論　文　　Technical Paper

ユーザモデルを利用した説明文生成プランニング

A Text Planning Mechanism Using Individualized User Models

垣内　隆志[*,*1]　槇本　英治[*]
Takashi Kakiuchi　Eiji Makimoto
上原　邦昭[*]　豊田　順一[*]
Kuniaki Uehara　Jun-ichi Toyoda

＊　大阪大学産業科学研究所
The Institute of Scientific and Industrial Research, Osaka University, Osaka 567, Japan.

1988年3月16日　受理

Keywords: text generation, user model, planning.

Summary

This paper describes an on-line help system, which is designed to produce multiparagragh explanations in response to user's questions about computer terminology. A major component of this system is a text planning mechanism, which tries to decide on what to say and how to say it for answering the user's question. In order for the system to have the ability to provide different responses in accordance with user's level of expertise, a user model is constructed throughout a session, and then utilized to vary response by cutting off an unimportant, redundant plan during the course of planning. Furthermore, a focusing mechanism is introduced to ensure that the generated explanation is coherent.

1. まえがき

最近，計算機を容易に利用できるようにするために，知的なインターフェイス機能を持つシステムの研究が盛んに行われている．このようなシステムの一つとしてマニュアルレスシステムがある．マニュアルレスシステムとは，ユーザがシステムを利用している際に生じる疑問に対して，自然言語で応答するシステムを総称したものである．

ユーザとの対話能力を重視するマニュアルレスシステムでは，ユーザの知識レベルに応じて応答を詳細にしたり簡潔にするといった動的な文章生成能力が要求される．また，この機能を実現する枠組として，ユーザの知識レベルに応じて「何をどのような順序で述べるか」を動的に決定するメカニズムが必要になる．さらに，理解容易な文章を生成するためには，一文からなる応答では表現能力が乏しく，文と文のつながりを重視した複数の文からなる文章を生成する能力が要求される．

以上のような考察に基づいて，我々はユーザが入力した質問に対して一貫した説明文を生成するマニュアルレスシステムASSISTを開発している．ASSISTでは，ユーザの知識レベルに応じた応答を実現するために，ユーザとの対話過程で得られる情報を利用して，ユーザの知識状態を動的にモデル化したユーザモデルを導入している．さらに，説明文生成過程を問題解決におけるプランニング過程とみなして，説明の内容やその順序を動的に決定する処理をプランニングによる統一的な枠組で実現している．このプランニング過程にユーザモデルの情報をフィードバックすることで，ユーザの知識レベルに合致した説明文が生成できるようになっている．

また，フレーム化された知識表現構造から，質問に応じて文章を再構成しているために，同一の知識構造から異なる文章を生成することが可能になっている．

＊1　現在，松下電器産業株式会社情報システム研究所

さらに，文章生成過程では，説明文中で現在話題の中心となっている事柄を明確にするために焦点の概念を導入して，話題の一貫性を考慮した文章を生成するようにしている．

2．説明文を生成するために必要な知識

2・1　質問の分類

ASSIST では，システムの対象領域として，計算機の利用方法や計算機用語についての質問に答えるように設計している．このような領域におけるユーザからの質問の型として，以下の5種類を取り扱っている．

What-is 型　:「アセンブラ言語とは何ですか？」
Difference 型 :「アセンブラ言語と機械語の違いは何ですか？」
Similarity 型 :「アセンブラ言語と機械語の共通点は何ですか？」
Relation 型　:「アセンブラ言語と機械語の関係は何ですか？」
How 型　　:「ファイルを消去するためにはどうすればいいですか？」

本論文では，このうち What-is 型，Difference 型の質問についてのみ考察する．なお，Similarity 型および Relation 型の質問についての応答は Difference 型や What-is 型の質問応答メカニズムを利用して実現できる．また，How 型の質問に答えるには，計算機の動作をシミュレートしたり，動作の因果関係を理由づけて提示するなど，本論文で提案する以外のメカニズムが必要になるため，別稿で改めて議論する[7]．

2・2　知識表現形式

上記の質問から説明文を生成するためには，予めシステムが対象領域に関する知識（すなわち，計算機用語に関する知識）を計算機内部で利用できるような形式で蓄えておく必要がある．さらに，その知識表現を利用して文を生成する場合，少なくとも2通りの文体が表現できなければならない．一つは「～である」という文体（be 動詞文と呼ぶ）で，他の一つは「～する」という文体（do 動詞文と呼ぶ）である．be 動詞文は概念の形態，性質，上位下位関係などを言及する場合に用いられる．

《例》　アセンブラ言語はプログラミング言語の一種である．

do 動詞文は対象概念間の動的な関係を表したものである．

《例》　アセンブラはアセンブラ言語で書かれたプログラムを機械語に変換する．

また，知識表現形式としては，①対象を的確に表現できること，②モジュール性が高いこと，③知識を構造化して表現できること，などの能力が要求される．

このような観点から，説明対象に関する知識を表現した知識構造として，概念フレームおよび文フレームと呼ぶ2種類のフレーム表現を用意している．概念フレームは，概念の静的な定義を記述するための枠組で，各概念の固有な性質に関する属性情報や，他の概念との間で成立する階層関係などを記述するために利用している．概念フレームで表現された知識は be 動詞文に変換される．一方，文フレームは，概念フレームで表現できない，概念間の動的な関係を記述するための枠組で，概念間の関係を格関係を用いて表現している．文フレーム中に現れる各概念は，概念フレームからポインタで結合されており，特定の概念フレームから関連する文フレームにアクセスできるようにしている．文フレームで表現された知識は do 動詞文に変換される．

