Vol. 30 No. 11 情報処理学会論文誌 Nov. 1989

LISP-PAL: プログラミング支援のための自然言語による質問応答システム†

上原三八†† 山本里枝子†† 小川知也††

質問応答システム LISP-PAL は，初心者のプログラマを対象に，熟練者のプログラミング知識を会話的に提示することによってプログラミングを支援する．たとえば，ユーザがプログラミングしたいことを日本語で問合せすると，システムは使えそうな関数やノウハウに関して，説明とプログラム例を提示する．本研究は，知識ベースを利用したソフトウェア開発環境の構築と，知識ベースの利用技術の開発を目的とした研究の一貫として行われた．本論文で述べる研究成果は，(1)日本語文間の意味関係を階層的なモデルを使って表すことにより，関連知識を効率良く検索する方法，(2)フレームに基づいた知識表現をオブジェクト指向言語でパターンマッチングを使って容易に扱うための技術，である．システムを実際に運用するために必要な機能として，プログラミング知識や未登録語をユーザが会話的にシステムに登録できる機能を導入した．プログラミング知識をテキスト形式で登録し，問合せできるシステムをワークステーションとパソコン上に実現し，システムの評価を，約50名のモニタによって試用された際の質問応答内容のログとアンケートの分析により行った．知識ベースは Common Lisp プログラミングに関する約 100 のノウハウ，600 の関数，100 の用語の知識を持つ．

1. はじめに

プログラミング技術の習得過程を考えてみると，教科書で一通り基本事項を学んだ後，参考となるプログラム例から多くの知識を習得していると思われる．さらに進んでプログラミングを行う際は，“こういった場合にどうすれば良いか？”といった知識を，参考となるプログラム例を探したり経験の多い人に尋ねて学んでいる．このような参考となるプログラム例や経験の多い人が身近に存在するかがプログラミング技術修得の鍵であるが，現状は必ずしもそうではない．

Programmer's Apprentice[1] はクリシェイと呼ばれるプログラミング知識に基づいて，プログラマの作業を支援する．また，LispTutor[2] はプログラミング知識を IF-THEN 形式でモデル化し，正しいモデルと誤りモデルを用いて初心者の書いたプログラムのエラーについて説明する教育システムである．これらのシステムの知識ベースを作るには複雑なプログラミング知識を定められた形式で抽出しなければならない．この作業は比較的明確であるプログラミング言語自身に関する知識以外，たとえばどうプログラミングすべきかという設計知識を扱おうとすると困難になってしまう．

そこで我々は，多種多様なプログラミング知識の知識表現化を行うより，先に述べたような実際にプログラマが上達する過程に着目し，上達に必要なプログラム例や熟練プログラマの持つ知識を必要に応じて文章の形で説明・提示できる環境を作成することが現実的で重要であると考えた．さらに，そのような環境にはユーザが考えていることを容易に表現できるという理由で自然言語インタフェースが適切と考えた．メニュー主体のインタフェースでは，何百もの見出し項目を扱う場合には不適当だからである．

このようなシステムの主な課題を挙げる．

①表現上異なる問合せを，同じであると認識する技術．たとえば，「関数定義の仕方は？」と「関数を定義したい」を同じゴール（目的）を持つ文と認識する技術．

②関連知識を検索する技術．特に，ユーザが問合せしたい内容を具体的に表現できない場合に対処する技術として，関連性のある知識を内部的に表現し，処理する技術．

③実際にシステムを利用する上で必要な機能の実現．たとえば，エンド・ユーザが知識ベースを容易に構築できるようなインタフェース．従来，見受けられるような，知識を知識記述言語でプログラミングすることは実際的なシステムを構築する上で障害となる．

本論文では2章でシステムの機能と構造の概要を述べた後，3章で課題②に対応する知識ベースの知識表現と高速に関連知識を検索する方法について説明する．4章では①と③に対応して，自然言語インタフェースの実現と，システムの実際的な利用に有効な機

† LISP-PAL: Natural Language Consultation in a Programming Environment by SANYA UEHARA, RIEKO YAMAMOTO and TOMOYA OGAWA (Software Laboratory, FUJITSU LABORATORIES LTD.).

†† (株)富士通研究所ソフトウェア研究部

能について述べる．5章ではモニタの試用をもとに行ったシステム評価について，6章でまとめを述べる．

2. システム概要

知識を扱うシステムでは，何よりも知識そのものが重要である．そこで，初級〜中級レベルの LISP プログラマが知っているべきプログラミング知識に対象を絞って，知識を収集した．知識源として，著名な LISP プログラミングの教科書数冊の内容を分析し，それらが教えようとしているプログラミング技術を抽出した．同時に，熟練プログラマからインタビューやアンケート形式によって収集した．これら知識収集作業においては，LispTutor での知識表現の基本である，“あるゴールを達成するにはどんな方法をとるか”の形式で行うと効率良く行えることがわかった．特に教科書は自由なスタイルで書かれているので，この形式で抽出することが“使える知識”を収集する上で重要である．

本論文でのプログラミングノウハウとは，あるゴール（目的）を達成するための方法であり，その内容は，プログラム例やプログラム・スキーマの説明と，プログラミングする上での注意事項から構成されている．また関数知識とは，あるゴールを達成するためにどの関数を使い，どうコーディングするかといった知識，用語知識とは，“リスト”，“インターン”などの用語の意味と使い方に関する知識である．以下に，これらの知識に基づいて実現したシステムの機能と構造について述べる．

2.1 システム利用例

図1に質問応答例を示す．質問「リストの全ての要素を処理するには？」はゴールを実現するノウハウの問合せで，システムは複数の知識が見つかったのでそれらのタイトルを表示してユーザに選ばせている．あるゴールをどう実現するかを問合せする質問を我々は“機能質問”と呼んでいる．ノウハウに関する機能質問に対して提示される内容はプログラム例やプログラム・スキーマと説明，関数に関する機能質問に対しては関数名である．もしユーザの問合せ文中に未知語が存在すると，システムは未知語登録を促す．この場で登録された未知語はユーザ個人の辞書に登録されるので他人へ影響することはない．共通辞書には管理者によりチェックされた後で登録される．この機能によって，すべてのユーザはいつでも自分で使いたい語彙で質問でき，そしてシステムに不足している語彙をユーザが使うことによって収集できる．ここでの特徴は同義語を一括して登録する方法である．その中に既知語が1つでもあれば，既知語と同じ意味を持つ語として登録し，すべて未知語であれば品詞や用法を尋ねて，新しい語として定義する．いずれの場合でも複数の語を容易に登録するこ

