TR-I-0322

対話構造解析プログラム
- 利用マニュアル -
Dialog Analysis Program User's Manual

大井 耕三　　西村 仁志*　　飯田 仁
Kozo OI　　Hitoshi NISHIMURA　　Hitoshi IIDA

1993.3

概要

本報告は、対話構造解析プログラムの利用方法について書いたものである。

対話構造解析処理の流れは次のとおりである。話題に関する知識に加え対話運用の知識を利用した階層型プラン認識により対話理解を行ない、その理解状態をスタックにより管理する。そして、そのスタックを参照することにより、次発話に関する抽象的な文脈情報を得ることができる。

ATR 自動翻訳電話研究所
ATR Interpreting Telephony Research Laboratories

*日本電子計算株式会社

概要

本書は、対話構造解析プログラムの利用マニュアルである。

対話構造解析処理の流れは次のとおりである。話題に関する知識に加え対話運用の知識を利用した階層型プラン認識により対話理解を行ない、その理解状態をスタックにより管理する。そして、そのスタックを参照することにより、次発話に関する抽象的な文脈情報を得る。

本書は利用マニュアルである。そのため、本プログラムの理論的説明について詳しくは述べていない。理論的説明については、飯田 仁, 有田 英一：“4 階層プラン認識モデルを使った対話の理解”, 情処学論, **31**, 6, pp.810-821(1990-6) や山岡 孝行, 飯田 仁: “階層型プラン認識モデルを利用した次発話予測手法 - 話し手の意図を表す表現についての音声認識結果曖昧性の解消”, 電子情報通信学会論文誌,vol J76-DII No.6,(1993-6) などに記述されているので、参照して欲しい。

また、階層型プラン認識よる対話理解は、プラン認識プログラム [21] の応用である。そのため、プラン認識プログラムの詳細な説明はそちらのマニュアルを参照して欲しい。

本マニュアルの構成を以下に示す。

1章 システム概要 では、対話構造解析プログラムの機能、動作環境、システム構成、ファイル構成、使用するデータについて説明している。

2章 利用方法 では、対話構造解析プログラムで利用できる関数の使用方法を説明している。

3章 操作例では、付録にあるサンプルデータを用いて、利用方法を具体的に説明している。

4章 関数リファレンスでは、対話構造解析プログラムの主な関数・マクロ・大域変数について説明している。

付録 A では、情報伝達行為認定システム (CATE) の説明をしている。

付録 B では、音声認識候補絞り込みシステムの説明をしている。

付録 C では、データ記述例ではサンプルデータが添付されている。

目次

第 1 章

システム概要

本章では対話構造解析プログラムの機能、動作環境、システム構成、ファイル構成、使用するデータについて説明する。

1.1 対話構造解析プログラムの機能

対話構造解析プログラムの機能は、次のとおりである。話題に関する知識に加え対話運用の知識を利用した階層型プラン認識により対話理解を行ない、その理解状態をスタックにより管理する。そして、そのスタックを参照することにより、次発話に関する抽象的な文脈情報を得る。

1.2 対話構造解析プログラムの稼働環境

表 1.1: 対話構造解析プログラムの稼働環境

マシン	リスプマシン、Sun
使用言語	Common Lisp

1.3 対話構造解析プログラムで扱う対話内の表現の分類と特徴

1.3.1 対話構造解析プログラムで扱う対話 (調的目標指向型対話)

本プログラムで扱う対話は、ある目的を達成するために協調的な情報交換が営まれる話し言葉対話である。まず，ここでは Allen のいう合理的な対話参加者 (rational agent)[9] を仮定する。合理的な対話参加者とは，対話に関する常識を備え，かつ行なわれている対話の目的や状況を的確に判断できる行為者である．対話が協調的に営まれるとは，直観的には，Grice[10] の会話の協働原則を狭義に解釈し，それに基づいた対話が行なわれることを仮定する．例えば，比喩的表現や皮肉による応答などは，これらは広義に協働原則を遵守しているかも知れないが，ここでは協調的であるとは

解釈しない．(目的に対する合理性や協調性の厳密な考察は，本稿で扱う範囲ではないので，これ以上は言及しない．)すなわち，協調的目標指向型対話とは，合理的な対話参加者が対話の目的を達成するために上記の意味で協調的な情報交換を行なう対話である．例えば，不要意な話題の逸脱や問い返しなどは，本システムのモデルでは扱わない．これらを，ここで提示する手法を基礎として，モデルに組み込むことは将来の課題となる．[1]

1.3.2 発話の表現の分類と特徴

協調的目標指向型対話における発話の表現は，その機能に着目して，下記のように分類することできる．(() の中は，その表現が最も一般に現れる文章中の位置を示している．)

- 一般的な陳述を行なう表現
  - 発話の意図を表す表現 (文末)
  - 発話の命題内容を構成する表現 (自立語)
  - 文章構成のための機能語 (付属語)
- 対話で固定化された表現 (断片的一発話)

一般に情報交換における陳述は，伝達したい情報の内容を表すの命題部分と，発話の目的に従った話し手の意図 (1.7参照) を表す部分とに，切り分けて考えることができる．命題内容は，述語とその格要素から構成される．これら構成要素のエンティティは，自立語として表現される．一方，発話の意図を表す表現は，文末の付属語や接続詞といった補助的な語で表現されることが多い．例えば、

対話例

事務局	「お名前を，お願いします．」	**(u1)**
質問者	「名前は，鈴木真弓です．」	**(u2-a)**
	「名前は，鈴木真弓ですか？」	**(u2-b)**
事務局	「わかりました．」	**(u3-a)**
	「ありました．」	**(u3-b)**

の **(u2)** では，「名前は鈴木真弓である」という命題についての情報を要求しているか提供しているかという話し手の意図が，文末に疑問の終助詞「か」の有無を決定している．協調的対話における話し手の意図は，情報伝達行為 [5] で抽象化される．

[1] このような問題を扱ったものとしては，Grosz and Sidner: "Discourse structure and the proper treatment of interruptions", IJCAI'85(1985), Perrault: "An application of default logic to speech act theory", in Intentions in Communication, The MIT Press (1990), Carberry: "A pragmatic-based approach to ellipsis resolution", Computational Linguistics, 15 (1989) 等がある．

対話で固定化された表現は，電話対話における「もしもし」や「わかりました」のようにある状況で習慣的に発せられる一発話である．これらは，『対話の開始』や『情報交換成立の確認』といった，それ自体固有の情報伝達行為を持つ．

このような表現の機能的特徴による分類は，1.9節で述べるプランによる予測の対象の違いに対応させることができる．そして，この分類と文章内での位置の対応は，複雑な構文解析を通さない容易な音声認識結果絞り込み処理 (付録 B参照) への，文脈予測情報の適用を可能にする．

1.4 システム構成

対話構造解析プログラムの基本的システム構成を図 1.1に示す。本システムは、対話理解、次発話の予測の 2 つのモジュールからなる。各モジュールの機能は，以下の通りである:

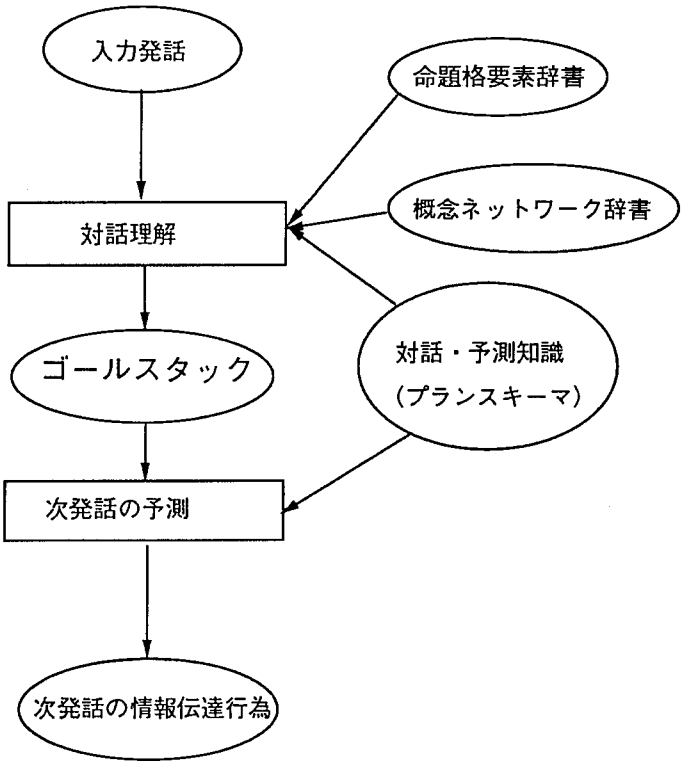

図 1.1: 対話構造解析プログラムの構成

(1) 対話理解: 階層型プラン認識モデルにより，発話の目的を解析し，対話構造を構築する． (1.8節及び [21] 参照)

(2) 次発話の予測: 構築された対話構造と対話の知識 (プランスキーマ) を利用して，次発話に関する情報を抽出する． (1.9節参照)

1.5 ファイル構成

対話構造解析プログラムのファイル構成とそのサンプルデータを表 1.2と表 1.3に示す。

表 1.2: 対話構造解析プログラムのファイル構成

ファイル名	内容
対話構造解析プログラムのロード (``AnaDialog/*'')	
load.lisp	対話構造解析プログラムのロード
load-data.lisp	対話構造解析プログラムのサンプルデータのロード
gp-evaluate.lisp	対話構造解析プログラムの処理結果のサマリーの表示
control-heuristics.lisp	プラン認識処理の制御
階層型プラン認識システム群 (``PlanRec/LAYLA/*'',``PlanRec/NP/*'')([21] 参照)	
予測プログラム群 (``AnaDialog/Prediction/*'')	
load.lisp	予測システムのロード
prediction.lisp	予測システム
ドキュメント (``AnaDialog/Doc/*'') 本マニュアル (LATEX で記述)	

表 1.3: サンプルデータ

ファイル名	内容
主プログラム群 (``PlanRec/Data/*'')	
case-grammar.lisp	命題格要素辞書
concept-nodes.lisp	概念ネットワーク辞書
kaiwa-*.data	入力行為データ
simple.plans	プランスキーマ 1(目標に関する行動の規定の記述が、副行為 (decomposition chain) のみのプランスキーマの例。推論制御モードがシンプルモードのとき使用する)
test.plans	プランスキーマ 2(simple.plans を含む。推論制御モードがシンプルモード以外のとき使用する)

1.6 対話構造解析プログラムで扱うデータ

1.6.1 入力データ

1.6.1.1 入力行為 (発話)

入力となる発話。以下の形で表現する。

(情報伝達行為 話し手 聞き手 主題 命題内容)

情報伝達行為の設定については，第1.7章で詳しく述べる．主題は，格助詞「は」相当句でマークされているか，そうでなければ文章の先頭にある，格要素である． 命題内容は，述語とその格要素で表す．例えば，対話例 (1.3.2参照) の u1 の発話の表現は，次のようになる：

```
(ASK-VALUE 事務局 質問者 名前
   ((predicate IS) (object 名前) (identifier ?id)))
```

1.6.2 知識ベース

1.6.2.1 命題格要素辞書

命題 (2.5.1参照) のうち、グラフユニフィケーション (命題間の格要素の数が異なっていても単一化できる) ができる命題を指定する辞書。([21] 参照)

1.6.2.2 プランスキーマ

プランスキーマは、知識ベース中のプランの記述である。プラン (plan) とは、ある目標実現に関する知識である。([21] 参照)

対話構造解析プログラムでは、以下に示すように 4 つのタイプのプラン (4 階層プラン) がある。

- 対話運用に関するプラン

  インタラクションプラン： 協調的な応答実行のための知識を記述したプラン.

  コミュニケーションプラン： ある話題についての協調的な対話を実現するための知識を記述したプラン.

  ダイアログプラン： 対話の大局的な展開の知識を記述したプラン.

- 話題領域固有のプラン

  ドメインプラン： 話題の行為達成のための行動を記述したプラン.

これらのプランは，実際の運用上以下のような階層関係を持つ：

インタラクションプラン ≪ コミュニケーションプラン ≪ ドメインプラン ≪ ダイアログプラン

より右辺のプランを下位のプランとする．プランは，スキーマの形で記述する．インタラクションプランの記述の図 1.2を示す (先頭に "?" のついている項は，変項を表す．):

```
((header
   (GET-VALUE-UNIT ?sp ?hr ?tpc
     ((predicate IS) (object ?obj) (identifier ?id))))
 (precondition
   (KNOW ?sp ?obj ?id))
 (decomposition
   (ASK-VALUE ?sp ?hr ?tpc
     ((predicate IS) (object ?obj) (identifier ?id)))
   (INFORM-VALUE ?sp ?hr ?tpc
     ((predicate IS) (object ?obj) (identifier ?id)))
   (CONFIRMATION ?sp ?hr ?tpc
     ((predicate IS) (object ?obj) (identifier ?id))))
 (effect
   (KNOW ?hr ?obj ?id)))
```

図 1.2: プランスキーマの記述例

1.6.2.3 概念ネットワーク辞書

集合としての同一性による単一化を行なうとき参照する知識ベースである。([21] 参照)

1.6.3 出力データ

1.6.3.1 対話構造 (ゴールスタック)

対話構造 (ゴールスタック) は、目標構造を保持・管理する構造体である。その内容は、表1.4に示す 4 つのリストよりなる。各々のスタックには、プランスキーマあるいは状態記述 (プランスキーマの effect スロットに記述される list のみ) が要素としてはいる。(詳細は、[21] 参照)

1.6.3.2 予測結果

ゴールスタックのスロット (prediction) に予測構造体 (4.4.1) としてセットされる。

表 1.4: 構造体ゴールスタックのスロット

スロット名	データタイプ	内容
incomplete	list of actions	未充足プラン
complete1	list of actions	充足 (完了) プラン
complete2	list of actions	未充足かつ談話セグメント完了プラン
statements	list of lists	共通理解事項 (状態記述)

1.7 意図表現に関する知識

本節では，特に話し手の意図を表わす表現の曖昧さを解消することに焦点を当て，それを解決するための知識として，情報伝達行為と表層表現の対応とインタラクションプランの整理を行なう。

1.7.1 情報伝達行為と発話対

情報伝達行為は，対話において，話し手が何らかの情報を聞き手に与える発話を行なう時に観察される行為である．情報伝達行為は，話し手の発話内容に関する信念と聞き手の信念に与える影響の違いに着目すれば，大きく次のようにに区別することができる．まず，聞き手に与える影響が，与えられた命題内容に関する聞き手の発話を促す要求 (Demand) の情報伝達行為と，それに促されて発話される，すなわち要求の発話に対する応答 (Response) の情報伝達行為とに分けられる．また，情報交換の成立の確認の意味で発せられる発話の情報伝達行為を，確認 (Confirm) の情報伝達行為とする．

ここで，協調的に行なわれる情報交換を仮定すれば，Demand の情報伝達行為に対する聞き手の応答の義務が生じる．すなわち情報交換成立のためには，聞き手が Demand により受けた信念に対して何らかの形で情報を与えること，すなわち Response の発話を行なわなければならない．このような，ある話題 (命題内容) についての情報交換成立のための，Demand と Response の情報伝達行為を持つ発話の対を発話対と呼ぶ．発話対には，要求した話者が情報交換成立に対する確認の意味で，Confirm の発話を行なうことがある．

発話対は，対話構造構築のための基本的構成要素になる．発話対は，インタラクションプランとして対話の知識ベースに記述する．

1.7.2 情報伝達行為、固定化表現、発話対の設定

情報伝達行為の設定 情報伝達行為は，いわゆる発話のモダリティの部分と話題[2]の属性で表現する．ここで話題の属性を導入する理由は，同様の意図による発話でも，その話題となる事柄の属性により，特に応答表現に違いが出ると考えるからである．例えば，「名前を，お願いします．」という表現に対して，「(名前は，) ...です．」と答えることができるが，「登録用紙を，お願いします」に対して，その様な表現を使った応答は少ない．これは，「(...を，) お願いします．」という，同様のモダリティを持つ要求の発話表現に対してではあるが，話題となっている事柄の違い，ここでは“名前の尋問”(オブジェクトの値) と“登録用紙の送付”(行為) によって，応答表現に違

[2]ここでの「話題」は，一般的な広義の意味で用いており，1.6.1.1節の「主題」とは，区別される．例えば，発話 u1(1.3.2参照) の主題は，“名前”であるが，話題は，名前を聞いていることとか名前の値を知ることといったぐらいの意味になる．従って，話題は発話の命題内容全体から導出されることになる．

いが出たと考えることができる.

表1.5に，発話のモダリティの部分を表わすタイプと話題の属性のタイプを示す. さらに，INFORMの下位に属すると考えられるが，対話での機能上特定の情報伝達行為有すると考えられるものを区別する.

表 1.5: 発話のモダリティと話題の属性の分類

1. 発話のモダリティのタイプ	
INFORM	: 事柄の事実について述べる。
	: (平叙文)
ASK	: 未知の事柄について質問する。
	: (疑問詞を伴う疑問文)
CONFIRM	: 事柄の真偽について質問する。
	: (疑問文)
REQUEST	: 相手に行動を依頼する。
	: (命令文)
OFFER	: 自分の行為を拘束する。
	: (動作動詞を述語とした平叙文)
GREETING	: あいさつ。
	: (おもに固定化された表現)
() 内は代表的な文形式を表している。	
2.INFORM の下位属性のタイプ	
AFFIRMATIVE	: 話題の内容を肯定する。
NEGATIVE	: 話題の内容を否定する。
ACCEPT	: 行為を受け入れる。
REJECT	: 行為を拒む。
CONFIRMATION	: 情報交換成立の確認。
3. 話題の属性のタイプ	
ACTION	: ある行為・行動についての話題
VALUE	: ある事物の属性についての話題
STATEMENT	: ある事物の状態をについての話題

表1.5のような分類から，表1.6のような情報伝達行為のタイプを設定する. また，これらの情報伝達行為に対応した代表的な表層表現を併せて示す. これら情報伝達行為のタイプに対応した表現を表層テーブルに記述しておくことにより，音声認識候補の選択に利用する (付録B参照)。情報伝達行為の解析・対応する表現の生成は，付録Aで述べる.

表 1.6: 情報伝達行為のタイプと表層表現

1.Demand Class:		
ASK-ACTION	:	「*ACT*は *WH*ですか。」
CONFIRM-ACTION	:	「*ACT*（する／できるの）ですか。」
REQUEST-ACTION	:	「*ACT*して下さい。」
	:	「*ACT*していただけますか。」
	:	「*ACT*していただきたいのですが。」
OFFER-ACTION	:	「*ACT*します。」
	:	「*ACT*させていただきます。」
ASK-VALUE	:	「*OBJ*は *WH*ですか。」
	:	「*OBJ*をお願いします。」
	:	「*OBJ*を教えて／聞かせて下さい。」
	:	「*OBJ*を聞くことができますか。」
	:	「*OBJ*をお聞きしたいのですが。」
CONFIRM-VALUE	:	「*OBJ*は *VAL*ですか。」
	:	「*OBJ*は *VAL*ですね。」
ASK-STATEMENT	:	「*STA*は *WH*ですか。」
CONFIRM-STATEMENT:		「*STA*ですか。」「*STA*ですね。」
GREETING-OPEN	:	「もしもし。」*
GREETING-CLOSE	:	「失礼します。」*「さようなら。」*
	:	「ありがとうございました。」
2.Response Class:		
INFORM-ACTION	:	「*ACT*して下さい。」
	:	「*ACT*しなくてはいけません。」
	:	「*ACT*です。」
INFORM-VALUE	:	「*OBJ*は *VAL*です。」
	:	「*VAL*を／が *STA*ます／です。」
INFORM-STATEMENT	:	「*STA*です（が）。」
	:	「*STA*したい（のですが）。」
AFFIRMATIVE	:	「はい。」*「そうです。」*
NEGATIVE	:	「いいえ。」*
	:	「（否定表現）。」
ACCEPT-ACTION	:	「わかりました。」*
	:	「*ACT*は問題ありません。」
REJECT-ACTION	:	「*ACT*できません。」
ACCEPT-OFFER	:	「ありがとうございます。」*
	:	「（よろしく）*ACT*願います。」
REJECT-OFFER	:	「（いいえ）結構です。」
GREETING-OPEN	:	「はい。」*
GREETING-CLOSE	:	「失礼します。」*「さようなら。」*
	:	「どういたしまして。」*
3.Confirm Class:		
CONFIRMATION	:	「わかりました。」*「そうですか。」*

表層表現内のイタリック文字は変項を表し、それぞれ *ACT* は ACTION に関する内容、*OBJ* は OBJECT に関する内容、*STA* は STATEMENT に関する内容、*WH* は疑問詞を表す。また、／は選言を表し、() 内は付いても付かなくても良い表現である。

表 1.7: 発話対の構成

Demand Class	Response Class
ASK-ACTION	INFORM-ACTION
CONFIRM-ACTION	AFFIRMATIVE,NEGATIVE INFORM-ACTION
REQUEST-ACTION	ACCEPT-ACTION,REJECT-ACTION
OFFER-ACTION	ACCEPT-OFFER,REJECT-OFFER
ASK-VALUE	INFORM-VALUE
CONFIRM-VALUE	AFFIRMATIVE,NEGATIVE INFORM-VALUE
ASK-STATEMENT	INFORM-STATEMENT
CONFIRM-STATEMENT	AFFIRMATIVE,NEGATIVE INFORM-STATEMENT
GREETING-CLOSE	GREETING-CLOSE

固定化表現の設定 さらに，問い合わせ電話対話などにおいて，慣用的に発話される固定化された表現 (以下固定化表現と呼ぶ) を設定する．(表 1.6右欄の "*" のついた表現)

固定化表現は，示した情報伝達行為による発話をなす時，対話中に断片的な一発話として現れることが多い．従ってこれらの表現が音声認識候補にある時は，入力文全体に対して優先的に予測情報との対応を見るようにする．このような表現を優先して処理することは，システムの処理効率の向上につながる．

発話対の設定 前述したように協調的なやり取りを仮定すれば，表 1.6で設定した情報伝達行為は，表 1.7のような発話対を構成する．これらをインタラクションプランのスキーマの decomposition スロットの値として，Demand Class，Response Class，Confirm Class の順で記述し，対話理解並びに次発話の予測に用いる．

1.8 対話理解と対話管理

対話構造解析プログラムのモデルは階層型プラン認識モデル [21] である。このモデルは，3 つの対話運用に関するプランと 1 つの話題領域固有のプラン (1.6.2.2節参照) を利用して，対象対話の対話構造を構築する [5]. 構築された対話構造は，システムの理解状態としてスタックにより管理する.

1.8.1 対話理解

1.6節で説明した入力発話、知識ベースを入力データとして対話構造を出力する。

階層型プラン認識モデルによる発話理解の処理は，発話の表現からより上位のプランへの連鎖を求めることである. 連鎖を求めるための推論規則として，decomposition chain, precondition chain, effect chain がある. 発話の表現とプランのスロット，あるいはプランのスロット間のマッチングは，単一化により行なう. このモデルによる処理により，発話理解において，プランの階層性による効率的なプラン探索と各発話の対話全体における関係付けの明確化が可能となる.

1.8.2 対話管理

対話構造と理解状態を管理するスタックには，以下のものを用意する:

C: 充足プランスタック

I: 未充足プランスタック

S: 理解事項スタック

スタック C, I の要素は，プランスキーマのインスタンスであり，プランの階層性と発話順に基づいた順序に基づいてプッシュされる. すなわち，最近の発話に関するより下位のプランのインスタンスが，スタックの上の方にあることになる. 発話のプラン認識によりすべてのスロットが満たされたインスタンスは，I から，C へ移す. このインスタンスがスタックの最上部のものでないときは，それまで I からポップしたインスタンスも C へ移す[3]. また，S には充足されたプランの effect に記述された命題 (前節のプランの例では, (KNOW 事務局 名前 ?id)) がはいる.

入力され，認識された発話の表現は，通常最も下位のプランであるインタラクションプランのインスタンスのスロットに単一化される. プランのインスタンス間の連鎖

[3]一般的な対話を対象とした場合，ポップした未充足プランを無条件に C へ移す処理は，柔軟ではない. この処理を避けるためには，不完全な (しかし焦点が当たっていない) インスタンスを一時的に格納しておく別のスタックを用意すれば良い. このスタックを参照するタイミングは，本文の I において連鎖を求められなかった場合や，「先ほどの件ですが」などの clue が認識された場合であろう. ここでは，協調的な対話の仮定において発話対の交差はないものと仮定し，この処理を行なう.

は，ポインタで指示する(下の例の，# で始まるアトム)[4]．スタック内のイメージを，図 1.3に示す．例えば，対話例の発話 u1 入力後の未充足プランスタックの一つの最も上部には，図 1.4のようなインスタンスがあることになる(#u1 の具体値は，前節の最後に示した発話の表現である):

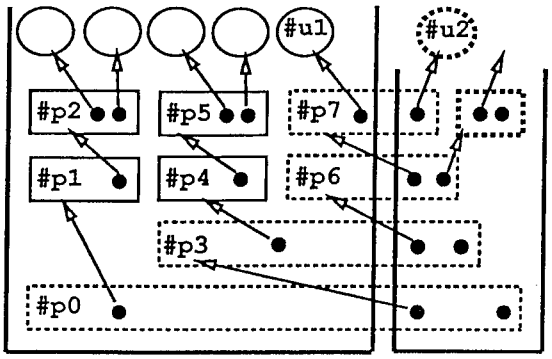

図 1.3: 対話管理のスタック

```
((header
   #p7(GET-VALUE-UNIT 事務局 質問者 名前
        ((predicate IS) (object 名前) (identifier ?id))))
 (precondition
   (KNOW 質問者 名前 ?id))
 (decomposition
   #u1
   (INFORM-VALUE 質問者 事務局 名前
     ((predicate IS) (object 名前) (identifier ?id)))
   (CONFIRMATION 事務局 質問者 名前
     ((predicate IS) (object 名前) (identifier ?id))))
 (effect
   (KNOW 事務局 名前 ?id)))
```

図 1.4: プランインスタンスの例

[4]ここでは便宜上，#p で始まるものをプランのインスタンス間の，#u で始まるものを発話の表現との連鎖を示すものとして記述する．

1.9 次発話の予測

協調的目標指向型対話を仮定すれば，上述のスタックIを参照することにより，次発話の抽象的内容を予測することが可能である [6][7]. なぜなら，未充足であるプランインスタンスの満たされていないスロットに対応する発話は，対話の目的達成のために必要であると考えられるので，後に発話されることが期待される.

各プランのもつ機能から，発話の意図に関する表現および対話で固定化された表現はインタラクションプランに記述された情報伝達行為のタイプ，命題内容を構成する表現はドメインプランに記述された主題 (あるいは格要素) から，抽象化された内容として予測される. 予測された情報は，プランの連鎖により下位のプランに伝搬される. 従って各発話の予測情報は，情報伝達行為のタイプと命題内容に関する話題との組で表される.

予測情報は，次発話の話し手により変化する. これは，予測された発話の表現の話し手のスロットを参照して得られる. 例えば，質問者の発話に対して前節のスタックの状態から最初に予測される情報は，情報伝達行為タイプ INFORM-VALUE と話題に関する『名前』の概念になる. 一方，次発話の話し手が事務局である場合は，概念『名前』に関する CONFIRMATION は予測情報として取り上げない. これは，応答の満たされていない発話対に対する確認は成立しないという，プラン適用に関する制約条件としてプランスキーマに記述する. [5]

次発話に関する予測情報は，発話順とプランの階層性による優先順位を持つ. これは，基本的にスタックIにプッシュされた順序にしたがう. すなわち，システムは最近の話題に関するやり取りを満たすような発話を期待していることになる.

---

[5]全てのインタラクションプランについては，この制約条件が適用できるが，より上位のプラン，特にドメインプラン，に対しては，このような制約条件は強過ぎる. また，前提条件 (precondition) スロットに関しても適用制約は考えられる. 従って，各スキーマの制約条件として記述しておく.

1.10 関連システム

対話構造解析プログラムは一つの独立したシステムである。しかし、本来、対話構造解析プログラムは音声言語処理システムのサブシステムを想定して作成された。

以下に、そこでの対話構造解析プログラムの位置を図1.5に示し、音声言語処理システムの概要を説明する。

なお、(3) 音声認識候補絞り込み、(5) 情報伝達行為解析システムについては本システムの付録として付いている。

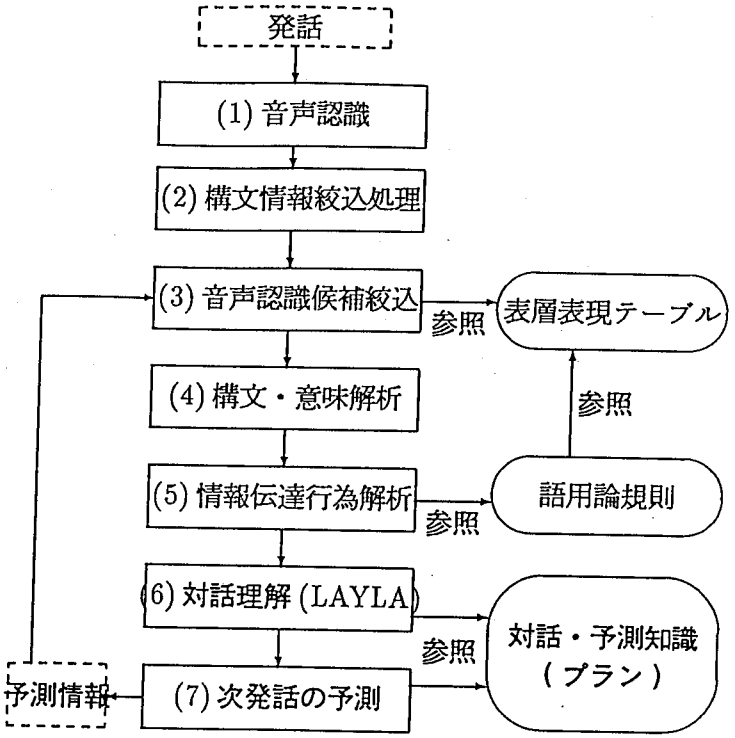

図 1.5: 音声言語処理システムの構成

(1) 音声認識: 文節区切り発声による音声入力を，HMM-LR 音声認識手法により音声認識を行ない，認識候補の文節ラティスを出力する [15].

(2) 構文情報による絞り込み処理: 係受けデータによる文章の整合性の評価により，文節候補を絞り込む [2].

(3) 音声認識候補絞り込み処理: 対話理解部からの予測情報を利用して，文節候補を選択する． B.1.6節で詳しく説明する.

(4) 構文・意味解析: ユニフィケーションベースパーサにより素性構造意味表現を出力する [16].