Fig. 1 に“アセンブラ言語”と“機械語”の概念フレーム，およびそれらに関連する文フレームを示す．“アセンブラ言語”の概念フレーム中にある self は自分

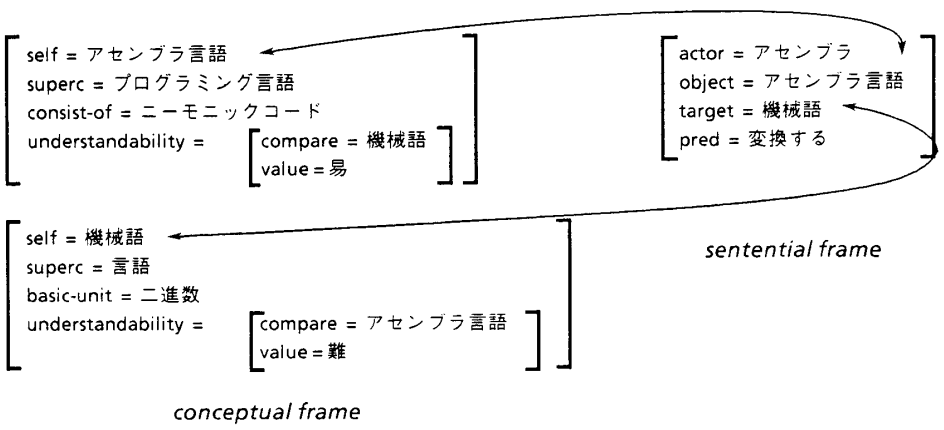

Fig. 1　Knowledge structures.

自身の概念を表し，supercは上位概念を表している．また，文フレーム中のactor, objectは，それぞれ文の主格，対格に相当し，述語はpredの属性値として記述されている．

3．説明文生成手法

3・1　基本的な考え方

通常，人間は発話する際に，真あるいは偽の命題を単純に言葉で言い表すだけでなく，聞き手に与える影響を考慮して，発話の内容や順序を決定していると考えられる．この聞き手に与える影響は，発話を遂行することで達成される効果ととらえることができ，発話を一種の行為とみなすことができる．また，人間はある行為を遂行する前に，意図する目標状態（ゴール）に達するための一連の手続き（プラン）を計画（プランニング）するものである．発話する場合も同様に，聞き手を納得させたり良い印象を与えるなどのゴールを達成するために，発話内容や発話順序をプランニングによって決定していると考えることができる[1]．

ASSISTでは，対話状況に応じて動的に説明内容を変更するために，このようなプランニングに基づく説明文生成手法を採用している．これを説明文生成プランニングと呼ぶ．説明文生成プランニングにおけるゴールとは，読み手の疑問を解消できるような説明文を生成することであり，ゴールを達成するためのプランは，説明文の内容や順序を規定した一連の手続きに対応している．

ASSISTは，説明の手順を形式的に記述したルール（オペレータと呼ぶ）*の集合を利用して，ゴールをサブゴール列に展開しながらプランニングを進めるようになっている．オペレータをプランニングメカニズムと分離しているために，説明の手順を追加，修正できると共に，読み手の知識状態をプランニング過程に反映させるなどの拡張が容易に行えるようになっている．

3・2　What-is型質問のためのプランニング

説明文生成プランニングを行うには，ユーザの質問内容に応じて達成すべきゴールを抽出しておく必要がある．この質問文から抽出されるゴールを初期ゴールと呼ぶ．初期ゴールを抽出するために，質問の型と初期ゴールの対を予め用意して，質問文から一意に初期ゴールが定まるようにしている．前章のWhat-is型質問の場合，「アセンブラ言語の定義を説明する」ことを表すDEFINE型ゴール“DEFINE（アセンブラ言語）”が抽出される．

```
OPERATOR1
   GOAL:        DEFINE(X)
   SUB GOALS:   EXPLAIN(X){superc = A}  (-1),                    ①
                EXPLAIN(X){consist-of = B}  (-1),                ②
                Explain X with its attributives  (-1),           ③
                Explain X with its descriptive information  (-1) ④
   CONDITION:   nil.
OPERATOR2
   GOAL:        EXPLAIN(X){R = Y}
   SUB GOALS:   OUTPUT(X){R = Y}  (0),                           ⑤
                DEFINE(Y)  (VY)                                  ⑥
   CONDITION:   ExistInKS(X){R = Y}
```

Fig. 2　Operators for DEFINE type goal.

DEFINE型ゴールは，Fig. 2のオペレータを用いて，概念の特徴や性質を明らかにするためのサブゴール列に展開される．オペレータは，ゴール部，サブゴール部，適用条件部からなる．ゴール部はオペレータを適用して達成されるゴールを示し，サブゴール部は，ゴールを達成するために実現すべきサブゴール列を示している．また，適用条件部はオペレータの適用可能性を表す条件を示している．

オペレータを適用して得られる各サブゴールには，さらに新たなオペレータが適用され，抽象レベルから具体レベルへと段階的に詳細化が施される．プラン展開は，すべてのゴールがプリミティブOUTPUT（文を生成するための手続き）に詳細化された段階で終了する．このプラン展開の結果，初期ゴールを頂点とし，各プリミティブが葉に対応する「プランの木構造」が得られる．

たとえば，オペレータ1にはDEFINE型ゴールを達成するために必要な手順として，「概念Xを説明するためには，①その上位概念と②構成要素について説明し，③他の属性があるときはそれらの内容についても述べ，④さらにXと関連する文フレームがあればその内容を提示せよ」ということが記述されている．したがって，初期ゴール“DEFINE（アセンブラ言語）”にオペレータ1を適用すると，概念フレーム“アセンブラ言語”から上位概念や構成要素に関する属性情報を取り出し，これらをさらに説明するためのサブゴール列に展開される．

サブゴール列のうち，属性情報を説明するためのサブゴールEXPLAINにはオペレータ2が適合する．オペレータ2には「概念Xに含まれる属性対R=Yを説明するためには，⑤属性Rについての説明文を生成した後，⑥新たな概念Yに話題を移して説明せよ」と

