TM-0841

仕様獲得実験システム

土田賢省，阪田全弘，
鈴木宏文，町田和浩（日本電気）

December, 1989

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

仕様獲得実験システム

土田賢省　　阪田全弘　　鈴木宏文 *　　町田和浩 *

日本電気(株) ソフトウェア生産技術開発本部
Software Engineering Development Laboratory, NEC Corporation

* 日本電気技術情報システム開発(株)
NEC Scientific Information System Development Coporation

概要

エキスパートシステムのユーザインタフェースを対象とした仕様獲得実験システムについて報告する。本システムでは、ユーザの要求から厳密な仕様を獲得する作業をユーザとの例示による対話を繰り返しながら仕様を詳細にしていくプロセスとしてモデル化した。このモデルに基づく仕様獲得を計算機上で可能とするために、ユーザが具体的イメージで例を与えることのできる例示入力機能、ユーザの与えた具体例から抽象度の高い仕様を獲得するための一般化機能、および獲得された仕様をシミュレーション実行する仕様確認機能を備えている。実験により診断型エキスパートシステムの質問と結論のユーザインタフェースを規定する仕様を例示から獲得できることが確かめられた。

A Prototyping System for Dialog Specifications with Acquisition

Kensei TSUCHIDA　　Masahiro SAKATA

Software Engineering Development Laboratory, NEC Corporation

11-5, Shibaura, 2-chome, Minato-ku, Tokyo 108, Japan

Hirofumi SUZUKI　　Kazuhiro MACHIDA

NEC Scientific Information System Development Corporation

ABSTRACT

A specification acquisition method of expert systems' user interfaces is described. This method uses examples as specification media with models of knowledge used in the specification refinement process between the designer and the user. The system's task knowledge can situate each example in an appropriate context of interaction. The interaction technique knowledge can provide *an vocabulary* in the example presentation and can constrain the presentation. A formal generalization method applied to example representation schemes can generalize them to specifications. An experimental system based on this method can acquire a schematic user interface of a real diagnostic expert system.

1 はじめに

一般にエキスパートシステムの開発では、ユーザインタフェース部の工数が全体の大部分を占め、その構築に多大な労力を必要としている [Smith 83][Smith 84]。また、既存の多くのエキスパートシステムを見ると、ユーザインタフェース部が推論部に比較して非常に変化に富んだものとなっている。例えば、推論は前向きや後向きなどの多くのエキスパートシステムに共通な推論方式が採用されるのに対し、ユーザインタフェースはウィンドウの大きさや形状、対話方式そして表示レイアウトなどがユーザの好みや稼働環境などにより個別システム毎に変わってくる。そして、ユーザインタフェースの仕様はユーザの好みにかかわることから、頻繁に仕様の変更や追加要求が起こりこの部分の後戻りコストが大きくなっている。そのため、エキスパートシステムの（特にユーザインタフェースに関する部分の）開発ではプロトタイピング方式がとられることが多い。このようなことからエキスパートシステムの開発の初期段階において、ユーザとの対話の繰り返しによりユーザインタフェースの厳密な仕様を固める作業をマシン上で支援できれば、ソフトウェアの生産性の観点からも効果が期待できる。

エキスパートシステムの開発プロセス　実際のエキスパートシステム開発作業 [Akao 89] では、次のようなことが起こる。

(1) 推論方式が決定されると、仕様定義者（ユーザ）は具体的に画面のスケッチなどの例によりユーザインタフェースの要求を開発者に提示する。

(2) 開発者は、提示された例からユーザの意図を理解して、ユーザインタフェース構築やプログラムに関する知識を用いて、ユーザの要求を満たすプログラムを作成する。

(3) ユーザは作成されたプログラムを実行して自分の要求が満たされているかを確認する。

(4) 確認のためのテスト対象範囲を広げていくうちに（他のデータ・状況でテストを行ううちに）仕様の変更あるいは追加の要求が出て来れば、再び例によって開発者に提示する。

このようなやりとりを繰り返しながら次第に最終的な仕様に収束していく。図1はエキスパートシステムの開発プロセスにおけるユーザインタフェース仕様の厳密化過程を示したものである。このような分析に基づき、ユーザインタフェースモデルをベースとした例示の一般化によるユーザインタフェース仕様獲得の（半）自動化を試みた。

