋め込まれた注釈ノードを活性化することにより、個々のプランに潜在的に備わっている注釈を状況に応じて顕在化させることが可能となっている。

注釈によって候補概念を一意に絞り込めなかった場合は、候補概念の一般性に着目して、システム知識のノードの活性化から一つの概念を選択するようにしている。例えば、mkdir コマンドと rm -r コマンドはどちらもディレクトリを消去するためのコマンドであるが、mkdir コマンドの方がより一般的に使用される。

#### 5.1 注釈による意図抽出と応答内容の選択

前述のように、注釈は階層的に取り扱い、それらは知識ネットワーク内に埋め込まれている。注釈の階層性と知識ネットワーク内への埋め込みの例をそれぞれ図5.1と図5.2に示す。

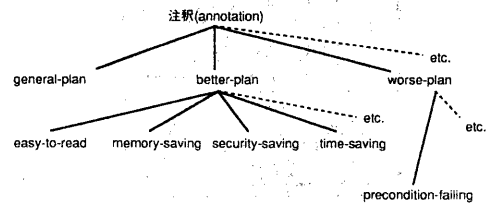


図 5.1 注釈の階層構造

図 5.1 では、注釈には general-plan, better-plan, worse-plan などがあり、さらに better-plan には easy-to-read, memory-saving などがあることが示されている。ここで、注釈 general-plan のみはネットワーク内に明示的には記述されていない。general-plan は長期記憶としてのシステム知識において、同一の目標に対してプラン

のノードに付加された活性度が最大のものである。すなわち、長期記憶としての活性度が大きいプランほど、より一般的に利用されているものとしている。

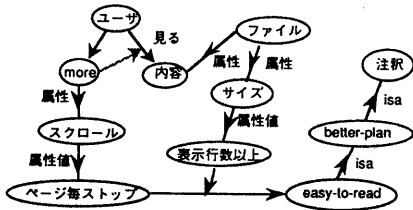


図 5.2 注釈を埋め込んだmore コマンドの表現

図 5.2 においては、「more」というノードに対して属性「スクロール」とその属性値「ページ毎ストップ」が与えられている。この属性値は、「ユーザが見るファイルの大きさが表示行数以上である」という前提状況において「読みやすい(easy-to-read)」ものとなるので、「表示行数以上」というノードが活性化された場合に限って注釈ノード「easy-to-read」が活性化され、moreコマンドに注釈が付加されたものとなる。

## 5.2 応答内容選択アルゴリズム

前述のような注釈を利用してユーザの質問意図を推定し、応答内容を選択するアルゴリズムを示す。

図 2.1 において、プランニング部がユーザの質問内容から構造写像により応答内容を検索する場合に、候補が複数存在するならば、プラン選択モジュールを利用して一つの候補を選択する。ここでプラン選択モジュールは、モジュール化された手続きとなっており、プランニング部が入力文のタイプと検索候補をメッセージとしてプラン選択モジュールに入力すると、一つの候補を選択して出力するというしくみになっている。検索候補から一つの応答内容を選択するためのプラン選択モジュールの手続きを図 5.3 に示す。

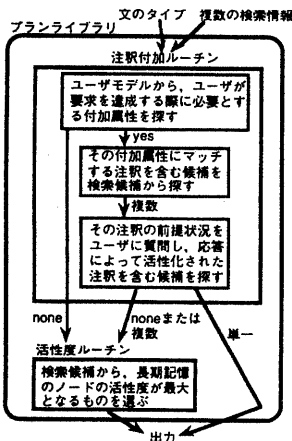


図 5.3 応答内容を選択するためのプラン選択モジュール

図 5.3 において、プラン選択モジュールに複数の検索候補が入力されると、ユーザにとって必要な付加属性を最優先するために、まず、注釈付加ルーチンによって候補の絞り込みを行う。注釈付加ルーチンは、まずユーザモデルからユーザにとって必

要な付加属性を検索する。続いて、その付加属性にマッチする注釈の前提状況が活性化されていないならば、前提状況が真かどうかをユーザに質問して、その候補の注釈を活性化させ、活性化された注釈を含む候補を選ぶ。ここでユーザが使用したプランと違う解が得られれば、その解は代替案となる。

注釈付加ルーチンによって候補が一意に絞り込めなかった場合は、活性度ルーチンにより候補を一意に決定する。活性度ルーチンは、検索候補から長期記憶のノードの活性度が最大となるものを選び、その検索候補のうち最も一般的に利用されるものを一意に決定することができる。

このようにして、プラン選択モジュールは複数の検索候補からユーザに提供すべき情報を一意に選択するようにしている。また、選択においては、最初に注釈付加ルーチンを呼び出すことにより、意味的ヒューリスティクスを優先するようにしている。