＊　事物を説明するための文章構造に関して，Rhetorical Predicate[6]などの幾つかの研究が行われているが，一般的な原則は発見されておらず，ASSISTで使用しているオペレータは，各種のマニュアル，計算機の入門書などを参考にして説明の手順を主観的に一般化したものに過ぎない．

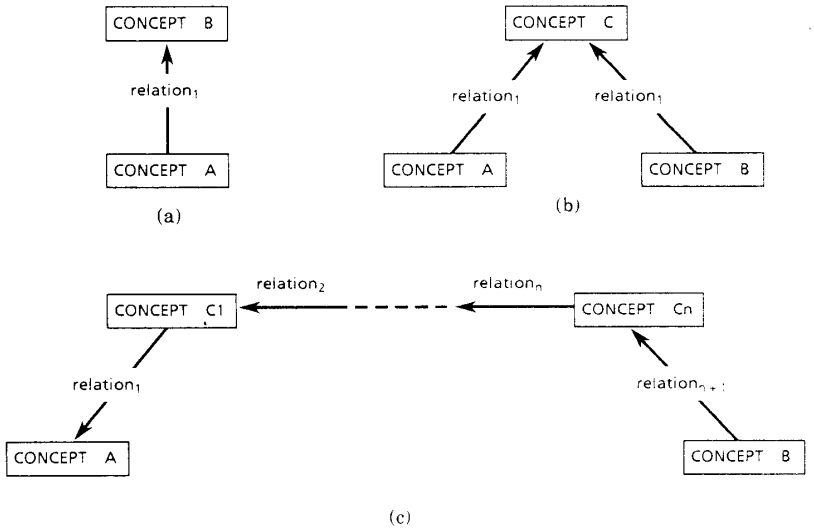

Fig. 3 Relations between two concepts.

いうことが記述されており，OUTPUT と DEFINE 型のゴールに展開される．以下同様にして，すべてのゴールがプリミティブ OUTPUT となるまで展開される．

3・3 Difference 型質問に答えるためのプランニング

Difference 型質問とは，「AとBの違いは？」のように，概念間の違いについて説明を求めるためのものである．この Difference 型質問からは DISTINGUISH 型ゴール“DISTINGUISH (A, B)”が抽出される．DISTINGUISH 型ゴールを達成するためには，概念 A, B 間で成り立つ関係に応じて比較・対照する属性を選択し，それに適した説明方法を使い分ける必要がある[5]．たとえば，共通の上位概念を持つ二つの概念“主記憶装置”と“補助記憶装置”の違いを説明する場合は，両者が共に“記憶装置”の範疇に属すること，属性“記憶容量”の値が互いに異なっていることなどを説明すればよい．一方，共通の範疇に属さない二つの概念“主記憶装置”と“中央処理装置”の違いを説明する場合には，比較すべき属性が存在しないために，「主記憶装置は中央処理装置の一部である」のように両者の機能的関係などを言及する必要がある．

以上のように，概念間で成立する関係に応じて説明方法を変えるために，概念間の関係を Fig. 3 の三つのパターンに分類している．パターン (a) は，概念 A, B の間で上位下位関係が成立する場合（例：“磁気ディスク装置”と“補助記憶装置”）を示している．パターン (b) は概念 A, B が共通の上位概念Cを持つ場合（例：“主記憶装置”と“補助記憶装置”）を示している．パターン (c) はその他の関係（例：“主記憶装置”と“中央処理装置”）を示したものである．

```
OPERATOR3
   GOAL:          DISTINGUISH(A,B)
   SUB GOALS:     OUTPUT(A){R = B}   (-1),                    1
                     Explain A with its attributives  (-1),   2
                     Explain B with its attributives  (-1)    3
   CONDITION:     ExistInKS(A){R = B}     and
                     ( R = is-a or R = instance-of )

OPERATOR4
   GOAL:          DISTINGUISH(A,B)
   SUB GOALS:     OUTPUT([A,B]){R = C}   (0),                 1
                     Explain C with its attributives  (-1),   2
                     Explain A with its attributives  (-1),   3
                     Explain B with its attributives  (-1)
   CONDITION:     ExistInKS(A){R = C}     and
                     ExistInKS(B){R = C}     and
                     ( R = is-a or R = instance-of )

OPERATOR5
   GOAL:          DISTINGUISH(A,B)
   SUB GOALS:     ExplainRelation(A,B)   (0),
   CONDITION:     not(   (   ExistInKS(A){R = B} or
                                (   ExistInKS(A){R = C}        and
                                   ExistInKS(B){R = C} )   )   and
                             ( R = is-a or R = instance-of )   )
```

Fig. 4 Operators for DISTINGUISH type goal.

このように分類された各パターンに対して，Fig. 4 に示された三つのオペレータを用意している．パターン (a) の関係を説明するには，まず，①二つの概念 A, B 間の関係を説明し，次に，②上位概念Aに固有な属性情報と③下位概念Bに固有な属性情報を対比して説明しなければならない．このような説明を行うためにオペレータ3が用いられる．オペレータ3の適用条件部には，概念 A, B 間で上位下位関係が成立することを示した論理式が記述されている．

同様に，パターン (b) の関係を説明するためには，①概念 A, B が共通の上位概念Cを持つこと，②上位概念Cに固有な属性情報，③下位概念 A, B に固有な属性情報を説明すればよい．この説明にはオペレータ4が用いられる．

通常，二つの概念の違いを尋ねる場合は，パターン(a)，(b)のように同一の範疇に属する概念について質問するものであるが，それ以外（パターン(c)）の場合には，概念の違いを特徴づける属性情報が存在しないため，オペレータ5を用いて二つの概念間の関係を説明するようにしている．オペレータ5は，オペレータ3，オペレータ4が適用不可能な場合にのみ適用され，各概念間の関係を説明するためのサブゴールEXPLAIN-RELATIONに展開されるようになっている．