(5) 情報伝達行為解析: 素性構造意味表現から，発話の意図 (情報伝達行為) と命題を抽出し，発話の表現を出力する． A節で詳しく説明する．

(6) 対話理解: 階層型プラン認識モデルにより，発話の目的を解析し，対話構造を構築する． (1.8節参照)

(7) 次発話の予測: 構築された対話構造と対話の知識 (4 階層プラン) を利用して，次発話に関する情報を抽出する． (1.9節参照)

基本動作　上記音声言語処理システム全体の基本動作は，予測情報の優先順位による優先処理を行なう．[6]すなわち，(3) でより優先度の高い予測情報により選択された文候補が (6) において成功すれば，(7) を行ない次発話 (1) へ，失敗すれば，(3) において残りの予測情報により，以上の処理を行なう．ここで，(6) における成功 とは，入力発話表現が，現在のスタック内で連鎖可能であることとする．

このアルゴリズムは，構文情報による文章として確からしい音声認識候補の中で，対話の流れに沿ったものを，文脈情報により選択し，優先的に解析を行なっていく，と捉えることができる．なお，(2) で候補が一つに定まった時は，それを正解とする．また，(3) で適切な解が見つからない時や複数が正解となる時は，全解を (4) に受け渡す[7]．これらの時，文脈情報による表現候補の修正 (correction) や提示は行なわない．

[6]これは，今のところ逐次的な処理を仮定していることによる．将来，並列処理が可能な装置の導入により，処理速度向上が期待できる．

[7]この構文解析機構内の音声認識候補選択については，永田，小暮: "音声言語日英翻訳実験システム SL-TRANS における日本語解析", 情処学自然言語処理技報,78-20 (1990-7) を参照されたい．

第 2 章

利用方法

本章では、第1章で述べた機能を実装したシステム対話構造解析プログラムの利用方法および関数について説明する。

2.1 対話構造解析プログラムのインストール

対話構造解析プログラムのインストールは、以下の通りである。

1. システムにログインする
   login: ユーザ名
   password: パスワード

2. 対話構造解析プログラムをインストールするディレクトリへ移動する
   cd ディレクトリ名

3. MT から DISK へコピーする (カレントディレクトリの下にディレクトリ AnaDialog と PlanRec を作り、コピーする)
   tar xvf デバイス名

4. パス名をセットする。
   ファイル **AnaDialog/load.lisp** を編集し、*AnaDialog-path* の値を 3でコピーした対話構造解析プログラムのディレクトリ名にする
   (defvar *AnaDialog-path* "ディレクトリ名 /AnaDialog/")

2.2 対話構造解析プログラムの起動の準備

1. LISP を立ち上げる

2. 対話構造解析プログラムのディレクトリへ移動する
   (cd "ディレクトリ名 /AnaDialog/")

3. 対話構造解析プログラムをロードする
   (load “load.lisp”)

4. パッケージを:gplanner にする
   (in-package :gplanner)

5. 対話構造解析プログラムをコンパイルする。(この処理は最初の準備のときのみ)

   (compile-AnaDialog)

2.3 データの用意

1. 入力データ、知識ベース (命題格要素辞書、プンスキーマ、概念ネットワーク辞書) を用意する。(2.5節、[21] 参照)

2. 1で定義したデータのファイル名を変数にセットする。
   load-data.lisp を編集する。(表 2.1参照)

3. データファイルをロードする。
   (load “load-data.lisp”)

表 2.1: データのファイル名をセットする変数

変数名	説明
プラン認識プログラム参照用 (パッケージは gplanner)(‘‘PlanRec/load-data.lisp’’)	
data-path	プランスキーマなど、データ・知識を記述したファイルのあるディレクトリ
schemata-filename	プランスキーマを記述したファイル名
np-concept-filename	概念ネットワーク辞書のファイル名
proposition-grammar-filename	命題格要素辞書のファイル名
kaiwa-input-filename	入力行為データのファイル名
対話構造解析プログラム参照用 (パッケージは user)(‘‘AnaDialog/load-data.lisp’’)	
simple-plan-filename	プラン認識の制御モードがシンプルモードのとき、使用するプランスキーマのファイル名

2.4 システムの操作

2.4.1 一括に対話理解、次発話の予測を行なう手順

入力データのすべての文に対して一括に次の処理を行なう。1）対話理解を行ない、2）次発話の予測をする。

(gp-test-file *&optional (filename* *kaiwa-input-filename**) *&key tree*)

tree が nil でないとき、対話構造を表示する。

2.4.2 逐次的に対話理解、次発話の予測を行なう手順

入力ファイルの文に対して逐次的に上述の1）、2）を行なう。

1. *filename*のファイルから入力命題をロードする。
   (gp-load-file *&optional (filename* *kaiwa-input-filename**))
2. 対話構造解析プログラムを初期化する。(目標構造・入力履歴の初期化を行なう。)

   (gp-reset)
3. プランニングの制御モード (表2.2参照) を指定する。

4. 指定された入力の ID に対して、上述の1）、2）の処理する。
   (gp-next 'ID)
5. 現在の理解状態の概観を *stream*へ表示する。
   (tprint-gsl *&optional (stream t)*)

2.4.3 その他の主な操作

- 知識ベースのロード
  - プランスキーマ (*file*) をロードする。
    それまでのプランスキーマはクリアされる。
    (load-schemata *&optional (file *schemata-filename**))
  - 概念知識ベースファイルをロードして、概念ネットワークを作る
    (np::load-concept-network *&key file package*)

- 知識ベースの表示
  - プランスキーマテーブルの内容全体を *stream*へ表示する.
    (print-schemata *&optional (stream t)*)
  - 概念ネットワークの全てのノードを端末出力へ木構造で表示する。
    (np::pprint-all-nodes *&key (start :bottom) (type-list nil) (network :concept)*)

表 2.2: 制御モードを指定する関数

指定するための関数	説明
(set-inference-rules rules)	適用推論規則モード rules をセットする
(set-schema-type schema-type)	検索対象モード schema-type をセットする
(layer-mode-on)	階層モードを layered にする。
(layer-mode-off)	階層モードを single にする。
(input-order-on)	入力順序モードを sequential にして、入力間の優先順位を設定した処理を行なう。
(input-order-off)	入力順序モードを parallel にする。
(goal-direction-br)	対象ゴールモードを br にする。
(goal-direction-dp)	対象ゴールモードを dp にする。
(chain-mode-direct)	直接間接モードを direct にする。
(chain-mode-indirect)	直接間接モードを indirect にする。
(set-indirect-depth num)	間接連鎖回数 num を設定する。
(first-hit-on)	最短優先モードで処理を行なう。
(first-hit-off)	最短優先で行なわない (すなわち求められる範囲のものは全て求める)
(simple-mode-on)	シンプルモードにする。
(simple-mode-off)	シンプルモードでないようにする。
(trace-on)	トレース表示を行なうようにする。
(trace-off)	トレース表示を行なわない。

- プランニング処理
  - プランニングの制御モードを表示する。
    (display-mode)
  - *ID*(2.5節参照) で指定された入力を処理した時点の状態に戻す。
    (reset-gplanner*&optional ID*)
  - 処理結果のサマリーを表示する。
    (print-gp-eval-all)
  - ロードされた入力命題の文 ID のリストを返す。
    (gp-input-id-list)

2.5 データ記法

2.5.1 記号の定義

- シンボル (*symbol*) は、"?" 以外で始まる LISP で利用できるシンボルに等しい。
- 任意変数 (*free-var*) は、"?" で始まる LISP シンボルで表す:

  free-var ::= *?symbol*

- タイプ付変数 (*typed-var*) は、"?" で始まり、その直後に LISP リスト (その要素はシンボルのみ) をとる:

  typed-var ::= *?(label type)*
  label ::= *symbol*
  type ::= ≪*concept-name*≫

  特に指定がなければ、任意変数・タイプ付変数を総じて変数 (*var*) と呼ぶ。

  var ::= *free-var* | *typed-var*

- リスト (*list*) は、シンボル・変数・リストを要素としてとる LISP リストである。

- 命題 (*prp*) は、特定の指定された述語 (*pred*) で始まるリストである。

  prp ::= *(pred case)*
  pred ::= *symbol*
  case ::= ≪ *concept* | *var* | *prp* ≫

  命題のみを要素とする LISP リストを、複合命題 (*complex-prp*) と呼ぶ。命題は、行為の見出しや状態を表現するために利用する。

  例:

```
(ASK-STATEMENT      SP2 SP1 ?TPCD1-5
                      (だ -IDENTICAL ?OBJED1-5 用件 -1)
                      "どのようなご用件でしょうか")
```

2.5.2 データの定義方法

1. 概念の定義 ([21] 参照)

   - 概念 (*concept*) は、概念ネットワーク辞書で管理されるデータであり、ラベルとリンクから構成される。

     concept ::= *('defconcept' concept-name link)*
     concept-name ::= *symbol*
     link ::= ≪*(link-name concept-name)*≫
     link-name ::= *symbol*

     'defconcept' は概念ノード定義用のマクロである。

   - 記述例:

```
(np::defconcept  住所 -1
                 (is-a 送り先 -1))
```

この例は、概念 “住所 -1” は概念 “送り先 -1” の is-a 関係による下位概念であることを表す。

2. 命題格要素の定義 ([21] 参照)

   - 命題格要素の定義は述語 (*pred*) と格要素 case (ここでの case は変数のみである) で指定されたリストである。

     prp ::= (pred case)
     pred ::= symbol
     case ::= ≪ var ≫

   - 記述例:

     ```
     (だ -IDENTICAL    OBJE IDEN)
     ```

3. 入力行為の定義 ([21] 参照)

   - 入力行為 (*input*) は、ユニークな ID を第 1 要素とする命題 (および複合命題) の LISP リストである:

     input ::= (ID ≪ prp | complex-prp ≫)
     ID ::= symbol

   - 記述例:

     ```
     (D1-5
       (
         (ASK-STATEMENT      SP2 SP1 ?TPCD1-5
                             (だ -IDENTICAL ?OBJED1-5 用件 -1)
                             "どのようなご用件でしょうか")
       )
     )
     ```

4. プランスキーマの定義 ([21] 参照)

   - プランスキーマは、以下のように記述する (内容については、?? 節参照):

     schema ::= ('defschema' type body)
     type ::= symbol
     body ::= "(':HEAD' prp ≪slot≫)"
     slot ::= slot-name slot-body
     slot-name ::= ':DECO', ':PREC', ':EFFE', ':DELE', ':CONS'
     slot-body ::= (≪prp≫)

     'defschema' はスキーマ定義用のマクロである。

   - 記述例:

```
(gplanner::defschema :DOMAIN-PLAN
  "(
  :HEAD (SEND-SOMETHING ?AGN ?RCP ?(SOMETHING object))
  :PREC ((HAVE ?AGN ?SOMETHING)
         (KNOW ?AGN (is ?(DEST 送り先-1) ?(VAL pronoun))))
  :DELE ((HAVE ?AGN ?SOMETHING))
  :EFFE ((HAVE ?RCP ?SOMETHING))
  :DECO ((INTRODUCE-ACTION ?AGN ?RCP ?TPC (SEND ?AGN ?RCP ?SOMETHING)))
  :CONS ()
  )"
)
```

この例は、対話において「(AGN が RCP に) 何か (SOMETHING) を送る」ことを実現するためのプランである。'()' は nil として解釈する。

第 3 章

操作例

サンプルデータ (入力行為データ、概念ネットワーク辞書、命題格要素辞書、プランスキーマ)(C参照) を使い操作例を示す。

なお、本例で使用したマシンは SunSPARCstation2 であり、 LISP は nemacs から Lucid Common Lisp を起動して使っている。

3.1 対話構造解析プログラムをインストールする

1. システムにログインする
   login:user
   password: パスワード

2. 対話構造解析プログラムをインストールするディレクトリへ移動する
   cd /home/usr

3. MT から DISK へコピーする (カレントディレクトリの下にディレクトリ AnaDialog と PlanRec を作り、コピーする)
   (デバイス名は /dev/rst0 とする)

   tar xvf /dev/rst0

4. パス名をセットする。
   ファイル **AnaDialog/load.lisp** を編集し、 *AnaDialog-path* の値を 3でコピーした対話構造解析プログラムのディレクトリ名にする

   (defvar *AnaDialog-path* "/home/user/AnaDialog/")

3.2 対話構造解析プログラムの起動の準備

1. LISP を nemacs から起動する

   `m-x lucid` を入力すると次のメッセージが表示される。

```
Starting /usr/local/bin/lisp ...
;;; Sun Common Lisp, Development Environment 4.0.0 , 6 July 1990
;;; Sun-4 Version for SunOS 4.0.x and sunOS 4.1
;;;
;;; Copyright (c) 1985, 1986, 1987, 1988, 1989, 1990
;;;            by Sun Microsystems, Inc.,  All Rights Reserved
;;; Copyright (c) 1985, 1986, 1987, 1988, 1989, 1990
;;;            by Lucid, Inc., All Rights Reserved
;;; This software product contains confidential and trade secret
;;; information belonging to Sun Microsystems, Inc.  It may not be copied
;;; for any reason other than for archival and backup purposes.
;;;
;;; Sun, Sun-4, and Sun Common Lisp are trademarks of Sun Microsystems Inc.

>
>
```

2. 対話構造解析プログラムのディレクトリへ移動する

```
> (cd "/home/user/AnaDialog/")
#P"/home/user/AnaDialog/"
>
```

3. 対話構造解析プログラムをロードする

```
> (load "load")
;;; Loading source file "load.lisp"
;;; Loading source file "../PlanRec/load.lisp"
;;; Warning: File "../PlanRec/load.lisp" does not begin with
IN-PACKAGE.  Loading into package "GPLANNER"
;;; Loading source file "../PlanRec/LAYLA/test.lisp"
;;; Loading source file "../PlanRec/NP/load.lisp"
;;; Loading binary file "../PlanRec/Utility/utility.sbin"
;;; Loading binary file "../PlanRec/LAYLA/tools.sbin"
;;; Loading binary file "../PlanRec/LAYLA/tree.sbin"
;;; Loading binary file "../PlanRec/LAYLA/display.sbin"
;;; Warning: Redefining MACRO MONITOR whose source-file was not recorded
;;; Loading binary file "../PlanRec/LAYLA/unify.sbin"
;;; Loading binary file "../PlanRec/LAYLA/unify2.sbin"
;;; Warning: Redefining FUNCTION UNIFY-1 which used to be defined in
 "/home/user/PlanRec/LAYLA/unify.lisp"
;;; Warning: Redefining FUNCTION VAR-UNIFY which used to be defined in
"/home/user/PlanRec/LAYLA/unify.lisp"
;;; Loading binary file "../PlanRec/LAYLA/schema.sbin"
```

```
                    --- 中略 ---

--- Control Modes of 0 ---
Inference Rules      = (DECOMPOSITION-CHAIN EFFECT-CHAIN PRECONDITION-CHAIN)
Schema Typee         = (INTERACTION-PLAN COMMUNICATION-PLAN
DOMAIN-PLAN DIALOGUE-PLAN)
Layer Mode           = T
Input Order          = T
Goal Direction       = BR
Chain Mode           = DIRECT
Indirect Depth       = NIL
First Hit            = T
Simple Mode          = NIL
Trace                = NIL
;;; Loading source file "gp-evaluate.lisp"
;;; Warning: File "gp-evaluate.lisp" does not begin with IN-PACKAGE.
 Loading into package "GPLANNER"
;;; Warning: Redefining FUNCTION SIMPLE-MODE-ON which used to be
defined in "/home/user/PlanRec/LAYLA/control.lisp"
;;; Warning: Redefining FUNCTION SIMPLE-MODE-OFF which used to be
defined in "/home/user/PlanRec/LAYLA/control.lisp"
#P"/home/user/AnaDialog/load.lisp"
>
```

4. パッケージを:gplanner にする

```
> (in-package :gplanner)
#<Package "GPLANNER" 6344B6>
>
```

5. 対話構造解析プログラムをコンパイルする。(この処理は最初の準備のときのみ)

```
> (compile-anadialog)
;;; You are using the compiler in development mode (compilation-speed = 3)
;;; If you want faster code at the expense of longer compile time,
;;; you should use the production mode of the compiler, which can be obtained
;;; by evaluating (proclaim '(optimize (compilation-speed 0)))
;;; Generation of full safety checking code is enabled (safety = 3)
;;; Optimization of tail calls is disabled (speed = 2)
;;; Reading source file "../PlanRec/Utility/utility.lisp"
;;; While compiling CHANGE-EJ
;;; Warning: Free variable *TPLAN-EJ-DICT* assumed to be special
```

```
;;; Writing binary file "../PlanRec/Utility/utility.sbin"

                --- 中略 ---

;;; Reading source file "Prediction/prediction.lisp"
;;; Writing binary file "Prediction/prediction.sbin"
;;; Loading binary file "Prediction/prediction.sbin"
NIL
>
```

3.3 データの用意

1. 入力データ、知識ベース (命題格要素辞書、プランスキーマ、概念ネットワーク辞書) を用意する。

   ここでは、付録のサンプルデータを使用する。

2. 定義したデータのファイル名を変数にセットする。
   ここでは、付録のサンプルデータを使用するので、編集しない。
   以下に、 PlanRec/load-data.lisp と AnaDialog/load-data.lisp の内容を示す。

   ファイル名: PlanRec/load-data.lisp

```
(in-package :gplanner)
;;; Data
(defvar *data-path*
    (concatenate 'string user::*PlanRec-path* "Data/"))
(defvar *schemata-filename*
  (merge-pathnames "test.plans" *data-path*))
(defvar *np-concept-filename*
  (merge-pathnames "concept-nodes.lisp" *data-path*))
(defvar *proposition-grammar-filename*
  (merge-pathnames "case-grammar.lisp" *data-path*))
(defvar *kaiwa-input-filename*
  (merge-pathnames "kaiwa-1.data" *data-path*))
;;;;;;;;;;;;
(init-all)
```

   ファイル名:AnaDialog/load-data.lisp

```
(in-package :user)
(load (concatenate 'string user::*PlanRec-path* "load-data"))
(setq *simple-plan-filename* "simple.plans")
```

3. データファイルをロードする。

```
> (load "load-data")
;;; Loading source file "load-data.lisp"
;;; Loading source file "../PlanRec/load-data.lisp"
;;; Loading source file "../PlanRec/Data/test.plans"
;;; Warning: File "../PlanRec/Data/test.plans" does not begin with
 IN-PACKAGE.  Loading into package "GPLANNER"
;;; Loading source file "../PlanRec/Data/simple.plans"
;;; Warning: File "../PlanRec/Data/simple.plans" does not begin with
IN-PACKAGE.  Loading into package "GPLANNER"
;;; Loading source file "../PlanRec/Data/concept-nodes.lisp"
;;; Warning: File "../PlanRec/Data/concept-nodes.lisp" does not begin
with IN-PACKAGE.  Loading into package "GPLANNER"
#P"/home/user/AnaDialog/load-data.lisp"
>
```

3.4 一括に対話理解、次発話の予測を行なう手順

入力データのすべての文に対して一括に次の処理を行なう。1）対話理解を行ない、2）次発話の予測を行なう。

```
> (gp-test-file "kaiwa-2.data" :tree t)
--- Control Modes of 0 ---
Inference Rules      = (DECOMPOSITION-CHAIN EFFECT-CHAIN PRECONDITION-CHAIN)
Schema Typee         = (INTERACTION-PLAN COMMUNICATION-PLAN DOMAIN-PLAN DIALOGUE-PLAN)
Layer Mode           = T
Input Order          = T
Goal Direction       = BR
Chain Mode           = DIRECT
Indirect Depth       = NIL
First Hit            = T
Simple Mode          = NIL
Trace                = NIL


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
---(D2-1)-------- Plan-inference with input 2
(GREETING-OPEN SP2 SP1 OPENING (OPEN-DIALOGUE))
(AFFIRMATIVE SP2 SP1 ?TPCD2-1 ?PRPD2-1)
Input order:
  1: (GREETING-OPEN AFFIRMATIVE)
  CHAIN              3    3.285
--------------- Result Number is 3
```

```
---- Prediction at the end of (D2-1) ----
  PREDICTION        3    0.003
 Predicted CATs::
  Next SP2: (INFORM-VALUE CONFIRM-VALUE-UNIT)
  Next SP1: (GREETING-OPEN)

+--DIALOGUE
    |--[D]: GREETING-OPEN-UNIT
    |   |--[E]: (KNOW SP1 (IS SP2 ?NAME-5))
    |   |--[D]: GREETING-OPEN (SP2) はい
    |   +--[D]: (INFORM-VALUE SP2 SP1 ?TPC-5 (IS SP2 ?NAME-5))
    |--[D]: (CONTENTS ?S3-40 ?S4-40 ?TOPIC-40)
    +--[D]: (GREETING-CLOSE-UNIT ?S5-40 ?S6-40 CLOSING)

+--DIALOGUE
    |--[D]: GREETING-OPEN-UNIT
    |   |--[D]: GREETING-OPEN (SP2) はい
    |   +--[D]: (CONFIRM-VALUE-UNIT SP2 SP1 ?NAME-6)
    |--[D]: (CONTENTS ?S3-40 ?S4-40 ?TOPIC-40)
    +--[D]: (GREETING-CLOSE-UNIT ?S5-40 ?S6-40 CLOSING)

+--DIALOGUE
    |--[D]: GREETING-OPEN-UNIT
    |   |--[D]: GREETING-OPEN (SP2) はい
    |   +--[D]: (GREETING-OPEN SP1 SP2 OPENING (OPEN-DIALOGUE))
    |--[D]: (CONTENTS ?S3-40 ?S4-40 ?TOPIC-40)
    +--[D]: (GREETING-CLOSE-UNIT ?S5-40 ?S6-40 CLOSING)

          ---------------- 中略 -------------------

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
---(D2-21)-------- Plan-inference with input 1
(GREETING-CLOSE SP2 SP1 CLOSING (CLOSE-DIALOGUE))
Input order:
  1: (GREETING-CLOSE)
  CHAIN             0    1.119
  UNRELATE          1    0.001
--------------- Result Number is 1

---- Prediction at the end of (D2-21) ----
  PREDICTION        1    0.002
 Predicted CATs::
  Next SP2: NIL
```

```
  Next SP1: (CONFIRMATION)

+--DIALOGUE
    |--[D]: GREETING-OPEN-UNIT
    |   |--[E]: (KNOW SP1 (IS SP2 会議事務局-1))
    |   |--[D]: GREETING-OPEN (SP2) はい
    |   +--[D]: INFORM-VALUE (SP2) こちらは会議事務局です
    |--[D]: CONTENTS
    |   |--[P]: (KNOW SP2 (IS 参加料-1 用件-1))
    |   +--[D]: PAY-FEE
    |       |--[E]: ASK-ACTION-UNIT
    |       |   |--[E]: (KNOW SP1 (支払う-1 SP1 SP2 参加料-1))
    |       |   |--[D]: ASK-ACTION (SP1) 参加料はどのようにお支払いしたらよいのですか
    |       |   |--[D]: INFORM-ACTION (SP2) 案内書に記載されている口座番号に振り込んで
下さい
    |       |   +--[D]: (CONFIRMATION SP1 SP2 参加料-1 (支払う-1 SP1 SP2 参加料-1))
    |       |--[P]: GET-VALUE-UNIT
    |       |   |--[E]: (KNOW SP1 (IS 参加料-1 三万五千-1))
    |       |   |--[D]: ASK-VALUE (SP1) 会議の参加料について教えて頂きたいのですが
    |       |   |--[D]: INFORM-VALUE (SP2) 参加料は現在お一人三万五千円です
    |       |   +--[D]: (CONFIRMATION SP1 SP2 参加料-1 (IS 参加料-1 三万五千-1))
    |       |--[P]: CONFIRM-ACTION-UNIT
    |       |   |--[E]: (KNOW SP1 (DO SP2 割引-1))
    |       |   |--[D]: CONFIRM-ACTION (SP1) 参加料の割引はないのですか
    |       |   |--[D]: NEGATIVE-U
    |       |   |   +--[D]: NEGATIVE (SP2) 今回は割引を行なっておりません
    |       |   +--[D]: CONFIRMATION (SP1) そうですか
    |       |--[P]: GET-VALUE-UNIT
    |       |   |--[E]: (KNOW SP1 (IS 期限-1 今年-1))
    |       |   |--[D]: (ASK-VALUE SP1 SP2 期限-1 (IS 期限-1 今年-1))
    |       |   |--[D]: INFORM-VALUE (SP2) また期限は今年いっぱいです
    |       |   +--[D]: CONFIRMATION (SP1) 分かりました
    |       +--[D]: (INTRODUCE-ACTION SP1 SP2 ?ACT-38 (PAY SP1 SP2 参加料-1))
    +--[D]: GREETING-CLOSE-UNIT
        |--[D]: GREETING-CLOSE (SP1) どうもありがとうございました
        +--[D]: GREETING-CLOSE (SP2) どういたしまして

+--DIALOGUE
    |--[D]: GREETING-OPEN-UNIT
    |   |--[E]: (KNOW SP1 (IS SP2 会議事務局-1))
    |   |--[D]: GREETING-OPEN (SP2) はい
    |   +--[D]: INFORM-VALUE (SP2) こちらは会議事務局です
    |--[D]: CONTENTS
    |   |--[P]: (KNOW SP2 (IS 参加料-1 用件-1))
```

```
    |   +--[D]: PAY-FEE
    |       |--[E]: ASK-ACTION-UNIT
    |       |   |--[E]: (KNOW SP1 (支払う -1 SP1 SP2 参加料 -1))
    |       |   |--[D]: ASK-ACTION (SP1) 参加料はどのようにお支払いしたらよいのですか
    |       |   |--[D]: INFORM-ACTION (SP2) 案内書に記載されている口座番号に振り込んで
下さい
    |       |   +--[D]: (CONFIRMATION SP1 SP2 参加料 -1 (支払う -1 SP1 SP2 参加料 -1))
    |       |--[P]: GET-VALUE-UNIT
    |       |   |--[E]: (KNOW SP1 (IS 参加料 -1 三万五千 -1))
    |       |   |--[D]: ASK-VALUE (SP1) 会議の参加料について教えて頂きたいのですが
    |       |   |--[D]: INFORM-VALUE (SP2) 参加料は現在お一人三万五千円です
    |       |   +--[D]: (CONFIRMATION SP1 SP2 参加料 -1 (IS 参加料 -1 三万五千 -1))
    |       |--[P]: CONFIRM-ACTION-UNIT
    |       |   |--[E]: (KNOW SP1 (DO SP2 割引 -1))
    |       |   |--[D]: CONFIRM-ACTION (SP1) 参加料の割引はないのですか
    |       |   |--[D]: NEGATIVE-U
    |       |   |   +--[D]: NEGATIVE (SP2) 今回は割引を行なっておりません
    |       |   +--[D]: CONFIRMATION (SP1) そうですか
    |       |--[P]: GET-VALUE-UNIT
    |       |   |--[E]: (KNOW SP1 (IS 期限 -1 今年 -1))
    |       |   |--[D]: (ASK-VALUE SP1 SP2 期限 -1 (IS 期限 -1 今年 -1))
    |       |   |--[D]: INFORM-VALUE (SP2) また期限は今年いっぱいです
    |       |   +--[D]: CONFIRMATION (SP1) 分かりました
    |       +--[D]: (INTRODUCE-ACTION SP1 SP2 ?ACT-38 (PAY SP1 SP2 参加料 -1))
    +--[D]: GREETING-CLOSE-UNIT
        |--[D]: GREETING-CLOSE (SP1) どうもありがとうございました
        +--[D]: GREETING-CLOSE (SP2) どういたしまして
```

```
;;; EVALUATIONs ;;;
 ID      & CAT   & SUC   & GS    & PLAN  & PRED  & UNIF   & NODE  & \\
 D2-1    & 2     & 1     & 3     & 1     & 1     & 1158   & 7     & \\
 D2-2    & 1     & 1     & 1     & 1     & 0     & 198    & 3     & \\
 D2-3    & 1     & 1     & 2     & 2     & 3     & 579    & 7     & \\
 D2-4    & 1     & 0     & 2     & 2     & 3     & 772    & 7     & \\
 D2-5    & 2     & 0     & 2     & 2     & 3     & 456    & 2     & \\
 D2-6    & 1     & 1     & 2     & 2     & 3     & 183    & 2     & \\
 D2-7    & 1     & 0     & 2     & 2     & 3     & 591    & 2     & \\
 D2-8    & 2     & 0     & 2     & 2     & 3     & 1544   & 3     & \\
 D2-9    & 1     & 0     & 2     & 2     & 3     & 629    & 2     & \\
 D2-10   & 3     & 1     & 6     & 3     & 3     & 1983   & 15    & \\
 D2-11   & 2     & 1     & 2     & 3     & 7/2   & 572    & 2     & \\
 D2-12   & 1     & 1     & 1     & 3     & 2     & 220    & 9     & \\
 D2-13   & 1     & 1     & 1     & 4     & 3     & 354    & 2     & \\
```

```
 D2-14  & 2     & 0     & 1     & 4     & 3     & 1235  & 4     & \\
 D2-15  & 2     & 1     & 1     & 4     & 3     & 156   & 2     & \\
 D2-16  & 1     & 1     & 1     & 5     & 3     & 563   & 3     & \\
 D2-17  & 2     & 1     & 1     & 5     & 2     & 218   & 9     & \\
 D2-18  & 2     & 1     & 2     & 6     & 3     & 746   & 4     & \\
 D2-19  & 1     & 1     & 1     & 6     & 2     & 156   & 2     & \\
 D2-20  & 2     & 0     & 1     & 6     & 2     & 1241  & 6     & \\
 D2-21  & 1     & 0     & 1     & 6     & 2     & 442   & 2     & \\
 21     & 32    & 13    & 37    & 71    & 107/2 & 13996 & 95    & \\
T
>
```

3.5 逐次的に対話理解、次発話の予測を行なう手順

入力ファイルの文に対して逐次的に、1）対話理解を行ない、2）次発話の予測を行なう。

1. 指定されたファイルの入力命題をロードする。

```
> (gp-load-file "kaiwa-3.data")
16
>
```

2. 対話構造解析プログラムを初期化する。(目標構造・入力履歴の初期化を行なう。)

```
> (gp-reset)
NIL
>
```

3. プランニングの制御モード (表 2.2参照) を指定する。

```
> (simple-mode-on)
;;; Loading source file "../PlanRec/Data/simple.plans"
;;; Warning: File "../PlanRec/Data/simple.plans" does not begin with
IN-PACKAGE.  Loading into package "GPLANNER"

--- Control Modes of 0 ---
Inference Rules      = (DECOMPOSITION-CHAIN EFFECT-CHAIN PRECONDITION-CHAIN)
Schema Typee         = (INTERACTION-PLAN COMMUNICATION-PLAN DOMAIN-PLAN DIALOGUE-PLAN)
Layer Mode           = T
Input Order          = T
Goal Direction       = BR
Chain Mode           = DIRECT
Indirect Depth       = NIL
```

```
First Hit              = T
Simple Mode            = T
Trace                  = NIL
NIL
>
```