## 5.3 具体的な例

次に、プラン選択モジュールにより応答内容を選択する簡単な例を示す。

ユーザ : ファイルの内容を見たい。

まず構造写像により、cat コマンドや more コマンド、さらに vi や emacs などのエディタなどの複数の候補が検索される。つぎにプラン選択モジュールが呼び出されるが、ユーザにとって必要な付加属性がユーザモデルより得られないので、注釈付加ルーチンが失敗する。このため、活性度ルーチンにより、長期記憶としてのノードの活性度が最大となる cat コマンドが、最も一般的に使用されるものとして選択される。

Assist 3 : cat コマンドを使いなさい。

ここでユーザは実際に cat コマンドを試みるが、ファイルのサイズが大きいため画面がスクロールしてしまい、一度には読めない。

ユーザ : 読みづらいですね。

ユーザの発話内容は「ファイルの内容を読みやすく表示させるためにはどうするのか」となるが、「読みにくい」というのは、画面のスクロールの他に、画面が暗い、漢字コードが対応していないなどの様々な状況が考えられる。したがって、プラン選択モジュールはどのような状況において「読みにくい」という事態が発生したのかを同定しなければならない。言い換えると、図 5.2 における more コマンドの注釈ノード「easy-to-read」を活性化させるために、前提状況となるノードを活性化させなければならないので、ユーザに対して次のような質問を行なう。

Assist 3 : ファイルの大きさが表示行数以上であるために読みづらいのですか。

ユーザ : そうです。

ここでノード「表示行数以上」が活性化され、それに伴って注釈ノード「easy-to-read」が活性化されるので、プラン選択モジュールはユーザに対して more コマンドを提供すればよいことが分かる。

応答 : more コマンドを使えばよい。more コマンドは cat コマンドと同様にファイルの内容を表示するコマンドですが、cat コマンドと違って1ページ毎にストップしてスクロールすることができます。

## 6 対話の履歴管理

### 6.1 ユーザモデル

ユーザの発話内容やシステムからの応答内容に該当するシステム知識内の各ローカルネットワークは、対話の履歴としてユーザモデル領域にコピーされる。コピーされたネットワークは、ユーザが既知に知っていると思われる概念の検索の他、後述する“忘却”の仕組みによって、現在の対話の焦点や話の前後関係を知るために利用される。例えば、ユーザから意味的曖昧性を伴う不完全な入力文が与えられる場合には、以下のような手段により対処している。

- (6.1) まず、対話の履歴から、入力内容が現在の話題に即して解釈できるかどうかを調べる。
- (6.2) 解釈できない場合には、対話履歴中の他の話題に即した解釈を試みる。
- (6.3) もし既定の選択基準では過去のどの対話内容に関係するのかを決定できないならば、内容が合致する最も最近の話題から解釈する。

### 6.2 ユーザモデルにおける “忘却”

ユーザモデル中のノードやアークは、一度活性化されると、次に活性化されるまで徐々に活性度が0に近づく。この過程を“忘却”と呼んでおり、対話において過去の内容が徐々に意識から遠ざかるという、人間の短期記憶に関する考察に基づいている。時刻  $t$  の活性度  $A(t)$  は、

$$A(t) = \alpha A(t-1) \quad (0 < \alpha < 1)$$

によって与えている。現在の Assist 3 では  $\alpha = 0.8$  としている。また、時間の単位としては、ユーザからの1回の入力を1単位時間としている。

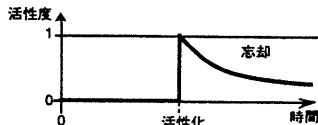


図 6.4 活性度の時間変化の例

忘却の仕組みにより、現在最も活性化されているノードを探すことによって、対話の焦点となっている概念をユーザモデルの中から見出すことができる。したがって、ユーザモデルは、Carberry [8] の “Context Model” としての働きを持っている。ただし、Context Model が、概念間の関係と対話の焦点を表すのみに重点を置いているのに対して、本システムのユーザモデルでは、活性度の高い要素から順にトレースしているために、対話の焦点の移り変わりを表すことができるという特徴がある。すなわち、現在の対話がどのような目的のために行われているのか、また、過去にどのような順序で何を話したのかを、ユーザモデルから推定することができる。例として、以下のような対話を考える。

- 例) ( ) 内の数値は、対話の各段階において、以下の概念を表す各ノードの活性度 (小数点以下2ケタまで) である。  
 [ F: ファイル, D: ディレクトリ, W: ライトパーミッション ]  
 ユーザ: ファイルを消したい。

(F = 1.00, D = 0, W = 0)

Assist 3: ファイルを消すには rm コマンドを使います。

(F = 1.00, D = 0, W = 0)

ユーザ: ディレクトリも消したい。

(F = 0.80, D = 1.00, W = 0)

Assist 3: ディレクトリを消すには rmdir コマンドを使います。

(F = 0.64, D = 1.00, W = 0)

ユーザ: 消えない。(※1)

(F = 0.51, D = 0.80, W = 0)

Assist 3: ライトパーミッションは設定されていますか?