4．ユーザモデル

4・1　基本的な考え方

マニュアルレスシステムでは，ユーザが新しい言葉や概念などを理解できるように，ユーザの知識レベルに合致した文章を生成する機能が必要である．たとえば，説明文を読み進むに従って，ユーザの知らない新たな概念が現れた場合，その概念に関する説明が補足されなければ，理解しづらいものとなる．逆に，ユーザが十分知っている場合には，補足される説明から目新しい情報が得られないために，煩わしく冗長な説明となる可能性がある．このように，個々のユーザの知識レベルに合わせた文章を生成するためには，説明に使用する概念をユーザが既に知っているかどうか，すなわちユーザの既知概念であるかどうかを，システムが知っておく必要がある．

一方，人間どうしの対話では，対話を積み重ねることで相手の知っていることや知らないことが徐々に明らかになっていく．しかしながら，初対面の人どうしでも，ある程度相手の知っていると思われることを考慮して，スムーズに対話を進めることができる．この現象は，不完全な情報しか存在しない場合に推論を助ける知識（デフォルト）が用意されており，対話の過程で得られる情報はそのデフォルトに追加されていくと考えることで説明できる．

以上の考察に基づき，ASSISTでは，平均的なユーザの知識状態を表すデフォルトモデルと，個々のユーザの知識状態とデフォルトモデルとの差異を表す個人モデルからなるユーザモデルを構築している．デフォルトモデルは，システムとの対話過程でまったく書き換えられることがない（静的モデル）のに対し，個人モデルは対話過程を通じて逐次更新される（動的モデル）．

具体的には，ASSISTのユーザモデルは以下の3要素から構成されている．

① ユーザの既知概念を記録したファクト集合

② 対象概念間の推論規則

③ ②の推論規則を解釈，実行するためのプログラム

①のファクト集合が個人モデルに，②の推論規則がデフォルトモデルに相当しており，これらはすべてPrologプログラムとして表現されている．このユーザモデルを利用してユーザの知識の欠落などを推定するために，さらにユーザモデルの診断モジュールとして，PDS (Program Diagnosis System)[8]を利用したユーザモデルインタプリタ(UMI)を導入している．PDSは内蔵するPrologインタプリタでPrologのゴールを実行し，実行結果が正しくない場合には，プログラムの正常な動作を知っているオラクル（通常はプログラマ）に問い合わせながら，プログラム中に存在する節の欠落や誤りなどのバグを検出するプログラム診断システムである．UMIは，PDSのPrologインタプリタ機能を利用して，Prologプログラムとして表現されたユーザモデルをトレースしながら，ユーザがある概念を知っているかどうかを推論するものである．PDSの本来の機能であるバグ検出機能をユーザモデルの診断に利用する方法については，7章で改めて議論する．

4・2　ユーザモデルの構成

ASSISTは，ユーザとの対話情報から得られる，①入力文の前提的意味要因（以後，前提と略す），②システムがユーザに示した説明内容，をユーザの既知概念として利用している．①の前提とは，ユーザの発話内容が有意味であるために真となるべき命題をさしている．質問「ファイルのアクセス権とは何ですか？」の場合，真となるべき命題は「ファイルはアクセス権という属性を持つ」である．この命題について「ユーザの発話に含まれる正しい陳述に関して，ユーザは既知である」と仮定し[4]，入力文の前提をユーザの既知概念とみなしている．

また，「ユーザは対話を介して説明された内容を忘れない」と仮定し，システムが提示した説明内容をユーザの既知概念として利用している．したがって，以下の対話の後では，“アセンブラ言語”が新たなユーザの既知概念とみなされる．

user　：アセンブラ言語とは何ですか？

system：アセンブラ言語はプログラミング言語の一種であり，ニーモニックコードからなる．ニーモニックコードはコードの一種であり，疑似コードと記号アドレスからなる．

以上のようなユーザの既知概念は，ファクト集合と

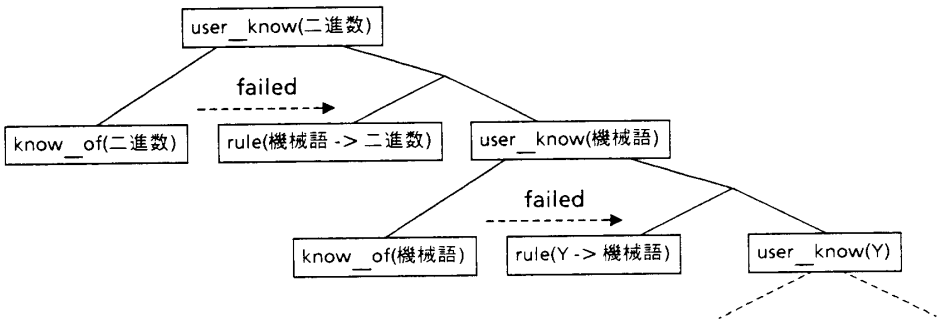

Fig. 5 Diagnosis process of a user model.

して表現されている．上例の前提「ファイルはアクセス権という属性を持つ」と，「ユーザはアセンブラ言語という概念を知っている」という事実は，それぞれ以下のファクトの形式で表現されている．

know-of (attribute (ファイル, アクセス権)).
know-of (アセンブラ言語).