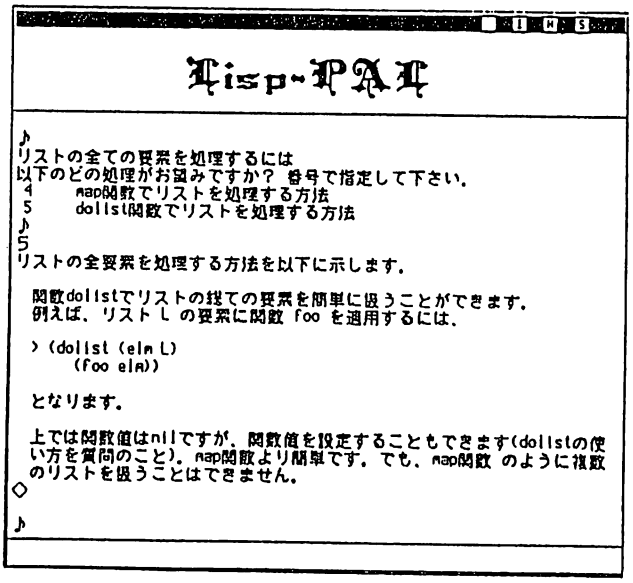


図 1 質問応答例
Fig. 1 Question-answer example.

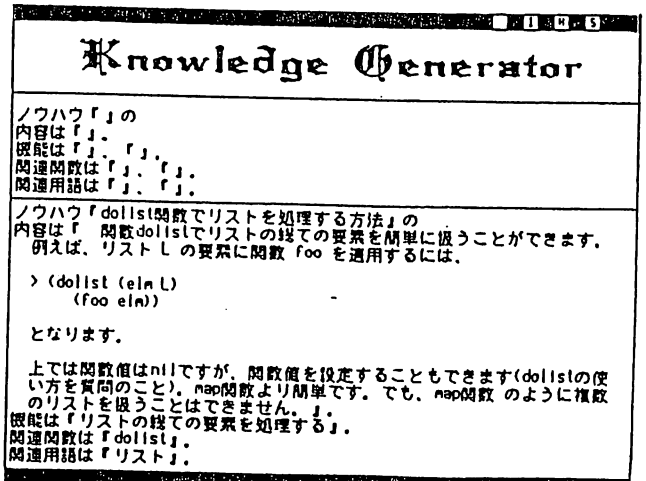


図 2 知識登録例
Fig. 2 Knowledge registration example.

とが可能である．

従来，知識ベースの作成と変更には高度な知識が必要とされた．しかし，これは実際面で大きな障害であり，知識を持つ専門家自身が容易に作業できるインタフェースが必要であることがわかってきた．図2は，熟練プログラマが知識を追加登録するインタフェースである．システムへ“知識追加（します）”と入力すると，図のようなウィンドウが生成される．ウィンドウには3種類の知識（ノウハウ，関数，用語）を定義するためのテンプレートが表示される．熟練プログラマは必要なテンプレートを入力行にコピーし，その中に必要事項を記入することで知識を登録する．必要事項のうち，「機能」は自然言語解釈され，3章で述べるような検索のための処理がなされる．「関連関数」や「関連用語」は，それぞれの関数や用語へのリンクが張られ，その他は入力された文章そのものの形で蓄積される．追加された知識はいったん個人用の知識ベースに登録され，管理者が内容をチェックした後に共通知識ベースへ登録される．

2.2 システム構成

図3に示されるように，システムはトランスレータと質問応答モジュールの2つの主要部分からなり，それらの間のインタフェースとして SEM (semantic representation) と呼ぶ知識表現形式を用いる．ユーザの問合せはすべて SEM に変換される．SEM の構文はフレーム表現に基づいて，その意味はプログラミング質問応答で必要な意味を表現できるように設計した．SEM については3.2節で，トランスレータの詳細は4.1節で述べる．

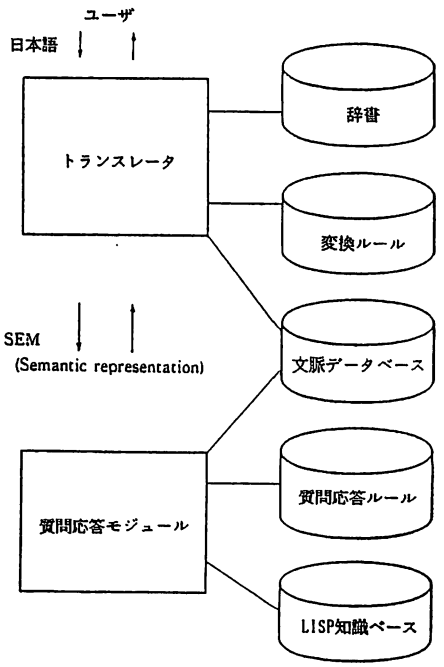


図3 システム構造
Fig. 3 System structure.

質問応答モジュールは受け取った SEM に対して，質問応答ルールを適用し，LISP 知識ベースから該当する知識を探し出して，応答を SEM 形式で作り上げる．トランスレータは，SEM から日本語の応答文を作り出す．知識の内容それ自体は，日本語の文章なので，ここでは応答文としてふさわしい体裁の文章を生成する．さらに，代名詞や省略語といった処理を行うために，質問応答の履歴を文脈データベースに SEM の形で蓄えて利用する．

3. 知識ベース

システム実現上の課題のひとつは，機能を言及する文（節）の意味関係をいかに表現し，かつそれをユーザの問合せに柔軟に活用するかである．たとえば，ユーザは連想リストを作る際に，リスト構造を目前にしているために「リストを作るには」と問合せしてしまい，本来のゴールを十分に言い尽くせないことが予想される．そのような場合，ユーザの問合せが意味的に含むようなゴールを実現する方法，すなわち連想リストを作る知識についても教える必要がある．本章では，これを実現するための知識表現と実現技術について述べる．

