TR-I-0256

談話構造解析モジュール DIANA
DIscourse ANAlyzor

山岡孝行　　菊井玄一郎
Takayuki YAMAOKA　　Gen-ichiro KIKUI

1992.3

概要

本稿では、対話翻訳システム *ASURA* において、談話理解を行ない文脈情報を提供するための談話構造解析モジュール DIANA(DIscourse ANAlyzor) の構成と処理の概要および簡単な利用方法について述べる。DIANA は、1) 語用論的知識に基づく情報伝達行為の認定、2) 発話対の知識を利用したプラン認識による発話クラスタの同定と発話の解釈、および 3) 談話構造の管理と文脈情報の提供を行なう。

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

目次

第 1 章

はじめに

対話を構成する各発話は本質的に文脈依存である。従って、高品質な対話翻訳の実現にはいわゆる文脈の処理が不可欠となる。ATR では、文脈に依存した高品質な翻訳を目指し、対話翻訳制御機構の実装を行なっている [菊井 92]。本稿では、ATR 対話翻訳システム (*ASURA*) の文脈処理機構 (談話構造解析モジュール DIANA) について述べる。

対象文脈の範囲　一口に“文脈”といってもその意味する範囲は広範であり、また一般的に不明確である。まず、DIANA の対象とする文脈を定義する。

- 発話対に基づく発話クラスタ [山岡 90][定延 90TR]

すなわち、ここで扱う文脈とは、対話における情報の授受の最小単位 (談話セグメント) のみである[1]。従って、ここでの文脈処理とは、発話の系列から上記の意味での発話クラスタを求めることであり、また (文脈における) 発話の理解とは、ある入力発話の属する発話クラスタを認定し、その範囲で発話を解釈することになる。

このように局所的な文脈のみを対象とする理由はいくつか上げられる:

1. 対象対話の性質
   - 機械翻訳や自然言語理解においてよく問題の対象となる省略や断片的発話は、局所的文脈で理解できるものが多い。
   - その局所的文脈の中における情報の授受の方法に関しても、質問－応答の埋め込み等、様々なバリエーションが見られる。
2. 処理能力
   - (話題の流れなどの知識である) いわゆるドメインプランは広範かつ繁雑な知識であり、一般的に記述が困難である。

[1]例外的に、対話の始まり終りに関しては、ダイアログプラン・コミュニケーションプラン [飯田 90] の一部を援用する。これは、この部分のみであれば、複雑な処理や知識を導入することなく談話構造解析が行なうことができ、発話の解釈の特定が容易である。

- 従って、ドメインプランを扱おうとすれば、探索の組合せ的爆発が起こり、現実的解を導けないことが多い。

このような見地に立てば、その記述が繁雑になるドメインプランを導入せずとも、文脈に依存した翻訳としてある程度満足のいく解答を求めかつ、現実的な処理が行うことが可能である。

理解された (局所的) 文脈は、対話構造 (談話構造) としてシステム内で保持・管理される。

文脈情報　対話翻訳制御機構からの要請と上の文脈の定義から、DIANA で提供する文脈情報を以下のものに限定する:

- 任意の発話についての
  - 所属する発話クラスタ,
  - その発話クラスタにおける発話クラス.

談話構造解析モジュールの機能　以上の要請から、DIANA の機能は以下のようになる:

1. 発話クラスタにおける入力発話の解釈,
2. 発話クラスタ (対話構造) の構築と管理,
3. 発話クラスタと発話クラス情報の提供.

アプローチ　DIANA では、以上の機能を実現を以下のようなアプローチを基に行なう:

- 一文単位での語用論的知識に基づく情報伝達行為の認定,
- 発話対の知識であるインタラクションプランによるプラン認識手法.