また，デフォルトモデルとして導入している推論規則は，ユーザが知っているとシステムが信じている，概念間での依存関係を表しており，人間が日常行っている常識的な推論を実現するためのものである．以下に示した推論規則の例は，「"機械語"について知っているユーザは，"二進数"についても知っているであろう」という依存関係を表現したものである．

rule (機械語−> 二進数)

一方，人間がある事象について判断を下す場合には，以前から持っている一般的な知識よりも，より新しい経験によって得られた特殊な知識が優先される[2]．同様に，最近の対話過程をもとに構築される個人モデルの情報は，予め用意されたデフォルトモデルの情報よりも優先されるのが自然である．このような考え方に従って，推論プログラムは，まず個人モデルのファクト集合を検索し (述語 P1)，ファクト集合にその事実が記録されていなければ，さらにデフォルトモデルの推論規則を参照して (述語 P2) ユーザの既知概念を推論するようになっている．

(P1) user-know (X) :− know-of (X).
(P2) user-know (X) :− rule (Y−> X), user-know (Y).

ユーザモデルインタプリタ (UMI) は，以上のユーザモデルを構成するプログラムの実行過程をトレースして，ある概念がユーザの既知概念であるかどうかを判断するようになっている．Fig. 5 は，UMI が概念"二進数"がユーザの既知概念であるかどうか，を調べる様子を示している．まず始めに，UMI はPrologのゴール

user-know (二進数) ①

の実行過程をトレースする．UMI は，推論プログラムの述語 P1 を用いて，ゴール①を以下のゴールに展開する．

know-of (二進数) ②

ゴール②は，ファクト集合に概念"二進数"が記録されていれば成功し，UMI はユーザが概念"二進数"を知っていると結論づける．もし記録されていない場合，UMI は述語 P2 を用いてさらに以下のゴール列に展開し，

rule (Y−> 二進数), user-know (Y) ③

"二進数"と依存関係が成立する概念Yを導き出す．以下同様にトレースし，最終的にゴール"user-know (Y)"が失敗すれば，UMI はユーザが概念"二進数"を知らないと結論づける*．このように，UMI はユーザモデルに対する問合せの実行過程をトレースし，成功した場合は既知，失敗した場合は未知であると結論づけるようになっている．

4・3 ユーザモデルを用いたプランニングの制御

3 章で述べたように，初期ゴールはオペレータによって段階的に詳細化されるために，「プランの木構造」が下位レベルへ展開されるに従って，説明内容が詳細化されると同時に，初期ゴールの概念との話題の関連性が低下していくことになる．このような話題の発散を防ぎ，ユーザの知識レベルに応じて説明の詳細さを変えるためには，「プランの木構造」から不必要と思われるプランの部分構造を省略する必要がある．

まず話題の発散を防ぐために，「プランの木構造」の各ノードに重要度と呼ぶ数値を与えている．「プランの木構造」の初期ゴールに対応するノードには重要度'4'が与えられている．それ以外のノードには，プ

* PDS を用いて診断するには，プログラムの正常な動作を PDS に伝えるオラクルが必要である．しかしながら，上で述べた推論規則と推論プログラムには本来バグが存在しないと仮定している．また，ユーザの既知概念を表すファクト集合にもバグが存在しないと仮定しているために，オラクルに相当するものは用意していない．