3.1 知識表現

日本語文の間に，表層的な，単語レベルで決定できるような意味的に含むか否か，という包含関係を導入する．この関係は IS-A 関係に類似している．これは，上例の“「リストを作る」は「連想リストを作る」を意味的に含む”を導くものである．この関係は次のような単純なルールで定義する．すなわち，文を構成する名詞と動詞の包含関係，そして前置詞や修飾句の有無による詳細度によりどの文が意味的に“含む”か“含まれる”か“無関係”かを定める．たとえば，名詞「コンピュータ」は「マイクロコンピュータ」を“意味的に含む”という単語間の包含関係を前提として与えることにより，「彼はコンピュータを手に入れる」は「彼はマイクロコンピュータを手に入れる」を“意味的に含む”と定める．包含関係を定める別の方法の例は，「日本語を翻訳する」は「日本語を英語に

翻訳する」を意味的に含む，である．後者は「英語に」が付け加わり，より詳細に意味を表しているからである．同様に，「翻訳する」は修飾語が付け加わった「早く翻訳する」を意味的に含む．以上のような包含関係は，一般に日本語文の意味を扱う上で厳密なものではないが，プログラミングに関する質問応答において，比較的単純な日本語の言い回しを扱う上で有効なものである．

包含関係は，フレーム構造と，スロットの値の両者により表現される．LISP 知識ベース中では，関数フレームは，関数名，構文，機能，使用例などのスロットからなる．このうち，機能スロットは，動詞フレームへのリンクを保持している．図4は関数 VECTOR の機能がゴール“ベクタを作る”を表現するために動詞“作る”をリンクしている様子を示している．ノウハウフレームの機能スロットも同様な構造をとる．

LISP 知識ベース中には，関数，ノウハウ，用語，スキル，動詞，修飾詞の各フレームが存在する．関数と用語とノウハウの各フレームのスロットには，プログラミング知識が文章の形で保持される．ゴールは動詞，名詞，修飾詞のフレームから構成される．図4は「ベクタを作る」が「ベクタ」と「作る」のフレームで表現される様子を示す．さらに，「配列を作る」は「ベクタを作る」を含んでいると表現されている．これは，用語のフレーム関係で，ベクタは配列に意味的に含まれると定義されているためである（Common Lisp ではベクタは一次元の配列であると定義されている[3]）．同様に，動詞「処理する」と「作る」との関係から「ベクタを処理する」は「ベクタを作る」を意味的に含む，という関係が作られる．

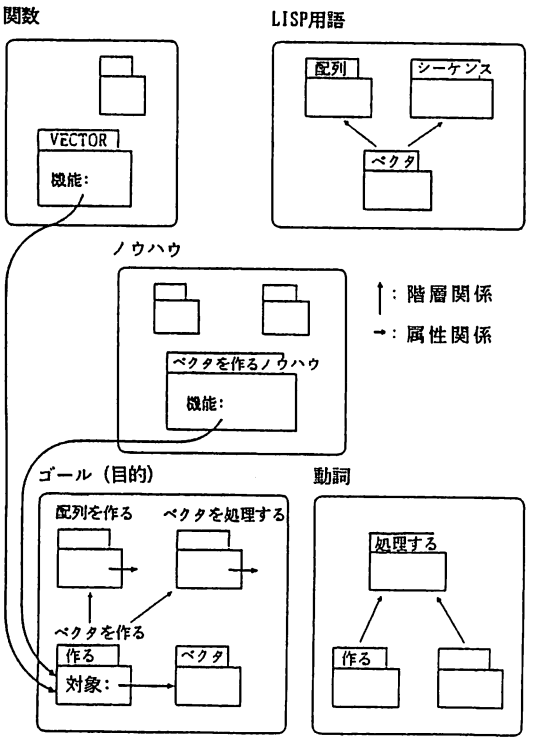


図4 LISP 知識ベースにおける“機能”のフレーム表現
Fig. 4 Frame representation of “function” in the LISP knowledge base.

ユーザが「配列を作るには」と問合せした場合には，システムは，問合せ文と同じか，あるいは意味的に含まれるゴールを探し，「ベクタを作る」知識（ノウハウと関数）を含めて応答する．また，対応する知識が見つからない場合には，階層の上方を探し，関係する可能性のある知識として提示する（経験的には，プログラム例中に，参考になる点が見つかることがある）．高速に検索を行うために，関数とノウハウフレームとゴールフレーム（動詞のインスタンス）間には双方向のリンクが，またゴールフレームの上位・下位間にも双方向リンクが張られる．

知識ベースとシステム本体は，LISP をベースにしたオブジェクト指向言語で実現した．個々のフレームはオブジェクトとして実現され，意味的な関係を示すリンクや，フレーム間相互の高速検索用リンクはそれぞれ特定のスロットに割り当てられる．

3.2 質問応答の実現

LISP 知識ベースは，図4中のリンク（矢印）で表される属性で関係付けされたオブジェクトで実現されている．これらのオブジェクトの操作をプログラミング言語で記述するためには，オブジェクトの属性や上位・下位を参照するプリミティブなオペレーションを使わなければならないが，オブジェクト間の関係が複雑になるに従いプログラムが煩雑になってしまう，という問題がある．一般に，プログラミング言語で属性と上位・下位で関係付けられたオブジェクトをひとまとまりの知識として扱おうとすると，記述が複雑になってしまう．

我々は意味のあるひとまとまりの複雑なオブジェクトをプログラム中で簡潔に，かつわかりやすく表現するために，フレーム表現に基づいたパターンでそれらを記述するよう工夫した[4]．我々はこのオブジェクトの構造を表現する記法を SEM と名付けた．

図5の3つの楕円内に，“リストAのある条件を満た

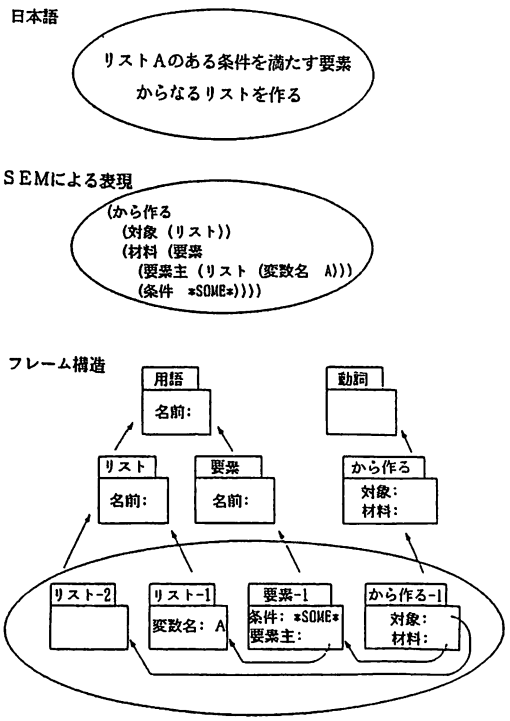


図 5 LISP-PAL における日本語の表現形式
Fig. 5 Representation for Japanese sentences in LISP-PAL.