(F = 0.41, D = 0.64, W = 1.00)

ユーザ: ライトパーミッションとは何ですか?

(F = 0.33, D = 0.51, W = 1.00)

Assist 3: ライトパーミッションとはアクセス権の一種です。

(F = 0.26, D = 0.41, W = 1.00)

ユーザ: 中身を見たい。(※2)

(F = 0.21, D = 0.33, W = 0.80)

Assist 3: ディレクトリの内容を見るには ls コマンドを使います。

(F = 0.17, D = 1.00, W = 0.64)

上記の例において、(※1)、(※2)の部分では、ユーザが指している対象を推定しなければならない。(※1)では、直前の対話内容から、対象が“ディレクトリ”であると推定するのが妥当である。しかし、(※2)では、何の“中身”を指しているのかを直前の対話内容から知ることはできない。Assist 3では、ユーザモデルから、“中身”と同様な意味の属性を持つノードを探す。上記の対話例では、(※2)の直前の主題である“ライトパーミッション”には“中身”に相当する属性が存在しないために、“中身”の類似概念である“内容”属性を持つ“ファイル”と“ディレクトリ”が候補として検索されている。“忘却”の仕組みにより、“ディレクトリ”の方がノードの活性度が高く、したがって、時間的により現在に近いことがわかる。“ファイル”と“ディレクトリ”の間には、他に比較できる情報が無いため、Assist 3は、(※2)の解釈が「ディレクトリの内容を見たい。」であると仮定して、応答内容を決定する。

## 7 応答文の構成

類推機構で得られた情報から、必要に応じてプラン選択モジュールを利用することにより応答内容が得られると、入力文のタイプに応じたスキーマによって、応答文の構成が決定される。

ユーザに対する説明文の構造については、Paris & McKeown [2] と McKeown [9] がある。文献 [2] では、説明文の要素として Identification (対象概念の同定)、Cause-effect (複数量要素間の因果関係の説明) など計7種類を挙げている。Assist 3では、これらの要素に加え、ユーザがトラブルに陥った場合の助言として、ユーザの知識不足や誤解の指摘および訂正を行なうようにしている。

本研究では、文献 [10]などを参考に、初心者ユーザと専門家との対話における説明文の構成について統計的に考察し、入力文のタイプに応じた応答文の構成をスキーマとして定義した。Assist 3では、入力文を以下の6種類のタイプに分類している。

- (7.1) what 型 (「…とは何? (what)」, 「…にはどのコマンドを使うのか? (which), etc.)
- (7.2) how 型 (「…するにはどうする? (how)」, 「…したい。(want)」, etc.)
- (7.3) why 型 (「…となるのは何故? (why)」, 「…できない (cannot)」, etc.)

- (7.4) yes-no 疑問型 (「…ですね? (yes-no)」, etc.)
- (7.5) yes/no 型 (「そうです。 (yes)」, 「いいえ (no)」, etc.)
- (7.6) fact 型 (「…しました。 (fact)」, etc.)

how 型の質問文に対しては、プラン選択モジュールにより応答文の構成が決定される。what 型、why 型の質問文については、以下のような構成で応答文が生成される。ただし、各項目は、該当する応答内容が得られなければ省略される。

- (7.7) [what 型質問文に対する応答文の構成]
  - (1) 概要 (簡単な説明)
  - (2) より詳しい説明
  - (3) 動作の流れや概念間の因果関係の説明
  - (4) 関連情報の補足
- (7.8) [why 型質問文に対する応答文の構成]
  - (1) ユーザの誤解の指摘と訂正
  - (2) ユーザの知識不足の指摘と知識の補完
  - (3) システムが与えた手段に不都合がある場合の修正案の提示
  - (4) より適切なプランがあれば提示
  - (5) 別の手段を採用する際の注意事項の提示

以下に、why 型質問文に対する応答例を示す。応答文中の番号は、上記 (7.8) において対応する番号を表す。

例)

ユーザ：ディレクトリを消すために rm コマンドを使ったが、消えない。何故か？  
 Assist 3：rm コマンドはファイルを消すためのコマンドです。…  
 (1) rm コマンドを使ってディレクトリを消すには -r オプションを付加しなければなりません。…(2) ディレクトリを消去するには rmdir コマンドを使用できます。書式は  
     rmdir ディレクトリ名  
 です。…(4) ただし、ディレクトリの階層下が空でなければなりません。…(5)