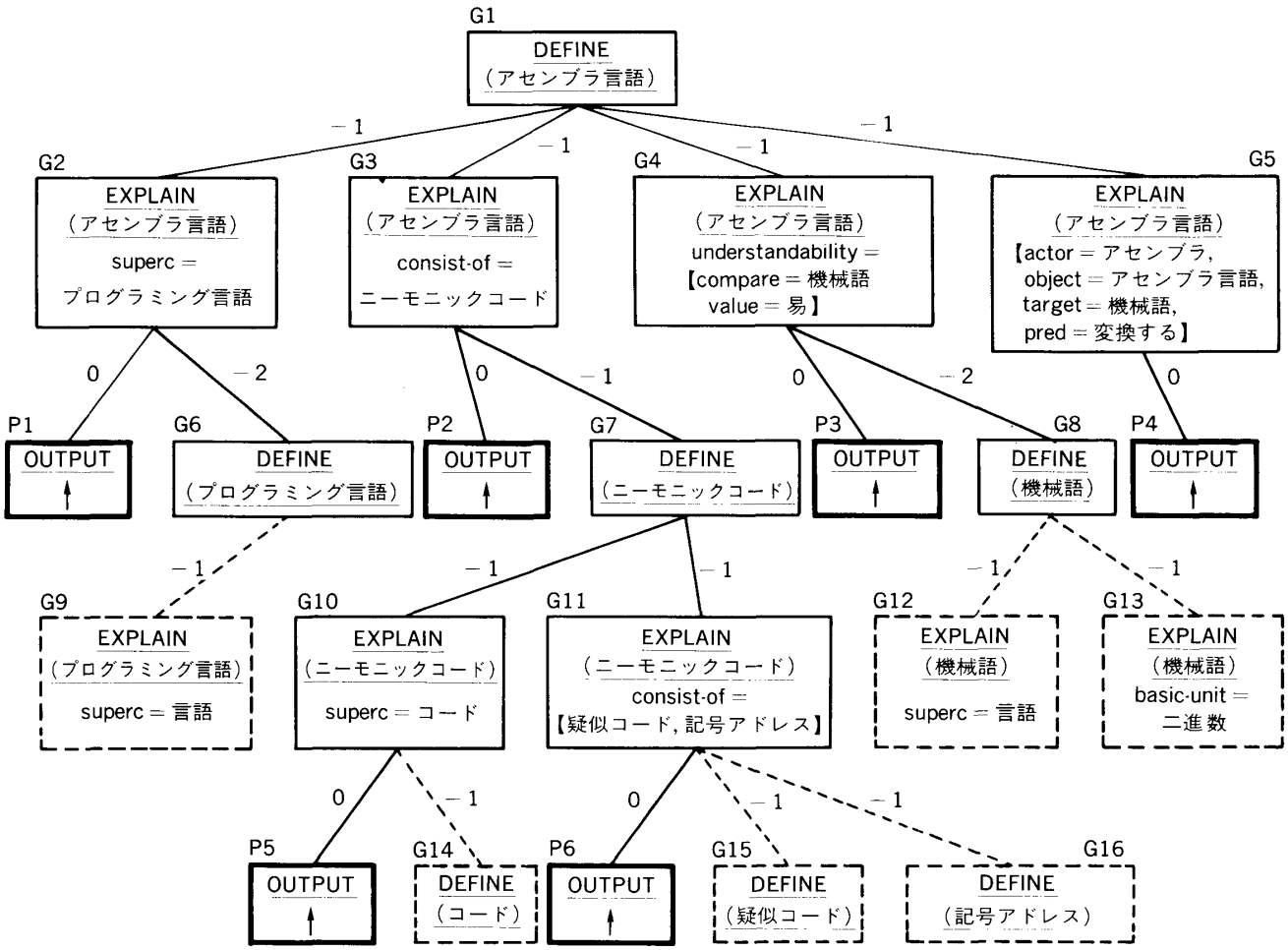

Fig. 6 A hierarchical planning structure.

ラン展開過程で初期ゴールの重要度とそのノードに至る各枝に付けられた重みの総和が計算され，それぞれ新たな重要度として各ノードに与えられる．この重みは，オペレータ中のサブゴールに付随して記述されており，負の値が割り当てられている．したがって，枝の重みの総和として計算される重要度は，下位レベルに至るに従って小さくなり，臨界値‘0’以下の重要度を持つノードは話題が発散しているものとみなされ，削除されるようになっている．

また，ユーザの知識レベルに応じて説明の詳細さを変えるために，プランニングで現れる新たな概念をユーザモデルの状態に応じてさらに詳細に言及するか，あるいは簡略化するかを決定するようにしている．オペレータ2のサブゴールDEFINEに付随させた重みVYは，この制御に用いるものである．ユーザモデルに問い合わせた結果，ユーザがその概念を知っていると推論された場合は，この重みに‘−2’が，そうでない場合は‘−1’が代入される．この枝の重みの相違によって，ユーザの既知概念について説明を行うゴールの重要度が相対的に小さくなり，不必要な説明が省略されるようにしている．

Fig. 6に，ゴール“DEFINE (アセンブラ言語)”を展開して得られる「プランの木構造」を示す．ここでは，ユーザモデルに問い合わせた結果，ユーザが“機械語”と“プログラミング言語”を知っており，“ニーモニックコード”については知らないことが推論されたものと仮定している．したがって，ノードG6, G8の枝には‘−2’が，G7の枝には‘−1’が割り当てられている．この結果，図中の破線で囲まれたノードの重要度が臨界値以下になるために，「プランの木構造」から削除されている．

5. 焦点の管理

説明文のような文章は，通常ある特定の事物について書かれている．しかしながら，文章中の個々の文について考えると，それらは常に同一の事物について言及しているわけではなく，関連する新たな概念が現れたり，以前に言及された概念が再び現れるなど，話題が次々と展開している．このように，文中で話題の中

心となっている概念を焦点と呼ぶ．人間が文章を読む際には，この焦点を発見しながら語の多義性の解消や照応指示の決定を行っている．

したがって，読みやすい文章を書くには，焦点の移動を明示して，一貫性のある文章を生成する必要がある．たとえば，文章中で同一の焦点が維持されるときには主語を代名詞化したり，焦点が移動される際には，そのことを示す接続詞を用いたり，新たに焦点となる概念を主語にするなど，焦点の管理を行った文を生成しなければならない．

ASSISTの説明文生成プランニングによって得られるゴールOUTPUTの系列には，システムが生成すべき説明内容と，その順序を規定する情報のみが含まれている．これらの情報から，つながりのある読みやすい文章を生成するために，上で述べた焦点の概念を導入している．焦点の推移過程は，現在の焦点を蓄えるレジスタCF (Current Focus)，次文で焦点となりうる概念を蓄えるリストPFL (Potential Focus List)，以前の焦点を蓄えるためのスタックFS (Focus Stack) からなる三つのデータ構造を用いて管理している[9]．ただし，PFLに蓄えられる焦点の候補となる概念とは，前文で出現した概念のうち，焦点の当たっていない概念すべてを指している．

以下では，Fig. 6の「プランの木構造」から得られるゴールOUTPUTの系列P1, P2, P5, P6, P3, P4を用いて，文生成過程で焦点の推移を管理する方法について述べる．まず，プランP1が“アセンブラ言語”についての記述であるため，CFには“アセンブラ言語”が，PFLには“アセンブラ言語”の上位概念“プログラミング言語”が代入される．また，FSには初期値として[ ](空リスト)が代入される．次にプランP1の{superc= プログラミング言語}からCFを主語とする文

「アセンブラ言語はプログラミング言語の一種である」

が生成される．プランP2もプランP1と同様に“アセンブラ言語”に関する記述であり，焦点が移動していないために，CFは“アセンブラ言語”のままである．また，PFLには[ニーモニックコード]が代入される．次に，プランP2の{consist−of= ニーモニックコード}から文

「アセンブラ言語はニーモニックコードからなる」

が生成される．最後に，上の2文は主語が同一であるので，1文目の述語を連用形にした上で，後の文の主語を削除して，重文化が行われる．プランP5, P6では，PFLの要素“ニーモニックコード”に焦点が移動し，前文の焦点“アセンブラ言語”はFSにプッシュされる．プランP5, P6でも同様に，CFが変化していないために重文化した文が生成される．次のプランP3では，再び“アセンブラ言語”について記述されているため，FSに蓄えられた概念“アセンブラ言語”がポップされる．さらに，文生成過程で話題の流れがとぎれていることを示すために段落分けが行われる．このようにして最終的に以下の説明文が生成される．なお，文生成メカニズムの詳細については別稿[3]に譲る．

「アセンブラ言語はプログラミング言語の一種であり，ニーモニックコードからなる．ニーモニックコードはコードの一種であり，疑似コードと記号アドレスからなる．アセンブラ言語は機械語と比較して理解性が高く，アセンブラによって機械語に変換される．」