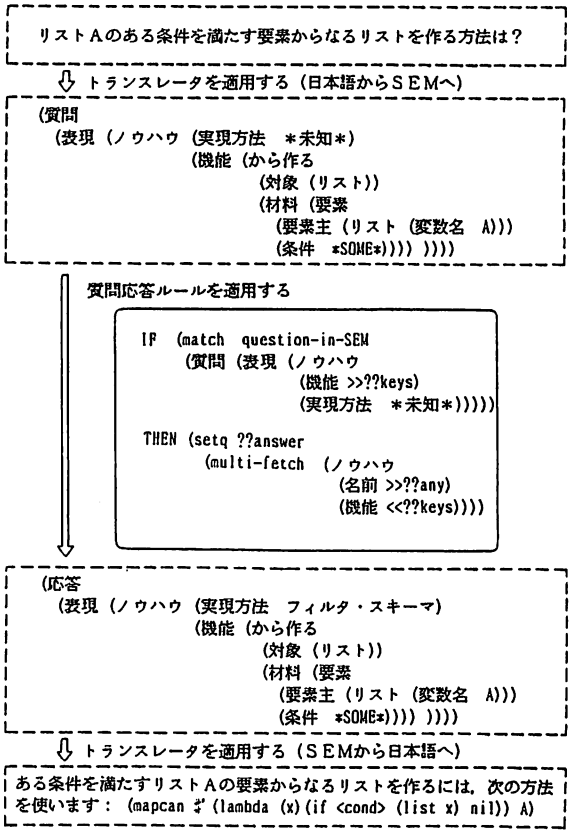


図 6 質問応答過程における SEM 表現
Fig. 6 SEM representation in the question-answer process.

す要素からなるリストを作る." の日本語表現，SEM による表現，そして基底となるフレーム構造を示す．この例のフレーム構造はやや複雑である．そこでプログラム中で意味のあるひとまとまりの複数のオブジェクトを扱う際には常に SEM で記述する．プログラムが実行される際には，SEM による記述はオブジェクト構造へと解釈・展開される．

図 6 は質問応答の処理内容を示す．トランスレータは質問文を SEM に変換する．次に，パターンマッチングを用いて，SEM に対応する質問応答ルールを発火させ，知識検索を行う．すなわち，質問応答ルールの THEN 部では，SEM 上のオペレーションである multi-fetch を用いて知識ベースから必要なノウハウ (skill) を探し出し SEM 形式で応答を作り出す．ここで，"??" で始まる変数はオブジェクトの集合を保持するものであることを示す．たとえば，THEN 部の ?? answer には知識ベースから検索された複数のノウハウに対応するオブジェクトが代入される．また，"≫" はオブジェクトの集合を直後に記述される変数に代入することを，"≪" はオブジェクトの集合を直後に記述される変数から参照することを示す．たとえば，IF 部では ≫?? keys とあるのでパターンマッチングによって，質問文の表現のうちノウハウの機能部分が変数 ?? keys に代入される．ここでもし，≫?? keys でなく ≪?? keys ならば変数 ?? keys の値が参照されて質問文の機能部分と等しいかのテストが行われる．これらの記法はオブジェクトの集合を簡潔に扱うことを可能とし，通常のオブジェクト指向言語でオブジェクトの個々を扱わなければならない煩わしさをなくしている．図 6 中央の IF-THEN プログラムの利点は，意味のあるひとまとまりの複数のオブジェクトがパターンで簡潔に記述されていることである．実行時にこのパターンを解釈しながら知識ベース中のオブジェクトが検索される．図 6 下では，SEM 形式の応答が日本語生成ルールにより日本語文に変換される様子を示している．

3.3 知識生成と高速検索の実現

いかに容易に知識を知識ベースに登録できるかは，

知識ベースの保守性の面から極めて重要である．3.1節に述べたように，本システムの知識ベースはオブジェクトが複雑に構造化されているため，プログラミング言語で直接コーディングやデバッグを行うのはほとんど不可能である．熟練プログラマ自身が知識登録を行えるためにも，また，知識の内容が日本語の文章であることから，知識は日本語で登録できることが望ましい．

図7は関数 DOLIST の知識が登録される様子を表している．関数 DOLIST の知識は関数定義テンプレートを用いて行われる．トランスレータは「機能(キー)」に記入された文「リストの総ての要素を処理する」を解析し SEM を作り出す．これをもとに3.1節で述べたゴールを表現する階層中のどこに配置するかを定める．もし同じ表現が既に存在すればそれを使用し，そうでなければ新規に階層中に作り出す．後者の場合，そのゴールを意味的に含む表現（図7では，「リストを処理する」）と意味的に含まれる機能表現（図7では「リストを変更する」）へのリンクを張る．

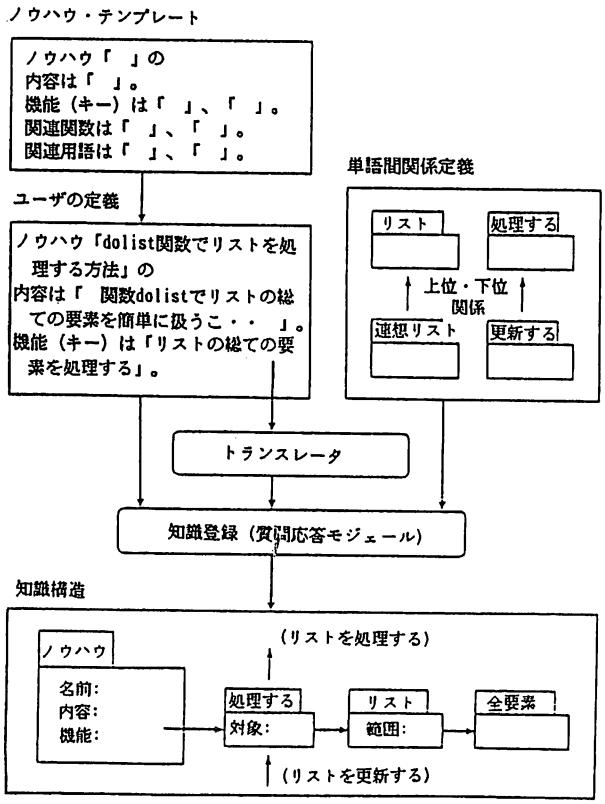


図7 テンプレートによる知識登録
Fig. 7 Defining knowledge with a template.