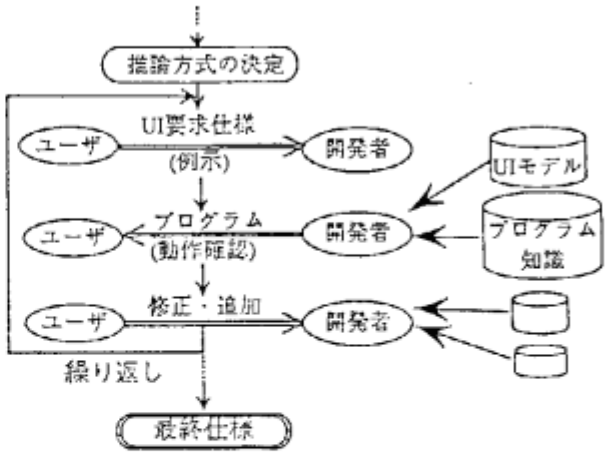

図 1: エキスパートシステム開発プロセス

2 ユーザインタフェース仕様獲得技法

エキスパートシステムのモデル　対象とするエキスパートシステムを次のようにモデル化した。エキスパートシステム全体は、知識ベース部、推論部、ユーザインタフェース部の3つの部分により構成される。知識ベースにはネットワーク表現された知識が格納される。さらに、ネットワーク表現された知識に対して基本的なアクセス関数群が用意されている。推論は、ネットワーク表現中のトークンや予約語（start, quit など）の系列を入力として、状態遷移ネットを生成する機械として定義される。ここで、状態とは推論の計算状態のことである。ユーザインタフェース部は、推論の各状態に対する表示あるいはユーザとのインタラクションをつかさどる。

ユーザインタフェース仕様　ユーザインタフェース仕様とは、前述のように、推論途中でユーザからの情報が必要になったときどのような対話手段（例えば、メニューのタイプ・メッセージの内容・大きさ・形状）によりどのような対話を行うか、あるいはユーザにどのように推論過程・結果を見せるか（どういうタイミングでどの種類の説明用ウィンドウとどの種類を見せるか、そしてこれらのレイアウトや操作方法）などを定義するものである。他の言葉で定義すると以下のようになる。ユーザインタフェース仕様は、表示内容を記述する意味的仕様と表示やインタラクションの方法を記述する物理的仕様とからなる。意味的仕様は、表示内容を内延的に定義するものであり、ネットワーク表現アクセス関数などを用いて規定される。物理的仕様は、アクション（ウィンドウのオープンなど）の系列を記述する動作仕様と表示物の大きさ・形状・位置関係を記述する空間仕様からなる。そして、エキスパートシステムのシミュ

レーション時にはユーザインタフェース仕様は推論の状態を記述したものとのペアで解釈される。

モデルと例示の役割　ユーザインタフェース仕様獲得において、仕様表現方法の表現能力の高さと表現のし易さとが重要である。表現方法が何らかの対話モデル[Green 86][Hartson 89][Myers 88b]に基づいていると、次のような利点がある：

- 表現能力の形式的限界が明確になる。
- もしモデルが目的とする対話にとって自然なモデルであれば、意図した仕様を容易に表現できる。

また、本論で述べるような対話仕様の自動獲得の観点から見ても、対話モデルを持つことには次のような利点がある。

- 獲得された仕様の妥当性をモデルに基づいて検証できる。
- モデルに基づいて必要な対話処理を予測したり、不適切な対話を抑制することができる。

一方、前述のようなエキスパートシステムの開発プロセスでは、ユーザ（要求者）と開発者との間に交換される情報や、それぞれが持つ知識にも注目する必要がある。開発者は、ユーザインタフェースに関する詳細な知識を持つ。一方、ユーザ（要求者）は、例を通じて要求仕様を提示する場合が多い。これは、ユーザインタフェースの形式的表現が、必ずしもユーザに理解し易いものではないからである。この非形式的な要求に答えて、開発者は、ユーザインタフェース部品の組合せ方やユーザインタラクションの制御などを決定する。確認のために開発者からユーザへ示される情報も、部分試作したプログラムの実行による画面イメージなどの例である場合が多い。