6. 実　行　例

本章では，付録に示したシステムの対話例を用いて，ASSISTにユーザモデルを導入した効果について考察する．まず対話例1では，ユーザモデルの個人モデルが空であるために，“計算機ハードウェア”とそれに関連する概念について詳細な説明が行われている．対話例2は，例1の質問を行う前に，“中央処理装置”について質問した場合のものである．初めに“中央処理装置”に関する説明が行われているため，個人モデルに“演算装置”や“制御装置”などが追加されている．したがって，後の対話ではこれらの概念に関する説明が省略されている．

このように，ユーザモデルを導入した結果，すでに説明した事柄やその事柄から容易に推測される事柄に関しては，説明のレベル(詳細さ)が低くなるようになっている．最後の対話例3は，Difference型質問の応答例である．

7. 他システムとの比較および評価

ASSISTと同様に，知的なマンマシンインターフェイスの実現を目指して開発されたシステムとして，データベースの機能を自然言語で説明するTEXT[6]がある．TEXTは，談話構造の定型的な枠組(スキーマ)を用いて文章構造を決定し，つながりのある文章を生成できるようになっている．しかしながら，スキーマによって文章構成が予め規定されているために，ユーザの知識レベルに合わせて動的に文章を生成できないという欠点がある．

また，UNIX のコマンド体系について説明するマニュアルレスシステム UC[10] は，ユーザモデルを利用し，ユーザの熟練度に応じて説明内容を変更する機能を備えている．しかしながら，UC で出力される説明文は予め用意されたもので，ユーザモデルの情報を文生成過程にフィードバックできないために，同じ質問を何度繰り返しても常に同一の応答しか得られないといった問題点がある．

一方，ASSIST は，ユーザとの対話を介してユーザモデルを構築し，その情報をプランニングにフィードバックさせて，ユーザの知っている概念をそうでない概念と比較して簡単に説明するようになっている．この機能は，ゴールの展開レベルを説明の詳細さのレベルとみなし，展開レベルが臨界値を上回っているかどうかを監視するという単純な枠組で実現されている．また，初期ゴールに与えられる重要度や重みの値は，試行実験を繰り返して，生成される説明文が適当な文数（5～7文程度*）で収まるように設定したものである．したがって，この枠組は説明の意味内容を考慮したものになっておらず，幾つかの問題が生じる場合がある．たとえば，ユーザが複雑な概念について詳細な説明を要求した場合にも，ゴールの展開レベルは一定の臨界値で制限されるために，十分な説明が得られないままで説明が打ち切られることになる．ユーザが本当に必要としている情報を的確に提示するには，ユーザの知らない事柄に関しては既知概念との関連性を説明して積極的に理解を促すなど，局面に応じて異なる説明の戦略を使い分けるための新たなメカニズムが必要になる．

また，ユーザモデルの診断モジュールは，PDS のインタプリタ機能のみを利用しており，本来の機能である節の欠落や誤りなどのバグを検出する機能は利用していない．しかしながら，ASSIST のユーザモデルは，①ユーザの知識を Prolog プログラムとして表現しているために，PDS が検出する節の欠落部分はユーザの知らない知識に，誤った節はユーザの誤った知識に対応していること，②「ユーザの知識は，システムが持つ知識の部分集合として表現できる」と仮定してユーザモデルを構築していること，などの点を考慮して設計されているために，さらに以下の能力を実現することが可能である．すなわち，システムが持っている知識を PDS に伝えるオラクルを実現すれば，ある概念がユーザの既知概念であるかどうかを単純に結論づけるだけでなく，その概念を知らない原因が，どの知識の欠落によるものなのか，あるいはどの知識の誤りによるものなのかを明らかにすることができる．これらの情報を有効に活用すれば，ユーザの欠落した知識についてさらに詳細に説明したり，ユーザの誤った知識（ユーザの誤解）を訂正するための説明を追加するなど，説明内容をさらに動的に変更できるようになる．

* 付録の対話例1は十数文からなるが，これはユーザモデルの個人モデルが空であるためである．対話例2は，個人モデルが更新され，説明文の長さが短くなっている様子を示している．

8. むすび

本論文では，マニュアルレスシステム ASSIST について述べた．以下では，ASSIST のシステム拡充のために残されている幾つかの課題について検討する．

(1) 前章で述べたように，対話状況やユーザの意図なども考慮して多様なプラン展開を実現するには，プランニング過程を常に監視しながら，状況に応じて適用すべき戦略を決定する．メタプランニングの制御メカニズムを導入する必要がある．

(2) 現在のユーザモデルは，既知概念の集合と少数の推論規則から構成されているが，今後は推論規則を増やすとともに，推論プログラムと推論規則で構成されるモデルの有効性を検討する必要がある．

(3) ASSIST には，質問からユーザの意図を抽出する機能が充実していないために，質問の型が限定されるという問題点がある．ユーザの質問を柔軟に解釈するためには，Wilensky らが行ったような意図抽出メカニズム[10]を開発しなければならない．

(4) ASSIST のユーザモデルは，「ユーザは対話を介して説明された内容を忘れない」など，システムを簡略化するための仮定に基づいて構築されている．したがって，仮定に反する振舞いを予測できないために，ユーザの知識状態を正確に把握できないという問題点がある．

この問題を解決するには，学習モデルや知的 CAI などの研究で得られた認知科学的な手法も取り入れる必要がある．

謝　辞

本論文の作成にあたり，PDS について深い示唆をいただいた査読者の方ならびに本学博士過程の池田満氏に感謝します．

◇参 考 文 献◇

(1) Cohen, P.R.and Perrault, C. R. : Elements of a Plan-based Theory of Speech Acts, *Cognitive Science*, Vol. 3, pp. 177-212 (1979).