ゴールフレームと関数知識のフレームとは双方向にリンクされる．

知識を検索する際には，定義する際と同様に階層中の適切な位置が探される．この時，まず質問と同じ動詞を用いるゴール表現をハッシングによって求めて，検索範囲を絞ってから行う．（ハッシング自体は LISP 処理系により実現される．）全く同じゴールが階層中に見つかった場合，その機能と，そのゴールの下方の（意味的に含む）ゴールが，検索対象となる．すなわち，それらのゴールからリンクする関数やノウハウが求める知識となる．しかし，質問の都度，ゴールの下方をトラバースして求めるのは，時間がかかるので，あらかじめすべてのゴールに対して，下方のゴールをすべて計算しておく方法をとっている．これにより，知識数とはほぼ無関係な応答時間が得られる．質問のゴールと一致するものが知識ベース中にない場合は，一致する機能表現を仮想的に階層中に作り，上記と同様な処理を行う．

4. 実用的システムとしての機能

本システムでは，利用しやすさと運用しやすさの面から必要な機能を導入した．2.1節で述べた知識追加や単語登録はそのような機能に含まれる．また，自然言語インタフェースはユーザが手軽に使える反面，どう質問して良いか，あるいは言い回しが正しく解析されないことがある，といった欠点を持つ．以下に，自然言語インタフェースとシステムをより実用的なものとする機能について述べる．

4.1 自然言語インタフェース

プログラミングに関する質問応答では，たとえば，文学作品のような細かいニュアンスを把握することは不要であることから，本システムで対象とする文型は単純なものに限定すべきだと考えた．しかし，問合せ中に変数名や式が（たとえば，「変数Aを1増やす」の中の変数名「A」と式「1」），さらに現実的には未知語を扱うことが必須である．また，同義語はもちろんのこと，語順の変化，言い回しの違い（例，「～を教えて下さい」と「～とは」，「関数定義の方法」と「関数で定義するには」）を吸収する必要がある．また，MMI の立場から，解析が高速に処理される必要もある．

以上により，①言い回しの自由度の吸収，②未

知語の柔軟な扱い，③効率が良い，という要求を満たすように，ELI[5]で示された技術を発展させて，文法と単語定義の方法および解析方法を考案し実現した．また，本設計を行う上で，データベース自然言語問合せインタフェース[6]における世界モデルの利用技術を参考にした．設計思想については文献 7)で述べた．

我々のアプローチは，単語間の意味／関係付け情報を使い，ヒューリスティックを用いた手続きを動作させることで解析を行う．単語の関係付け情報とは，次の 4 つである．

（1） 単語がどのような属性を持つか
（2） どの格と関係付け可能か
（3） どのようなクラスの単語と関係付け可能か
（4） 語順と単語間の距離

この方法では格の脱落や語順の変化を解析法の中で吸収する．文法は品詞に相当する単語のクラス定義が中心となる．クラス定義はフレームにより階層的に行われ，単語属性を示すスロットには係受け的な関係付けの可能性が格やクラスで限定される．属性，格，クラスはフレーム形式で定義されるので追加や変更が容易である．クラスを使うと，分野の専門語の結び付けを制限することも可能となる．

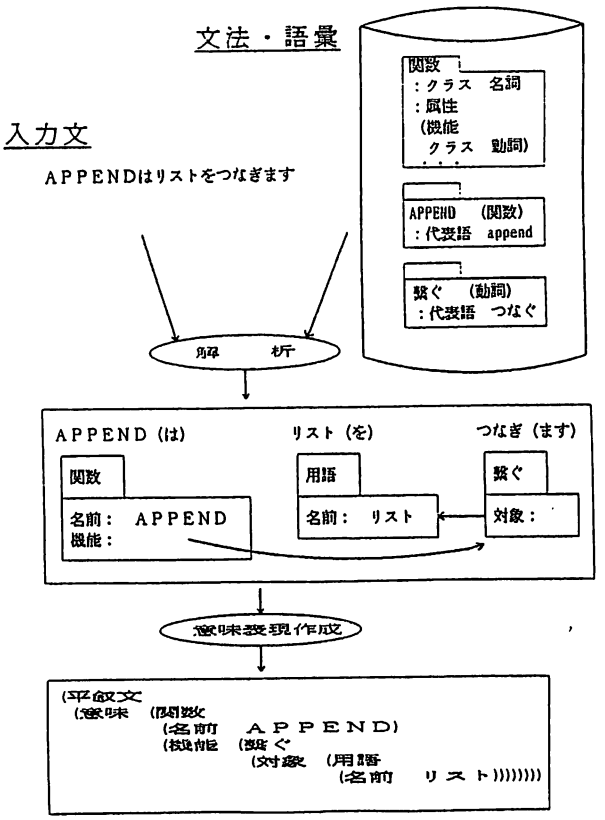


図 8 自然言語処理
Fig. 8 Natural language processing.