従って、ユーザインタフェース仕様獲得を（半）自動化する方法として以下の様なものが考えられる：ユーザから対話の例示を受け、その妥当性を判定し、それから一般的な仕様を推論して、その結果を再び例示によりユーザに確認させる。この場合、ユーザインタフェースモデルは、妥当性の判定や一般化において重要な役割を果たす。

ユーザインタフェースモデル　診断型エキスパートシステムにおける対話のように、システムと人とが協調的に問題解決を行うための対話の機能として、次のようなものが本質的である：システムが情報不足のため計算（推論）を継続できなくなり、計算再開のため人の情報を獲得する機能である。

このような機能に注目した場合、計算過程を監視して、いかなる情報が不足しているために計算が継続できないかを判断できなくてはならない。そこで、システムの内部状態を示すものとして推論を特徴付けるパラメータを定義し、そのパラメータの値をウォッチできるようにする。

計算は、システムの内部状態と入力とにより完全に決定されるものとする。外部には、計算が中断した時点毎の内部状態（の表現）を見せる。この内部状態から計算の停止理由を推論する。また、内部状態がシステムからのメッセージを構成するための最小単位の語となる。

システムや人にとって必要な情報を基本的なトークンを組み合わせて生成しなければならないことがある。このトークンを組み合わせて計算に必要な情報を構成することがユーザインタラクションにおける主な計算である。また、妥当性のないトークンを拒否したり、脱出処理の制御もユーザインタラクションの一部と考えることもできる。以降、計算状態を情報を構成する文脈を与える推論の計算状態とトークン単位に処理していく個々の状態を記述するユーザインタラクションの計算状態のマクロとミクロに分けて考える。

ユーザインタフェースは、対話メディアを通してトークンの取り出し・格納を繰り返すものと見なす。取り出し・格納などは、それぞれ基本対話処理を組み合わせて実現する。従って、ユーザインタフェースを、基本対話処理の具体化としてモデル化する。これは、人によるユーザインタフェース仕様の獲得における基本対話処理（およびそれらから構成される複合対話処理）をいかなる順番に結合して、要求仕様を実現するかを決定してゆく過程のモデル化になっている。そして基本対話処理をノードとするグラフを処理フローといい、このグラフを獲得することをユーザインタフェース仕様の獲得とみなす。

基本対話処理は、開発者が持つ対話処理知識の抽象化である。基本対話処理には、計画問題における動作記述と同様な構造を与える：

- 処理記述、
- その処理を呼び出すための必要条件を記述した前提条件、
- その処理を実行した場合（文脈に依存せずに）ユーザインタフェースの計算状態をどのように変更するかを記述した変換記述、

- その処理を実行した後に満足されているべき遷移後条件。

対話処理の逐次制御は、処理フロー上の道（パス）として表現される。

例示と一般化　ユーザからの例示により、処理フローの具体例を獲得する。例示は、基本対話処理を逐次的に具体化することにより行う。基本対話処理は、以下のように具体化される：

(1) ユーザが一つの基本対話処理を選択する。

(2) 現在の計算状態がその処理の前提条件を充足しているかをチェックする。

(3) 充足している場合、前提条件を現在の計算状態で置換する。以降、置換した計算状態の記述を、その処理を呼び出した理由・説明と解釈する。

(4) 処理記述に従って（インタプリタが）処理を実行し、計算状態を変更する。

(5) 遷移後条件をチェックする。実際には、計算状態の表現に対して、充足のチェックや変更を行う。従って、(3) や (4) において計算状態の表現が条件を充足していない場合、その処理は具体化されない。

変換は、一般的には、ユーザインタフェースの計算状態をのみ変更する。従って、一つの処理フロー内では、推論の計算状態は不変である。

対話が必要になった時点で、既に獲得された処理フローの中に、呼び出し理由（もしくはそれを一般化したもの）を現時点の計算状態が充足しているものがある場合、その処理フローを現在の対話処理の候補として選択する。そして、ユーザと協調的に基本対話処理単位でその妥当性を検証してゆく。この場合、処理の呼び出し理由を現在の計算状態が充足できるまで一般化する。もし、このような処理フローが存在していて、ユーザが明示的に分岐を指示した場合、候補の処理フローの呼び出し理由と現在の計算状態（新たに例示された処理フローの呼び出し理由となる）を弁別して、それぞれの呼び出し理由を特殊化する。