(2) Holland, J. H., Holyoak, K. J., Nisbett, R. E. and Thagard, P. : Induction (Processes of Inference, Learning, and Discovery), The MIT Press (1986).

(3) 垣内, 上原, 豊田 : 焦点の概念を導入したマニュアルレスシステムの説明機能について, Proc. of the Logic Programming Conference '86, pp. 19-26 (1986).

(4) Luria, M. : Dividing up the Question Answering Process, Proc. of AAAI-82, pp. 71-74 (1982).

(5) McKeown, K.R.: Generating Relevant Explanations : Natural Language Responses to Questions about Database Structure, Proc. of AAAI-80, pp. 306-309 (1980).

(6) McKeown, K. R. : Text Generation, Cambridge University Press (1985).

(7) 槇本, 上原, 豊田 : ユーザモデルと定性的シミュレーション法を利用した説明文生成プランニング, 情報処理学会および人工知能学会共催, 「人工知能システムの枠組み」シンポジウム, pp. 121-130 (1987).

(8) Shapiro, E. L. : Algorithmic Program Debugging, MIT Press (1982).

(9) Sidner, C. L. : Focusing in the Comprehension of Definite Anaphora, in M. Brady, *et al.* (eds.), Computational Models of Discourse, pp. 267-330, MIT Press (1983).

(10) Wilensky, R., Arens, Y. and Chin, D. : Talking to UNIX in English : An Overview of UC, *Commun. of ACM*, Vol. 27, No. 6, pp. 574-593 (1984).

〔担当編集委員・査読者 : 田中　穂積〕

◇付　　録◇

(対話例1)

・入力文>　計算機ハードウェアについて教えてください.

計算機ハードウェアは電子機器の一種であり, 中央処理装置や周辺機器からなる. 中央処理装置は演算装置, 制御装置, 主記憶装置からなる. 演算装置はレジスタや演算回路からなり, 算術演算や論理演算を実行する.

制御装置はプログラム制御装置からなり, 演算装置, 主記憶装置, 周辺装置を制御する. 主記憶装置は記憶装置の一種であり, IC メモリからなる. 補助記憶装置と比較すると主記憶装置は容量が小さく, アクセス時間が小さい.

中央処理装置はプログラムを実行する.

周辺機器は補助記憶装置, 入力装置, 出力装置からなる. 補助記憶装置は記憶装置の一種である.

入力装置にはキーボードやカードリーダがある. 入力装置はデータやプログラムを内部形式に変換する.

出力装置には CRT ディスプレイやラインプリンタがある. 出力装置は計算機が加工したデータを書き出す.

計算機ハードウェアはディジタル計算機システムを構成する. ディジタル計算機システムは計算機システムの一種である.

(対話例2)

・入力文>　中央処理装置とは何ですか.

中央処理装置は演算装置, 制御装置, 主記憶装置からなる. 演算装置はレジスタや演算回路からなり, 算術演算や論理演算を実行する. 算術演算は演算の一種である. 算術演算には加算, 減算, 乗算, 除算がある.

論理演算は演算の一種である. 論理演算には OR, AND, NOT がある.

制御装置はプログラム制御装置からなり, 演算装置, 主記憶装置, 周辺機器を制御する. 主記憶装置は記憶装置の一種であり, IC メモリからなる. 補助記憶装置と比較すると主記憶装置は容量が小さく, アクセス時間が小さい.

周辺機器は補助記憶装置, 入力装置, 出力装置からなる.

中央処理装置はプログラムを実行する.

・入力文>　計算機ハードウェアは.

計算機ハードウェアは電子機器の一種であり, 中央処理装置や周辺機器からなる. 中央処理装置は演算装置, 制御装置, 主記憶装置からなる.

周辺機器は補助記憶装置，入力装置，出力装置からなる．

計算機ハードウェアはディジタル計算機システムを構成する．ディジタル計算機システムは計算機システムの一種である．

（対話例 3）

・入力文＞　アセンブラ言語と機械語の違いを教えてください．

アセンブラ言語と機械語はプログラミング言語の一種である．アセンブラ言語はニーモニックコードからなる．ニーモニックコードはコードの一種であり，疑似コードと記号アドレスからなる．ニーモニックコードは機械コードに対応する．

アセンブラ言語は機械語と比較して理解性が高い．機械語は機械コードで表現される．アセンブラがアセンブラ言語で書かれたプログラムを機械語に変換する．

著　者　紹　介

垣内　隆志

1985 年大阪大学基礎工学部情報工学科卒業．1987 年同大学院基礎工学研究科前期課程卒業．同年，松下電器産業（株）情報システム研究所勤務．現在，マルチメディア処理，文書処理に従事．自然言語理解，マンマシンインターフェイスに興味を持つ．情報処理学会会員．

上原　邦昭（正会員）

1978 年大阪大学基礎工学部情報工学科卒業．1983 年同大学院基礎工学研究科博士後期課程退学．同年，大阪大学産業科学研究所勤務．現在，同研究所講師．工学博士．人工知能，特に自然言語理解，および自動プログラム合成の研究に従事．ACM，電子情報通信学会，計量国語学会，情報処理学会，日本ソフトウェア科学会各会員．

槇本　英治

1986 年大阪大学基礎工学部情報工学科卒業．1988 年同大学院基礎工学研究科前期課程卒業．同年，住友金属工業（株）勤務．自然言語理解に興味を持つ．情報処理学会会員．

豊田　順一（正会員）

1961 年大阪大学工学部通信工学科卒業．1966 年同大学院博士後期課程単位取得退学．同年，大阪大学基礎工学部助手．1969 年助教授．1982 年大阪大学産業科学研究所教授．工学博士．現在，主として，自然言語理解，画像理解，文書画像処理，および ICAI システムなどの研究に従事．電子情報通信学会，日本認知科学会，情報処理学会各会員．