4. 指定された入力の ID に対して、1）プラン認識を行ない、2）次発話の予測を行なう。

```
> (gp-next 'd3-1)
---(D3-1)-------- Plan-inference with input 2
(GREETING-OPEN SP2 SP1 OPENING (OPEN-DIALOGUE))
(AFFIRMATIVE SP2 SP1 ?TPCD3-1 ?PRPD3-1)
Input order:
  1: (GREETING-OPEN)
  2: (AFFIRMATIVE)
  CHAIN              3    0.507
--------------- Result Number is 3

---- Prediction at the end of (D3-1) ----
  PREDICTION         3    0.003
 Predicted CATs::
  Next SP2: (INFORM-VALUE CONFIRM-VALUE-UNIT)
  Next SP1: (GREETING-OPEN)
(D3-1 2 1 3 1 1 442 3)
>
```

5. 現在の理解状態の概観を表示する。

```
> (tprint-gsl)
+--DIALOGUE
    |--[D]: GREETING-OPEN-UNIT
    |   |--[E]: (KNOW SP1 (IS SP2 ?NAME-2))
    |   |--[D]: GREETING-OPEN (SP2) はい
    |   +--[D]: (INFORM-VALUE SP2 SP1 ?TPC-2 (IS SP2 ?NAME-2))
    |--[D]: (CONTENTS ?S3-360 ?S4-360 ?TOPIC-360)
    +--[D]: (GREETING-CLOSE-UNIT ?S5-360 ?S6-360 CLOSING)

+--DIALOGUE
    |--[D]: GREETING-OPEN-UNIT
    |   |--[D]: GREETING-OPEN (SP2) はい
    |   +--[D]: (CONFIRM-VALUE-UNIT SP2 SP1 ?NAME-3)
    |--[D]: (CONTENTS ?S3-360 ?S4-360 ?TOPIC-360)
    +--[D]: (GREETING-CLOSE-UNIT ?S5-360 ?S6-360 CLOSING)
```

```
+--DIALOGUE
    |--[D]: GREETING-OPEN-UNIT
    |   |--[D]: GREETING-OPEN (SP2) はい
    |   +--[D]: (GREETING-OPEN SP1 SP2 OPENING (OPEN-DIALOGUE))
    |--[D]: (CONTENTS ?S3-360 ?S4-360 ?TOPIC-360)
    +--[D]: (GREETING-CLOSE-UNIT ?S5-360 ?S6-360 CLOSING)
(NIL NIL NIL)
>
```

3.6 その他の主な操作

知識ベースのロード

- プランスキーマをロードする。
  それまでのプランスキーマはクリアされる。

```
> (load-schemata)
;;; Loading source file "../PlanRec/Data/test.plans"
;;; Warning: File "../PlanRec/Data/test.plans" does not begin with
IN-PACKAGE.  Loading into package "GPLANNER"
;;; Loading source file "../PlanRec/Data/simple.plans"
;;; Warning: File "../PlanRec/Data/simple.plans" does not begin with
IN-PACKAGE.  Loading into package "GPLANNER"
#P"/home/user/PlanRec/Data/test.plans"
>
```

- 概念知識ベースファイルをロードして、概念ネットワークを作る

```
> (np::load-concept-network)

;;; Loading source file "../PlanRec/Data/concept-nodes.lisp"
;;; Warning: File "../PlanRec/Data/concept-nodes.lisp" does not begin
with IN-PACKAGE.  Loading into package "GPLANNER"
472
>
```

知識ベースの表示

- プランスキーマテーブルの内容全体を表示する.

```
> (print-schemata)
:DOMAIN-PLAN
  PRESENT-PAPER          :(G1798)
  MAKE-REGISTRATION      :(G1797)
```

```
    PAY-FEE                 :(G1796)
    JOIN-EVENT              :(G1795)
    SEND-SOMETHING          :(G1794)
    WRITE-SOMETHING         :(G1793)

  :COMMUNICATION-PLAN
    EXPLAIN-STATEMENT       :(G1792)
    EXECUTE-ACTION          :(G1791 G1790)
    INTRODUCE-ACTION        :(G1789)

  :INTERACTION-PLAN
    INFORM-WANT-UNIT        :(G1788)
    GET-VALUE-UNIT          :(G1787)
    CONFIRM-VALUE-UNIT      :(G1786 G1785 G1784)
    AFFIRMATIVE-U           :(G1783 G1782)
    NEGATIVE-U              :(G1781 G1780)
    REQUEST-ACTION-UNIT     :(G1779 G1778)
    OFFER-ACTION-UNIT       :(G1777 G1776)
    ASK-ACTION-UNIT         :(G1775)
    CONFIRM-ACTION-UNIT     :(G1774 G1773 G1772)
    ASK-STATEMENT-UNIT      :(G1771)
    CONFIRM-STATEMENT-UNIT :(G1770 G1769 G1768)
    GREETING-CLOSE-UNIT     :(G1767 G1766)
    GREETING-OPEN-UNIT      :(G1765 G1764 G1763)

  :DIALOGUE-PLAN
    CONTENTS                :(G1762 G1761 G1760)
  NIL
  >
```

- 概念ネットワークの全てのノードを端末出力へ木構造で表示する。

```
> (np::pprint-all-nodes :start :top :type-list '(is-a))

+--(お願い)

+--(もの)

+--(こと)

+--(他)
```

---------- 中略 ----------

```
+--(ASK-PRAG)

+--(CAT)
    |--IS-A(DEMAND)
    |   |--IS-A(ASK)
    |   |   |--IS-A(ASK-ACTION)
    |   |   |--IS-A(ASK-VALUE)
    |   |   +--IS-A(ASK-STATEMENT)
    |   |--IS-A(REQUEST)
    |   |--IS-A(CONFIRM)
    |   |   |--IS-A(CONFIRM-ACTION)
    |   |   |--IS-A(CONFIRM-VALUE)
    |   |   +--IS-A(CONFIRM-STATEMENT)
    |   |--IS-A(REQUEST-ACTION)
    |   |--IS-A(OFFER-ACTION)
    |   |--IS-A(GREETING-OPEN)
    |   +--IS-A(GREETING-CLOSE)
    |--IS-A(RESPONSE)
    |   |--IS-A(INFORM)
    |   |   |--IS-A(INFORM-ACTION)
    |   |   |--IS-A(INFORM-VALUE)
    |   |   |--IS-A(INFORM-STATEMENT)
    |   |   |--IS-A(INFORM-WANT)
    |   |   |--IS-A(AFFIRMATIVE)
    |   |   |   +--IS-A(AFFIRMATIVE-U)
    |   |   +--IS-A(NEGATIVE)
    |   |       +--IS-A(NEGATIVE-U)
    |   |--IS-A(ACCEPT-ACTION)
    |   |--IS-A(REJECT-ACTION)
    |   |--IS-A(ACCEPT-OFFER)
    |   |--IS-A(REJECT-OFFER)
    |   |--IS-A(GREETING-OPEN)
    |   +--IS-A(GREETING-CLOSE)
    +--IS-A(ACKNOWLEDGE)
        +--IS-A(CONFIRMATION)
NIL
>
```

プランニング処理

- プランニングの制御モードを表示する。

```
> (display-mode)
--- Control Modes of D3-1 ---
```

```
Inference Rules      = (DECOMPOSITION-CHAIN EFFECT-CHAIN PRECONDITION-CHAIN)
Schema Typee         = (INTERACTION-PLAN COMMUNICATION-PLAN DOMAIN-PLAN DIALOGUE-PLAN)
Layer Mode           = T
Input Order          = T
Goal Direction       = BR
Search Mode          = BR
Chain Mode           = DIRECT
Indirect Depth       = NIL
First Hit            = T
Simple Mode          = T
Trace                = NIL
NIL
>
```

- *ID*(2.5節参照) で指定された入力を処理した時点の状態に戻す。

```
> (reset-gplanner 'd3-5)
D3-5
>
```

- 処理結果のサマリーを表示する。

```
> (print-gp-eval-all)

;;; EVALUATIONs ;;;
 ID     & CAT   & SUC   & GS    & PLAN  & PRED  & UNIF  & NODE  & \\
 D3-1   & 2     & 1     & 3     & 1     & 1     & 442   & 3     & \\
 1      & 2     & 1     & 3     & 1     & 1     & 442   & 3     & \\
NIL
>
```

- ロードされた入力命題の文 ID のリストを返す。

```
> (gp-input-id-list)
(D3-1 D3-2 D3-3 D3-4 D3-5 D3-6 D3-7 D3-8 D3-9 D3-10 D3-11 D3-12 D3-13
D3-14 D3-15 D3-16)
>
```

引数: なし
リターン値: 最近に、処理を行なった入力発話の ID

最近に、処理を行なった入力発話の ID を返す。

---

gp-reset *&optional id* [*Function*]

引数:

1. id (&optional)

リターン値: id

*ID*で指定された入力を処理した時点の状態に戻す。

---

gp-test-list *list &optional tree* [*Function*]

引数:

1. list
2. tree (&optional)

リターン値: nil

入力発話 ID のリスト (list) の入力発話の対話理解、次発話の予測、音声入力候補の絞込をする。
tree が nil でないなら、ID に対する対話構造を表示する。

---

gp-test-file *file &key tree* [*Function*]

引数:

1. tree (&key)
2. file (&key)

リターン値: t

指定されたファイルの入力発話をロードし、そのすべての入力発話に対して、入力発話の対話理解、次発話の予測、音声入力候補の絞込をする。

simple-mode-on *&optional (control *default-control*)* [*Function*]

引数:

1. (control *default-control*) (&optional)

リターン値: nil

プラン認識の制御モードをシンプルモードオンする。(LAYLA の関数に、シンプルモード用のプランスキーマ (*simple-plan-filename*) をロードするように、パッチをあてている。)

simple-mode-off *&optional (control *default-control*)* [*Function*]

引数:

1. (control *default-control*) (&optional)

リターン値: nil

プラン認識の制御モードをシンプルモードオフにする。(LAYLA の関数に、デフォルト (*schemata-filename*) のプランスキーマをロードするように、パッチをあてている。)

4.2 データのロード (AnaDialog/load-data.lisp)

パッケージは user。

4.2.1 大域変数

simple-plan-filename [*Variable*]

初期値: "simple.plans"

プラン認識の制御モードがシンプルモードのとき、使用するプランスキーマ。

4.3 処理結果の評価 (AnaDialog/gp-evaluate.lisp)

パッケージは gplanner。

4.3.1 大域変数

---

gp-eval [*Variable*]

初期値: nil

処理結果のリストを蓄える大域変数。処理結果は次の項目のリストである。

```
ID   : 発話 ID
CAT  : 情報伝達行為解析の結果出力された発話の表現の数
SUC  : プラン認識により認識された発話の表現数
GS   : その発話の認識終了時の対話構造数
PLAN: 利用されたインタラクションプランの累積 (発話対数)
PRED: 予測情報伝達行為タイプの数の平均 (次発話の話者について)
PLEV: 予測の最大レベル
CAND: 音声認識候補の数 (入力)
SELE: 選択候補数
UNIF: 単一化の回数
NODE: プラン認識時に作ったノードの数
```

4.3.2 関数

---

gp-eval-data *id time &optional (gsl *goal-stack-list*)* [*Function*]

引数:

1. id
2. time
3. (gsl *goal-stack-list*) (&optional)

リターン値: *gp-eval*

id で指定された文の処理結果を *gp-eval* にセットする。

---

gp-eval-data-list *id time &optional (gsl *goal-stack-list*)* [*Function*]

引数:

1. id
2. time
3. (gsl *goal-stack-list*) (&optional)

リターン値: id で指定された文の処理結果。

id で指定された文の処理結果を返す。

---

gp-eval-succeed-cats *id &optional (gsl *goal-stack-list*)* [*Function*]

引数:

1. id
2. (gsl *goal-stack-list*) (&optional)

リターン値: プラン認識により認識された発話の表現数

プラン認識により認識された発話の表現数を返す。

---

print-gp-eval-all *&optional (str t)* [*Function*]

引数:

1. (str t) (&optional)

リターン値: nil すべての文の処理結果を表示する

---

print-gp-eval *list &optional (str t)* [*Function*]

引数:

1. list
2. (str t) (&optional)

リターン値: nil

1文の処理結果を表示する

---

gp-eval-sumup *lists* [*Function*]

引数:

1. lists

リターン値: すべての文の処理結果の合計

すべての文の処理結果の合計を返す

4.4 次発話の予測 (AnaDialog/Prediction/prediction.lisp)

パッケージは prediction。

4.4.1 大域変数

---

trace [*Variable*]

初期値: nil

この値が nil 以外であれば、次発話の予測システムのトレースを取る。

---

trace-stream [*Variable*]

初期値: *monitor-stream*

関数 with-trace で表示するストリーム

---

prediction [*Structure*]

- utterance : 初期値: nil
- type : 初期値: nil

- action : 初期値: nil
- gs : 初期値: nil
- speaker : 初期値: nil
- level : 初期値: nil
- info : 初期値: nil

予測の内部データ構造 (予測構造体).

---

with-trace *return string &rest rest* [*Macro*]

引数:

1. return
2. string
3. rest (&rest)

リターン値: return

関数 trace-on で予測のトレースを表示するようにしておけば、rest のリターン値を string の形式でストリーム *trace-stream* に表示する。

4.4.2 関数

---

print-prediction *pred stream depth* [*Function*]

引数:

1. pred
2. stream
3. depth

リターン値: nil

予測構造体のプリント関数

---

prediction-cat *prediction* [*Function*]

引数:

1. prediction

リターン値:

予測構造体からその情報伝達行為を取り出す。

---

trace-on [*Function*]

引数: なし
リターン値: t

予測システムのトレースを表示するようにする。

---

trace-off [*Function*]

引数: なし
リターン値: nil

予測システムのトレースを表示しないようにする。

---

prediction-set *gsl &optional (control *default-control*)* [*Function*]

引数:

1. gsl
2. (control *default-control*) (&optional)

リターン値: gsl

対話構造のリスト gsl を入力にして、予測をし、その結果 (関数 prediction のリターン値) を gsl にセットする。

---

prediction *gs &optional (control *default-control*)* [*Function*]

引数:

1. gs

2. (control *default-control*) (&optional)

リターン値: 予測

```
(list (list 話者1 予測構造体... 予測構造体)
      (list 話者2 予測構造体... 予測構造体))
```

対話構造 gs を入力にして、予測をし、その結果を返す

---

prediction-alist-by-sp *list* [*Function*]

引数:

1. list

リターン値: 話者と予測構造体の A リストのリスト

予測構造体のリストを話者により、分類し、話者と予測構造体の A リストにする。

---

prediction-main *goals gs &aux (level 1)* [*Function*]

引数:

1. goals
2. gs
3. (level 1) (&aux)

リターン値: 予測構造体

未充足プランのリスト goals を充足させるための入力を予測する。

---

interaction-plan-p *goal* [*Function*]

引数:

1. goal

リターン値:

t(goal がインターラクションプランである)
又は
nil(goal がインターラクションプランである)

goal がインターラクションプランかどうかを調べる。

---

domain-plan-p *goal* [*Function*]

引数:

1. goal

リターン値:

t(goal がドメインプランである)
又は
nil(goal がドメインプランである)

goal がドメインプランかどうかを調べる。

---

complete-goal-p *goal* [*Function*]

引数:

1. goal

リターン値:

t(goal が充足プランである)
又は
nil(goal が充足プランである)

goal が充足プランかどうかを調べる。

---

predict-from-decomposition *action gs level* [*Function*]

引数:

1. action
2. gs
3. level

リターン値: 予測構造体

decomposition で連鎖する入力を予測する

---

predict-from-precondition *action gs level* [*Function*]

引数:

1. action
2. gs
3. level

リターン値: 予測構造体

precondition で連鎖する入力を予測する

---

prediction-1 *prp rest action gs type level* [*Function*]

引数:

1. prp
2. rest
3. action
4. gs
5. type
6. level

リターン値: 予測構造体

命題 prp で連鎖する入力を予測する

---

predictable-proposition-p *prp rest* [*Function*]

引数:

1. prp
2. rest

リターン値:

t(命題 prp で連鎖する)
又は
nil(命題 prp で連鎖しない)

命題 prp で連鎖するかどうかを調べる。

---

chained-cdr *list* [*Function*]

引数:

1. list

リターン値:

t(命題 prp で連鎖する)
又は
nil(命題 prp で連鎖しない)

list の cdr 部が連鎖するかどうかを調べる。

---

collect-predicted-cats *gsl sp* [*Function*]

引数:

1. gsl
2. sp

リターン値: 情報伝達行為のリスト

対話構造のリスト gsl から話者 sp に対して予測される入力の情報伝達行為のリストを返す。

---

gs-predicted-cat-list *gs sp* [*Function*]

引数:

1. gs
2. sp

リターン値: 情報伝達行為のリスト

対話構造 gs から話者 sp に対して予測される入力の情報伝達行為のリストを返す。

---

gs-get-predictions-by-sp *gs sp* [*Function*]

引数:

1. gs
2. sp

リターン値: 予測構造体のリスト

対話構造 gs から話者 sp に対して予測される予測構造体のリストを返す。

---

transmatrix *list* [*Function*]

引数:

1. list

リターン値: list を転置したリスト

行列を転置する。
例:

```
(prediction::transmatrix '((a b x)(c d)(d e)))
((A C D) (B D E) (X))
```

付録 A

情報伝達行為解析システム (CATE)

A.1 システム概要

A.1.1 機能

情報伝達行為解析システム (CATE:Communicative Act Type Extractor) は、対話構造解析プログラムを自動翻訳電話システムにおける文脈処理機構として組み込む際の、文解析 – 対話構造解析インタフェースである。

つまり、CATE は日本語解析システム NADINE[16] の出力である素性構造意味表現 (sem 素性) 記述 (prag 素性: 語用論素性を含む) のみから、発話の解釈として可能な限りの発話の表現を抽出するものである。

A.1.2 稼働環境

表 A.1: CATE 稼働環境

マシン	Sun
使用言語	Common Lisp

なお、本システムを利用するには推論エンジンの素性構造書換えシステム [22] が必要である。

A.1.3 システム構成

図 A.1に、CATE の基本的なモジュール構成を示す。

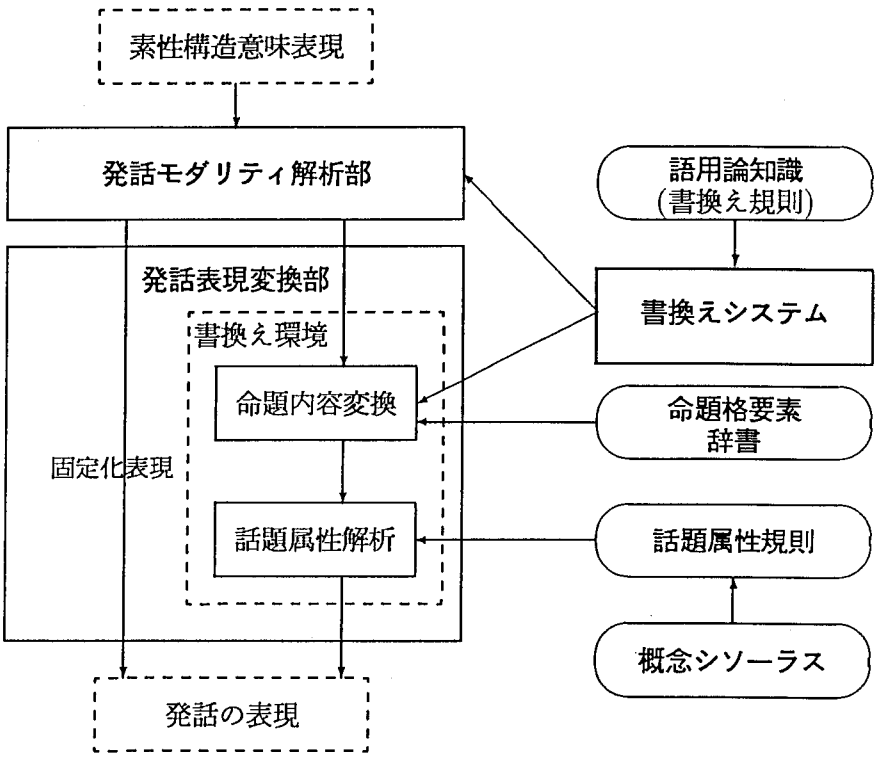

図 A.1: CATE のシステム構成

処理の説明

1. 発話モダリティ解析部 (素性構造書き換え処理)
   入力素性構造意味表現を、書換え規則の適用により書換え、発話行為のタイプあるいは情報伝達行為タイプを含む素性構造 (中間素性構造 (A.1.5.3) 参照) を出力する。

2. 発話表現変換部
   中間素性構造を発話の表現のデータ構造 (リスト) に変換する。

A.1.4 ファイル構成

情報伝達行為解析システムのファイル構成とその内容を表 A.2と表 A.3に示す。

表 A.2: 情報伝達行為解析システムのファイル構成

内容	ファイル名
プログラムソース	''AnaDialog/CATE/''
システムロード用	load.lisp
発話モダリティ変換部	cate-main.lisp
発話表現変換部	output.lisp
話題属性パラメータ設定	parameters.lisp
素性構造操作用	fs.lisp
実験用インタフェース	examine.lisp

表 A.3: 情報伝達行為解析システムのサンプルデータファイル構成

内容	ファイル名
語用論知識 (書換え規則)	''AnaDialog/Rules/''
固定化表現	frozen.lisp
文末表現	intention.lisp
命題前処理	proposition-ignore.lisp
トピック抽出	topic.lisp
オブジェクト変換	object-general.lisp
オブジェクト疑問詞	object-wh.lisp
前処理	preedit.lisp
後処理	postedit.lisp
話題属性規則	output-rule.lisp
その他の知識ベース 1	''AnaDialog/Data/''
スピーカテーブル	speaker.list
その他の知識ベース 2	''PlanRec/Data/''
命題格要素辞書	case-grammar.lisp
概念ネットワーク辞書	concept-node.lisp
入力データ	''AnaDialog/CATE_Data/''
話題属性・概念集合設定	parameter-set.lisp
素生構造 (サンプルデータ)	sem-*.fs

A.1.5 CATE で扱うデータ

A.1.5.1 入力データ

素性構造 CATE が扱う素性構造は、素性構造書換えシステムで定義された素性構造である。素性構造の内部構造については、参考文献 [22] を参照。

素性構造中の任意の素性パタンは、そのタイプにより以下の意味を持たせる:

:atomic	何らかの概念を示すシンボル (symbol)
:leaf	任意の構造と単一化可能な変項 (variable: 次節参照)
:variable	:leaf に同じ
:complex	上記 2 項の組合せによる構造体

CATE の処理では、入力として、文解析 NADINE から出力される素性構造の意味表現 (sem 素性) および語用論素性 (prag 素性) のみを扱う。

A.1.5.2 知識ベース

CATE が利用する知識・データは以下のようになる。

これらの知識・データは、ユーザや対象問題によりカスタマイズできることが望ましい。つまり、CATE を生かすも殺すもこれらの知識やデータの記述にかかってくる。各々の知識・データについて、現在どのようなものを準備しているか、またどのようにカスタマイズするかは、第 A.1.8.1章で詳説する。

語用論知識 (発話モダリティに関する語用論知識) 語用論知識は、発話モダリティと情報伝達行為タイプを対応づける知識 である。発話表現のモダリティ部分は、比較的反射的な表現が行なわれていると考えられる。つまり、命題内容のように、構成要素の配列や格助詞の利用といった構成的な知識が用いられず、話し手が持っているある意図に対応した表現を直観的に引き出しているということになる。もちろん文法的可否により判断される部分は多くあろうが、日本語の文末表現のように補助動詞・助動詞・接続助詞さらには活用変化といった複雑な構造と組合せを、真面目に解析することは効率的ではない。従って、CATE では、このような対応知識を 網羅的に書き下す という方針をとる。

語用論知識は、書換え規則のフォーマリズムで記述 する。従って、情報伝達行為タイプの特定は、それに対応する表現パターンと入力との書換えシステムで定義されたパターンマッチングによりなされる。この処理は、決定的な処理ではない。従って、複数のパターンにマッチすれば、潜在的に複数の情報伝達行為を持っていると解釈し、そのすべてを出力する。[1]

[1]現在の素性構造書換えシステムでは、マッチングの競合において、その規則記述中の素性パスの長い規則が優先されるようである。それはそれで、よい。

話題属性規則 (話題の属性による変換規則) 話題の属性による規則は、IF-THEN の形で記述する。前節までで述べたように、CATE での情報伝達行為抽出では、話題の属性を考慮に入れるが、

1. 対話に現われる述語や概念をすべて書換え規則として設定するのが面倒であること、

2. たとえそれらを設定したとしても多量の規則の元で動作するため処理効率が低下すること、

3. 命題の格要素に関するデータベースや概念のネットワークが対話構造解析機構と共有できること、

といった理由から、書換え規則による記述は行なわない。(以下の 2 つのデータベースについてもこれらの理由から特有のものを利用するに至った。) もちろん、特定の環境下での規則適用という考えは、継承する。すなわち、この規則は独立して成立するものではなく、発話モダリティ解析の結果に従った記述になる。

基本的に、語や語の意味に関する分析は、文解析の仕事である。解析や変換部の辞書から自動的に命題内容や話題の属性に関する書換え規則を設定する道具を作ることができれば統一性の高いシステム実現が可能である。しかし、ここではそこまで行なわない。[2]

命題格要素辞書 命題格要素データは、単純に述語とその格要素のラベルのリストで表現する。これらは、命題内容の表現を行なう際に必要な構成素を取り出してくる時に利用する。[3]

概念ネットワーク辞書 (概念シソーラス) 概念シソーラスは、述語 (動詞) や格要素・話題 (名詞句) となる概念の階層関係を表した知識である。これは、ネットワークで表現する [14]。

概念の階層関係は、具体的には、命題内容や話題の属性を決定する時に利用する。例えば、情報交換自体の行為を意味する述語、「教える」などは“体面動作”という概念の下位概念であり、この概念に属する命題内容は、その目的格の属性によって、情報伝達行為タイプに変化を与える。例えば、「お名前を教えて下さい」は質問になるが、「英語を教えて下さい」などは (行為) 依頼になる。

概念シソーラスのより詳しい説明と利用は、[21] に譲る。

---

[2]旧 CATE では、これらはすべて書換え規則として記述されていた。

[3]実は、このような規定を行なう最大の理由は、対話構造解析のプラン認識におけるマッチングメカニズム (単一化) がグラフ構造を扱えないことである (term unify である)。つまり、同じ命題内容は、同じ構造として入力されてくることを期待している。今後改良すべきである。

A.1.5.3 中間データ

中間素生構造 第A.1.3節に示すように、CATEの処理には2つのフェイズがあり、その間で素性構造の受渡しが行なわれる。この時点での素性構造(以後中間素性構造と呼ぶ)は、入力意味表現を抽象化したものになっている。従って、ここで、そのデータを表現する際に新しい素性を導入する。

CATEで新しく導入する素性の素性名は、3文字のアルファベットで表す。(例外もある)表A.4に、その中で重要なものの一覧を示す。なお、表中データタイプのキーワードは、素性構造書換えシステムの素性構造のタイプに対応する。また、variableは:leaf/:variableを、– はいずれのタイプでも良いことを表す。

表A.4: CATE固有の素性

素性名	データタイプ	内容
CAT	:atomic	情報伝達行為タイプ／発話行為のタイプ
TPC	:atomic/variable	トピック
PRP	–	命題内容の素性構造
VAL	:atomic	(複合)概念の主辞
MOD	:atomic	発話モダリティのエキストラ情報 (例えば、否定(nagate)・欲求(want)等)

- CATには、第A.1.7.2節で定義した情報伝達行為タイプが入る。文解析の失敗などで、入力素性構造の発話モダリティ解析ができない時(すなわち発話行為のタイプが求められない時)は、:unknownにする。[4]

- TPCには、トピック(A.1.5.4節)があれば、その主辞の概念を表すシンボルが入る。でなければ、変項である。

- PRPには、発話の命題内容部分の素性構造が入る。これはしばしば、名詞句を表現しているものになることがある。そのような時は:atomicタイプになる。命題内容部分は存在しない時は、変項である。

- VALは、主に名詞句の主辞を表すことに用いる。素性構造意味表現における名詞句の表現は、様々な情報を包含し表す形になっているので、複雑である。とりあえず、対話構造解析への情報としては、名詞句の中心的な意味のみで十分であると考え、その主辞の情報のみを受け渡すことにする。

  この素性には、具体的には、"登録用紙-1"といった概念(を表現している記号:RELN素性, 関係名)が入り、素性構造意味表現でいうENTITYの値ではない。

  もちろん、発話の厳密な解釈を行なうには、これだけの情報では物足りない場合の方が多い。付加語や修飾語の情報の取り扱いは、今後の課題である。

[4]:unknownは変項ではない。

- MOD には、いわゆる発話のボイス・ムードといった情報を入れる。これらは、通常発話モダリティ解析のための情報として利用されるべきであるが、現在の CATE が、モダリティ部分を完全に解析できる能力を備えていないために、このような素性を持っておき、後々利用できるようにする。この素性は以下のような場合、中間素性構造に付加される:
  - 発話モダリティ解析において、そのムードに顕著な特徴がある時;
    例えば、 A.1.7.2節で触れた INFORM-STATMENT の欲求:WANT や、否定:NEGATE など。
  - 発話モダリティ解析の対象に入らず、かつ命題内容部分でないボイスなど:
    これらは、 A.1.6で触れるが、命題内容変換の前処理の際に抽出する。例えば、可能:POSSIBLE, 受益:PASSIVE など。

これらの素性構造は、入力素性構造の発話表現への変換への中間構造として、情報の受渡しをするのみならず、話題の属性による解析や (我々の) 発話行為の解釈モデルに含まれない情報 (表の MOD 素性など) を利用した情報伝達行為タイプの絞り込みに対しての、情報提供を行なっている。逆にいえば、発話表現変換部 (A.1.3節) では、単にデータの変換を行なうに留まらず、このような情報を利用してより的確な情報伝達行為・発話の表現を生成していることになる。

A.1.5.4 出力データ

発話の表現 CATE の出力である発話の表現は、階層型プラン認識モデルの入力であり、以下の構造で表現される:

> 発話の表現 ::= (情報伝達行為タイプ 話し手 聞き手 トピック 命題内容)

- トピック
  トピックは、発話表現中で「は」相当格によってマークされた概念である。[5]

  トピックは、対話構造解析において話題領域に関する制約として利用できる。具体的には、ドメインプラン探索の探索空間の制限を可能とする [?]。また、情報伝達行為解析においては、話題の属性の決定に貢献する。[6]例えば、「締切は、3月15日です。」の「締切」は具体的値を持つオブジェクトについての話題であるが、「支払は、銀行振込です。」の「支払」はそれよりも行為自体を話題としている。従って、後者は、単なる情報提示と言うよりも、より行為依頼に近い情報伝達行為を持つと認定できる。

[5]階層型プラン認識モデルでは、「は」相当格に該当するものがない場合、発話の先頭の格要素を取るという定義が為されているが、現在の意味表現からこれを行なうことは困難である。

[6]現在の CATE では、このトピックとさきに述べた話題の属性における話題の間に強い相関を持たせていない。(強い相関を持たせることは危険である。)