計算状態は、一階のリテラル（0 変数の命題リテラルも含む）の集合（連接と解釈する）として表現する。述語引き数としては、アトムもしくはアトムのリストもしくは変数を取る。各引き数は、型を持つ。一般化は、型の階層に沿って行う。即ち、arg1:type1 と arg2:type2 とは、type1 と type2 との共通の型 type12 が存在する場合に限り、arg12:type12 に一般化する。arg1 = arg2 の場合、arg12 = arg1。arg1 ≠ arg2 の場合、arg12 =変数である。実際の計算状態と基本対話処理に付随した前提条件（具体化された基本対話処理の呼び出し条件）とから以下のような 6 種のリテラル集合を生成する：

A 完全に一致するリテラル

B 実際の計算状態の方が一般的なリテラル（通常は空集合）

C 呼び出し状態の方が一般的なリテラル

D 一般化により単一化できるリテラル

E 実際の計算状態にのみ存在するリテラル

F 呼び出し状態にのみ存在するリテラル

実際の計算状態においてある呼び出し条件を持つ対話処理を呼び出す（あるいは、具体化できる）ためには、F が空集合でなければならない。また両者が弁別できるためには、B から F の集合の内少なくとも一つが非空集合である必要がある。

ある計算状態において、二つ以上の処理フローが呼び出し可能であった場合、いずれか一つを選択するために、上述の 6 種のリテラル集合の状態から決定されるヒューリスティックな順序関係を利用する。

【例】以下の様な診断知識を考える：

```
q1:a & q2:b => d1.
```
(q1 の答えが a かつ q2 の答えが b ならば d1)

```
q1:a & q2:c => d2.
```
(q1 の答えが a かつ q2 の答えが c ならば d2)

.
.
.

そして、これらを図 2のような決定木の形式に翻訳する。

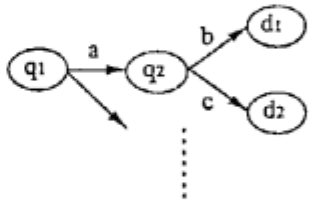

図 2: 決定木

いま、質問 q1 により、回答 a を得て、q2 に関する情報を得るために計算を停止したとする。この場合、推論の計算状態として、例えば、リテラル集合

```
{
  is_in_query_state,
  is_current_node(q2),
  is_current_branch([b,c])
}
```

を持つ（型は無視する）。既に、質問q1の段階で、質問から回答を得て再び推論を行うまでの例示が与えられていた場合、その処理フローの呼び出し条件は、

```
{
  is_in_query_state,
  is_current_node(q1),
  is_current_branch([a,...])
}
```

なる形式である。もし例示された処理フローを質問q2においても適用するとすると、呼び出し条件を一般化して、

```
{
  is_in_query_state,
  is_current_node(X),
  is_current_branch(Y)
}
```

なる形にする（X、Yは変数）。更に推論が進行して、結論まで到達したとする。その時点での計算状態は、

```
{
  is_in_conclusive_state,
  is_current_node(r),
  is_current_branch([])
}
```

のような形である。もしこれまでの処理フローが、結論における対話として不適切であるとして、新たに処理フローが例示された場合、その処理フローの呼び出し条件は、この計算状態そのものである。そして、ヒューリスティックな順序付けにより、結論では新たに例示された処理フローを選択するようになる。

3 実験システム

上述のユーザインタフェース仕様獲得方法の妥当性を検討するため、Prologマシン PSI-II においてSIMPOSのウィンドウシステム上に実験システムを作成した。この実験システムを用いて、診断型エキスパートシステムのユーザインタフェース仕様の獲得を行った。

システムの基本構成　図3に仕様獲得実験システムの基本構成を示す。ユーザは、例示モニタにより各推論状態に応じた対話方法や推論過程の説明用画面などを構成する。例示モニタでは、推論の状態を表すパラメータの値が表示されている。パラメータなどの計算資源やシステムで用意しているユーザインタフェースの媒体（部品）を用いてユーザインタフェース（処理フロー）の具体例を入力する。一般化機構は、前提条件の異なる処理フローから両者に共通する弱い前提条件とそれに対応する処理フローを計算して仕様DBに格納する。他の推論状態に移行したとき、仕様DBの中から最も近い状態の仕様に対して仕様インタプリタによりその状態に合わせた具体化を行い、例示モニタの画面を通じて実行結果を提示する。システムには、ユーザインタフェースモデルと推論モデル（エキスパートシステムのモデル）の知識が組み込まれている。