図 8 に「APPEND はリストをつなぎます」の例を使って，文法と単語定義，解析手順の概要を示す．日本語インタフェースは 2 つのフェーズからなっている．第一フェーズは単語間の関係を決め，フレーム構造を作り出す．第二フェーズはフレーム構造のパターンから，文種や文の中心概念を決め，意味表現を作り出す．たとえば，「パッケージとは何」という文は疑問文で意味は説明を知りたい，と解析してそのような表現，

```
（意味（用語
        （名前 パッケージ）
        （説明 *未知*）））
```

へ変換する．第一フェーズは汎用な日本語インタフェースとしてツール化できる部分と考えられる．

4.2 知識集

知識ベース中の知識をテーマ別に見たいユーザや，どう質問したら良いかが全くわからないユーザのために，教科書のように目次に従って見せる機能を実現した．この特徴は，目次がデータとして自由に書き換えができる点にある．システムは，ユーザの指定によって選ばれた項目，たとえば，「属性リスト」や「デバッグを効率良く行うには」という名詞やゴールをもとに知識ベース検索して得られた知識を提示する．したがって，目次を変更するだけで，知識ベースの見せ方を自由に変えることができる上に，知識ベースに新規に登録された知識も何ら手を加えることなく提示させることができる．

4.3 ヘルプ機能

質問文が解釈できない（質問応答ルールにマッチしない）場合，システムは極力言い回しがおかしな点や未知語を示して再入力を促す．それでも解釈できない文が入力されると，システムが受付け可能な代表的な言い回しの例をユーザに提示する．また，ユーザがどう表現してよいか迷って「何が聞けるの」などと問合せすると，知識ベースにある内容を集約して見せる．これは，質問文が何回か続けて解釈できなかった時にもヘルプメッセージとして提示される．

また，質問文のうち答えられなかったものはログフ

ァイルに記録され，知識ベース管理者が後で知ることができる．また，ユーザが使用中に考えた意見（たとえば，知識の内容に関するもの）やバグ情報をログに記録する方法も取り入れた．これは，問合せ文の先頭に「；」記号を置くことによって行われる．従来では，要望・障害レポートを記入するという面倒な方法がとられていた．このようなログ情報を用いることで，システムの改良を効率良く行うことができる．

5. システム評価

システム実現の妥当性とプログラミング支援の有効性の評価を目的とした実験を行った．システムの実現の妥当性の評価は，入門レベルの LISP プログラミング知識を有する約 40 名の被験者を使って行った．システムの使い方と機能についてあらかじめ説明した後，(1)LISP でプログラミングするために必要と思われる事柄について自由に質問する，(2)システムの機能を一通り使ってみる，の2点を条件として1人当たり約 30 分～1時間システムを使用した後，付録1のアンケートに答えてもらった．評価はアンケートと約 2,000 の質問応答文を含むログを分析して行った．

プログラミング支援の有効性の評価は，初級レベルの8人の LISP プログラミング経験者を使って行った．3つの小課題（付録2参照）を与え，課題を解きながら，(1)問題を解く上で必要と思われることを自由に質問する，(2)質問することは強制でなく，質問したいことだけを質問する，を条件として，システムを使用してもらった．課題終了後，質問応答文のログと回答を分析して評価を行った．課題は，LISP プログラミングとして標準的なものを選ぶことにより，問題の内容により利用結果がなるべく片寄らないようにした．また重要なこととして，知識ベースには問題を解く上で必要となりそうな知識を実験の事前に追加しなかった．その理由は，もしそうすると知識ベースや辞書の充実度について正しい評価が行えないからである．なおこれらの評価を通じて，今後の知識ベースと辞書の充実の計画を立てるために内容別（データ構造，関数・マクロ，アルゴリズム，入出力，設計法，デバッグ法など）に，質問の片寄り，回答率，知識ベース中の知識数の統計をとった．

5.1 システム妥当性の評価

システムの有益性に対するアンケートの集計を図 9 に示す．モニタの 60% が「自分にとり有益なシステムである」と答え，15% が「まだ完成度が低い」，その他 25% が「何とも言えない」や「他の支援形態が必要」と答えた．「システムを使ってみたいか？」のアンケートに約 70% ものモニタが肯定的であったことからも，システムのニーズと実現の妥当性が認められた，と考えられる．一方，モニタの否定的な意見は，質問に対するシステムの回答率が十分でないことに起因していると思われる．図 10 に回答率と回答できなかった原因を示す．図中の知識を提示できた中には，「属性リストの使用例」のようにほとんど必ず答えられる質問が含まれており，本システムのねらいである機能質問のみの回答率は約 40% であった．5.2 節で述べるように質問の中には LISP プログラミングの中級程度までの範疇に含まれない質問も多いため，知識だけを考慮しても 100% の回答率は望めないが，しかしより高い回答率への改善の必要性は高いと判断される．

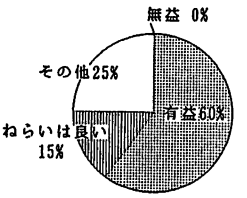


図 9 システムの有効性（アンケート）
Fig. 9 System usefulness (questionnaire).

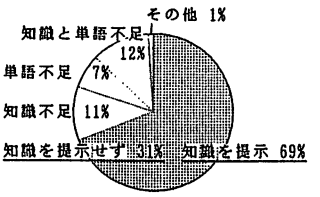


図 10 質問応答の回答率と失敗原因（ログ分析）
Fig. 10 Success and failure analysis in question-answer (log analysis).

ゴールの階層表現が機能質問に有効であることがログより確認された．機能質問でシステムが知識提示できた場合のほぼ全文が，機能の階層を使って応答知識を増やしており，40%は階層を利用しないと応答できないものであった．たとえば，質問「コンパイルする方法は」に対して，「関数 compile-file と compile」と応答した．知識ベースの定義は，compile-file の機能は「ファイルをコンパイルする」，compile の機能は「ラムダ式をコンパイルする」であり，階層を利用して初めて答えることができた．

日本語インタフェースの解析率については，約半数が「改善すべきである」と答えたが，残り半数の人は「ほぼ妥当」と答えた．インタビューにより，後者を答

表 1 文型と回答率（ログ分析）
Table 1 Sentence pattern and correctly answered ratio (log analysis).