トピックの導出及びトピックと話題の属性の厳密な相関モデルは、今後の課題である。

- 話し手・聞き手
  話し手及び聞き手は、言語外情報であり、システム全体の中で何らかの形で得られると仮定する (CATE では、スピーカテーブルで与えている)。

上述のように、一つの発話の表現は、一本のリストで表す。このリストは、発話の一つの解釈に相当する。従って、一つの発話・一つの素性構造意味表現から、ただ一つのリストが導かれるわけではない。発話の表現のリストの構造を図 A.5に示す。(発話文字列は、オプショナルである。)

表 A.5: 発話の表現の構造

要素	データタイプ	内容
1 情報伝達行為タイプ (cat)	symbol	第 A.1.7.2節で定義したもの
2 話し手 (sp)	symbol	話し手を表す任意のシンボル
3 聞き手 (hr)	symbol	聞き手を表す任意のシンボル
4 トピック (tpc)	symbol/variable	上述参照
5 命題内容 (prp)	list/variable	下記参照
(6 発話文字列 (str)	string	発話文の文字列

発話の表現のリストには、情報伝達行為タイプ・話し手／聞き手・概念を表すシンボルの他に、以下のようなデータが入りえる:

- 変項
  変項 (variable) は、対話構造解析において任意の構造と単一化可能なデータ構造である。CATE では、入力素性構造において以下の条件を満たすものを変項として扱う:
  - 素性値が :leaf/:variable タイプの素性構造
  - 疑問詞相当句が主辞である格要素

  以下具体的変項の記述は、接頭辞 "?" を付して表す。

- 命題内容リスト

  命題内容は、一本のリストで表す。命題内容のリストは、命題格要素データの記述にしたがった長さ (1 以上) を持つ。命題内容の先頭要素は、その述語を表すシンボルである。その他の要素 (cdr) は、その格要素の並びである。格要素の並びの順序は、命題格要素データにより規定されている。(命題格要素データの構造については、第 A.1.8.3節参照) 任意の格要素は変項になり得る。

  命題内容リストは、以下のような場合、それ自体変項になり得る:

- 発話モダリティ解析の結果、命題内容に相当する素性構造が残らなかった時
  (多くは、「はい」「そうです」等の固定化表現)
- 話題の属性による解析により、従来の命題内容が消滅した時
  (例えば、「参加方法について教えて下さい」など。これは、「参加方法」がトピックとして残るのみである。)

図 A.2に「登録用紙をお送りします」の命題内容部分の素性構造意味表現、命題内容リスト表現、および「送る」の命題格要素データの記述例を示す。

```
;;; 素性構造意味表現 ;;;
[[RELN  送る -1]
 [AGEN []]
 [RECP []]
 [ASPT  UNRL]
 [OBJE [[PARM !X12[]]
        [RESTR [[RELN  登録用紙 -1]
                [ENTITY !X12]]]]]]

;;; 命題内容リスト ;;;
(送る -1 ?agen ?recp  登録用紙 -1)

;;; 命題格要素データ ;;;
(送る -1  AGEN  RECP  OBJE)
```

図 A.2: 命題内容リストの例

また、図 A.3に「登録用紙を送ってください」の素性構造意味表現と発話の表現のリストの記述例を示す。(この例では、情報伝達行為に多義が出る。)

```
;;; 素性構造意味表現 ;;;
[[SEM [[RELN  下さい-REQUEST]
       [AGEN [[LABEL  *SPEAKER*]]]
       [RECP !X02[[LABEL  *HEARER*]]]
       [ASPT  UNRL]
       [OBJE [[RELN  送る-1]
              [AGEN !X02]
              [RECP []]
              [OBJE [[PARM !X01[]]
                     [RESTR [[RELN  登録用紙-1]
                             [ENTITY !X01]]]]]]]]]]

;;; 発話の表現 ;;;
(REQUEST-ACTION *SPEAKER* *HEARER* ?TPC (送る-1 *HEARER* ?RECP  登録用紙-1))
(INFORM-ACTION  *SPEAKER* *HEARER* ?TPC (送る-1 *HEARER* ?RECP  登録用紙-1))
```

図 A.3: 発話の表現のリストの例

A.1.6 処理の概要

CATE の処理は、大まかに、発話モダリティ解析と発話表現変換の 2 つのフェイズに分かれる:

A.1.6.1 発話モダリティ解析

入力素性構造意味表現を、書換え規則の適用により書換え、発話行為のタイプあるいは情報伝達行為タイプを含む素性構造 (中間素性構造 (A.1.5.3節参照)) を出力する。

A.1.6.2 発話表現変換

このフェイズでは、中間素性構造を発話の表現のデータ構造 (リスト) に変換する。その過程において、以下のことを順次行ない、発話の表現のリストを構成し、出力する:

1. 命題内容変換
   中間素性構造の命題内容部分 (PRP 素性) を、命題格要素データを参照して、命題内容リストに変換する。

   前段の発話モダリティ解析では、発話の意図に関する部分しか変換されず、特定の発話行為における命題内容を除いては、素性構造で受け渡される。現在、対話構造解析機構が素性構造そのものを扱うことができないため、この命題内容部分の素性構造を先に述べた命題内容リストに変換せねばならない。

   この処理では、命題内容素性構造の主辞 (述語節の関係名) を主述語として、命題格要素データに記述された格要素だけを、対話構造解析への入力として抽出する。結果として、命題内容素性構造が構造を持つものであれば、命題内容リストが返る。

2. 話題属性解析
   話題属性による再解析を行なう。ここでは、発話モダリティ解析により A.1.7.2節で定義した情報伝達行為タイプが出なかったもの、すなわち発話行為タイプの抽出に留まったものが、話題属性規則により再解析される。

   話題属性規則は、発話行為タイプに特有のものであり、その中で可能な情報伝達行為タイプを決定する規則である。つまり発話行為タイプを環境と見て、話題となっている部分をキーに書き換える操作に等しい。話題属性規則内部での話題の属性の判断は、概念シソーラスを参照して行なう。

   ここで、発話行為タイプとは、久米ら [17] の表層発話行為タイプ (SIFT) に近いものであり、以下のようなものがある (詳細な表現の例については、語用論規則 (付録 C.2) 参照):

タイプ名	文型(表層表現の例)
INFORM	平叙文(... です, ... ます)
CONFIRM	yes/no 疑問文(... か?, ... ね?)
ASK	(疑問詞のある)疑問文(... は何ですか?)
REQUEST	命令文, 依頼文(... をお願いします, ... して下さい)

また、話題の属性には、A.1.7.2節で設定した3つ(細かく分ければ5つ)のものがある。そして、発話中で話題となり得るエンティティについての階層関係の知識が、概念シソーラスに記述されている。

従って、話題属性規則には、上の発話行為タイプに対応する4つの規則がある。発話行為タイプと可能な情報伝達行為タイプの対応の詳細と、その処理規則については、A.1.8.2節で詳説する。

A.1.6.3 その他の処理

CATEの中心となる処理は以上で述べたが、その他に以下のような処理を行なっている。これらは、現状の環境の制約から来るものが多く、将来は統一的な形で改善・消滅していくことが望まれる:

1. **命題内容素性構造の前処理**
   語用論知識(発話モダリティ書き換え規則)適用後、命題内容に付加している情報を変換する。知識適用後の命題内容部分 PRP 素性に対する、素性構造書き換え処理である。

   1.6節 MOD 素性のところで触れたように、現在の CATE の語用論知識では、すべてのモダリティ部分を処理できない。従って、命題内容部分に、助動詞・補助動詞などの情報が残っている場合がある。このよう名名だいないよう部分から、純粋な命題内容部分を取り出すための処理を前処理として行なう。ここでは、その対象によって、MOD 素性を追加する操作を行なうことがある。

   現状では、「...でしょう」の う -GUESS や NEGATE、*-SHOULD 等の助動詞に関する 12 規則がある。

2. **オブジェクトの処理**
   主に名詞句の素性構造意味表現から、その主辞の概念(関係名)を取り出すための書き換え操作である。基本的には、以下の2つの書き換え規則による書換えである:

```
(rws:defrwschema2 gobj-0 gobj main
  "on <RELN> :unspecified
   in :Phase :object :Type :general
      in= [[RELN   ?val]
```

```
            [ENTITY ?x]
            ?rest]
       out= ?val
   end")
(rws:defrwschema2 gobj-1 gobj main
  "on <RESTR RELN> :unspecified
   in :Phase :object :Type :general
      in= [[PARM  ?ent]
           [RESTR ?obj]
           ?rest]
      if ?ent is []
        then => ?obj
      out= ?obj
        else => ?ent
      out= ?ent
      endif
   end")
```

なお、この他に、関係節や並立助詞・形式名詞などに対応する特定の規則を設定している。

また、疑問詞に対しては、上の環境と別環境に設定している。これは、疑問詞に関する書き換えは、発話モダリティ解析の際の条件として利用しているからである。従って、上のタイミングで疑問詞の書き換えは行なわない。

3. **トピック抽出**
   トピックは、基本的に、入力素性構造の prag 素性の topic 素性を参照する。この処理は、素性構造書換えであり、発話モダリティ解析の後処理として行なう。この書き換え規則は単に、 topic 素性のオブジェクトの内容を取り出すのみである。

4. **話し手・聞き手情報**
   対話構造解析において、話し手・聞き手は、区別された agent として各々ユニークに表されなければならない。なぜなら、典型的な発話対は、違った話者による情報交換であるからである。話し手・聞き手の情報は、システム外から与えられることが期待されるが、現状の NADINE では、その区別が明示的ではない。

   現状の CATE では、話し手・聞き手は、 SP1/SP2 という 2 つのシンボルを与えている。そして、サンプル会話について、その中の発話 ID と話し手の対応テーブルを準備しておき、それを参照することにより、話し手のユニークなシンボ

ルを獲得する。また、聞き手は、得られた話し手の逆のシンボルを与える。従って、現状のCATE話し手・聞き手情報を扱う際には、以下の注意が必要である:

(a) 処理対象の発話IDは指定されたものでなければならない

この情報の獲得は、発話表現変換の始めに行なう。

5. **書き換え前処理・後処理**
現在の素性構造書換えシステム[22]では、その規則定義中の素性値(素性パスの値)に"*"を含むシンボルを書くことができない。従って、書き換え前処理として、"*"を含むような素性構造(実際には*SPEAKER*, *HEARER*)をそれを含まないユニークな素性構造に変換している。(情報内容に変化はない。)

また、書き換え後処理として、複数命題を含む一つの素性構造を、その命題内容の数に振り分ける処理を行なっている。

A.1.6.4 処理のながれ

現状のCATEでは、以上説明した処理モジュールの処理順序をまとめると、以下のようになっている:

1. **発話モダリティ解析部**(素性構造書き換え処理)
   (a) 書き換え前処理
   (b) 発話モダリティ解析(語用論知識適用)
   (c) 命題内容素性構造前処理
   (d) オブジェクト変換処理
   (e) トピック抽出
   (f) 書き換え後処理
2. **発話表現変換部**
   (a) 話し手・聞き手情報獲得
   (b) 命題内容変換
   (c) 話題属性解析

発話モダリティ解析部は、素性構造書き換え処理であり、その書き換え環境を表A.6のように設定して、規則適用の効率化と規則管理のモジュラリティ向上を狙っている:

各処理部の詳細と実装については、付録A.4: 関数リファレンス参照。

表 A.6: 書き換え環境一覧

処理	環境名 :Phase	環境名 :Type	規則数
1. 書き換え前処理	:preedit	:speaker	2
2. 発話モダリティ解析	:intention	:general, :default	49
3. 命題内容素性構造前処理	:proposition	:ignore	12
4. オブジェクト変換処理	:object	:general, :wh-phrase	28
5. トピック抽出	:topic	:generral	2
6. 書き換え後処理	:postedit	:topic	4

A.1.7 情報伝達行為

CATE の目的は、一発話の意味表現から情報伝達行為を抽出することである。本節では、その出力目的の中心となる情報伝達行為の概念について再考し、ATR サンプル対話のための情報伝達行為と発話の表現のモデルを設定する。

A.1.7.1 協調的目標指向型対話における情報伝達行為とは

協調的目標指向型対話は、ある目標の達成に向けて、対話参加者が協調的に情報交換を行なう対話である。

情報伝達行為は、協調的目標指向型対話において話し手が何らかの情報を聞き手に与える発話を行なう時に観察される発話行為である。情報伝達行為は、話し手の信念に基づく意図の遂行の形態である。したがって、それは話し手の信念の内容と聞き手の信念に与える影響に関する話し手の信念の内容からで記述することができる。

情報伝達行為のクラス 情報伝達行為は、その聞き手に与える影響 (効果:effect) により大きく 2 つに区別することができる。『要求 (Demand)』と『応答 (Response)』である。『要求』の情報伝達行為とは、effect が聞き手の与えられた命題内容に関する発話を行なうこと (動機:motivation) を促すものである。ここで、協調的に行なわれる情報交換の仮定において、『要求』の情報伝達行為の受け手 (すなわち聞き手) には、応答の義務が生じる。すなわち、受け手が『要求』により受けた motivation に対して、何らかの形で情報を与えることにより、情報交換が成立する。この『応答の義務』は (協調的) 会話の協働原則により保証される。『要求』の情報伝達行為に対し、それが伝達している命題内容について、純粋に言明する発話の情報伝達行為を、『応答』の情報伝達行為とする。すなわち、この定義においては、ある『要求』の effect がそのまま motivation となる受け手の発話がその『要求』に対する『応答』になる。

例えば、質問 (要求の発話)「お名前をお願いします」に対する「はい」は応答ではない。なぜなら、前の発話の持つ effect

```
(BELIEVE( hearer WANT( speaker KNOW(speaker ?name))))
```

に対して、この発話は何ら貢献していないと考えられるからである。(敢えていえば、"I believe." の意味での発話かも知れないが、情報交換は成立しない。) この質問に対する『応答』の発話とは、KNOW(speaker ?name) を成就する発話でなければならない。

すべての発話が、『要求』・『応答』でとらえられるとは限らない。情報交換の成立の確認の意味で発せられる発話の情報伝達行為を、『確認 (Acknowledgement)』index 確認の情報伝達行為とする。その他、上記 3 つに含まれない発話の情報伝達行為を『陳述 (inform)』の情報伝達行為とする。

このようにここでは、情報伝達行為のクラスとして『要求』『応答』『確認』『陳述』の 4 つを考える。

発話対　協調的目標指向型対話では、『要求』と『応答』の発話により、ある命題内容についての情報交換が成立することになる。この2つの発話の作る構造を『発話対 (utterance pair)』と呼ぶ。発話対によって成立する情報交換を、対話中の最小の談話セグメント[7]としてとらえ、それらの上位目標への関係により表される構造が、対話構造である。従って、我々の文脈理解は、発話の情報伝達行為と命題内容から、対話構造を構築することが目標となる。

本報告の1つの目標は、対象対話における情報伝達行為の分類にある。そして、その分類に際しての基本的立場は、『情報交換を成就する発話対の構成』である。よって、分類された情報伝達行為は、『要求』から『応答』への写像関係 (必ずしも1対1でない) を持つものでなければならない。

以下では、以上の考えに基づいて、ATR サンプル対話を対象に情報伝達行為の分類を行なう。

A.1.7.2　情報伝達行為タイプの設定

情報伝達行為タイプは、発話モダリティのタイプと話題の属性で表現する。

発話モダリティのタイプ　表 A.7(表 1.5 と同様) に、発話モダリティのタイプを示す。さらに、INFORM の下位に属すると考えられるが、対話での機能上特定の情報伝達行為有すると考えられるものを区別する。

表 A.7: 発話モダリティのタイプの分類

INFORM	: 事柄の事実について述べる。	(平叙文)
ASK	: 未知の事柄について質問する。	(疑問詞を伴う疑問文)
CONFIRM	: 事柄の真偽について質問する。	(yes/no 疑問文)
REQUEST	: 相手に行動を依頼する。	(命令文)
OFFER	: 自分の行為を拘束する。	(動作動詞を述語とした平叙文)
GREETING	: あいさつ。	(おもに固定化された表現)
() 内は代表的な文形式を表している。		
INFORM の下位属性のタイプ		
AFFIRMATIVE	: 命題内容を肯定する。	
NEGATIVE	: 命題内容を否定する。	
ACCEPT	: 行為を受け入れる。	
REJECT	: 行為を拒む。	
CONFIRMATION	: 情報交換成立の確認。	

話題の属性タイプ　表 A.8(表 1.5 と同様) に、話題の属性タイプの分類を示す。

[7]この談話セグメント (発話クラスタ) は、『確認』の発話を伴う場合がある。

表 A.8: 話題の属性タイプの分類

ACTION	: ある行為・行動についての話題
VALUE	: ある事物の属性についての話題
STATEMENT	: ある事物の状態についての話題
(WANT)	: 話し手の欲求状態について
(GREETING)	: 対話の開始・終了

なお、WANT は STATEMENT の下位であるが、先に述べたように対話中で、通常の状態提示とはことなる、重要な振舞いを見せるため区別した。また、GREETING は第1章で述べたように、特定の発話行為 (上の発話モダリティのタイプ GREETING) で成立する話題としてここにあげた。これは、具体値として、OPEN(ing), CLOSE(ing) を設定する。

情報伝達行為タイプ 表 A.7と表 A.8のような分類から、表 1.6のような情報伝達行為のタイプが設定できる。また、これらの情報伝達行為に対応した代表的な表層表現を併せて示す。

固定化表現の設定 問い合わせ電話対話などにおいては、慣用的に発話される固定化された表現が良く現れる。これらを『固定化表現 (frozen term)』と呼び、効率的な解析の手助けとする。(表 1.6右欄で "*" のついた表現)

固定化表現は、表で示した情報伝達行為による発話をなす時、対話中に断片的な一発話として現れることが多い。従って、発話モダリティの書換え規則として記述する際の負担も大きくない。

A.1.8 知識ベース解説

A.1.8.1 語用論知識

発話のモダリティ表現部分と情報伝達行為タイプ (あるいは表層発話行為タイプ) を対応付ける知識である語用論知識は、素性構造書換えシステムの書換え規則のフォーマリズム [22] で記述する。

現在の CATE では、実際の素性構造意味表現と対話の流れの対応分析から 49 の規則を設定している。(巻末付録 C.2にその一覧を掲げる)

文末表現規則 文末表現に対する規則の基本的な記述戦略は:

- 発話モダリティ部分の書換え環境は、'((:Phase :intention)) とする。
- 素性パスには、できる限り深い位置にありかつ特定的な素性値を記述する。
- 対象入力素性構造パタン (in=) には、入力素性構造意味表現の発話モダリティ部分を記述する。[8]
- 適用結果素性構造 (out=) には:
  - CAT 素性に、情報伝達行為タイプ (あるいは表層発話行為タイプ) の明示的な値 (:atomic) を,
  - TPC 素性に、入力の prag 素性構造を (これは後のトピック抽出で書き換えられる),
  - PRP 素性に、発話モダリティの OBJE 素性構造を

  を記述するようにする。

発話モダリティを表す文末表現 “ください” と “(疑問詞) ですか?” に対する規則の例を図 A.4 に示す。

[8]例外として、“疑問詞 + 教える (対面動作動詞)” は、ASK を表す発話モダリティとして考えている。

```
;;; 規則例 1 ;;;
(rws:defrwschema2 req-2-1 req kudasai
  "on <SEM RELN>   ください-REQUEST              ;; 素性パスと素性値
   in :Phase :intention :Type :general          ;; 環境設定
      in= [[SEM [[RELN  ください-REQUEST]       ;;
                [AGEN [[LABEL  SPEAKER]]]      ;; 行為者は話し手
                [RECP [[LABEL  HEARER]]]       ;; 受益者は聞き手
                [OBJE ?prp]                    ;; 命題内容
                ?rest]]
           [PRAG  ?prag]
           ?rests]
      out= [[CAT REQUEST]                       ;; 表層発話行為タイプ
            [PRP ?prp]                          ;; 命題内容
            [TPC ?prag]]                        ;; トピック
   end")

;;; 規則例 2 ;;;
(rws:defrwschema2 ask-1 ask INFORMREF
  "on <SEM OBJE RELN> INFORMREF                 ;; 特定の素性値で深いものは
                                                ;; INFORMREF である
   in :Phase :intention :Type :general          ;; 環境設定
      in= [[SEM [[RELN  S-REQUEST]
                [AGEN !SP[[LABEL  SPEAKER]]]
                [RECP !HR[[LABEL  HEARER]]]
                [OBJE [[RELN  INFORMREF]
                       [AGEN !HR]
                       [RECP !SP]
                       [OBJE [[PARM ?wh]        ;; 疑問詞であること
                              [RESTR ?prp]      ;; 命題内容
                              ?rest2]]
                       ?rest3]]
                 ?rest4]]
           [PRAG ?prag]
           ?rests]
      ==> ?wh  with :Phase :object  :Type :wh-phrase
                                                ;; 上の ?wh が真に疑問詞で
                                                ;; あることを確認する
      out= [[CAT ASK]                           ;; 表層発話行為タイプ
    [WH  ?wh]                                ;; 疑問詞の具体的値（エキストラ情報）
            [TPC ?prag]                         ;; トピック
            [PRP ?prp]]                         ;; 命題内容
   end")
```

図 A.4: 語用論規則の記述例

図 A.4の規則例 1 を以下の入力素性構造 (発話文:「登録用紙で手続きをして下さい」) に適用すれば[9];

```
[[SEM [[RELN  ください-REQUEST]
       [ASPT  UNRL]
       [AGEN !X04[[LABEL  *SPEAKER*]]]
       [RECP !X03[[LABEL  *HEARER*]]]
       [OBJE [[RELN  する-1]
              [AGEN !X03]
              [OBJE [[PARM !X06[]]
                     [RESTR [[RELN  手続き-1]
                             [ENTITY !X06]]]]]
              [INST [[PARM !X05[]]
                     [RESTR [[RELN  登録用紙-1]
                             [ENTITY !X05]]]]]]]]
 [PRAG [[RESTR [[IN [[FIRST [[RELN  RESPECT]
                             [AGEN !X04]
                             [RECP !X03]]]
                    [REST [[FIRST [[RELN  POLITE]
                                   [AGEN !X04]
                                   [RECP !X03]]]
                           [REST !X02[]]]]]]
                [OUT !X02]]]
        [TOPIC [[IN []]
                [OUT []]]]
        [PRSP-TERMS [[IN []]
                     [OUT []]]]
        [SPEAKER !X04]
        [HEARER !X03]
        [ASPE [[IN !X01[]]
               [OUT !X01]]]]]]
```

以下の出力素性構造が得られる;

```
[[CAT REQUEST]
 [PRP [[AGEN !X4[[LABEL HEARER]]]
       [INST [[PARM !X1[]]
              [RESTR [[ENTITY !X1]
                      [RELN 登録用紙-1]]]]]
       [OBJE [[PARM !X2[]]
              [RESTR [[ENTITY !X2]
                      [RELN 手続き-1]]]]]
       [RELN する-1]]]
```

[9]この中の *SPEAKER*, *HEARER* は、書き換え前処理により、それぞれSPEAKER, HEARER に書き換わっている。

```
[TPC [[ASPE [[IN !X3[]]
             [OUT !X3]]]
      [HEARER !X4]
      [PRSP-TERMS [[IN []]
                   [OUT []]]]
      [RESTR [[IN [[FIRST [[AGEN !X6[[LABEL SPEAKER]]]
                          [RECP !X4]
                          [RELN RESPECT]]]
                  [REST [[FIRST [[AGEN !X6]
                                 [RECP !X4]
                                 [RELN POLITE]]]
                        [REST !X5[]]]]]]
              [OUT !X5]]]
      [SPEAKER !X6]
      [TOPIC [[IN []]
              [OUT []]]]]]]
```

ちなみに、A.1.5.3節で述べたその他の処理を行なった後の中間素性構造出力は以下のようになる;

```
[[CAT REQUEST]
 [PRP [[AGEN HEARER]
       [INST 登録用紙 -1]
       [OBJE 手続き -1]
       [RELN する -1]]]
 [TPC []]]
```

同様に、図の規則例2の発話文:「どのようなご用件でしょうか?」への適用例を以下に示す;

```
;;; 入力素性構造 ;;;
[[SEM [[RELN  S-REQUEST]
       [AGEN !X02[[LABEL  *SPEAKER*]]]
       [RECP !X03[[LABEL  *HEARER*]]]
       [OBJE [[RELN  INFORMREF]
              [AGEN !X03]
              [RECP !X02]
              [OBJE [[PARM !X05[[RELN  どのような -1]
                                [ARG-1 [[PARM !X04[]]
                                        [RESTR [[RELN  用件 -1]
                                                [ENTITY !X04]]]]]]]
                     [RESTR [[RELN  う -GUESS]
                             [ASPT  STAT]
                             [EXPR !X02]
                             [OBJE [[RELN  だ -IDENTICAL]
                                    [OBJE !X06[]]
                                    [IDEN !X05]]]]]]]]]]
```

```
 [PRAG [[RESTR [[IN [[FIRST [[RELN  POLITE]
                            [AGEN !X02]
                            [RECP !X03]]]
                    [REST !X01[]]]]
               [OUT !X01]]]
        [TOPIC [[IN []]
               [OUT []]]]
        [PRSP-TERMS [[IN []]
                    [OUT []]]]
        [SPEAKER !X02]
        [HEARER !X03]
        [ASPE [[IN []]
              [OUT []]]]]]]

;;; 適用結果 ;;;
 [[CAT ASK]
  [PRP [[ASPT STAT]
        [EXPR !X3[[LABEL SPEAKER]]]
        [OBJE [[IDEN !X4 用件-1]
               [OBJE []]
               [RELN だ-IDENTICAL]]]
        [RELN う-GUESS]]]                    ;; 命題内容前処理により落ちる
  [TPC [[ASPE [[IN []]
               [OUT []]]]
        [HEARER !X2[[LABEL HEARER]]]
        [PRSP-TERMS [[IN []]
                    [OUT []]]]
        [RESTR [[IN [[FIRST [[AGEN !X3]
                            [RECP !X2]
                            [RELN POLITE]]]
                    [REST !X1[]]]]
               [OUT !X1]]]
        [SPEAKER !X3]
        [TOPIC [[IN []]
               [OUT []]]]]
  [WH !X4]]

;;; 中間素性構造 ;;;
 [[CAT ASK]
  [PRP [[IDEN !X1 用件-1]
        [OBJE []]
        [RELN だ-IDENTICAL]]]
  [TPC []]
  [WH !X1]]
```

固定化表現規則 また、固定化表現に対する規則については:

- 固定化表現は、入力素性構造そのもの (全体) が対象であり、発話のモダリティであると考える。
- 命題内容 (トピック) が特定的なものについては、適用結果素性構造にその値を記述する (現在は、GREETING のみ)。

固定化表現「もしもし」の規則記述例を図 A.5に示す。

```
;;; 規則例 3 ;;;
(rws:defrwschema2 gop-1 gop moshimoshi
  "on <SEM RELN> もしもし-OPEN_DIALOGUE
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  もしもし-OPEN_DIALOGUE]
                [AGEN [[LABEL  SPEAKER]]]
                [RECP [[LABEL  HEARER]]]
                ?rest]]
           ?prag]
      out= [[CAT GREETING-OPEN]
            [TPC opening]                    ;; 特定のトピックと
            [PRP [[RELN open-dialogue]]]]    ;; 命題内容を設定
   end")
```

図 A.5: 固定化表現の規則記述例

デフォールト規則 発話文の中には、その素性構造意味表現に発話モダリティ部分を持たないような発話がある。例えば、以下のような素性構造は、sem 素性の直下に命題内容部分が来ている;

```
;;; それでは登録用紙をお送り致します
[[SEM [[RELN  送る-1]
       [ASPT  UNRL]
       [AGEN !X04[[LABEL  *SPEAKER*]]]
       [RECP !X05[[LABEL  *HEARER*]]]
       [OBJE [[PARM !X03[]]
              [RESTR [[RELN  登録用紙-1]
                      [ENTITY !X03]]]]]
       [INST [[PARM !X02[]]
              [RESTR [[RELN  それ-1]
                      [ENTITY !X02]]]]]]]
```

このような例には、上のような規則を設定することができないので、情報伝達行為タイプがでないことになる。従って、このような入力に対する救済措置が必要となる。そこで、(sem 素性に) 発話モダリティ部分のない素性構造に対する規則として、デフォールト規則を設ける。現在、基本的デフォールト規則には 2 つのものがある (詳細は付録 C.2.1.3参照);

1. 命題内容の行為者 AGEN 素性が話し手かつアスペクトが UNRL なものは、行為拘束 (OFFER-ACTION) である。

2. 1 以外は、単に陳述的 INFORM である。

A.1.8.2 話題属性規則

話題属性規則は、発話の話題の属性により、より詳細な情報伝達行為タイプ導出を行なうための、知識記述である。発話モダリティ解析の結果として、情報伝達行為タイプの他に 4 つの (表層) 発話行為タイプが現れる可能性があることは、A.1.6.2節で述べた。最終的には、これらも情報伝達行為タイプに変換する必要がある。ここでは、これら 4 つの表層発話行為タイプに対応するための規則の内容について説明する。

情報伝達行為分析 まず、対象対話中の発話の、表層発話行為タイプと A.1.7.2節の情報伝達行為タイプとの対応について分析を行なった。その対応の一部を、表 A.9に示した。

概念の集合 発話の話題の属性の値に関する情報は、名詞句概念ネットワーク [14] の考えを述語その他の概念にも拡張した知識 (**概念シソーラス**) から得る (概念シソーラスの詳細については、[21] 参照)。各発話の話題属性の認定は、ここから得られた集合を参照することにより行なう。

ここでは、話題とトピックは異なる概念として捉えている (A.1.5.4節)。ここでいう話題には、命題内容の構成要素全般が含まれる。つまり、焦点があたっているオブジェクトのみならず、述語の概念 (どのようなことを使用としているのか)、行為者・対象の概念なども参照する。表 A.10に、名詞句・疑問詞・述語別の話題の属性の記述を示す。これらの具体的要素は、概念ネットワークインタフェース ([21] 参照) を利用することによって、得ることができる。[11]

[11]現在は、処理効率化のため、これらの集合はリストとして各々の集合を示す大域変数に保持している。これらの変数の設定は、システム初期化時に行なっている。従って、概念シソーラスを変更した際は、システムの初期かが必要となる。