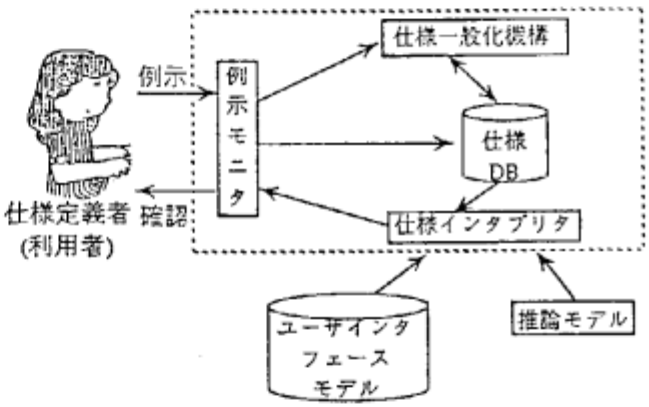

図 3: システム構成

実行例　本システムによるユーザインタフェース仕様獲得の実行イメージを説明する。対象は診断型エキスパートシステムである。

例示入力　図4は、推論の最初の状態に対応した画面である。左のウィンドウには、その時点での推論の状態を示す内部状態パラメータが表示されている。これは、推論の過程を示す情報、あるいはワークメモリの内容に対応するものなどであり、診断型のモデルを反映したものとなっている。この部分は推論が進むにつれてシステムにより自動的に書き変えられていく。ユーザはこの状況に最もふさわしい対話法を作成しようする。まず、ユーザインタフェース部品から適切なものを選択する。図5では、メッセージ付きのメニュー部品を選択している。そして、この部品の大きさと表示位置を決定する。次に、状態

パラメータなどのトークンを資源として表示内容を具体的に指定する。図6では、状態パラメータ2の内容をメッセージ部に、メニューの項目に状態パラメータ3の内容を設定しようとしている。さらに、こうして構成された媒体手段に対してとるべきアクションを指定する。これには、どの情報を推論側に渡すかという内容も含まれる。図7では、メニュー項目の一つである「飛ばない」というトークンを推論側に渡すことを指定している。このようにして1つのセッション（処理フロー）の例示を終え、次の推論状態に遷移する。

（注）実験システムでは、例示入力のために次のようなウィンドウに基づくユーザインタフェース部品が用意されている。それぞれに特有の基本対話処理が定義される。

- テキストウィンドウは、矩形状表示エリアを画面上に開き、文字列を適当な位置に表示する機能を提供する。テキストウィンドウ特有の基本対話処理として、生成・消去・表示停止・表示再開・文字列表示・位置移動・サイズ変更などがある。
- 表題付きテキストウィンドウは、テキストウィンドウの機能に加えて、セッション中固定される表題を表示する機能を提供する。特有の基本対話処理としては、テキストウィンドウのそれに加えて、表題表示がある。
- メニューは、矩形状表示エリアを画面上に開き、文字列のリストを一覧表示して、マウスで選択しトークンを出力する機能を提供する。特有の基本対話処理としては、生成・消去・表示停止・表示再開・文字列のリストの一覧表示・マウス選択・位置移動などがある。
- メッセージ付きメニューは、メニューの機能に加えて、セッション中固定されるメッセージを表示する機能を提供する。特有の基本対話処理としては、メニューのそれに加えて、メッセージ表示がある。更に、計算状態それぞれをトークン保持用のエリアに取り出す処理群。ウィンドウ／メニューにトークンを表示する処理、メニューからトークンを選択する処理、トークンを推論側に渡す処理などがある。

一般化　推論状態が遷移したとき、既に定義したユーザインタフェース仕様を利用してユーザとの対話を進めることが可能である。このとき、仕様の一般化が行われる。また、遷移した状態がこれまでの状態