## 8 結論と今後の課題

本論文では、静的な概念と動的な概念を1つのネットワーク上に記述できる ACT-NET モデルを提案した。このモデルにより、従来の対話型システムでは、各々の概念表現に対して、同種の処理を別々のルールとして記述していたところを、単一の処理として取り扱うことが可能になった。

また、類推という統一的手法を採用することにより、質問文のタイプによらず、構造写像を用いて必要な情報が抽出できることを示した。さらに、抽出された情報と“注釈”を利用することにより、アナロジーによる分かりやすい説明や、ユーザの誤解の認識を可能にした。

今後の課題としては、以下のような点が挙げられる。まず、現在は、全てのシステム知識を人間が ACT-NET 表現に変換して定義しているが、ユーザモデル内には、システム自身が入力文を ACT-NET 表現に変換して格納している。したがって、この処理部分を拡張することにより、教師からシステム知識を自動的に学習できるようにすべきである。ただし、文の構造に極度に依存したネットワークにならないよう、様々な文から、できる限り冗長性のない統一的なネットワーク構造に変換するアルゴリズムを設計する必要がある。しかしながら、これらはエキスパートシステムにおける知識獲得の問題に相当するため、現時

点では、教師からの入力文にある程度の制限を与えた上で、知識獲得の自動化を考えている。

一方、応答内容の候補の中で、複数の候補に同一の注釈が付加された場合、意味的に妥当な選択手段が用意されていない。そこで、注釈以外の選択基準として、コヒーレンスを導入することを考えている。すなわち、同じ注釈を持つ候補概念が、現在までの対話内容に各々どの程度の関連性を持っているかを計算することによって、各候補に優先順位を与える。今後は、ACT-NET モデルに対して妥当なコヒーレンスの計算方法を設計しなければならない。

また、現時点では、“忘却”による活性化の減衰率が時間のみをパラメータとして定義されており、ユーザモデルにおける概念間の関連性が考慮されていない。対話の目的としてユーザが念頭においている概念は、たとえしばらくの間直接には触れていなくとも、関連性のある概念が話題となっている限り、強く意識されているはずである。したがって、現在の話題の焦点となっている概念に対して関連性の高い概念ほど、活性化の減衰率が低くなるように活性化の減衰率を定義すべきである。そのように定義できれば、ユーザモデルにおける活性化が、意味的に妥当性のあるコヒーレンスの計算に利用できると考えている。今後は、“概念間の関連性”を各概念を表すノード間の“距離”としてとらえ、ACT-NET モデルにおけるノード間の“距離”の定義と、コヒーレンスの計算要素としての活性化の導入について考えていく予定である。

## 参考文献

- [1] 細見, 上原, 前川: 日本語 Tree Adjoining Grammar と語順変化の制御, 第 43 回情報処理学会全国大会予稿集, 4G-1 (1991).
- [2] Paris, C. L., Mckeown, K. R.: Discourse Strategies for Describing Complex Physical Objects in G. Kempen (ed.), Natural Language Generation, Martinus Nijhoff Publishers, pp.97-115 (1987).
- [3] Gentner, D.: Structure-Mapping: A Theoretical Framework for Analogy, Cognitive Science, Vol.7, No.2.: pp.155-170 (1983).
- [4] Falkenhainer, B., Forbus, K. D., and Gentner, D.: The Structure-Mapping Engine: Algorithm and Examples, Technical Report UIUCDCS-R-87-1361, University of Illinois at Urbana-Champaign (1987).
- [5] Hobbs, J. R., Stickel, M., Martin, P. and Edwards, W.: Interpretation as Abduction, Proc. of the 26th Annual Meeting of the Association for Computational Linguistics, pp.95-103 (1988).
- [6] Ng, H. T., and Mooney, R. J.: Abductive Explanation in Text Understanding: Some Problems and Solutions, Technical Report AI89-116, Department of Computer Sciences, University of Texas at Austin (1989).
- [7] Beek, P. V. and Cohen, R.: Resolving Plan Ambiguity for Cooperative Response Generation, Proc. of the 12 th International Joint Conference on Artificial Intelligence, pp.938-944 (1991).
- [8] Carberry, S.: Modeling the User's Plans and Goals, Computational Linguistics, Vol.14, No.3, pp.23-37 (1988).
- [9] Mckeown, K. R.: Discourse Strategies for Generating Natural-Language Text, Artificial Intelligence, Vol.27, pp.1-42 (1985).
- [10] 大野 浩之: root 1/1000000, Super ASCII, アスキー出版, Vol.2, No.3 - Vol.3, No.1 (1991 - 1992).