表 A.9: 発話行為タイプと情報伝達行為タイプの対応

発話行為タイプ	情報伝達行為タイプ	表現の例
ASK	ASK-VALUE	参加料はいくらですか?
	ASK-STATEMENT	どのようなご用件ですか?
10	ASK-ACTION	どうすればよろしいでしょうか?
CONFIRM	CONFIRM-VALUE	そちらは会議事務局ですか?
	CONFIRM-STATEMENT	登録用紙はお持ちですか?
	CONFIRM-ACTION	参加なさいますか?
	ASK-VALUE	お名前をお伺いできますか?
	ASK-ACTION	参加方法をお伺いできますか?*
REQUEST	REQUEST/INFORM-ACTION	用紙を送って下さい
	ASK-VALUE	お名前をお願いします
	ASK-ACTION	参加方法を教えて頂けますか?*
	ACCEPT-OFFER	(よろしく) お願いします
INFORM	INFORM-VALUE	名前は鈴木真弓です
	INFORM-STATEMENT	京都プリンスホテルが近いんですが
	INFORM-ACTION	こちらで審査を行ないます
	ASK-VALUE	参加料について教えて頂きたいのですが
	ASK-ACTION	参加方法について教えて頂きたいのですが
	(INFORM-WANT)	会議に申し込みたいのですが

表現に * を付したものは、サンプル対話には出てこない

表 A.10: 話題属性記述名

略号	: 内容	例
NP-value	: 特定の値を持つオブジェクトを表す名詞句概念の集合	名前 -1, 参加料 -1
NP-behavior	: 動作・行動を表す名詞句概念の集合	参加 -1, 交通手段 -1
WH-value	: 特定の値を尋ねる疑問詞の集合	いくら -1, いつ -1
WH-behavior	: 動作・行動の内容・方法を尋ねる疑問詞の集合	どのように -1, どう -1
V-action	: 行為動詞概念の集合	する -1, 送る -1
V-statement	: 状態を表す述語概念の集合	近い -1, 持つ -1
V-interaction	: 対面動作を表す述語概念の集合	教える -1, 聞く -3
IS-verb	: いわゆるダ文を表す述語概念の集合	だ -IDENTICAL

話題属性規則詳説 以上のような分析から現状の CATE に設定した環境 (表層発話行為タイプ) 別の話題属性規則の処理の詳細を以下で説明する:

パラメータ記述 話題属性規則処理の対象 (入力) は中間素性構造と命題内容変換時までにでき上がっている一時的な発話の表現である。以下はそれら入力データに含まれている情報であり、説明では以下の略号を利用する (これらを入力から獲得する時のパラメータ設定については、付録 A.4.3節: 話題属性関連参照);

略号:	内容
prp :	命題内容リスト
pred:	命題内容の主述語
mod :	中間素性構造の MOD 素性値 (発話ムード: エキストラ情報)
wh :	中間素性構造の WH 素性値 (疑問詞対象概念: エキストラ情報)
aspt:	命題内容素性構造の ASPT 素性値 (命題アスペクト)

また、概念シソーラスから求められた特定の話題の属性の概念の集合については、 A.1.8.2節の記述を利用する。

ASK 規則 発話モダリティ解析結果表層発話行為タイプが ASK の時;

1. pred が IS-verb であり、かつ wh が WH-value(あるいは NP-value) であれば、
   → ASK-VALUE,
   (例): “参加料はいくらですか?”
2. pred が V-action であり、かつ wh が WH-behavior(あるいは NP-behavior) であり、かつ mod が義務的 (must, should) であれば、
   → ASK-ACTION,
   (例): “参加料はどのようにお支払いしたらよろしいのでしょうか?”
3. それ以外の時 (このステップは複数出力が可能);
   (a) wh が WH-value(あるいは NP-value) であれば ASK-VALUE を与える,
   (例): “誰と出席なさいますか?”
   (b) wh が WH-behavior(あるいは NP-behavior) であれば ASK-ACTION を与える,
   (例): “それはどのような方法でしょうか?”
   (c) pred が V-statement であれば ASK-STATEMENT を与える.
   (例): “どちらが近いですか?”
4. 以上の処理で情報伝達行為タイプがでないものには、デフォールトとして、ASK-VALUE, ASK-ACTION, ASK-STATEMENT を与える。

CONFIRM 規則 発話モダリティ解析結果表層発話行為タイプが CONFIRM の時;

1. pred が IS-verb であり、その値 (IDEN 格) が変項でなければ、
   → CONFIRM-VALUE,
   (例): “そちらは会議事務局ですか?”
   (a) さらに、その値が NP-behavior であれば、
   → CONFIRM-ACTION,
   (例): “支払いは銀行振り込みですか?”
2. mod が可能的 (POSSIBLE) である時 (複数出力可能);
   (a) pred が V-interaction であり、かつその対象格 (目的格) の値が変項でないとき;
   i. その値が NP-value であれば、
   → ASK-VALUE,
   (例): “参加料について教えて頂けますか?”
   ii. その値が NP-behavior であれば、
   → ASK-ACTION,
   (例): “支払い方法について教えて頂けますか?”
   (b) pred が V-behavior であれば、
   → CONFIRM-ACTION,
   (例): “まだ参加できますか?”
3. それ以外の時 (複数出力可能);
   (a) aspt が UNRL かつ pred が V-action であれば、CONFIRM-ACTION を与える,
   (例): “参加なさいますか?”
   (b) aspt が UNRL 以外であるか、または pred が V-statement であれば、CONFIRM-STATEMENT を与える,
   (例): “登録用紙はお持ちですか?”
4. 以上の処理で情報伝達行為タイプがでないものには、デフォールトとして、CONFIRM-VALUE, CONFIRM-ACTION, CONFIRM-STATEMENT を与える。

REQUEST 規則 発話モダリティ解析結果表層発話行為タイプが REQUEST の時;

1. prp が変項であれば、
   → REQUEST-ACTION/ACCEPT-OFFER,
   (例): “(よろしく) お願いします”
2. prp が名詞句命題内容の時 (複数出力可能) [12];

[12] この時 (名詞句命題内容の時) は、当該名詞句をトピックに移して命題内容は変項にする。

(a) その値 (prp) が NP-value であれば、
→ ASK-VALUE,
(例): “ご住所をお願いします”

(b) その値が NP-behavior であれば、
→ REQUEST-ACTION,
(例): “支払をお願いします”

3. pred が V-interaction であり、かつその対象格 (目的格) の値が変項でないとき (複数出力可能);

(a) その値 (prp) が NP-value であれば、
→ ASK-VALUE,
(例): “ご住所を教えて下さい”

(b) その値が NP-behavior であれば、
→ ASK-ACTION,
(例): “支払方法を教えて下さい”

4. 以上の処理で情報伝達行為タイプがでないものには、デフォールトとして、REQUEST-ACTION, INFORM-ACTION を与える。

INFORM 規則 発話モダリティ解析結果表層発話行為タイプが INFORM の時;

1. pred が IS-verb であり、その値 (IDEN 格) が変項でなければ、
→ INFORM-VALUE,
(例): “こちらは会議事務局です”

(a) さらに、その値が NP-behavior であれば、
→ INFORM-ACTION,
(例): “支払いは銀行振り込みです”

2. mod が欲求 (WANT) であれば (複数出力可能)、
→ INFORM-WANT,
(例): “会議に参加したいのですが”
さらにこの場合;

(a) pred が V-interaction であり、かつその対象格 (目的格) の値が変項でないとき (複数出力可能)

i. その値が NP-value であれば、
→ ASK-VALUE,
(例): “ご住所を伺いたいのですが”

ii. その値が NP-behavior であれば、
→ ASK-ACTION,
(例): “支払方法について教えて欲しいのですが”

3. mod が可能的であり、かつ pred が V-action であれば、
   または、 aspt が UNRL であり、かつ pred が V-action であれば、
   → INFORM-ACTION,
   (例): “参加できます”, “こちらで審査を行ないます”
4. それ以外の時 (複数出力可能);
   (a) mod が否定的 (NEGATE) であれば、
   NEGATIVE を与える,
   (例): “持っていません”
   (b) aspt が UNRL 以外であるか、または pred が V-statement であれば、 INFORM-STATEMENT を与える,
   (例): “持っています”
5. 以上の処理で情報伝達行為タイプがでないものには、デフォールトとして、INFORM-STATEMENT, INFORM-VALUE, INFORM-ACTION を与える。

A.1.8.3 命題格要素辞書

命題格要素データは、命題内容の主述語とその格要素の関係を定義したリストデータである。この定義に従い、命題内容部分の素性構造が命題内容リストに変換される。変換された命題内容は、定義にしたがった (格の位置が一定となっている) リスト構造となり、プラン認識機構で単一化可能なデータ構造である命題内容リストとして出力される。

この辞書は、本来不要である。プラン認識機構の単一化を柔軟にすれば、解決する問題である。

しかし、現在の文脈処理機構では必要なので、付録 C.2.4にサンプル会話をカバーするデータの一覧を掲げておく。

A.2 利用方法

本節では、情報伝達行為解析システムの利用方法および関数について説明する。

A.2.1 情報伝達行為解析システムのインストール

1. 対話構造解析プログラムをインストールする。(情報伝達行為解析システムもインストールされる。) (2.1節参照)

2. ファイル **AnaDialog/CATE/load.lisp** を編集し、*AnaDialog-path* の値を2.1節でコピーした対話構造解析プログラムのディレクトリ名にする
(defvar *AnaDialog-path* “ディレクトリ名 /AnaDialog/”)

A.2.2 情報伝達行為解析システムの起動の準備

1. LISP を立ち上げる

2. 素性構造書換えシステムをロードする。([22] 参照。素性構造書換えシステムは本システムには含まれていない。)

3. 情報伝達行為解析システムのディレクトリへ移動する
(cd “ディレクトリ名 /AnaDialog/CATE”)

4. 情報伝達行為解析システムをロードする
(load “load.lisp”)

5. パッケージを:cate にする
(in-package :cate)

6. 情報伝達行為解析システムをコンパイルする。(この処理は最初の準備のときのみ)
(compile-cate)

A.2.3 データの用意

1. 入力データ、知識ベースを用意する。
(A.1.5、A.1.8.1、A.2.5節参照)。

2. 1で定義したデータのファイル名を変数にセットする。
load-data.lisp を編集する。(表 A.11参照)

3. データファイルをロードする。
(load “load-data.lisp”)

表 A.11: データのファイル名をセットする変数

変数名	説明
(''AnaDialog/CATE/load-data.lisp'')	
user::*PlanRec-Data-path*	概念ネットワーク辞書と命題格要素辞書のあるディレクトリ
user::*Pred-Data-path*	スピーカテーブルのあるディレクトリ
CATE::*data-path*	話題属性解析用パラメータと入力素生構造のあるディレクトリ
CATE::*rule-path*	変換規則のあるディレクトリ
CATE::*np-concept-filename*	概念ネットワーク辞書のファイル
CATE::*proposition-grammar-filename*	命題格要素辞書のファイル
CATE::*parameter-set-filename*	話題属性解析用パラメータ
CATE::*speaker-filename*	スピーカテーブル
CATE::*kaiwa-input-path*	入力素性構造のあるディレクトリ
CATE::*kaiwa-input-file*	入力素性構造ファイル

A.2.4 システムの操作

A.2.4.1 一括に情報伝達行為解析を行なう手順

(examin-file *&optional (filename* **kaiwa-input-file***))

A.2.4.2 逐次的に情報伝達行為解析を行なう手順

1. *filename*のファイルから入力素生構造ををロードする。
   (load-kaiwa-input-file *&optional (filename* **kaiwa-input-file***))
2. 指定された入力の ID に対して、発話モダリティ解析を行なう。
   (tran 'ID)
3. 指定された入力の ID に対して、発話表現変換を行なう。
   (urep 'ID)

A.2.5 データ記法

1. 語用論知識 (書換え規則)

例

```
(rws:defrwschema2 afm-1 afm hai
  "on <SEM RELN> はい -AFFIRMATIVE
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  はい -AFFIRMATIVE]
;;                [ASPT  -]
```

```
                    [AGEN [[LABEL  SPEAKER]]]
                    [RECP [[LABEL  HEARER]]]
                    ?rest]]
              ?prag]
        out= [[CAT AFFIRMATIVE]
              [TPC []]
              [PRP []]]
    end")
```

2. スピーカテーブル
   音声認識候補絞り込みシステム (B.2.5節参照) と同様。

3. 命題格要素辞書

   対話構造解析プログラム (2.5.2節参照) と同様。

4. 概念ネットワーク辞書

   対話構造解析プログラム (2.5.2節参照) と同様。

5. 話題属性規則

   付録 C.2.2参照。

6. 素生構造 (サンプルデータ)

   例

```
D1-1 SP1 もしもし
[[SEM [[RELN  もしもし-OPEN_DIALOGUE]
       [ASPT  -]
       [AGEN !X01[[LABEL  *SPEAKER*]]]
       [RECP !X02[[LABEL  *HEARER*]]]]]
 [PRAG [[SPEAKER !X01]
        [HEARER !X02]]]]
```

A.3 操作例

サンプルデータ (付録参照) を使い操作例を示す。

なお、本例で使用したマシンは SunSPARCstation2 であり、LISP は nemacs から Lucid Common Lisp を起動して使っている。

A.3.1 情報伝達行為解析システムのインストール

1. 対話構造解析プログラムをインストールする。(情報伝達行為解析システムもインストールされる) (3.1節参照)

2. パス名をセットする。
   ファイル **AnaDialog/CATE/load.lisp** を編集し、*AnaDialog-path* の値を3.1節でコピーした対話構造解析プログラムのディレクトリ名にする

   (defvar *AnaDialog-path* "/home/user/AnaDialog/")

A.3.2 情報伝達行為解析システムの起動の準備

1. LISP を立ち上げる

   m-x lucid を入力すると次のメッセージが表示される。

```
Starting /usr/local/bin/lisp ...
;;; Sun Common Lisp, Development Environment 4.0.0 , 6 July 1990
;;; Sun-4 Version for SunOS 4.0.x and sunOS 4.1
;;;
;;; Copyright (c) 1985, 1986, 1987, 1988, 1989, 1990
;;;            by Sun Microsystems, Inc.,  All Rights Reserved
;;; Copyright (c) 1985, 1986, 1987, 1988, 1989, 1990
;;;            by Lucid, Inc., All Rights Reserved
;;; This software product contains confidential and trade secret
;;; information belonging to Sun Microsystems, Inc.  It may not be copied
;;; for any reason other than for archival and backup purposes.
;;;
;;; Sun, Sun-4, and Sun Common Lisp are trademarks of Sun Microsystems Inc.

>
>
```

2. 素性構造書換えシステムをロードする。([22] 参照。素性構造書換えシステムは本システムには含まれていない。)

```
> (load  "/home/transfer/rws-v2/load")
```

```
;;; Loading source file "/home/transfer/rws-v2/load.lisp"
;;; Loading source file "engine/load.lisp"
loading rewriting engine.
;;; Loading binary file "engine/declare.sbin"
;;; Loading binary file "engine/basic.sbin"
;;; Loading binary file "engine/apply-rule.sbin"

            --------  中略 ---------

;;; Warning: Redefining FUNCTION FS-CHECK-ERROR which used to be
defined in "rws-v2/engine/fs.lisp"
;;; Loading binary file "rws-v2/tools/help.sbin"
#P"/home/transfer/rws-v2/load.lisp"
>
```

3. 情報伝達行為解析システムのディレクトリへ移動する

```
> (cd "/home/user/AnaDialog/CATE")
#P"/home/user/AnaDialog/CATE/"
>
```

4. 情報伝達行為解析システムをロードする

```
> (load "load")
;;; Loading source file "load.lisp"
--------- CATE SYSTEM 2.0 ---------
;;; Loading source file "../../PlanRec/NP/load.lisp"
;;; Loading binary file "../../PlanRec/Utility/utility.sbin"
;;; Loading binary file "../../PlanRec/NP/network.sbin"
;;; Loading binary file "../../PlanRec/NP/display-network.sbin"
;;; Loading binary file "../CATE/parameters.sbin"
;;; Loading binary file "../CATE/cate-main.sbin"
;;; Loading binary file "../CATE/fs.sbin"
;;; Loading binary file "../CATE/output.sbin"
;;; Loading binary file "../CATE/examine.sbin"
#P"/home/user/AnaDialog/CATE/load.lisp"
>
```

5. パッケージを:cate にする

```
> (in-package :cate)
#<Package "CATE" 6344E6>
>
```

6. 情報伝達行為解析システムをコンパイルする。(この処理は最初の準備のときのみ)

```
> (compile-cate)
;;; You are using the compiler in development mode (compilation-speed = 3)
;;; If you want faster code at the expense of longer compile time,
;;; you should use the production mode of the compiler, which can be obtained
;;; by evaluating (proclaim '(optimize (compilation-speed 0)))
;;; Generation of full safety checking code is enabled (safety = 3)
;;; Optimization of tail calls is disabled (speed = 2)
;;; Reading source file "/home/user/AnaDialog/CATE/parameters.lisp"

                -------- 中略 --------

;;; Warning: Free variable *WH* assumed to be special
;;; Writing binary file "/home/user/AnaDialog/CATE/output.sbin"
;;; Loading binary file "/home/user/AnaDialog/CATE/output.sbin"
;;; Reading source file "/home/user/AnaDialog/CATE/examine.lisp"
;;; Writing binary file "/home/user/AnaDialog/CATE/examine.sbin"
;;; Loading binary file "/home/user/AnaDialog/CATE/examine.sbin"
NIL
>
```

A.3.3 データの用意

1. 入力データ、知識ベースを用意する。

   ここでは、付録のサンプルデータを使用する。

2. 1で定義したデータのファイル名を変数にセットする。
   load-data.lisp を編集する。

   ここでは、付録のサンプルデータを使用するので、編集しない。

3. データファイルをロードする。

```
> (load "load-data")
;;; Loading source file "load-data.lisp"
;;; Warning: Ignoring an unmatched right parenthesis
;;; Loading source file "preedit.lisp"
;;; Warning: File "preedit.lisp" does not begin with IN-PACKAGE.
Loading into package "CATE"
:UNSPECIFIED
:UNSPECIFIED
;;; Loading source file "frozen.lisp"
;;; Warning: File "frozen.lisp" does not begin with IN-PACKAGE.
```

```
Loading into package "CATE"
はい -AFFIRMATIVE
そう -1
そう -1

        ------------ 中略 -------------

;;; Loading source file "../../PlanRec/Data/concept-nodes.lisp"
;;; Warning: File "../../PlanRec/Data/concept-nodes.lisp" does not
begin with IN-PACKAGE.  Loading into package "CATE"
;;; Loading source file "../CATE_Data/parameter-set.lisp"
;;; Warning: File "../CATE_Data/parameter-set.lisp" does not begin
with IN-PACKAGE.  Loading into package "CATE"
;;; Loading source file "../Rules/output-rule.lisp"
;;; Warning: File "../Rules/output-rule.lisp" does not begin with
IN-PACKAGE.  Loading into package "CATE"

----- CATE 2.0 for demonstration Dialogue 1-10 ----
#P"/home/user/AnaDialog/CATE/load-data.lisp"
>
```

A.3.4 一括に情報伝達行為解析を行なう手順

```
(examine-file)
D1-1: もしもし,
D1-2: そちらは会議事務局ですか,
D1-3: はい,
D1-4: そうです,
D1-5: どのようなご用件でしょうか,
D1-6: 会議に申し込みたいのですが, 2 3
D1-7: どのような手続きをすればよろしいのでしょうか, 2 3
D1-8: 登録用紙で手続きをして下さい, 2 3
D1-9: 登録用紙は既にお持ちでしょうか,
D1-10: いいえ,
D1-11: まだです,
D1-12: 分かりました, 2 3 4
D1-13: それでは登録用紙をお送り致します, 2
D1-14: ご住所とお名前をお願いします,
D1-15: 住所は大阪市北区茶屋町二十三です,
D1-16: 名前は鈴木真弓です,
D1-17: 分かりました, 2 3 4
D1-18: 登録用紙を至急送らせて頂きます,
D1-19: よろしくお願いします,
D1-20: それでは失礼します, 2
```

```
D1-1 もしもし
;;; Result 1
 [[CAT GREETING-OPEN]
  [PRP [[RELN OPEN-DIALOGUE]]]
  [TPC OPENING]]


D1-2 そちらは会議事務局ですか
;;; Result 1
 [[CAT CONFIRM]
  [PRP [[ASPT STAT]
        [IDEN 会議事務局-1]
        [OBJE !X1 そちら-1]
        [RELN だ-IDENTICAL]]]
  [TPC !X1]]

    ---------------- 中略 -----------------

(D1-19
  (
    (REQUEST-ACTION     SP1 SP2 ?TPCD1-19
                        ?PRPD1-19
                        "よろしくお願いします")
  )
  (
    (ACCEPT-OFFER       SP1 SP2 ?TPCD1-19
                        ?PRPD1-19
                        "よろしくお願いします")
  )
)

(D1-20
  (
    (GREETING-CLOSE     SP1 SP2 CLOSING
                        (CLOSE-DIALOGUE)
                        "それでは失礼します")
  )
)
T
>
```

A.3.5 逐次的に情報伝達行為解析を行なう手順

1. ファイルから入力素生構造ををロードする。

```
> (load-kaiwa-input-file "sem-1.fs")
```

```
D1-1: もしもし,
D1-2: そちらは会議事務局ですか,
D1-3: はい,
D1-4: そうです,
D1-5: どのようなご用件でしょうか,
D1-6: 会議に申し込みたいのですが, 2 3
D1-7: どのような手続きをすればよろしいのでしょうか, 2 3
D1-8: 登録用紙で手続きをして下さい, 2 3
D1-9: 登録用紙は既にお持ちでしょうか,
D1-10: いいえ,
D1-11: まだです,
D1-12: 分かりました, 2 3 4
D1-13: それでは登録用紙をお送り致します, 2
D1-14: ご住所とお名前をお願いします,
D1-15: 住所は大阪市北区茶屋町二十三です,
D1-16: 名前は鈴木真弓です,
D1-17: 分かりました, 2 3 4
D1-18: 登録用紙を至急送らせて頂きます,
D1-19: よろしくお願いします,
D1-20: それでは失礼します, 2
#<Hash-Table D411D6>
>
```

2. 指定された入力の ID に対して、発話モダリティ解析を行なう。

```
> (tran 'd1-9)
D1-9 登録用紙は既にお持ちでしょうか
;;; Result 1
 [[CAT CONFIRM]
  [PRP [[AGEN []]
        [OBJE !X1 登録用紙 -1]
        [RELN 持つ -1]]]
  [TPC !X1]]
#<Structure SENTENCE-OBJ D9DB56>
>
```

3. 指定された入力の ID に対して、発話表現変換を行なう。

```
> (urep 'd1-9)
(((CONFIRM-STATEMENT SP2 SP1 登録用紙 -1 (持つ -1 ?AGEND1-9 登録用紙 -1)
 "登録用紙は既にお持ちでしょうか")))
>
```

A.4 関数リファレンス

ここでは、CATE の主な内部関数のいくつかを紹介する。

A.4.1 主関数

---

load-cate [*Function*]

引数: なし
リターン値:

CATE をロードする. (load.lisp をロードすればこれは自動的に行なわれる。) (load.lisp)

---

compile-cate [*Function*]

引数: なし
リターン値:

CATE のソースファイルをコンパイルする. 同時にそのコンパイルコードをロードする. (load.lisp)

---

initialize-cate [*Function*]

引数: なし
リターン値: T

CATE の初期化. 以下のことを行なう:

1. 書換え規則のロード (renew-rws-rules)
2. 概念ネットワークの設定 (load-concept)
3. 命題格要素データの設定 (load-proposition-grammar)
4. 規則パラメータの設定 (parameter-set)
5. 話題の属性規則のロード (load-output-rules)

(load.lisp)

A.4.2 大域変数

---

program-path [*Variable*]

タイプ: string
初期値: "AnaDialog/CATE/"

プログラムソース・バイナリファイルのパス名. (`load.lisp`)

---

data-path [*Variable*]

タイプ: string
初期値: "AnaDialog/Data/"

データファイルのパス名. (`load-data.lisp`)

---

file-list [*Variable*]

タイプ: list of strings
初期値: (list "parameters" "cate-main" "fs" "output" "examine")

プログラムソースファイル名のリスト. (`load.lisp`)

A.4.3 知識・データ・規則

書換え規則

---

rule-path [*Variable*]

タイプ: string/pathname
初期値: "AnaDialog/Rules/"

書換え規則ファイルのあるパス名. (`load-data.lisp`)

---

rule-file-list [*Variable*]

タイプ: list of strings
初期値: (list “preedit” “frozen” “intention” “proposition-ignore” “object-general” “object-wh” “topic” “postedit”)

書換え規則ファイル名のリスト. (`load-data.lisp`)

---

load-rule-file *file &optional (dir *rule-path*)* [*Function*]

引数:

1. *file* : 規則ファイル名
2. *dir (&optional)*: 規則ファイルのパス名

リターン値:

引数で指定された規則ファイルをロードする. (`cate-main.lisp`)

---

load-rules *&optional (dir *rule-path*)* [*Function*]

引数:

1. *dir (&optional)*: 規則ファイルのパス名

リターン値:

書換え規則の全ロード. 引数 *dir* の, 大域変数 ***rule-file-list*** の要素で指定されたファイルすべてをロードする. (`cate-main.lisp`)

---

clear-rules [*Macro*]

引数: なし
リターン値:

書換え規則の全削除. (関数 **rws::remove-all-rw-rules2** に同じ) (`cate-main.lisp`)

---

renew-rws-rules *&optional (dir *rule-path*)* [*Function*]

引数:

1. *dir (&optional)*: 規則ファイルのパス名

リターン値:

規則の再ロード. 現在定義されている規則を全削除し, 新たに load-rules を行なう. (cate-main.lisp)

命題格要素データ

---

proposition-grammar-filename [*Variable*]

タイプ: string/pathname
初期値: "PlanRec/Data/case-grammar.lisp"

命題格要素データベースのファイル名. (load-data.lisp)

---

proposition-grammar [*Variable*]

タイプ: alist
初期値: nil

命題格要素のデータベース. (output.lisp)

---

load-proposition-grammar [*Function*]

引数: なし
リターン値:

命題格要素データベースをファイル:*proposition-grammar-filename*)
からロードし、 *proposition-grammar* に設定する。 (output.lisp)

---

get-pg *pred* [*Function*]

引数:

1. *pred* : 述語名

リターン値: list (命題格要素データ)

pred の命題格要素データが *proposition-grammar* にあれば、そのデータを返す。それ以外は nil を返す。(output.lisp)

話題属性関連

(概念・表現ネットーワーク検索システム本体については、参考文献 [21] 参照)

---

np-load-filename [*Variable*]

タイプ: string/pathname
初期値: "PlanRec/NP/load"

概念・表現ネットーワーク検索システムロード用のファイル名. (load.lisp)

---

np-concept-filename [*Variable*]

タイプ: string/pathname
初期値: "PlanRec/Data/concept-nodes.lisp"

概念ネットワークの辞書ファイル名. (load-data.lisp)

---

load-concept *&optional (file *np-concept-filename*)* [*Function*]

引数:

1. *file (&optional)*: 概念ネットワーク辞書ファイル名

リターン値: numeric(設定したネットワークのノード数)

file をロードし, 概念ネットワークを設定する. 引数は optional であり, 明示的に与えられなければ, *np-concept-filename* の値が利用される。話題属性解析に利用される概念集合の変数は更新しないので、そこまで更新する場合は、(parameter-set) を行なう必要がある。(load.lisp)

---

parameter-set-filename [*Variable*]

タイプ: string/pathname
初期値: "AnaDialog/CATE_Data/parameter-set.lisp"

話題属性パラメータ初期設定用のファイル名. (load-data.lisp)

---

parameter-set *&optional (package *read-package*)* [*Function*]

引数:

1. *package (&optional)*: 読み込みパッケージ名[13]

リターン値: loaded file's path

話題属性解析用のパラメータを設定する。実際には、*package* 下において、*parameter-set-filename* をロードするだけである。解析の入力となる中間素性構造を参照するためのパラメータについては、以下の表を参照. (parameters.lisp)

[13]この『読み込みパッケージ』は、CATE, LAYLA では、しばしば現れる。何のために利用されているかというと、パッケージの違うシステムで、同じシンボルを同じパッケージのものとして扱うために外部のデータを読み込む際に利用している。システムごとにパッケージが異なっていることがこの手間をつくっている。

中間素性構造参照用パラメータ 変数名	現在の値 (備考)
cate-cat-path	'(CAT) ; 情報伝達行為タイプ (表層発話行為タイプ) の素性名
cate-topic-path	'(TPC) ; トピックの素性名
cate-proposition-path	'(PRP) ; 命題内容の素性名
cate-sem-feature-path	'(SEM) ; 意味表現の素性名 (情報伝達行為が出ていない時利用)
cate-object-main-feature	'VAL ; 主に名詞句概念の主辞を表す素性名
cate-predicate-feature-name	'RELN ; 命題内容の主述語を表す素性名
cate-agent-value-list	'(SPEAKER HEARER) ; 対話参加者を表す素性値の集合
cate-speaker-value	'(SPEAKER) ; 発話の話し手を表す素性値の集合
cate-hearer-value	'(HEARER) ; 発話の聞き手を表す素性値の集合

output-rule-filename [*Variable*]

タイプ: string/pathname
初期値: 初期値: "AnaDialog/Rules/output-rules.lisp"

話題の属性を利用した抽出規則のソースファイル. (現在は LISP code で書かれている.) (`load-data.lisp`)

load-output-rules *&optional (filename *output-rule-filename*)* [*Function*]

引数:

1. *filename (&optional)*: 話題属性規則ファイル名

リターン値:

話題属性規則をロードする. 現在の CATE では、この規則は LISP code なので、単に *filename* をロードするのみ。(`output.lisp`)

話し手

---

speaker-filename [*Variable*]

タイプ: string/pathname
初期値: "PlanRec/Data/speaker.list"

サンプル会話内の発話の ID と話し手の対応データのファイル名. (load-data.lisp)

---

cate-speaker-list [*Function*]

引数: なし
リターン値: alist

speaker-filename から、発話 ID・話しての対応データを読み込み、
その A リストを返す。(parameters.lisp)

---

get-utterance-speaker *snum* [*Function*]

引数:

1. *snum* : 発話 ID

リターン値: symbol (現在は、SP1/SP2)

snum に対応する発話の話し手を表すシンボルを返す。そのようなものが
なければ nil. (parameters.lisp)

---

get-utterance-hearer *snum* [*Function*]

引数:

1. *snum* : 発話 ID

リターン値: symbol (現在は、SP1/SP2)

snum に対応する発話の聞き手を表すシンボルを返す。そのようなものが
なければ nil. (parameters.lisp)

cate-speaker-p *fs* [*Function*]

引数:

1. *fs* : 素性構造

リターン値: T/nil

fs が話し手を表すシンボル (:atomic 型素性構造) であれば T それ以外であれば nil を返す。(parameters.lisp)

A.4.4 発話モダリティ変換

transfer *node* [*Function*]

引数:

1. *node* : 素性構造

リターン値: 中間素性構造のリストのリスト

node の発話モダリティ解析を行なう。(A.1.6.4節 1 のすべての処理を行なう。) リターン値は, 以下の構造である:

```
(list (list 命題1についての中間素性構造1 命題1についての中間素性構造2 ...)
      ...
      (list 命題nについての中間素性構造1 命題nについての中間素性構造2 ...))
```

つまり, 入力発話が複数命題を持つものでなければ, リターン値の長さは 1 である.