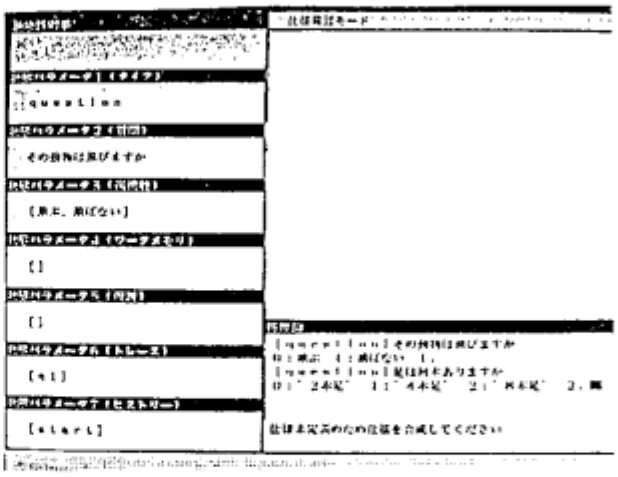

図 4: 例示入力 (1)

図 5: 例示入力 (2)

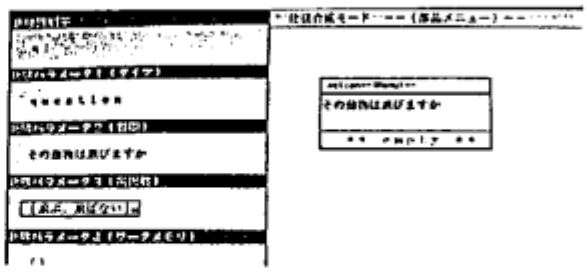

図 6: 例示入力 (3)

図 7: 例示入力 (4)

とは区別すべきとユーザが判断すれば、この場面に適切なユーザインタフェースを新たに定義することが可能である。（図8参照）

仕様確認　典型的な状況での例示が与えられて、ユーザインタフェース仕様が一通り定義されたとみなす。そこで、再びセッションを繰り返し他の推論状態でも、獲得された仕様が適合するかを確認する。さらに、知識ベースを変えた他のエキスパートシステムに適用しシミュレーションすることが可能である。また、獲得された仕様を図式的に表示する機能も備えている。（図9参照）

この実験システムにおいて、次のようなユーザインタフェース仕様を獲得することができた：エキスパートシステムから質問し、それに対してユーザの回答（複数候補からの選択）を得る。それに基づいて推論を繰り返し、結論に到達したらその結論とそれまでの質疑の履歴を表示して終了する。

また、獲得したユーザインタフェース仕様を仕様インタプリタにより（半）自動実行することによりプロトタイパとしての機能を確認できた。

4　議論

実装により確認されたように、プロトタイピングの一方法として本方法を利用できる。

現実装では、獲得された対話仕様をインタプリタが解釈実行している。この場合、個々の処理フローの呼び出し条件の内で、相互に弁別する部分のみが制御にとり意味がある。この部分のみを取り出すことにより、イベントハンドラ群として仕様をコンパイルすることも可能である。

本方法では、対話メディアとその機能を既存のものとしてそれらを組み合わせて対話処理を構成する。

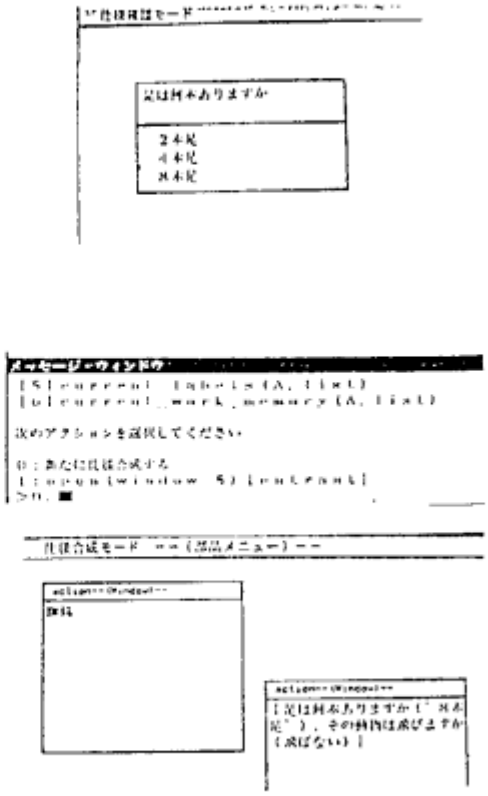

図 8: 一般化

しかし、ユーザインタフェースの仕様化においては、その組み合せによる複雑さが問題となる。それに対しては、例示と一般化が、早期に効率的に絞り込む役割を果していると考えられる。