	文型	例文	質問の割合	知識提示率
1	〈名詞〉とは (?)	リストとは nconc ?	19%	85%
2	〈名詞〉の〈名詞〉は(?)	文字列の例は append の構文?	21	86
3	～には (?)	文字列を扱うには 関数の定義方法は	29	49
4	～関数は (?)	リストを作る関数 文字列処理の関数	3	25
5	その他の日本語文	read のとる引数は さよなら	6	5
6	メニュー番号	1 2	21	—
7	コマンド	HELP 何が聞けるの	1	—

・文型 3 と 4 は機能質問（名詞，動詞等からなる）．
・文型 1～4 の名詞と動詞には，修飾詞が許される．

えたモニタは，日本語インタフェースをプログラミング支援において高度にしても際限がなく，むしろユーザが日本語の解釈のし方を想像できる程度にとどめた方が安心して使える，と考えていることがわかった．

知識の内容に対するアンケートの集計によると，質・量とも「おおむね適当」であった．量については，機能質問で答えられるように補うこと，また質については，説明やプログラム例の解説をより丁寧にわかりやすくするという要望が主であった．

5.2 プログラミング支援の評価

プログラミング課題を解きながら，モニタがシステムに対して行った質問応答の文型と知識の提示率を表 1 に示す．モニタは機能質問で使えそうな関数を見つけた後，その使用例を見たり，属性リストなどのデータ構造の使い方を見てどう使うかを参考にする，といった使い方をした．表 1 で，「～関数は」の質問が少ないのは，「～には」の機能質問でもノウハウと一緒に関数が提示されるために後者の文型が使われたことによる．

モニタの質問をログより分析すると，機能質問で課題に近い内容を質問していることがわかった．もちろん，知識ベースにはそのような知識はないため，システムは「知識は有りません」と応答する．質問がプログラミング言語に近いレベルになると，プログラム例が提示され始め，モニタは参考になるかを判断する．ここでプログラム例を利用できるかは，モニタのプログラミング経験に大きく依存することがわかった．ログと最終的に作られたプログラム例をつき合わせて，プログラミング過程でどのようなデシジョンを行う際に役立ったかを分析した結果を図 11 に示す．提示された知識のうち，約 60% がプログラミング作業に役立っていると推定された．その内容は

どの関数を使うか
関数をどう使うか
データ構造をどう扱うか（作成，変更，削除など）

であり，使えるプログラム例やプログラム・スキーマが見つかると作業が大幅に軽減される例が観察された．

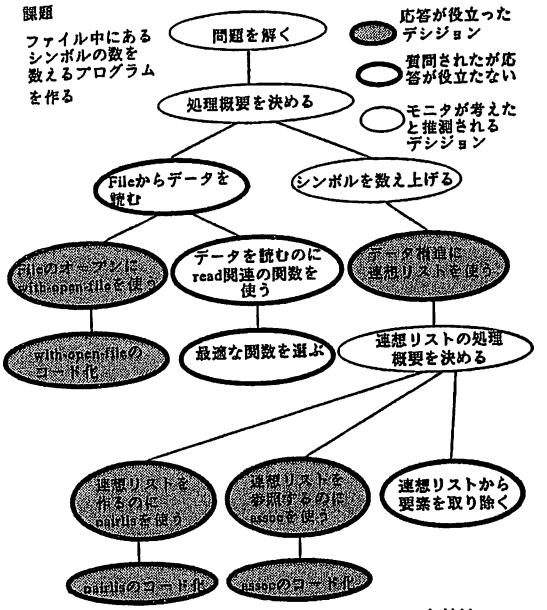


図 11 デシジョン分析によるシステムの有効性
Fig. 11 Usefulness analysis in programming decision.

6. まとめ

本論文で，自然言語インタフェースにより，LISP プログラミング知識を教えるシステムを述べた．特に，フレームに基づいて，日本語の意味を階層的に表現するための知識表現法と，それを用いて関連するプログラミング知識の高速検索を行う方法を述べた．また，知識追加と単語追加の運用上の必要性と実現例を示した．

システムを実現する技術として，属性関係により関係付けられた複数のオブジェクトの集合をパターンマッチングによって扱う方法を考案した．これらは，プログラムの記述性と理解性を高める．たとえば，質

問応答戦略の変更・拡張は極めて容易であった．この実現技術は，Unix Consultant[8] における技術をオブジェクト指向パラダイムに適用したものである．

システムは LISP プログラミング環境に統合されており，質問応答の結果を見ながら，LISP プログラミングが行える．現在，知識ベースは約 100 のノウハウ，600 の LISP 関数，100 の LISP 用語についてのプログラミング知識を持ち，UNIX ワークステーション，および FM/R シリーズパソコンで稼働している．応答時間は質問の種類で異なるが，2〜20 秒である．

モニタを使ったシステム評価によって，アプローチの有効性やシステムの有効性が確かめられた．応答できなかった質問はログ情報として残されるので，知識ベースや辞書の改良に利用できる．今後の主要な課題の1つは，知識に対する適切な日本語文の機能検索キーをどう見つけるかである．この重要性は，応答できない機能質問を分析した結果，知識・単語不足の原因を除いた主因が機能検索キーの不足または不適切であったことが裏付けている．

謝辞 日頃，御討論いただく(株)富士通研究所の諸兄，評価にモニタとして参加していただいた社内の諸氏，ならびに実際的な利用面に関して有益な討論をしていただいた富士通株式会社システム本部 SE テクニカルセンターの村上憲稔課長，藤井則夫，銀林純の諸氏に深く感謝いたします．

参考文献

1) Waters, R. C.: The Programmer's Apprentice: A Session with KBEmacs, *IEEE Trans. Softw. Eng.*, Vol. 11, No. 11, pp. 1296-1320 (1985).
2) Anderson, J. R. et al.: The Automated Tutoring of Introductory Computer Programming, *Comm. ACM*, Vol. 29, No. 9, pp. 842-849 (1986).
3) Steele Jr., G. L.: *Common LISP: The Language*, p. 465, Digital Press (1984).
4) Uehara, S. et al.: Implementing Programming Consultation System LISP-PAL: Framework for Handling Frame-Based Knowledge in Object-Oriented Paradigm, *CIIAM 86 Second Int. AI Conf.*, pp. 409-423 (1986).
5) Schank, R. C. et al.: *Inside Computer Understanding*, Lawrence Erlbaum Associates (1981).
6) Ishikawa, H. et al.: Designing a Knowledge-Based Natural Language Interface, *IEEE Expert*, Vol. 2, No. 2, pp. 56-71 (1987).
7) 上原ほか：LISP プログラミング知識に基づく質問応答システム，日本ソフトウェア科学会第4回大会論文集，pp. 367-370 (1987).
8) Chin, D. N.: Case Study of Knowledge Representation in UC, *IJCAI*, pp. 388-390 (1983).