A.4.5 発話表現変換

utterance-representation *snum fs &optional (string "none")* [*Function*]

引数:

1. *snum* : 発話 ID
2. *fs* : 素性構造

3. *string (&optional)*: 発話文字列

リターン値: list (発話の表現)

fs(素性構造) を発話の表現 (リスト) に変換したものを返す.
現在のところ, この中で, 話題の属性による情報伝達行為の解析 (get-cats-from-fs) を行なっている.

string は、オプショナルであり、発話の表現のリストの第 6 要素に加える。
(主にデバッグ表示用に利用) (output.lisp)

A.4.5.1 命題内容変換

---

get-proposition-from-fs *fs* [*Function*]

引数:

1. *fs* : 素性構造 (命題内容)

リターン値: symbol/variable/list (命題内容リスト) /nil

命題内容部分素性構造 *fs* を発話表現リスト用のデータ構造 (命題内容リスト) に変換する。 (output.lisp)

---

fs-to-urep-list-with-pg *pg fs* [*Function*]

引数:

1. *pg* : 命題格要素データ
2. *fs* : 素性構造 (命題内容)

リターン値: list (命題内容リスト)

pg の記述に従って、素性構造 *fs* から命題内容リストを構成する。 (output.lisp)

A.4.5.2 話題属性解析

---

get-cats-from-fs *fs cdr* [*Function*]

引数:

1. *fs* : 素性構造 (中間素性構造)
2. *cdr* : 発話の表現の cdr

リターン値: list (発話の表現)

話題の属性による書換えを行なった結果の発話の表現を返す。

引数 *cdr*は、話題の属性を鑑みる以前の発話表現のデータである。 (output.lisp)

---

get-caterule *cat* [*Function*]

引数:

1. *cat* : 表層発話行為タイプ

リターン値: function (規則の関数定義)/nil

cat の話題の属性の規則を定義した関数があれば、それを返す. それ以外は nil. (output.lisp)

---

urep-with-apply-caterule *rule fs urep* [*Function*]

引数:

1. *rule* : 話題属性規則
2. *fs* : 素性構造 (中間素性構造)
3. *urep* : 発話の表現リスト

リターン値: list (発話表現リスト)

fs, urep に *rule* を 適用した後の発話の表現リストを返す。 (output.lisp)

---

urep-with-atomic-prp *urep* [*Function*]

引数:

1. *urep* : 発話の表現リスト

リターン値: list (発話の表現)

urep の命題内容部分が命題データで表されていないものについて、データの変換を行なう。

具体的には、 atomic な命題内容 (例えば、“銀行振り込み”) を変項とし、その値をトピックの位置に置き換えている。 (output.lisp)

A.4.5.3 オブジェクト抽出

---

fs-to-urep-value *fs feature &key (without-sp nil)* [*Function*]

引数:

1. *fs* : 素性構造
2. *feature* : 素性パス
3. *without-sp (&key)*:

リターン値: symbol/variable (概念の主辞)

fs の *feature* で指定された素性値の主辞のシンボル (多くの場合は、***cate-object-main-feature*** で与えられた素性構造の素性値) を返す。

キーワード *without-sp* が nil 以外であれば、***cate-speaker-value***, ***cate-hearer-value*** に指定されたシンボルを、システム特有の話し手・聞き手を表すシンボル (SP1/SP2) に変換して返す。それ以外であれば変換しない。 (output.lisp)

変項

---

genvar *&optional string-list* [*Function*]

引数:

1. *string-list (&optional)*: 文字列のリスト

リターン値: variable (変項を表すシンボル)

変項を表すシンボルを生成し、それを返す。現在 CATE(LAYLA) では、prefix として "?" を付したシンボルを変項としている。

string-list が明示的に与えられれば、その要素である文字列を連結した文字列を変項名 (prefix に続く文字の並び) として変項を生成する。与えられなければ、内部的に文字列を生成し、それを変項名とする。 (output.lisp)

---

urep-var-p *atom* [*Function*]

引数:

1. *atom* : 任意のデータ

リターン値: T/nil

atom が変項であれば、T を、それ以外ならば nil を返す。(output.lisp)

A.4.6 実験用インタフェース

ここでは、CATE を単体で実験する際の環境と、便利な関数について説明する。なお説明注で明示的にファイル名が提示されていないものは、examine.lisp にある。

A.4.6.1 入出力インタフェース

内部データ

---

sentence-obj [*Structure*]

CATE 実験用の内部データ構造で、以下のようなスロットに内部処理における情報を格納している:

スロット名	初期値; 内容
number	nil ; 発話 ID(symbol)
speaker	nil ; 話し手 (symbol)
string	nil ; 発話文文字列 (string)
input-fs	nil ; 入力素性構造 (日本語解析結果: list of fs)
results	nil ; 中間素性構造 (発話モダリティ解析結果: list of fs)
cat-list	nil ; 発話表現リスト (最終的結果: list of list)

本実験用インタフェースの多くの関数では、基本的にこの内部データを対象として扱う。

---

kaiwa-input-data [*Variable*]

タイプ: hash-table
初期値: (make-hash-table)

内部データ sentence-obj を発話 ID をキーとして保持するデータプール。

---

init-kaiwa-input-data [*Function*]

引数: なし

kaiwa-input-data の全ての内部データを消去する。(初期化)

---

kaiwa-input-id-list [*Macro*]

引数: なし
リターン値: list (発話 ID のリスト)

kaiwa-input-data 内に保持されている全ての内部データの発話 ID のリストを返す。

---

make-inputdata *id sp string node-list* [*Macro*]

引数:

1. *id* : 発話 ID
2. *sp* : 話し手
3. *string* : 発話文文字列
4. *node-list* : 素性構造のリスト

リターン値: 内部データ構造 sentence-obj

引数を入力の情報として、内部データ構造 sentence-obj を作成し、それを返す。この時点では、結果 (results, cat-list スロット) は何もない。

---

find-inputdata *id* [*Function*]

引数:

1. *id* : 発話 ID

リターン値: 内部データ構造 sentence-obj

id に対応する内部データが *kaiwa-input-data* に存在すれば、その内部データを返す。そうでなければ nil.

入力

---

kaiwa-input-path [*Variable*]

タイプ: string/pathname
初期値: "AnaDialog/CATE_Data/"

サンプル会話の日本語解析結果素性構造出力ファイルがあるパス名. (`load-data.lisp`)

---

kaiwa-input-file [*Variable*]

タイプ: string
初期値: "sem-1.fs"

デフォールト処理対象となる会話の素性構造ファイル名.(`load-data.lisp`)

---

kaiwa-input-file-list [*Variable*]

タイプ: list of strings
初期値: (list "sem-1.fs" "sem-2.fs" "sem-3.fs" "sem-4.fs" "sem-5.fs" "sem-6.fs" "sem-7.fs" "sem-8.fs" "sem-9.fs" "sem-10.fs" "sem-A.fs" "sem-B.fs")

サンプル会話の日本語解析結果素性構造出力ファイル名のリスト. 現在は、以上の 12 会話。 (`load-data.lisp`)

---

load-kaiwa-input-file *&optional (file *kaiwa-input-file*) &key (print t)* [*Function*]

引数:

1. *file (&optional)* : 入力ファイル名
2. *print (&key)* : 表示フラグ

リターン値: *kaiwa-input-data*

file のデータを読み込み (素性構造書換えシステムの rws::push-fs-from-file を利用) 発話ごとに内部データを作成して、***kaiwa-input-data*** に格納する。

もし、*print* に nil 以外が与えられれば、作成された内部データの発話 ID と文字列を、端末出力に表示する。

ここでは、内部環境 ***kaiwa-input-data*** の初期化を行なわないので、その時点までで保持されている内部データに追加する形で、データが読み込まれる。従って、それまでの内部データをクリアしたい場合は、明示的に **(init-kaiwa-input-data)** を行なう必要がある。

出力

---

kaiwa-output-path [*Variable*]

タイプ: string/pathname
初期値: "AnaDialog/CATE_Data/"

情報伝達行為解析後の出力, 発話の表現のリスト (のリスト), を書き出す際のパス名. (`load-data.lisp`)

---

kaiwa-data *&optional file* [*Function*]

引数:

1. *file (&optional)* : 出力ファイル名

リターン値: T

その時点で保持されている全ての内部データ ((kaiwa-input-id-list) で得られる発話 ID) の結果を、階層型プラン認識システム入力用のデータフォーマットにして、出力する。

出力先は、*file* に明示的にファイル名が与えられれば、***kaiwa-output-path*** で指定されたディレクトリの下のそのファイル、そうでなければ、端末出力である。

---

make-and-print-a-urep *snum &optional (stream t)* [*Function*]

引数:

1. *snum* : 発話 ID
2. *stream (&optional)*出力ストリーム

snum に対応する内部データを、発話表現変換を行なった後、階層型プラン認識システム入力用のデータフォーマットにして、*stream* に出力する。

A.4.6.2 解析用関数

---

tran *id* [*Function*]

引数:

1. *id* : 発話 ID

リターン値: 内部データ

id に対応する内部データについて、発話モダリティの解析を行なう。解析結果の中間素性構造のリストは、results スロットに setf する。

この関数は、内部データの input-fs の全ての入力素性構造について、解析を行なった後に、重複する結果を削除したリストを格納する。

---

tran-1 *id* [*Function*]

引数:

1. *id* : 発話 ID

リターン値: 内部データ

id に対応する内部データについて、発話モダリティの解析を行なう。解析結果の中間素性構造のリストは、results スロットに setf する。

この関数は、内部データの input-fs の先頭 (car) の入力素性構造一つについてのみ、解析を行なった後に、重複する結果を削除したリストを格納する。

---

urep *id* [*Function*]

引数:

1. *id* : 発話 ID

リターン値: 発話表現リスト (のリスト)

id に対応する内部データについて、発話表現変換 (話題属性解析を含む) を行なう。結果の発話表現のリストは、cat-list スロットに setf する。

この関数は、内部データの results の全ての中間素性構造について、変換を行なった後に、重複する結果を削除したリストを格納する。

A.4.6.3 表示

---

show-input-fs *sentence-obj* [*Function*]

引数:

1. *sentence-obj* : 内部データ / 発話 ID

sentence-obj の全ての入力素性構造 (input-fs スロット) を、端末出力に表示する。引数 *sentence-obj* は、それが内部データであればそれを、発話 ID であればそれに対応する内部データが利用される。

---

show-results *sentence-obj* [*Function*]

引数:

1. *sentence-obj* : 内部データ / 発話 ID

sentence-obj の全ての中間素性構造 (results スロット) を、端末出力に表示する。引数 *sentence-obj* は、それが内部データであればそれを、発話 ID であればそれに対応する内部データが利用される。

---

show-urep *id* [*Function*]

引数:

1. *id* : 内部データ / 発話 ID

id の全ての入力素性構造 (cat-list スロット) を、端末出力に表示する。引数 *sentence-obj* は、それが内部データであればそれを、発話 ID であればそれに対応する内部データが利用される。

---

show-kaiwa [*Function*]

引数: なし

その時点で保持されている全ての内部データの、発話 ID と発話文文字列を、端末出力に表示する。

A.4.6.4 バッチ処理

---

examine-file *&optional (file *kaiwa-input-file*)* [*Function*]

引数:

1. *file (&optional)* : 入力ファイル名

file のデータを読み込み、CATE の全ての解析を行なった後に、その結果を LAYLA 入力用データフォーマットにして、端末出力に出力する。

付録 B

音声認識候補絞り込みシステム

B.1 システム概要

B.1.1 機能

対話理解部からの予測情報 (情報伝達行為) を利用して、音声認識候補を選択する。

B.1.2 稼働環境

表 B.1: 音声認識候補絞り込みシステム稼働環境

マシン	リスプマシン、Sun
使用言語	Common Lisp

B.1.3 システム構成

音声認識候補絞り込みシステムの基本的システム構成を図 B.1に示す。

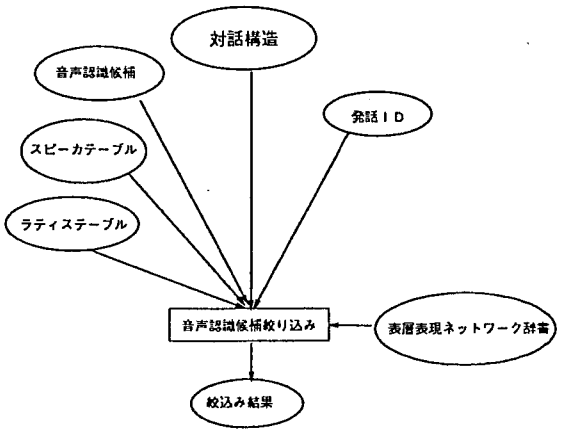

図 B.1: 音声認識候補絞り込みシステムの構成

B.1.4 ファイル構成

音声認識候補絞り込みシステムのファイル構成とその内容を表 B.2に示す。

表 B.2: 音声認識候補絞り込みシステムのファイル構成

ファイル名	内容
選択プログラム群 (``AnaDialog/Select/*'')	
load.lisp	選択システムロード
inputlattice.lisp	文節ラティスを扱うユティリティ
make-string.lisp	ラティスの値から入力文字列をつくる
np-predict.lisp	名詞句を予測する
select.lisp	選択システム本体

表 B.3: サンプルデータ

ファイル名	内容
(``AnaDialog/Data/*'')	
kakari-kekka.lisp	音声認識候補
id-table.lisp	ラティステーブル
speaker.list	スピーカテーブル
word-nodes.lisp	表層表現ネットワーク辞書

B.1.5 音声認識候補絞り込みシステムで扱うデータ

B.1.5.1 入力データ

対話構造 システムが理解した対話の状態を表す一表現 (1.6.3.1参照)

発話 ID 発話に対して付いている ID

音声認識候補 音声認識候補の文節ラティス

スピーカテーブル 話し手と発話 ID の対応テーブル

ラティステーブル 発話 ID と文節ラティスの対応テーブル

B.1.5.2 知識ベース

表層表現ネットワーク辞書 語用論的知識のモダリティに関する規則から作った発話意図を表す表層表現テーブル。

B.1.5.3 出力データ

絞り込み結果 ゴールスタックのスロット (selection) に音声認識絞込結果構造体 (B.4.1.1) としてセットされる。

B.1.6 音声認識候補絞り込み処理

文脈情報による音声認識候補絞り込み処理のモジュール構成を図 B.2に示す．以下では，各モジュールの処理について説明する．

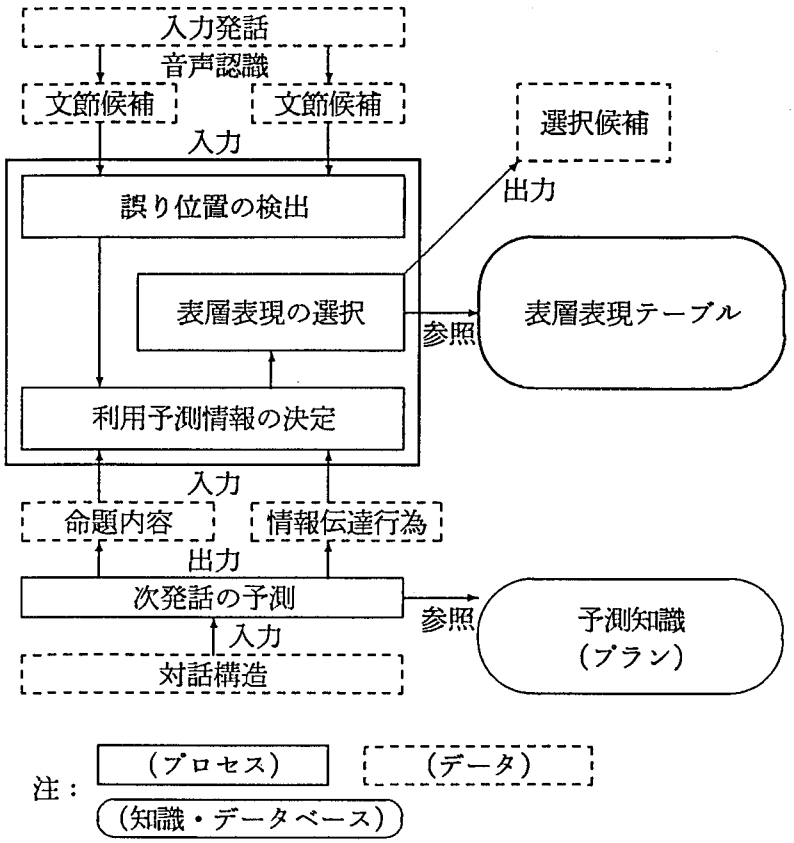

図 B.2: 文脈情報による音声認識候補選択処理

(1) 誤り位置の検出

文脈情報を適用した文節候補選択を行なうために，まず音声認識結果の誤り位置を検出する．絞り込み処理部へは，音声認識の出力の文節候補のラティスが入力となる．誤り位置は，各文節候補の自立語に誤りがあるか付属語に誤りがあるかで区別する．さらに，最終文節にあるか否かも区別する．すなわち，誤り位置は以下の 3 つの位置で区別される:

- 文中の自立語の誤り．
- 文中の付属語の誤り．
- 最終文節 (文末の表現) の誤り．

このような位置的区別で表現との関係付けを行なうことは，MINDS のなどの構文パタンの利用や，SPURT-I の ASP のような専用パーサに比べ，構文的・意味的な精密さは落ちる．しかし，構文解析でなく，音声認識結果の曖昧さ解消のための独立した処理としては，簡便な処理が望ましい．ここでの処理は，文節・単語ラティスによる入力と，発話の表現の分類と位置の対応関係から，簡便な処理により，ある程度，表現の意味的区別が可能である．

(2) 利用情報の決定

上記誤り位置により，利用する予測情報を決定する．すなわち，誤り位置が，最終文節であれば情報伝達行為のタイプに関する予測情報，自立語であれば命題内容に関する情報となる．なお，文中の付属語に関しては，現在扱っていない．

誤り位置が複数ある時は，最終文節の誤りからの処理を優先する．これは，プランの階層性に従っている．なお，音声認識候補に固定化表現を含む時は，それを一文単位で優先的に処理する．

(3) 表層表現の選択

最後に，予測情報に対応付けられる表層表現のものをを候補の中から選択する．予測情報と表層表現の対応知識は，表層テーブルで管理する．これらの知識の具体的値は，??節で述べる語用論的知識を表した書き換え規則から決定する．

このシステムは，発話の分類と利用する情報の種類の違いから，発話の意図に関する表現を扱う部分と，命題内容に関しての表現について扱う部分を分けて開発を進めている．しかし，上記の処理は双方に関して共通である．ただ，最後の表層表現の選択に関して，表現の機能に依存した知識の設定・処理を考えることになる．本稿では，以下で話し手の意図の表現に関する知識について述べ，実験結果を示す．(命題内容の構成要素，特に名詞句，の選択に関しては，参考文献 [14] を参照．)

B.2 利用方法

本節では、音声認識候補絞り込みシステムの利用方法および関数について説明する。

B.2.1 音声認識候補絞り込みシステムのインストール

対話構造解析プログラムをインストールすれば、音声認識候補絞り込みシステムもインストールされる。

B.2.2 音声認識候補絞り込みシステムの起動の準備

1. LISP を立ち上げる
2. 対話構造解析プログラムの起動の準備をする。(2.2節参照)
3. 対話構造解析プログラムのディレクトリへ移動する
   (cd “ディレクトリ名/AnaDialog/”)
4. 音声認識候補絞り込みシステムをロードする (ロードすることにより、対話構造解析プログラムの関数 gp-next-main にパッチを当てている。その結果、対話構造解析プログラムと同様の操作で、対話構造解析プログラムの対話構造解析処理に引続き、音声認識絞り込み処理がなされるようになる。)
   (load “Select/load.lisp”)
5. パッケージを:gplanner にする
   (in-package :gplanner)
6. 音声認識候補絞り込みシステムをコンパイルする。(この処理は最初の準備のときのみ)
   (select::compile-select)

B.2.3 データの用意

1. 対話構造解析プログラムのデータの用意をする。(2.3節参照)
   (対話構造解析用データをロードする)
2. 入力データ、知識ベースを用意する。(B.2.5節参照)。対話構造 (対話構造は対話構造解析で得られた結果を使う。)、発話 ID、ラティステーブル、スピーカテーブル、表層表現辞書を用意する。
3. 2で定義したデータのファイル名を変数にセットする。
   AnaDialog/Select/load-data.lisp を編集する。(表 B.4参照)

4. 音声認識候補絞り込み用データをロードする。
   (load “Select/load-data”)

表 B.4: データのファイル名をセットする変数

変数名	説明
音声認識候補絞り込みプラングラム参照用 (パッケージは select)(‘‘AnaDialog/Select/load-data.lisp’’)	
data-path	文脈情報絞込用をデータファイルのあるディレクトリ
inputlattice-filename	音声認識候補 (係受け出力データ) のファイル名
id-table-filename	ラティステーブルのファイル名
speaker-filename	スピーカテーブルのファイル名
np-word-filename	表層表現ネットワーク辞書のファイル名
プラン認識用データのロード (パッケージは user)(‘‘AnaDialog/load-data.lisp’’)	
simple-plan-filename	プラン認識の制御モードがシンプルモード のとき使用するプランスキーマのファイル名

B.2.4 システムの操作

対話構造解析プログラムの時と、同様。 (2.4節参照)

B.2.5 データ記法

1. 発話 ID
   例

   ```
   d2-5
   ```

2. 音声認識候補

   音声認識候補は、以下のように文節ラティスとラティスセンテンスの二つで記述する。
   例

   ```
   (DEF-BUNSETSU-LATTICE L1                          ; 文節ラティス
     V1 V999
     ((E1 V1 V999 "もしもし" (:PROB 0.875027))
      (E2 V1 V2 "五")
      (E3 V2 V3 "十")
      (E4 V3 V999 "四" (:PROB 0.109378)))
     (:NFRAMES 205 :INPUT "MOSHIMOSHI"))

   (DEF-LATTICE-SENTENCE S1                          ; ラティスセンテンス
     (L1 ))
   ```

3. スピーカテーブル

スピーカテーブルは、以下の例ように記述する。
例

```
(
;;; Kaiwa A
(Da-1 . SP1) ; もしもし
(Da-2 . SP1) ; そちらは会議事務局ですか
(Da-3 . SP2) ; はい
(Da-4 . SP2) ; そうです
(Da-5 . SP1) ; 会議に申込みたいのですが"
          ----- 中略 -----
(D10-35 . SP2) ; 分かりました
(D10-36 . SP2) ; 京都プリンスホテルに八月四日から八日まで一人部屋
をお取りしました
(D10-37 . SP1) ; どうもありがとうございました
(D10-38 . SP1) ; 失礼します
)
```

4. ラティステーブル

ラティステーブルは、以下の例のように記述する。
例

```
(
;;; Kaiwa A
(Da-1 . BUNSETSULATTICE::S1) ; もしもし
(Da-2 . BUNSETSULATTICE::S2) ; そちらは会議事務局ですか
(Da-3 . BUNSETSULATTICE::S3) ; はい
          ----- 中略 -----


)
```

5. 表層表現辞書

- 表層表現辞書 (*word*) は、概念と同様のネットワーク知識ベースで管理されるデータであり、ラベルとリンクから構成される。

  word ::= *('defword' word-name link)*
  word-name ::= *symbol*
  link ::= ≪*(link-name word-name)*≫
  link-name ::= *symbol*

'defword' は表層表現ノード定義用のマクロである。
表層表現の例:

```
(np::defword NEGATIVE-prag
  (EQ-FROZEN
    いいえ
    まだです)
  (EQ-TAIL
    まだです
    ません))
```

B.3 操作例

サンプルデータ (対話構造解析プログラム用データ: 入力行為データ、概念ネットワーク辞書、命題格要素辞書、プランスキーマ) と (音声認識候補絞り込みシステム用データ: 対話構造 (対話構造解析プログラムの出力)、発話 ID、ラティステーブル、スピーカテーブル、表層表現辞書)(C参照) を使い操作例を示す。

なお、本例で使用したマシンは SunSPARCstation2 であり、 LISP は nemacs から Lucid Common Lisp を起動して使っている。

B.3.1 音声認識候補絞り込みシステムのインストール

対話構造解析プログラムをインストールすれば、音声認識候補絞り込みシステムもインストールされる。

B.3.2 音声認識候補絞り込みシステムの起動の準備

1. LISP を立ち上げる

   m-x lucid を入力すると次のメッセージが表示される。

```
Starting /usr/local/bin/lisp ...
;;; Sun Common Lisp, Development Environment 4.0.0 , 6 July 1990
;;; Sun-4 Version for SunOS 4.0.x and sunOS 4.1
;;;
;;; Copyright (c) 1985, 1986, 1987, 1988, 1989, 1990
;;;            by Sun Microsystems, Inc.,  All Rights Reserved
;;; Copyright (c) 1985, 1986, 1987, 1988, 1989, 1990
;;;            by Lucid, Inc., All Rights Reserved
;;; This software product contains confidential and trade secret
;;; information belonging to Sun Microsystems, Inc.  It may not be copied
;;; for any reason other than for archival and backup purposes.
;;;
;;; Sun, Sun-4, and Sun Common Lisp are trademarks of Sun Microsystems Inc.

>
>
```

2. 対話構造解析プログラムの起動の準備をする。(3.2節参照)

3. 対話構造解析プログラムのディレクトリへ移動する

```
> (cd "/home/user/AnaDialog/")
#P"/home/user/AnaDialog/"
>
```

4. 音声認識候補絞り込みシステムをロードする。

```
> (load "Select/load")
;;; Loading source file "Select/load.lisp"
;;; Loading source file "Select/inputlattice.lisp"
;;; Loading source file "Select/make-string.lisp"
;;; Loading source file "Select/np-predict.lisp"
;;; Loading source file "Select/select.lisp"
;;; Loading source file "Select/gp-evaluate-patch.lisp"
;;; Warning: Redefining FUNCTION GP-EVAL-DATA-LIST which used to be
defined in "gp-evaluate.lisp"
;;; Warning: Redefining FUNCTION GP-EVAL-GS-LIST which used to be
defined in "gp-evaluate.lisp"
;;; Warning: Redefining FUNCTION PRINT-GP-EVAL-ALL which used to be
 defined in "gp-evaluate.lisp"
;;; Warning: Redefining FUNCTION GP-NEXT-MAIN which used to be defined
in "load.lisp"
#P"/home/user/AnaDialog/Select/load.lisp"
>
```

5. パッケージを:gplanner にする。

```
> (in-package :gplanner)
#<Package "GPLANNER" 6344B6>
>
```

音声認識候補絞り込みシステムをコンパイルする。(この処理は最初の準備のときのみ)

```
> (select::compile-select)
;;; You are using the compiler in development mode (compilation-speed = 3)
;;; If you want faster code at the expense of longer compile time,
;;; you should use the production mode of the compiler, which can be obtained
;;; by evaluating (proclaim '(optimize (compilation-speed 0)))
;;; Generation of full safety checking code is enabled (safety = 3)
;;; Optimization of tail calls is disabled (speed = 2)
;;; Reading source file "Select/inputlattice.lisp"
;;; Writing binary file "Select/inputlattice.sbin"
;;; Loading binary file "Select/inputlattice.sbin"

          -------- 中略 --------

;;; Reading source file "Select/np-predict.lisp"
;;; Writing binary file "Select/np-predict.sbin"
```

```
;;; Loading binary file "Select/np-predict.sbin"
;;; Reading source file "Select/select.lisp"
;;; While compiling GET-OWNER
;;; Warning: Variable UTTERANCE is bound but not referenced
;;; Writing binary file "Select/select.sbin"
;;; Loading binary file "Select/select.sbin"
NIL
>
```

B.3.3 データの用意

1. 対話構造解析プログラムのデータを用意する。(3.3節参照)
   (対話構造解析用データをロードする)

2. 入力データ、知識ベースを用意する。(B.2.5節参照)。対話構造 (対話構造は対話構造解析で得られた結果を使う。)、発話 ID、ラティステーブル、スピーカテーブル、表層表現辞書を用意する。
   ここでは、付録のサンプルデータを使用する。

3. 定義したデータのファイル名を変数にセットする。
   ここでは、付録のサンプルデータを使用するので、編集しない。
   以下に、 AnaDialog/Select/load-data.lisp の内容を示す。

   ファイル名: AnaDialog/Select/load-data.lisp

```
(IN-PACKAGE :select)
(defvar *data-path*
  (concatenate 'string user::*AnaDialog-path* "Data/"))
(defvar *inputlattice-filename*
  (merge-pathnames "kakari-kekka.lisp" *data-path*))
(defvar *id-table-filename*
  (merge-pathnames "id-table.list" *data-path*))
(defvar *speaker-filename*
  (merge-pathnames "speaker.list" *data-path*))
(defvar *np-word-filename*
  (merge-pathnames "word-nodes.lisp" *data-path*))

---------------------- 以下省略 --------------------------
```

4. 音声認識候補絞り込み用データをロードする。

```
> (load "Select/load-data")
;;; Loading source file "Select/load-data.lisp"
;;; Loading source file "Data/kakari-kekka.lisp"
;;; Loading source file "Data/word-nodes.lisp"
```

```
;;; Warning: File "Data/word-nodes.lisp" does not begin with
IN-PACKAGE.  Loading into package "GPLANNER"
#P"/home/user/AnaDialog/Select/load-data.lisp"
>
```