しかし、現段階では計算（推論）に関する知識や対話に関する知識の知識表現に問題が残る。前者に関しては、計算が中断したことを判定できるようなモデルを与えたにすぎない。従って、計算部からの喚起処理もユーザが例示しなければならない。また、一般化も非常に単純なレベルに留まっている。即ち、ある条件（リテラルの集合）をある事象（リテラルの集合）が充足していることを集合間の包含関係が成立することと同一視している。計算に関する知識を積極的に持ち込むことにより、喚起要求などを自動合成することができるようになると思われる。また、一般化の精度をあげることにより、より細かい制御を例示から獲得できるようになると思われる。これに関連して、「最小汎化」[Plotkin 70] が注目される。

5　結論

本報告では、インタフェースモデルに基づいた例示と一般化によるユーザインタフェース仕様獲得方

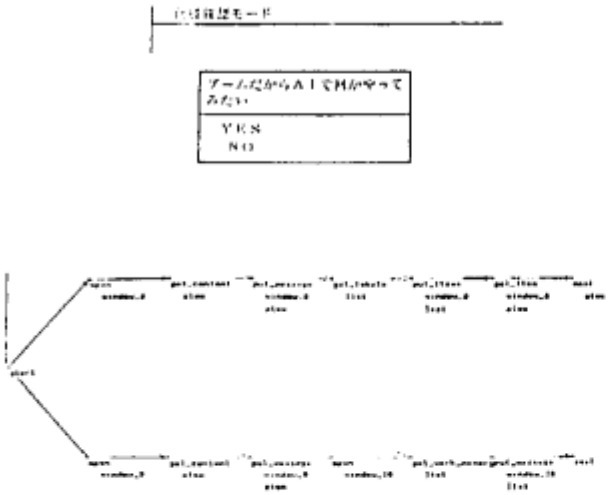

図 9: 仕様確認

法を提案した。また、方法論の部分的な実装である実験システムの実行例と、その方法論の妥当性を検討した結果を報告した。

ユーザインタフェース仕様の獲得と実現とは、非常にコストが掛かる。システムの利用し易さの大部分を決定する因子がユーザインタフェース部にあるとすると、システム開発においていかに効率的にユーザインタフェース仕様を獲得するかが重要になる。そのため、ユーザインタフェース仕様化の工程を開発方法論の一部として組み入れることも提案されている [Hartson 89]。この場合、プロトタイピング (とそれによる検証) を併せもつことも重要であるとされる。このような課題を解決する一方法として、本システムで行っているような仕様獲得方法が有効ではないかと思われる。

謝辞　本研究は新世代コンピュータ技術開発機構 (ICOT) の委託研究として進められました。御支援頂きました長谷川 ICOT 第一研究室室長を初めとする方々に深く感謝いたします。また、研究にあたり、御指導下さったソフトウェア生産技術開発本部川越課長ならびに C & C システム研究所宮下課長に感謝いたします。

参考文献

[Akao 89] Akao, Y. C., K. Tsuchida, and K. Imai (1989) "Debug Expert System for Switching Systems Software," NEC Research and Developments No.95 (to appear).

[Green 86] Green, M. (1986) "A Survey of Three Dialog Models," ACM Transaction on Graphics 5:3, pp.244-275.

[Hartson 89] Hartson, H. R. and D. Hix.(1989) "Human-Computer Interface Development: Concepts and Systems for Its Management." ACM Computing Surveys 21:1, pp.5-92.

[Myers 88b] Myers, B. A. (1986) *Creating User Interfaces by Demonstration*. Academic Press, Inc.

[Plotkin 70] Plotkin, G. D. (1970) "A Note on Inductive Generalization," Machine Intelligence 5, pp.153-163.

[Smith 83] Smith, R.G. and Baker, J.D. (1983) "The Dipmeter Advisor System – A Case Study in Commercial Expert System Development," Proc. 8th IJCAI, pp.122-129.

[Smith 84] Smith, R.G. (1984) "On the Development of Commercial Expert System," AI MAGAZINE, pp.61-73.

[Stefik 86] Stefik, M. and Bobrow, D.G. (1986) "Object-Oriented Programming- Themes and Variations," the AI MAGAZINE, pp.40-62.