付録 1 アンケート

(次 頁)

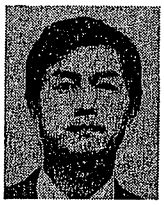
付録 2 プログラミング課題

課題1 「配列（ベクタ）の各要素の和を求める関数を定義しなさい．」

課題2 「リスト中の隣り合う，同じ(eqlで等しい)要素を1つにまとめる関数を定義しなさい．」

課題3 「ファイル中にあるシンボルの数を数える関数を定義しなさい．」

(昭和64年1月5日受付)
(平成元年9月12日採録)

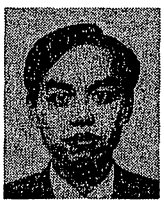


上原 三八（正会員）

1977年東京工業大学情報科学科卒業．1978年より1年間，ワシントン大学計算機科学科大学院留学．1980年東京工業大学大学院情報科学修士課程修了．同年(株)富士通研究所入社．以来，ソフトウェア工学の研究に従事．現在，情報処理研究部門ソフトウェア研究部第三研究室に所属．ソフトウェア工学への人工知能の応用，プログラミング方法論，オブジェクト指向ソフトウェア開発，システムモデリングなどに興味を持つ．ACM，IEEE 各会員．

山本里枝子（正会員）

1960年生．1983年早稲田大学理工学部電子通信学科卒業．同年(株)富士通研究所入社．以来，ソフトウェア工学，人工知能，視覚的ソフトウェア開発環境の研究に従事．



小川 知也（正会員）

1960年生．1983年東京工業大学制御工学科卒業．1985年同大学院システム科学科修士課程修了．同年(株)富士通研究所入社．現在，ソフト工学，人工知能，知的 CAI の研究に従事．

アンケート

ＬＩＳＰ―ＰＡＬ評価票

御名前＿＿＿＿＿＿＿＿＿＿　所属＿＿＿＿＿＿＿＿＿＿　㊞＿＿＿＿

モニター日＿＿＿＿年＿＿＿月＿＿＿日

1. ①あなたの，Ｌｉｓｐプログラミングの経験をお聞かせ下さい．
   - □なし
   - □教科書などを読んだことがある
   - □簡単なプログラム作成
   - □本格的に使った　（使用経験　　年　月）
     （マシン：＿＿＿＿，言語：COMMONLISP　UTILISP　GCLISP　＿＿＿＿）

   ②他の言語でのプログラミング経験についてお聞かせ下さい。
   - □なし
   - □簡単なプログラム作成　（言語＿＿＿＿＿＿＿＿）
   - □本格的に使った　（言語＿＿＿＿＿＿＿＿）

2. システム全体に関して

   ①全体の印象として使い易いシステムでしょうか？　使い難い ├─┼─┼─┼─┤ 使い易い　（妥当）

   特に，どのような点が使い易い，または，使い難いでしょうか？

   ＿＿＿＿＿＿＿＿＿＿

   ②このシステムは，あなたにとって有益でしょうか？
   - □　有益である
   - □　無益である（理由＿＿＿＿＿＿＿＿）
   - □　システムの狙いは良いが，このシステムの完成度が低い
   - □　その他

   ＿＿＿＿＿＿＿＿＿＿

   ③このシステムは，＂○○○するには？＂という形式の質問応答を可能としています。
   この機能は有益でしょうか？　無益 ├─┼─┼─┼─┤ 有益
   コメント：

   ＿＿＿＿＿＿＿＿＿＿

   ④今後，このシステムを使ってみたいと思いますか？
   - □　はい
   - □　いいえ
   - □　改良されるならば使いたい（改良を望む点＿＿＿＿＿＿＿＿）
   - □　その他

   ＿＿＿＿＿＿＿＿＿＿

3. 日本語インタフェースに関して

   ①日本語はインタフェースとして有効だと思いますか？
   - □　それだけで有効である
   - □　それだけでは不充分である（＿＿＿＿＿＿＿＿を併用すれば良い）
   - □　コマンドやメニューで充分である
   - □　その他

   ＿＿＿＿＿＿＿＿＿＿

   ②このシステムの日本語の認識度をどのように思いましたか？　（妥当）
   不満である ├─┼─┼─┼─┤ 満足できる

   ③単語登録は，システムの誘導に沿って，同義語を羅列して行いますが，この単語登録はし易いと思いますか？
   - □　し易い
   - □　し難い（し難い点：＿＿＿＿＿＿＿＿）
   - □　使わなかった
   - □　その他

   ＿＿＿＿＿＿＿＿＿＿

4. 質問応答に関して

   ①質問応答の操作（キーの使い方，画面上での操作）はし易いと思いますか？
   し難い ├─┼─┼─┼─┤ し易い　（妥当）

   し難い点など：＿＿＿＿＿＿＿＿＿＿

   ②質問応答の応答速度は妥当だと思いますか？　遅い ├─┼─┼─┼─┤ 速い　（妥当）

5. 知識登録に関して（使用した方のみお答え下さい）

   ①このシステムで知識登録をするには，ＧＭＡＣＳのバッファを用意してから，知識登録用テンプレートを表示し，それを利用して行います。この知識登録は，し易いと思いますか？
   し難い ├─┼─┼─┼─┤ し易い　（妥当）

   し難い点など：＿＿＿＿＿＿＿＿＿＿

   ②知識登録の応答速度は妥当だと思いますか？　遅い ├─┼─┼─┼─┤ 速い　（妥当）

6. 知識集に関して（使用した方のみお答え下さい）

   システムに「知識集」と入力すると，メニューに沿って知識ベースの内容を提示する状態になります。

   ①このような機能は，あなたにとって有益だと思いますか？
   - □　有益である
   - □　無益である（理由＿＿＿＿＿＿＿＿）
   - □　この機能の狙いは良いが，完成度が低い
   - □　その他

   ＿＿＿＿＿＿＿＿＿＿

   ②この機能の操作はし易いと思いますか？
   - □　し易い
   - □　し難い（し難い点：＿＿＿＿＿＿＿＿）
   - □　その他

   ＿＿＿＿＿＿＿＿＿＿

7. 知識に関して

   ①このシステムの知識の内容のレベルは，適当でしょうか？
   - あなたにとって，　□　適当である　□　高い　□　低い
   - 初心者にとって，　□　適当である　□　高い　□　低い
   - □　その他

   ＿＿＿＿＿＿＿＿＿＿

   ②このシステムの知識の量は，適当でしょうか？
   - あなたにとって，　□　適当である　□　多い　□　少ない
   - 初心者にとって，　□　適当である　□　多い　□　少ない
   - □　その他

   ＿＿＿＿＿＿＿＿＿＿

8. その他，御意見，御希望がありましたら，御記入下さい。