B.3.4 一括に対話理解、次発話の予測、音声認識候補の絞り込みを行なう手順

入力データのすべての文に対して一括に次の処理を行なう。1）対話理解を行ない、2）次発話の予測を行ない、3）音声認識候補の絞り込みをする。

```
> (gp-test-file "kaiwa-2.data" :tree t)

     ---------- 中略 (対話構造解析プログラムと同様)----------

---(D2-1)-------- Plan-inference with input 2
(GREETING-OPEN SP2 SP1 OPENING (OPEN-DIALOGUE))
(AFFIRMATIVE SP2 SP1 ?TPCD2-1 ?PRPD2-1)
Input order:
  1: (GREETING-OPEN AFFIRMATIVE)
  CHAIN             3    3.356
--------------- Result Number is 3

---- Prediction at the end of (D2-1) ----
  PREDICTION        3    0.002
 Predicted CATs::
  Next SP2: (INFORM-VALUE CONFIRM-VALUE-UNIT)
  Next SP1: (GREETING-OPEN)

---- Selection of (S59) uttered by SP2 ------     ;;;
Candicates:: (Failure types are (TAIL))          ;;; 音声認識候補絞り込み
    "こちらは会議事務局です"                      ;;; 処理による表示
    "こちらは会議事務局ですか"                    ;;;
                                                 ;;;
  SELECTION         3    0.087                   ;;;
Selected::  1
  "こちらは会議事務局です" by [1]INFORM-VALUE(2)

     ---------- 中略 (対話構造解析プログラムと同様)----------

;;; EVALUATIONs ;;;
 ID     & CAT   & SUC   & GS    & PLAN  & PRED  & CAND  & SELE  & UNIF  & NODE  & \\
 D2-1   & 2     & 1     & 3     & 1     & 2/3   & 2     & 1     & 1158  & 7     & \\
 D2-2   & 1     & 1     & 1     & 1     & 0     & 2     & 2     & 198   & 3     & \\
```

```
D2-3    & 1    & 1    & 2    & 2    & 2     & 2    & 2    & 579   & 7    & \\
D2-4    & 1    & 0    & 2    & 2    & 1     & 2    & 2    & 772   & 7    & \\
D2-5    & 2    & 0    & 2    & 2    & 1     & 2    & 2    & 456   & 2    & \\
D2-6    & 1    & 1    & 2    & 2    & 0     & 2    & 2    & 183   & 2    & \\
D2-7    & 1    & 0    & 2    & 2    & 0     & 2    & 2    & 591   & 2    & \\
D2-8    & 2    & 0    & 2    & 2    & 3     & 2    & 2    & 1544  & 3    & \\
D2-9    & 1    & 0    & 2    & 2    & 3     & 2    & 2    & 629   & 2    & \\
D2-10   & 3    & 1    & 6    & 3    & 1     & 2    & 2    & 1983  & 15   & \\
D2-11   & 2    & 1    & 2    & 3    & 3     & 2    & 1    & 572   & 2    & \\
D2-12   & 1    & 1    & 1    & 3    & 2     & 2    & 2    & 220   & 9    & \\
D2-13   & 1    & 1    & 1    & 4    & 1     & 2    & 1    & 354   & 2    & \\
D2-14   & 2    & 0    & 1    & 4    & 1     & 1    & 1    & 1235  & 4    & \\
D2-15   & 2    & 1    & 1    & 4    & 0     & 4    & 4    & 156   & 2    & \\
D2-16   & 1    & 1    & 1    & 5    & 3     & 1    & 1    & 563   & 3    & \\
D2-17   & 2    & 1    & 1    & 5    & 2     & 1    & 1    & 218   & 9    & \\
D2-18   & 2    & 1    & 2    & 6    & 1/2   & 2    & 2    & 746   & 4    & \\
D2-19   & 1    & 1    & 1    & 6    & 0     & 2    & 2    & 156   & 2    & \\
D2-20   & 2    & 0    & 1    & 6    & 0     & 2    & 2    & 1241  & 6    & \\
D2-21   & 1    & 0    & 1    & 6    & 0     & 2    & 2    & 442   & 2    & \\
21      & 32   & 13   & 37   & 71   & 145/6 & 41   & 38   & 13996 & 95   & \\
T
>
```

B.3.5 逐次的に対話理解、次発話の予測、音声認識候補の絞り込みを行なう手順

入力ファイルの文に対して逐次的に、1）対話理解を行ない、2）次発話の予測を行ない、3）音声認識候補の絞り込みを行なう。

1. ファイルから入力命題をロードする。
   対話構造解析プログラムと同様。

2. 対話構造解析プログラムを初期化する。(目標構造・入力履歴の初期化を行なう。)
   対話構造解析プログラムと同様。

3. プランニングの制御モード (表 2.2参照) を指定する。
   対話構造解析プログラムと同様。

4. 指定された入力の ID に対して、1）プラン認識を行ない、2）次発話の予測を行ない、3）音声認識候補の絞り込みをする。

```
> (gp-next 'd3-1)
---(D3-1)-------- Plan-inference with input 2
(GREETING-OPEN SP2 SP1 OPENING (OPEN-DIALOGUE))
```

```
(AFFIRMATIVE SP2 SP1 ?TPCD3-1 ?PRPD3-1)
Input order:
  1: (GREETING-OPEN)
  2: (AFFIRMATIVE)
  CHAIN              3    0.372
--------------- Result Number is 3

---- Prediction at the end of (D3-1) ----
  PREDICTION         3    0.002
 Predicted CATs::
  Next SP2: (INFORM-VALUE CONFIRM-VALUE-UNIT)
  Next SP1: (GREETING-OPEN)

---- Selection of (S80) uttered by SP2 ------      ;;;
Candicates:: (Failure types are (FUZOKU))         ;;; 音声認識候補絞り込み
    "こちら八一二六です"                           ;;; 処理による表示
    "こちらは八一二六です"                         ;;;

  SELECTION          3    0.039
Selected::  0
(D3-1 2 1 3 1 2/3 2 2 442 3)
>
```

5. 現在の理解状態の概観を表示する。
   対話構造解析プログラムと同様。

B.4 関数リファレンス

音声認識候補絞り込みシステムの主な関数・マクロ・大域変数について、説明する。なお、音声認識候補絞り込みシステムのパッケージは次のとおりである。音声認識候補絞込は “select”、文節ラティスの処理は “bunsetsulattice” である。注意されたい。

B.4.1 音声認識絞込処理 (AnaDialog/Select/select.lisp)

パッケージは select。

B.4.1.1 大域変数

---

select-type-order [*Variable*]

初期値: (quote (frozen tail jititsu))

音声認識結果の誤り一が複数あるとき、利用する予測情報の優先順位を指定する変数。

---

trace [*Variable*]

初期値: nil

この値が nil 以外であれば、音声認識絞込処理システムのトレースを取る。

---

trace-stream [*Variable*]

初期値: *monitor-stream*

関数 with-trace で表示するストリーム

---

selection [*Structure*]

- utterance : 初期値: nil
- type : 初期値: nil
- used : 初期値: nil
- info : 初期値: nil

音声認識絞込結果構造体

B.4.1.2 マクロ

---

with-trace *string &rest rest* [*Macro*]

引数:

1. string
2. rest (&rest)

リターン値: nil

関数 trace-on で絞り込みのトレースを表示するようにしておけば、rest のリターン値を string の形式でストリーム *trace-stream* に表示する。

B.4.1.3 関数

---

print-selection *sele stream depth* [*Function*]

引数:

1. sele
2. stream
3. depth

リターン値: nil

音声認識絞込結果構造体のプリント関数

---

trace-on [*Function*]

引数: なし
リターン値: t

絞込システムのトレースを表示するようにする。

---

trace-off [*Function*]

引数: なし
リターン値: nil

絞込システムのトレースを表示しないようにする。

---

selection-set *gsl id* [*Function*]

引数:

1. gsl
2. id

リターン値: gsl

対話構造のリスト gsl を入力にして、絞り込みをし、その結果 (関数 prediction のリターン値) を gsl にセットする。

---

selection *strings ftypes next-sp gs* [*Function*]

引数:

1. strings
2. ftypes
3. next-sp
4. gs

リターン値: 音声認識絞込結果構造体

対話構造 gs を入力にして、絞り込みをし、その結果を返す

---

select-next-utterance *pred strings types &aux ans* [*Function*]

引数:

1. pred
2. strings
3. types
4. ans (&aux)

リターン値: 音声認識絞込結果構造体

予測構造体、音声認識結果、誤り位置を入力して、絞り込みを行なう。

---

select-candidate-1 *type input-strings prediction-strings* [*Function*]

引数:

1. type
2. input-strings
3. prediction-strings

リターン値: 絞り込まれた入力発話のリスト

誤り位置 type と予測された入力表層表現のリスト prediction-strings により、入力発話のリスト input-strings を絞り込む。

---

select-candidate-with-type *type input-string predictions* [*Function*]

引数:

1. type
2. input-string
3. predictions

リターン値: 絞り込まれた入力発話

誤り位置 type と予測された入力表層表現のリスト predictions により、入力発話 input-string を絞り込む。

---

frozen-candidate-p *pred input-string* [*Function*]

引数:

1. pred
2. input-string

リターン値:

t 予測された表層表現である nil 予測された表層表現でない

音声認識の誤り位置が固定化表現である入力発話 input-string が、予測された表層表現 pred であるかどうかを調べる。

---

tail-candidate-p *pred input-string* [*Function*]

引数:

1.
2. input-string

リターン値:

t 予測された表層表現である nil 予測された表層表現でない

音声認識の誤り位置が語尾である入力発話 input-string が、予測された表層表現 pred であるかどうかを調べる。

---

jiritsu-candidate-p *pred input-string* [*Function*]

引数:

1. pred
2. input-string

リターン値:

t 予測された表層表現である nil 予測された表層表現でない

音声認識の誤り位置が自立語である入力発話 input-string が、予測された表層表現 pred であるかどうかを調べる。

---

nouns-to-string *nouns* [*Function*]

引数:

1. nouns

リターン値: 名詞句のストリングのリスト

名詞句のリストをストリングにする

---

get-owner *utterance* [*Function*]

引数:

1. utterance

リターン値:nil

所有者を返す。(今のところ、常に nil)

---

get-possible-strings *pred type* [*Function*]

引数:

1. pred
2. type

リターン値: 表層表現のストリングのリスト

予測構造体と誤り位置より、それに相当する表層表現を返す

---

get-possible-strings-by-objs *utterance* [*Function*]

引数:

1. utterance

リターン値: 発話表現のストリングのリスト

発話の命題からそれに相当する発話表現のストリングのリストを返す

---

get-possible-strings-by-cat *utterance* [*Function*]

引数:

1. utterance

リターン値: 発話表現のストリングのリスト

発話の情報伝達行為からそれに相当する発話表現のストリングのリストを返す

B.4.2 ラティスに対する処理 (AnaDialog/Select/inputlattice.lisp)

パッケージは BUNSETSULATTICE。

B.4.2.1 マクロ

---

def-bunsetsu-lattice *name &body body* [*Macro*]

引数:

1. name
2. body (&body)

リターン値: name

文節ラティスを定義する
例

```
(DEF-BUNSETSU-LATTICE L2
  V1 V999
  ((E1 V1 V999 "そちら" (:PROB 0.013672))
   (E2 V1 V2 "そちら")
   (E3 V2 V999 "は" (:PROB 0.875027)))
  (:NFRAMES 229 :INPUT "SOCHIRAWA"))
```

---

def-lattice-sentence *name lattices* [*Macro*]

引数:

1. name
2. lattices

リターン値: name

文節ラティス文を定義する。
例

```
(DEF-LATTICE-SENTENCE S2
  (L2 L3 ))
```

B.4.2.2 関数

---

distribute-bunsetsu-lattice *bunsetsu* [*Function*]

引数:

1. bunsetsu

リターン値: 自立語・付属語の文字列

文節ラティスから、自立語・付属語の文字列を作る。
文節ラティス:(LIST EDGE-ID START-V END-V STRING [SCORE])

---

bunsetsu-lattice-to-strings *bnum* [*Function*]

引数:

1. bnum

リターン値: 文節番号に対応する文節候補の文字列のリスト

文節番号に対応する文節候補の文字列のリストを返す。

---

sentence-lattice-to-strings *snum* [*Function*]

引数:

1. snum

B.4.4.1 関数

---

predict-noun *concept owner sp* [*Function*]

引数:

1. concept
2. owner
3. sp

リターン値: 概念 concept に相当する名詞句。

概念 concept に相当する名詞句をくを返す。

---

inspect-owner *owner sp* [*Function*]

引数:

1. owner
2. sp

リターン値:

:speaker owner が話者である。
:other owner が話者でない。

owner が話者かどうかを判定する。

---

inspect-viewpoint *owner sp* [*Function*]

引数:

1. owner
2. sp

リターン値: いまのところ、常に:same-viewpoint。

owner の視点を返す。

---

get-prag-sefs *concept* [*Function*]

引数:

1. concept

リターン値: 概念 concept に相当する表層表現。

概念 concept に相当する代表的表層表現をのリスト返す。

---

make-preferable-sefs *sefs* [*Function*]

引数:

1. sefs

リターン値: 代表的表層表現のリスト sefs と同意の表層表現のリスト。

代表的表層表現のリスト sefs と同意の表層表現のリストを返す。

---

make-eq-if-polite-link-sefs *sefs* [*Function*]

引数:

1. sefs

リターン値: 代表的表層表現のリスト sefs と同意で敬語的表層表現のリスト。

代表的表層表現のリスト sefs と同意で敬語的表層表現のリストを返す。

---

make-upper-concept-sefs *sefs* [*Function*]

引数:

1. sefs

リターン値: 表層表現のリスト sefs と同じ (is-a リンクで結ばれる) 概念の表層表現。

表層表現のリスト sefs と同じ (is-a リンクで結ばれる) 概念の表層表現を返す。

---

make-is-a-link-concepts *concepts* [*Function*]

引数:

1. concepts

リターン値: 表層表現のリスト sefs と同じ (is-a リンクで結ばれる) 概念。

表層表現のリスト sefs と同じ (is-a リンクで結ばれる) 概念を返す。

---

replace-concept-network-with-antecedent *concept value* [*Function*]

引数:

1. concept
2. value

リターン値:

概念 concept の value リンクの値を value にする。

---

sef-antecedent-p *sef* [*Function*]

引数:

1. sef

リターン値:

t sef が変数である。
nil sef が変数でない。

sef が変数かどうかを調べる。

付録 C

データ記述例

C.1 対話構造解析プログラム用

C.1.1 入力行為

階層型プラン認識システム [21] 参照

C.1.2 概念ネットワーク辞書

階層型プラン認識システム [21] 参照

C.1.3 命題格要素辞書

階層型プラン認識システム [21] 参照

C.1.4 プランスキーマ

階層型プラン認識システム [21] 参照

C.2 情報伝達行為解析システム用

以下に、CATEで使用するデータ (話題属性規則、話題属性規則、スピーカテーブル 、命題格要素辞書、概念ネットワーク辞書、素生構造) を示す。

C.2.1 語用論知識

C.2.1.1 固定化表現規則

ファイル名:AnaDialog/Rules/frozen.lisp

```
; Response Class ;
;;
;; AFFIRMATIVE
;;
;;; 1. はい*
(rws:defrwschema2 afm-1 afm hai
  "on <SEM RELN> はい-AFFIRMATIVE
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  はい-AFFIRMATIVE]
                 [AGEN [[LABEL  SPEAKER]]]
                 [RECP [[LABEL  HEARER]]]
                 ?rest]]
           ?prag]
      out= [[CAT AFFIRMATIVE]
            [TPC []]
            [PRP []]]
   end")
;;; 2. そうです*
(rws:defrwschema2 afm-2-1 afm soudesu
  "on <SEM MANN RESTR RELN> そう-1
   in :Phase :intention :Type :general
      in= [[SEM [[RELN だ-STATEMENT]
;;                 [ASPT  STAT]
                 [MANN [[RESTR [[RELN そう-1]
                                ?rest1]]
                        ?rest2]]
                 ?rest3]]
           ?prag]
      out= [[CAT AFFIRMATIVE]
            [TPC []]
            [PRP []]]
   end")
(rws:defrwschema2 afm-2-2 afm soudesu
  "on <SEM IDEN RESTR RELN> そう-1
   in :Phase :intention :Type :general
      in= [[SEM [[RELN だ-STATEMENT]
;;                 [ASPT  STAT]
                 [IDEN [[RESTR [[RELN そう-1]
                                ?rest1]]
                        ?rest2]]
                 ?rest3]]
           ?prag]
      out= [[CAT AFFIRMATIVE]
            [TPC []]
            [PRP []]]
   end")

;;
;; NEGATIVE
;;
```

```
;;; 1. いいえ *
(rws:defrwschema2 ngt-1 ngt iie
  "on <SEM RELN> いいえ -NEGATIVE
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  いいえ -NEGATIVE]
                [AGEN [[LABEL  SPEAKER]]]
                [RECP [[LABEL  HEARER]]]
                ?rest]]
           ?prag]
      out= [[CAT NEGATIVE]
            [TPC []]
            [PRP []]]
   end")

;;; 2. まだです *
(rws:defrwschema2 ngt-2 ngt madadesu
  "on <SEM TLOC RESTR RELN> まだ -1
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  だ -STATEMENT]
;;                [ASPT  STAT]
                [TLOC [[RESTR [[RELN  まだ -1]
                               ?rest1]]
                      ?rest2]]
                ?rest3]]
           ?prag]
      out= [[CAT NEGATIVE]
            [TPC []]
            [PRP []]]
   end")
(rws:defrwschema2 ngt-2-2 ngt madadesu
  "on <SEM IDEN RESTR RELN> まだ -1
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  だ -STATEMENT]
;;                [ASPT  STAT]
                [IDEN [[RESTR [[RELN  まだ -1]
                               ?rest1]]
                      ?rest2]]
                ?rest3]]
           ?prag]
      out= [[CAT NEGATIVE]
            [TPC []]
            [PRP []]]
   end")

;;
;; ACCEPT-ACTION
;;
;;; 1. 分かりました *
(rws:defrwschema2 aca-1-1 aca wakarimashita
  "on <SEM RELN> 分かる -1
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  分かる -1]
                [ASPT ?asp]
                [OBJE ?obj]
                ?rest1]]
           [PRAG ?prag]
           ?rests]
      if ?asp is $UNRL then fail endif
      out= [[CAT ACCEPT-ACTION]
            [TPC ?prag]
            [PRP ?obj]]
   end")
(rws:defrwschema2 aca-1-2 aca wakarimashita
  "on <SEM RELN> 分かる -2
   in :Phase :intention :Type :general
```

```
      in= [[SEM [[RELN  分かる -2]
                [ASPT ?asp]
                [OBJE ?obj]
                ?rest1]]
           [PRAG ?prag]
           ?rests]
      if ?asp is $UNRL then fail endif
      out= [[CAT ACCEPT-ACTION]
            [TPC ?prag]
            [PRP ?obj]]
   end")

;;; 2. 問題ありません *
(rws:defrwschema2 aca-2 aca mondainai
  "on <SEM OBJE RELN> 問題ある -1
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  NEGATE]
                [ASPT  STAT]
                [OBJE [[RELN  問題ある -1]
                       [LOCT ?prp]
                       ?rest1]]
                ?rest2]]
           [PRAG  ?prag]
           ?rests]
      out= [[CAT ACCEPT-ACTION]
            [TPC ?prag]
            [PRP ?prp]]
   end")

;;; 3. いいですよ
(rws:defrwschema2 aca-3 aca iidesuyo
  "on <SEM RELN> いいですよ -CONFIRMATION
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  いいですよ -CONFIRMATION]
                [ASPT  -]
                [AGEN [[LABEL  SPEAKER]]]
                [RECP [[LABEL  HEARER]]]]]
           ?prag]
      out= [[CAT ACCEPT-ACTION]
            [PRP []]
            [TPC []]]
   end")

;;
;; ACCEPT-OFFER
;;
;;; 1. ありがとう *
(rws:defrwschema2 aco-1 aco thanks
  "on <SEM RELN> ありがとう -THANKING
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  ありがとう -THANKING]
                [ASPT  -]
                [AGEN [[LABEL  SPEAKER]]]
                [RECP [[LABEL  HEARER]]]
                ?rest]]
           [PRAG  ?prag]
           ?rests]
      out= [[CAT ACCEPT-OFFER]
            [PRP []]
            [TPC []]]
   end")

;; 2. お願いします *
(rws:defrwschema2 aco-2 aco please
  "on <SEM RELN> 願う -REQUEST
```

```
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  願う-REQUEST]
;;                  [ASPT  UNRL]
                  [AGEN [[LABEL  SPEAKER]]]
;;                  [RECP [[LABEL  HEARER]]]
                  [OBJE []]
                  ?rest1]]
            ?rests]
      out= [[CAT ACCEPT-OFFER]
            [PRP []]
            [TPC []]]
   end")

; CONFIRM Class ;
;;
;; CONFIRMATION
;;
;;; 1. 分かりました*
(rws:defrwschema2 cfm-1 cfm wakarimashita
  "on <SEM RELN> 分かる-1
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  分かる-1]
                  [ASPT ?asp]
                  [OBJE ?obj]
                  ?rest1]]
           [PRAG ?prag]
           ?rests]
      if ?asp is $UNRL then fail endif
      out= [[CAT CONFIRMATION]
            [TPC ?prag]
            [PRP ?obj]]
   end")
(rws:defrwschema2 cfm-2 cfm wakarimashita
  "on <SEM RELN> 分かる-2
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  分かる-2]
                  [ASPT ?asp]
                  [OBJE ?obj]
                  ?rest1]]
           [PRAG ?prag]
           ?rests]
      if ?asp is $UNRL then fail endif
      out= [[CAT CONFIRMATION]
            [TPC ?prag]
            [PRP ?obj]]
   end")

;;; 2. そうですか*
(rws:defrwschema2 cfm-3 cfm soudesuka
  "on <SEM OBJE OBJE MANN RESTR RELN> そう-1
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  S-REQUEST]
                  [AGEN !SP[[LABEL  SPEAKER]]]
                  [RECP !HR[[LABEL  HEARER]]]
                  [OBJE [[RELN  INFORMIF]
                         [AGEN !HR]
                         [RECP !SP]
                         [OBJE [[RELN  だ-STATEMENT]
                                [MANN [[RESTR [[RELN  そう-1]
                                               ?rest0]]
                                       ?rest1]]
                                ?rest2]]
                         ?rest3]]
                  ?rest4]]
           ?prag]
```

```
      out= [[CAT CONFIRMATION]
            [TPC []]
            [PRP []]]
   end")
(rws:defrwschema2 cfm-3-1 cfm soudesuka
  "on <SEM OBJE OBJE IDEN RESTR RELN> そう-1
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  S-REQUEST]
                 [AGEN !SP[[LABEL  SPEAKER]]]
                 [RECP !HR[[LABEL  HEARER]]]
                 [OBJE [[RELN  INFORMIF]
                        [AGEN !HR]
                        [RECP !SP]
                        [OBJE [[RELN  だ-STATEMENT]
                               [IDEN [[RESTR [[RELN  そう-1]
                                              ?rest0]]
                                      ?rest1]]
                              ?rest2]]
                       ?rest3]]
                 ?rest4]]
           ?prag]
      out= [[CAT CONFIRMATION]
            [TPC []]
            [PRP []]]
   end")

; OTHERS ;
;;
;; GREETING-OPEN
;;
;;; 1 もしもし*
(rws:defrwschema2 gop-1 gop moshimoshi
  "on <SEM RELN> もしもし-OPEN_DIALOGUE
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  もしもし-OPEN_DIALOGUE]
                 [AGEN [[LABEL  SPEAKER]]]
                 [RECP [[LABEL  HEARER]]]
                 ?rest]]
           ?prag]
      out= [[CAT GREETING-OPEN]
            [TPC opening]
            [PRP [[RELN open-dialogue]]]]
   end")

;;; 2. はい*
(rws:defrwschema2 gop-2 gop hai
  "on <SEM RELN> はい-AFFIRMATIVE
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  はい-AFFIRMATIVE]
                 [AGEN [[LABEL  SPEAKER]]]
                 [RECP [[LABEL  HEARER]]]
                 ?rest]]
           ?prag]
      out= [[CAT GREETING-OPEN]
            [TPC opening]
            [PRP [[RELN open-dialogue]]]]
   end")

;;
;; GREETING-CLOSE
;;
;;; 1. さようなら*
(rws:defrwschema2 gcl-1 gcl sayonara
  "on <SEM RELN> さようなら-CLOSE_DIALOGUE
   in :Phase :intention :Type :general
```

```
      in= [[SEM [[RELN  さようなら-CLOSE_DIALOGUE]
                 [AGEN [[LABEL  SPEAKER]]]
                 [RECP [[LABEL  HEARER]]]
                 ?rest]]
           ?prag]
      out= [[CAT GREETING-CLOSE]
            [TPC closing]
            [PRP [[RELN close-dialogue]]]]
   end")

;;; 2. 失礼します*
(rws:defrwschema2 gcl-2 gcl shitureisuru
  "on <SEM RELN> 失礼する-CLOSE_DIALOGUE
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  失礼する-CLOSE_DIALOGUE]
                 [AGEN [[LABEL  SPEAKER]]]
                 [RECP [[LABEL  HEARER]]]
                 ?rest]]
           ?prag]
      out= [[CAT GREETING-CLOSE]
            [TPC closing]
            [PRP [[RELN close-dialogue]]]]
   end")

;;; 3. ありがとうございました*
(rws:defrwschema2 gcl-3 gcl arigatou
  "on <SEM RELN> ありがとう-THANKING
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  ありがとう-THANKING]
                 [AGEN [[LABEL  SPEAKER]]]
                 [RECP [[LABEL  HEARER]]]
                 ?rest]]
           ?prag]
      out= [[CAT GREETING-CLOSE]
            [TPC closing]
            [PRP [[RELN close-dialogue]]]]
   end")

;;; 4. どういたしまして*
(rws:defrwschema2 gcl-4 gcl douitashimashite
  "on <SEM RELN> どういたしまして-YOUR-WELCOME
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  どういたしまして-YOUR-WELCOME]
                 [AGEN [[LABEL  SPEAKER]]]
                 [RECP [[LABEL  HEARER]]]
                 ?rest]]
           ?prag]
      out= [[CAT GREETING-CLOSE]
            [TPC closing]
            [PRP [[RELN close-dialogue]]]]
   end")

;;;; end of frozen expressions ;;;
```

C.2.1.2 文末表現規則

ファイル名:AnaDialog/Rules/intention.lisp

```
; DEMAND Class ;
;;
;; ASK (SIFT)
;;
;;; 1. SUBJ は WH ですか?
(rws:defrwschema2 ask-1 ask INFORMREF
  "on <SEM OBJE RELN> INFORMREF
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  S-REQUEST]
                [AGEN !SP[[LABEL  SPEAKER]]]
                [RECP !HR[[LABEL  HEARER]]]
                [OBJE [[RELN  INFORMREF]
                       [AGEN !HR]
                       [RECP !SP]
                       [OBJE [[PARM ?wh]
                              [RESTR ?prp]
                              ?rest2]]
                       ?rest3]]
                 ?rest4]]
           [PRAG ?prag]
           ?rests]
      ==> ?wh  with :Phase :object  :Type :wh-phrase
      out= [[CAT ASK]
    [WH  ?wh]
           [TPC ?prag]
           [PRP ?prp]]
   end")

;;; 2. WH か, 教えて欲しいのですが *
(rws:defrwschema2 ask-2 ask tellme
  "on <SEM OBJE RELN> 欲しい-DESIRE
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  が-MODERATE]
                [OBJE [[RELN  欲しい-DESIRE]
                       [EXPR !SP[[LABEL  SPEAKER]]]
                       [RECP !HR[[LABEL  HEARER]]]
                       [OBJE [[RELN  教える-1]
                              [AGEN !HR]
                              [OBJE [[RELN  INFORMREF]
                                     [OBJE [[PARM  ?wh]
                                            [RESTR ?prp]
                                            ?rest1]]
                                     ?rest2]]
                              ?rest3]]
                        ?rest4]]
                ?rest5]]
           [PRAG ?prag]
           ?rests]
      ==> ?wh  with :Phase :object      :Type :wh-phrase
      out= [[CAT ASK]
            [WH  ?wh]
            [TPC ?prag]
    [PRP ?prp]]
   end")

;;
;; CONFIRM (SIFT)
;;
;;; 1. です{か|ね}?
(rws:defrwschema2 cof-1 cof INFORMIF
  "on <SEM OBJE RELN> INFORMIF
```

```
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  S-REQUEST]
                [AGEN !SP[[LABEL  SPEAKER]]]
                [RECP !HR[[LABEL  HEARER]]]
                [OBJE [[RELN  INFORMIF]
                       [AGEN !HR]
                       [RECP !SP]
                       [OBJE ?prp]
                       ?rest2]]
                ?rest3]]
           [PRAG ?prag]
           ?rests]
      out= [[CAT CONFIRM]
            [TPC ?prag]
            [PRP ?prp]]
   end")
(rws:defrwschema2 cof-2 cof ne
  "on <SEM RELN> ね-CONFIRMATION
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  ね-CONFIRMATION]
                [OBJE ?prp]
                ?rest]]
           [PRAG ?prag]
           ?rests]
      out= [[CAT CONFIRM]
            [TPC ?prag]
            [PRP ?prp]]
   end")

;;; 2. できますか?
(rws:defrwschema2 cof-3-1 cof dekiruka
  "on <SEM OBJE OBJE RELN> できる-POSSIBLE
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  S-REQUEST]
                [AGEN !SP[[LABEL  SPEAKER]]]
                [RECP !HR[[LABEL  HEARER]]]
                [OBJE [[RELN  INFORMIF]
                       [AGEN !HR]
                       [RECP !SP]
                       [OBJE [[RELN  できる-POSSIBLE]
                              [OBJE ?prp]
                              ?rest2]]
                       ?rest3]]
                ?rest4]]
           [PRAG ?prag]
           ?rests]
      out= [[CAT CONFIRM]
            [MOD POSSIBLE]
            [TPC ?prag]
            [PRP ?prp]]
   end")
(rws:defrwschema2 cof-3-2 cof kiku
  "on <SEM OBJE OBJE OBJE RELN> できる-POSSIBLE
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  S-REQUEST]
                [AGEN !SP[[LABEL  SPEAKER]]]
                [RECP !HR[[LABEL  HEARER]]]
                [OBJE [[RELN  INFORMIF]
                       [AGEN !HR]
                       [RECP !SP]
                       [OBJE [[RELN  う-GUESS]
                              [OBJE [[RELN  できる-POSSIBLE]
                                     [OBJE ?prp]
                                     ?rest1]]
                              ?rest2]]
```

```
                    ?rest3]]
               ?rest4]]
          [PRAG ?prag]
          ?rests]
     out= [[CAT CONFIRM]
           [MOD POSSIBLE]
           [TPC ?prag]
           [PRP ?prp]]
  end")

;;
;; REQUEST (SIFT)
;;
;;; 1. OBJ をお願いします
(rws:defrwschema2 req-1 req negau
  "on <SEM RELN> 願う-REQUEST
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  願う-REQUEST]
;                [ASPT  UNRL]
                 [AGEN [[LABEL  SPEAKER]]]
                 [OBJE ?obj]
                 ?rest]]
          [PRAG ?prag]
          ?rests]
      out= [[CAT REQUEST]
            [TPC ?prag]
    [PRP ?obj]]
   end")

;; 2. ACT (して) ください
(rws:defrwschema2 req-2-1 req kudasai
  "on <SEM RELN>   ください-REQUEST
. in :Phase :intention :Type :general
      in= [[SEM [[RELN  ください-REQUEST]
                 [AGEN [[LABEL  SPEAKER]]]
                 [RECP [[LABEL  HEARER]]]
;                 [ASPT  UNRL]
                 [OBJE ?prp]
                 ?rest]]
           [PRAG  ?prag]
           ?rests]
      out= [[CAT REQUEST]
            [PRP ?prp]
            [TPC ?prag]]
   end")
(rws:defrwschema2 req-2-2 req kudasai
  "on <SEM RELN>   下さい-REQUEST
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  下さい-REQUEST]
                 [AGEN [[LABEL  SPEAKER]]]
                 [RECP [[LABEL  HEARER]]]
;                 [ASPT  UNRL]
                 [OBJE ?prp]
                 ?rest]]
           [PRAG  ?prag]
           ?rests]
      out= [[CAT REQUEST]
            [PRP ?prp]
            [TPC ?prag]]
   end")

;;; 3. ACT してもらう
(rws:defrwschema2 req-3-0 req receive
  "on <SEM OBJE OBJE RELN> てもらう-RECEIVE_FAVOR
   in :Phase :intention :Type :general
```

```
      in= [[SEM [[RELN  が-MODERATE]
                [OBJE [[RELN  たい-DESIRE]
                       [OBJE [[RELN  てもらう-RECEIVE_FAVOR]
                              [OBJE ?prp]
                              ?rest2]]
                       ?rest3]]
                ?rest4]]
           [PRAG ?prag]
           ?rests]
      out= [[CAT REQUEST]
            [TPC ?prag]
            [PRP ?prp]]
   end")
(rws:defrwschema2 req-3-1 req sitemoraemasuka
  "on <SEM OBJE OBJE OBJE RELN>   てもらう-RECEIVE_FAVOR
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  S-REQUEST]
                [AGEN !SP[[LABEL  SPEAKER]]]
                [RECP !HR[[LABEL  HEARER]]]
                [OBJE [[RELN  INFORMIF]
                       [AGEN !HR]
                       [RECP !SP]
                       [OBJE [[RELN  える-POSSIBLE]
                              [OBJE [[RELN  てもらう-RECEIVE_FAVOR]
                                     [OBJE ?prp]
                                     ?rest1]]
                              ?rest2]]
                       ?rest3]]
                ?rest4]]
           [PRAG  ?prag]
           ?rests]
      out= [[CAT REQUEST]
            [PRP ?prp]
            [TPC ?prag]]
   end")
(rws:defrwschema2 req-3-2 req sitemoraemasuka
  "on <SEM OBJE OBJE OBJE OBJE RELN>   てもらう-RECEIVE_FAVOR
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  S-REQUEST]
                [AGEN !SP[[LABEL  SPEAKER]]]
                [RECP !HR[[LABEL  HEARER]]]
                [OBJE [[RELN  INFORMIF]
                       [AGEN !HR]
                       [RECP !SP]
                       [OBJE [[RELN  NEGATE]
                              [OBJE [[RELN  える-POSSIBLE]
                                     [OBJE [[RELN  てもらう-RECEIVE_FAVOR]
                                            [OBJE ?prp]
                                            ?rest2]]
                                     ?rest3]]
                              ?rest4]]
                       ?rest5]]
                ?rest6]]
           [PRAG  ?prag]
           ?rests]
      out= [[CAT REQUEST]
            [PRP ?prp]
            [TPC ?prag]]
   end")
(rws:defrwschema2 req-3-3 req sitemoraenaidesuka
  "on <SEM OBJE OBJE OBJE OBJE OBJE RELN>   てもらう-RECEIVE_FAVOR
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  S-REQUEST]
                [AGEN !SP[[LABEL  SPEAKER]]]
                [RECP !HR[[LABEL  HEARER]]]
```

```
                [OBJE [[RELN  INFORMIF]
                       [AGEN !HR]
                       [RECP !SP]
                       [OBJE [[RELN  う-GUESS]
                              [OBJE [[RELN  NEGATE]
                                     [OBJE [[RELN  える-POSSIBLE]
                                            [OBJE [[RELN  てもらう-RECEIVE_FAVOR]
                                                   [OBJE ?prp]
                                                   ?rest1]]
                                            ?rest2]]
                                     ?rest3]]
                              ?rest4]]
                       ?rest5]]
                ?rest6]]
           [PRAG  ?prag]
           ?rests]
      out= [[CAT REQUEST]
            [PRP ?prp]
            [TPC ?prag]]
   end")

;;
;; Offer
;;
;;; 1. ACT させていただきます
(rws:defrwschema2 ofa-2 ofa saseteitadakimasu
  "on <SEM RELN>  てもらう-RECEIVE_FAVOR
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  てもらう-RECEIVE_FAVOR]
;;                [RECP !HR[[LABEL  HEARER]]]
                  [OBJE [[RELN  させる-PERMISSIVE]
;;                           [AGEN !HR]
;;                           [RECP [[LABEL  SPEAKER]]]
                         [OBJE ?prp]
                         ?rest1]]
                ?rest2]]
           [PRAG  ?prag]
           ?rests]
      out= [[CAT OFFER-ACTION]
            [PRP ?prp]
            [TPC ?prag]]
   end")

; RESPONSE Class
;;
;; Inform
;;
;;; 1. したい(のですが)
(rws:defrwschema2 inf-2-1 inf want
  "on <SEM OBJE RELN> たい-DESIRE
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  が-MODERATE]
                [OBJE [[RELN  たい-DESIRE]
                       [OBJE ?prp]
                       ?rest2]]
                ?rest3]]
           [PRAG ?prag]
           ?rests]
      out= [[CAT INFORM]
            [MOD WANT]
            [TPC ?prag]
            [PRP ?prp]]
   end")
(rws:defrwschema2 inf-2-2 inf WANT
  "on <SEM OBJE RELN>  たい-DESIRE
```

```
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  思う-1]
                [OBJE [[RELN  たい-DESIRE]
                       [OBJE ?prp]
                       ?rest2]]
                ?rest3]]
          [PRAG ?prag]
          ?rests]
      out= [[CAT INFORM]
            [MOD WANT]
            [TPC ?prag]
            [PRP ?prp]]
   end")
(rws:defrwschema2 inf-2-3 inf WANT
  "on <SEM OBJE OBJE RELN>  たい-DESIRE
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  が-MODERATE]
                [OBJE [[RELN  思う-1]
                       [OBJE [[RELN  たい-DESIRE]
                              [OBJE ?prp]
                              ?rest1]]
                       ?rest2]]
                ?rest3]]
          [PRAG ?prag]
          ?rests]
      out= [[CAT INFORM]
            [MOD WANT]
            [TPC ?prag]
            [PRP ?prp]]
   end")
;; the deference from the above is CONT feature
(rws:defrwschema2 inf-2-4 inf WANT
  "on <SEM OBJE CONT RELN>  たい-DESIRE
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  が-MODERATE]
                [OBJE [[RELN  思う-1]
                       [CONT  [[RELN  たい-DESIRE]
                              [OBJE ?prp]
                              ?rest1]]
                       ?rest2]]
                ?rest3]]
          [PRAG ?prag]
          ?rests]
      out= [[CAT INFORM]
            [MOD WANT]
            [TPC ?prag]
            [PRP ?prp]]
   end")
(rws:defrwschema2 inf-2-6 inf WANT
  "on <SEM OBJE OBJE RESTR RELN>  たい-DESIRE
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  が-MODERATE]
                [OBJE [[RELN  ある-1]
                       [OBJE [[RESTR [[RELN  たい-DESIRE]
                                      [OBJE ?prp]
                                      ?rest0]]
                              ?rest1]]
                       ?rest2]]
                ?rest3]]
          [PRAG ?prag]
          ?rests]
      out= [[CAT INFORM]
            [MOD WANT]
            [TPC ?prag]
            [PRP ?prp]]
```

```
   end")

;; 2. しなければなりません
(rws:defrwschema2 inf-3-1 inf must
  "on <SEM OBJE RELN>  なくてはいけない-MUST
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  が-MODERATE]
                [OBJE [[RELN  なくてはいけない-MUST]
                       [OBJE ?prp]
                       ?rest1]]
                ?rest2]]
           [PRAG  ?prag]
           ?rests]
      out= [[CAT INFORM]
            [MOD MUST]
            [PRP ?prp]
            [TPC ?prag]]
   end")
(rws:defrwschema2 inf-3-2 inf must
  "on <SEM RELN>  なくてはいけない-MUST
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  なくてはいけない-MUST]
                [OBJE ?prp]
                ?rest2]]
           [PRAG  ?prag]
           ?rests]
      out= [[CAT INFORM]
            [MOD MUST]
            [PRP ?prp]
            [TPC ?prag]]
   end")

;;
;; REJECT
;;
;;; 1. できません
(rws:defrwschema2 rja-1 rja dekinai
  "on <SEM OBJE RELN>  できる-POSSIBLE
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  NEGATE]
                [OBJE [[RELN  できる-POSSIBLE]
                       [OBJE ?prp]
                       ?rest1]]
                ?rest2]]
           [PRAG  ?prag]
           ?rests]
      out= [[CAT REJECT-ACTION]
            [PRP ?prp]
            [TPC ?prag]]
   end")

;;; OTHERS ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(rws:defrwschema2 etc-1 etc compound-s
  "on <SEM RELN>  連用形接続
   in :Phase :intention :Type :general
      in= [[SEM [[RELN  連用形接続]
                [ARG1 ?arg1]
                [ARG2 ?arg2]
                ?rest1]]
           [PRAG  ?prag]
           ?rests]
      set ?new1 to [[SEM ?arg1] [PRAG ?prag]]
      set ?new2 to [[SEM ?arg2] [PRAG ?prag]]
      out= [[CAT COMPOUND-CONT]
            [ARG-1 ?new1]
```

```
        [ARG-2 ?new2]
        [TPC  ?prag]]
end")
```

C.2.1.3 デフォールト規則

ファイル名:AnaDialog/Rules/intention.lisp(環境名 :Type = :default)

```
;;
;; Type :DEFAULT
;;
;;; OFFER-ACTION ; ACT します
(rws:defrwschema2 ofa-1 ofa simasu
  "on <SEM AGEN LABEL> SPEAKER
   in :Phase :intention :Type :default
      in= [[SEM [[ASPT  UNRL]
                [AGEN [[LABEL  SPEAKER]]]
 ?rest1]]
          [PRAG  ?prag]
          ?rests]
      set ?prp to ?input.sem
      out= [[CAT OFFER-ACTION]
            [PRP ?prp]
    [TPC ?prag]]
   end")

;;; INFORM (SIFT)
(rws:defrwschema2 inf-1-1 inf inform
  "on <SEM RELN>  :unspecified
   in :Phase :intention :Type :default
      in= [[SEM ?prp]
           [PRAG ?prag]
           ?rests]
      out= [[CAT INFORM]
            [TPC ?prag]
            [PRP ?prp]]
   end")
(rws:defrwschema2 inf-1-2 inf inform
  "on <SEM RELN>  が-MODERATE
   in :Phase :intention :Type :default
      in= [[SEM [[RELN  が-MODERATE]
                [OBJE  ?prp]
                ?rest2]]
           [PRAG ?prag]
           ?rests]
      if input.SEM has ASPT then
        set ?aspt to input.SEM.ASPT
        add {[ASPT ?aspt]} to ?prp
      endif
      set ?output to [[SEM  ?prp]
                      [PRAG ?prag]]
      => ?output
      out= ?output
   end")

;; NEGATE ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(rws:defrwschema2 ngm-1 ngt NEGATE
  "on <SEM RELN> NEGATE
   in :Phase :intention :Type :default
      in= [[SEM [[RELN  NEGATE]
                [OBJE  ?prp]
                ?rest]]
           [PRAG ?prag]
           ?rests]
      if input.SEM has ASPT then
        set ?aspt to input.SEM.ASPT
        add {[ASPT ?aspt]} to ?prp
      endif
      set ?output to [[SEM  ?prp]
```

```
                    [PRAG ?prag]]
    => ?output
    add {[MOD NEGATE]} to ?output
    out= ?output
  end")

;;;; END RWS RULES ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

C.2.2 話題属性規則

```
;;;
;;; Transform rules of cat and prp
;;;

(setq cate::*cate-output-rules*
      '((ask      . cate-ask-rule)
(confirm . cate-confirm-rule)
(request . cate-request-rule)
(inform  . cate-inform-rule)
(:unknown . cate-unknown-rule)
))

(defun cate-map-urep-list (cats urep)
  (mapcar #'(lambda (cat) (cons cat (cdr urep)))
  cats))
(defmacro push-if-not-find (x list)
  `(when (not (find ,x ,list :test #'equal))
    (push ,x ,list)))
(defun cate-prp-case-value (pred cases prp &aux temp posi)
  (dolist (case cases temp)
    (when (setq posi (search (list case) (cate::get-pg pred)))
      (if (cate::urep-var-p (nth posi prp))
  (setq temp (nth posi prp))
  (return (nth posi prp))))))
;;
;; Each rule
;;
(defun cate-unknown-rule (fs urep)
  (declare (ignore fs))
  (list (cons :unknown (cdr urep))))
;; ask
(defun cate-ask-rule (fs urep)
  (let* ((wh   (cate::fs-to-urep-value fs '(wh) :without-sp t))
 (cmod (cate::fs-to-urep-value fs '(mod) :without-sp t))
 (prp  (nth 4 urep))
 (pred (if (listp prp) (car prp) prp))
 ans)
    (cond
      ;;  Strong: if pred is "be-verb" and wh-phase is valuable
      ;;   then ASK-VALUE
      ((and (find pred cate::*is-verb*)
    (find wh   (append cate::*wh-value* cate::*np-value*)))
       (push-if-not-find
(cons 'ASK-VALUE (cdr urep)) ans))
      ((and (find pred cate::*v-action*)
    (find wh   (append cate::*wh-behavior* cate::*np-behavior*))
    (find cmod cate::*mod-oblig*))
       (push-if-not-find
(cons 'ASK-ACTION (cdr urep)) ans))
      (t
       (when (find wh (append cate::*wh-value* cate::*np-value*))
 (push-if-not-find
  (cons 'ASK-VALUE (cdr urep)) ans))
       (when (find wh (append cate::*wh-behavior* cate::*np-behavior*))
 (push-if-not-find
  (cons 'ASK-ACTION (cdr urep)) ans))
       (when (find pred cate::*v-statement*)
 (push-if-not-find
  (cons 'ASK-STATEMENT (cdr urep)) ans))))
    (if ans ans
(cate-map-urep-list
 '(ASK-VALUE ASK-STATEMENT ASK-ACTION)
 urep))))
```

---------- 中略 ----------

```
;; transform proposition
(defun cate-interction-prp-rule (pred urep)
  (setf (nth 4 urep)
(cate-prp-case-value pred '(RANG OBJE) (nth 4 urep)))
  (cate::urep-with-atomic-prp urep))

;;; end of file ;;;
```

C.2.3 スピーカテーブル

音声認識候補絞り込みシステム参照。(C.3.3参照)

C.2.4 命題格要素辞書

階層型プラン認識システム [21] 参照

C.2.5 概念ネットワーク辞書

階層型プラン認識システム [21] 参照

C.2.6 素生構造

```
D1-1 SP1 もしもし
[[SEM [[RELN  もしもし-OPEN_DIALOGUE]
      [ASPT  -]
      [AGEN !X01[[LABEL  *SPEAKER*]]]
      [RECP !X02[[LABEL  *HEARER*]]]]]
 [PRAG [[SPEAKER !X01]
        [HEARER !X02]]]]

D1-2 SP1 そちらは会議事務局ですか
[[SEM [[RELN  S-REQUEST]
       [AGEN !X05[[LABEL  *SPEAKER*]]]
       [RECP !X06[[LABEL  *HEARER*]]]
       [OBJE [[RELN  INFORMIF]
              [AGEN !X06]
              [RECP !X05]
              [OBJE !X07[[RELN  だ-IDENTICAL]
                         [ASPT  STAT]
                         [OBJE !X04[[PARM !X03[]]
                                    [RESTR [[RELN  そちら-1]
                                            [ENTITY !X03]]]]]
                         [IDEN [[PARM !X02[]]
                                [RESTR [[RELN  NAMED]
                                        [ENTITY !X02]
                                        [IDEN  会議事務局-1]]]]]]]]]]
 [PRAG [[RESTR [[IN [[FIRST [[RELN  POLITE]
                             [AGEN !X05]
                             [RECP !X06]]]
                    [REST !X01[]]]]
               [OUT !X01]]]
        [TOPIC [[IN [[FIRST [[FOCUS !X04]
                             [TOPIC-MOD  HA]
                             [SCOPE !X07]]]
                    [REST []]]]
               [OUT []]]]
        [PRSP-TERMS [[IN []]
                    [OUT []]]]
        [SPEAKER !X05]
        [HEARER !X06]
        [ASPE [[IN []]
               [OUT []]]]]]]

       ---------- 中略 ----------


D1-20 SP1 それでは失礼します
[[SEM [[RELN  失礼する-CLOSE_DIALOGUE]
       [ASPT  UNRL]
       [AGEN !X04[[LABEL  *SPEAKER*]]]
       [RECP !X03[[LABEL  *HEARER*]]]
       [INST [[PARM !X05[]]
              [RESTR [[RELN  それ-1]
                      [ENTITY !X05]]]]]]]
 [PRAG [[RESTR [[IN [[FIRST [[RELN  POLITE]
                             [AGEN !X04]
                             [RECP !X03]]]
                    [REST !X02[]]]]
               [OUT !X02]]]
        [TOPIC [[IN [[FIRST [[FOCUS []]
                             [TOPIC-MOD  HA]
                             [SCOPE []]]]
                    [REST []]]]
               [OUT []]]]
        [PRSP-TERMS [[IN []]
```

```
                    [OUT []]]]
        [SPEAKER !X04]
        [HEARER !X03]
        [ASPE [[IN !X01[]]
               [OUT !X01]]]]]]
[[SEM !X13[[RELN  失礼する -CLOSE_DIALOGUE]
          [ASPT  UNRL]
          [AGEN !X03[[LABEL  *SPEAKER*]]]
          [RECP !X07[[LABEL  *HEARER*]]]
          [CONECT !X48[[PARM !X04[]]
                       [RESTR [[RELN  それでは -1]
                               [ENTITY !X04]]]]]]]
 [PRAG [[RESTR [[IN !X23[[FIRST [[RELN  POLITE]
                                 [AGEN !X03]
                                 [RECP !X07]]]
                         [REST !X08[]]]]
               [OUT !X08]]]
        [TOPIC [[IN !X53[]]
                [OUT !X44[]]]]
        [PRSP-TERMS [[IN !X54[]]
                     [OUT !X45[]]]]
        [SPEAKER !X03]
        [HEARER !X07]
        [ASPE [[IN !X09[]]
               [OUT !X09]]]]]]
```

C.3 音声認識候補絞り込みシステム用

C.3.1 音声認識候補

以下に、ATR サンプル対話に対する音声認識結果の一部を掲載する。

```
;;;
;;; kakariuke analysis results for inputs of Select
;;;
;   modifying lattice-sentence number for uniforming with DIANA
;
(IN-PACKAGE "BUNSETSULATTICE")

(DEF-BUNSETSU-LATTICE L1
  V1 V999
  ((E1 V1 V999 "もしもし" (:PROB 0.875027))
   (E2 V1 V2 "五")
   (E3 V2 V3 "十")
   (E4 V3 V999 "四" (:PROB 0.109378)))
  (:NFRAMES 205 :INPUT "MOSHIMOSHI"))

(DEF-LATTICE-SENTENCE S1
  (L1 ))

()

(DEF-BUNSETSU-LATTICE L2
  V1 V999
  ((E1 V1 V999 "そちら" (:PROB 0.013672))
   (E2 V1 V2 "そちら")
   (E3 V2 V999 "は" (:PROB 0.875027)))
  (:NFRAMES 229 :INPUT "SOCHIRAWA"))

(DEF-BUNSETSU-LATTICE L3
  V1 V999
  ((E1 V1 V2 "会議事務局")
   (E2 V2 V3 "です")
   (E3 V3 V999 "か" (:PROB 0.875027)))
  (:NFRAMES 421 :INPUT "KAIGIJIMUKYOKUDESUKA"))

(DEF-LATTICE-SENTENCE S2
  (L2 L3 ))

()
```

```
(DEF-BUNSETSU-LATTICE L4
  V1 V999
  ((E1 V1 V999 "八" (:PROB 0.838311))
   (E2 V1 V999 "はい" (:PROB 0.129431)))
  (:NFRAMES 93 :INPUT "HAI"))

(DEF-LATTICE-SENTENCE S3
  (L4 ))

()


(DEF-BUNSETSU-LATTICE L5
  V1 V999
  ((E1 V1 V2 "そう")
   (E2 V2 V999 "です" (:PROB 0.875027))
   (E3 V2 V3 "です")
   (E4 V3 V999 "か" (:PROB 0.109378)))
  (:NFRAMES 229 :INPUT "SOUDESU"))

(DEF-LATTICE-SENTENCE S4
  (L5 ))

()
```

----- 中略 -----

```
(DEF-BUNSETSU-LATTICE 1353
  v1 v999
  ((e1 v1 v2 "失礼")
   (e2 v2 v3 "し")
   (e3 v3 v999 "ます" (:prob 0.875027)))
  (:nframes 299 :input "shitsureishimasu"))

(DEF-LATTICE-SENTENCE s138
  (1352 1353 ))

()

;;; end kakari-data ;;;
```

C.3.2 ラティステーブル

以下に、ATR サンプル対話に対する発話 ID と文節ラティスの対応テーブルの一部を掲載する。

```
;;
;; sentence id correspondances
;;
(
;;; Kaiwa A
(Da-1 . BUNSETSULATTICE::S1) ; もしもし
(Da-2 . BUNSETSULATTICE::S2) ; そちらは会議事務局ですか
(Da-3 . BUNSETSULATTICE::S3) ; はい
(Da-4 . BUNSETSULATTICE::S4) ; そうです
(Da-5 . BUNSETSULATTICE::S5) ; 会議に申込みたいのですが"
(Da-6 . BUNSETSULATTICE::S6) ; 登録用紙は既にお持ちでしょうか
(Da-7 . BUNSETSULATTICE::S7) ; いいえ
(Da-8 . BUNSETSULATTICE::S8) ; まだです
(Da-9 . BUNSETSULATTICE::S9) ; 分かりました
(Da-10 . BUNSETSULATTICE::S10) ; それでは登録用紙をお送り致します
(Da-11 . BUNSETSULATTICE::S11) ; ご住所とお名前をお願いします
(Da-12 . BUNSETSULATTICE::S12) ; 住所は大阪市北区茶屋町二十三です
(Da-13 . BUNSETSULATTICE::S13) ; 名前は鈴木真弓です
(Da-14 . BUNSETSULATTICE::S14) ; 分かりました
(Da-15 . BUNSETSULATTICE::S15) ; 登録用紙を至急送らせていただきます
(Da-16 . BUNSETSULATTICE::S16) ; 分からない点がございましたらいつでもお聞き下さい
(Da-17 . BUNSETSULATTICE::S17) ; 有難うございます
(Da-18 . BUNSETSULATTICE::S18) ; それでは失礼します
(Da-19 . BUNSETSULATTICE::S19) ; どうも失礼致します
;;; Kaiwa B
(Db-1 . BUNSETSULATTICE::S20)    ; もしもし
(Db-2 . BUNSETSULATTICE::S21) ; こちらは会議事務局です
(Db-3 . BUNSETSULATTICE::S22) ; 会議に参加したいのですが
(Db-4 . BUNSETSULATTICE::S23) ; どうすればよろしいですか
(Db-5 . BUNSETSULATTICE::S24) ; まず登録用紙で手続きをしていただかなくてはなりませんが
(Db-6 . BUNSETSULATTICE::S25) ; もう登録用紙はお持ちでしょうか
(Db-7 . BUNSETSULATTICE::S26) ; まだです


                    ----- 中略 -----

)
;;; end table ;;;
```

C.3.3 スピーカテーブル

以下に、ATR サンプル対話に対する話し手と発話 ID の対応テーブルの一部を掲載する。

```
;;
;; SPeakers list
;;
(
;;; Kaiwa A
(Da-1 . SP1) ; もしもし
(Da-2 . SP1) ; そちらは会議事務局ですか
(Da-3 . SP2) ; はい
(Da-4 . SP2) ; そうです
(Da-5 . SP1) ; 会議に申込みたいのですが"
(Da-6 . SP2) ; 登録用紙は既にお持ちでしょうか
(Da-7 . SP1) ; いいえ
(Da-8 . SP1) ; まだです
(Da-9 . SP2) ; 分かりました
(Da-10 . SP2) ; それでは登録用紙をお送り致します
(Da-11 . SP2) ; ご住所とお名前をお願いします
(Da-12 . SP1) ; 住所は大阪市北区茶屋町二十三です
(Da-13 . SP1) ; 名前は鈴木真弓です
(Da-14 . SP2) ; 分かりました
(Da-15 . SP2) ; 登録用紙を至急送らせていただきます
(Da-16 . SP2) ; 分からない点がございましたらいつでもお聞き下さい
(Da-17 . SP1) ; 有難うございます
(Da-18 . SP1) ; それでは失礼します
(Da-19 . SP2) ; どうも失礼致します
;;; Kaiwa B
(Db-1 . SP2)    ; もしもし
(Db-2 . SP2) ; こちらは会議事務局です

                    ----- 中略 -----

(D10-35 . SP2) ; 分かりました
(D10-36 . SP2) ; 京都プリンスホテルに八月四日から八日まで一人部屋をお取りしました
(D10-37 . SP1) ; どうもありがとうございました
(D10-38 . SP1) ; 失礼します
)
```

C.3.4 表層表現ネットワーク辞書

以下に、ATR サンプル対話に対する表層表現ネットワーク辞書の一部を掲載する。

```
;;;
;;; Word nodes
;;;


(np::defword 住所と名前
    (eq-if-polite ご住所とお名前)
    )

(np::defword 住所
    (eq-if-polite 御住所 ご住所)
    )


(np::defword 名前
    (eq-if-polite お名前)
    )


(np::defword 用紙
    (eq-short 登録用紙)
    )


(np::defword ASK-ACTION-prag
    (eq-tail
     ですか でしょうか
     教えて下さい        ;; there is no corresponding rule
     ))


(np::defword ASK-VALUE-prag
  (eq-TAIL
    お願いします
    お願い致します
    ですか でしょうか
    教えて下さい
    聞くことができますか
    お伺いできますか
    お伺いできますでしょうか
    お聞きしたいのですが))
```

```
(np::defword ASK-STATEMENT-prag
  (eq-TAIL
    ですか
    でしょうか))


(np::defword CONFIRM-ACTION-prag
  (eq-TAIL
    できますか
    ばよいのですか
    よろしいでしょうか   ;; diff is :prp :action
))

            ----- 中略 -----


(np::defword ACCEPT-ACTION-prag
  (EQ-FROZEN
    分かりました
    わかりました)
  (EQ-TAIL
    ます        ;; there is no corresponding rule
    問題ありません))


(np::defword REJECT-ACTION-prag
  (EQ-TAIL
    できません))


(np::defword ACCEPT-OFFER-prag
  (EQ-TAIL
    お願いします
    有難うございます))


(np::defword REJECT-OFFER-prag
  (EQ-TAIL
    結構です))
```

```
(np::defword CONFIRMATION-prag
  (eq-FROZEN
    分かりました
    わかりました
    そうですか))

(np::defword GREETING-OPEN-prag
  (eq-FROZEN
    もしもし
    はい))

(np::defword GREETING-CLOSE-prag
  (eq-FROZEN
    さようなら
    失礼します
    失礼いたします
    失礼致します)
  (eq-TAIL
    失礼します
    失礼いたします
    失礼致します
    有難うございました
    どういたしまして
    ありがとうございました))
```

参考文献

[1] 好田 正紀：“音声認識における言語処理”, 人工知能学会誌,**3**, 4, pp.424-430(1988-7)

[2] 柿ヶ原 康二, 森元 ：“SL-TRANS における文節候補の削減 – 係受け関係を用いた文節候補の選択 –”, 第 39 回情処学全大, 4G-6 (1989-10)

[3] Hauptmann, A. G., Young, S. R. and Ward, W. H.: “Using Dialog-Level Knowledge Sources to Improve Speech Recognition”, Proceedings of AAAI'88, pp.729-733 (1988-8)

[4] 堀 雅洋, 辻野 克彦, 溝口 理一郎, 角所 収：“音声理解システム SPURT-I – 動的クラスタリング方式と文節発声による性能評価 –”, 信学論 (D-II),**J72-D-II**, 8, pp.1291-1298(1989-8)

[5] 飯田 仁, 有田 英一：“4 階層プラン認識モデルを使った対話の理解”, 情処学論, **31**, 6, pp.810-821(1990-6)

[6] 山岡 孝行, 有田 英一, 飯田 仁：“階層型プラン認識モデルによる対話理解と次発話の予測手法”, 情処学「談話理解とその応用」シンポジウム, pp.53-64 (1989-11)

[7] Yamaoka,T. and Iida,H.: “A Method to Predict the Next Utterance Using a Four-layered Plan Recognition Model”, Proceedings of ECAI'90, pp.726-731 (1990-8)

[8] 山岡 孝行, 飯田 仁：“文脈を考慮した音声認識絞り込み手法”, 情処学自然言語処理技報,NL-78-16 (1990-7)

[9] Allen,J.F. and Perrault,C.R.: “Analyzing Intention in Uterances”, Artificial Intelligence, **15**, pp.143-178 (1980)

[10] Grice,H.P.: “Logic and Conversation”, Syntax and Semantics vol.3, Academic Press (1975)

[11] Hasegawa,T.: “A Rule Application Control Method in a Lexicon-driven Transfer Model of a Dialogue Translation System”, Proceedings of ECAI'90, pp.336-338 (1990-8)

[12] Kume,M.,Sato,G.K. and Yoshimoto,K.: "A Descriptive Framework for Translating Speaker's Meaning", Proceedings of European Chapter of ACL'89, pp.264-271 (1989-4)

[13] 野垣内 出, 飯田 仁: "キーボード会話における名詞句の同一性の理解", 情処学自然言語処理技報,NL-72-1 (1989-5)

[14] 有田 英一, 山岡 孝行, 飯田 仁: "電話対話における次発話内の名詞句表現の予測", 情処学自然言語処理技法, NL-81-13 (1991-1)

[15] 北 研二, 坂野 俊哉, 保坂 順子, 川端 豪: "SL-TRANS における文節音声認識 – HMM 音韻認識と LR 構文解析法による文節音声認識 –", 第 39 回情処学全大, 4G-5 (1989-10)

[16] 小暮 潔, 堂坂 浩二, 加藤 進: "SL-TRANS における言語解析", 第 39 回情処学全大, 4G-7 (1989-10)

[17] 久米 雅子, 小暮 潔: "発話意図の翻訳のための発話行為推論部", 第 38 回情処学全大, (1989)

[18] Zajac, R.: "A Transfer Model Using a Typed Feature Structure Rewriting System with Inheritance", 27th ACL, (1989)

[19] Iida,H., Yamaoka,T. and Arita,H.: "Predicting the Next Utterance Linguistic Expressions Using Contextual Information", IEICE TRANS. INF.&SYST.,vol.E76-D,NO.1,(1993)

[20] 山岡 孝行, 飯田 仁: "階層型プラン認識モデルを利用した次発話予測手法 - 話し手の意図を表す表現についての音声認識結果曖昧性の解消", 電子情報通信学会論文誌,vol.J76-DII,No.6,(1993-6)

[21] 大井 耕三, 西村 仁志, 飯田 仁: "プラン認識プログラム **LAYLA**- 利用マニュアル -", ATR テクニカルレポート, TR-I-0323,(1993)

[22] 鈴木 雅実, 古崎 博久: "言語変換処理系解説書 - 素性構造書き換えシステム 改良版とユーザーズマニュアル -", ATR テクニカルレポート, TR-I-0330,(1993)

索引