以下では、第2章で DIANA のデータ構造と大まかなシステム構成を示した後、第3章で内部の処理の概要について述べる。第4章は、DIANA を利用する際のリファレンスである。さらに、付録として、DIANA 単体で処理実験を行なう際のプログラムとその利用方法について説明する。[2]

[2]文脈処理を担当する方は、少なくともこの付録に与えられている機能については、十分に理解しておくことが望まれる。

第 2 章

システム構成

DIANA のシステム構成を図 2.1 に示す。図の太線は、DIANA のサブシステムを、円柱は知識ベース・データベースを表している。さらに、対話翻訳制御部 (DTM) との情報授受のための、インタフェースを備えている。また、受け渡されるデータは、semantic representation は意味表現素性構造[1]、CAT&PRP は発話の表現 (情報伝達行為と命題内容) を、contextual information は文脈情報を表す。各々について、詳細を以下で説明する。

2.1 データ構造

DIANA 内部で扱われるデータは、基本的に以下の 4 つである:

- 素性構造,
- 発話の表現,
- インタラクションプラン,
- ゴールスタック.

2.1.1 素性構造

DIANA の入力は、日本語解析機構 NADINE の出力である素性構造意味表現である。この意味表現の詳細については、参考文献 [久米 90TR] 参照。また、入力となる素性構造は、長谷川素性構造書換えシステム [長谷川 90TR] により扱い可能なものでなければならない。このデータ構造の取り扱いについては、[長谷川 90TR] または [山岡 92TR1] 参照。

[1]正確には、sem 素性と prag 素性

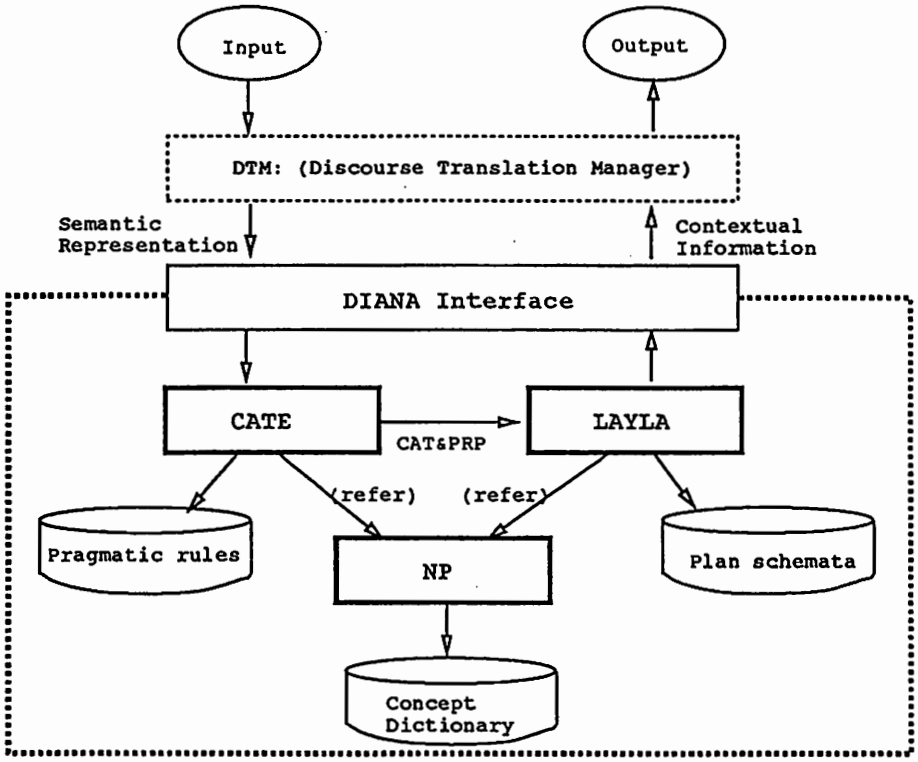

図 2.1: DIANA の構成

表 2.1: 情報伝達行為一覧 (発話対の構成)

Demand Class	Response Class	Acknowledge Class
ASK-ACTION	INFORM-ACTION	Confirmation
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-OPEN	(GREETING-OPEN)	
GREETING-CLOSE	GREETING-CLOSE	

2.1.2 発話の表現

発話の表現は、ある発話の一つの解釈を表すものである。これは、情報伝達行為解析の結果として出力し、談話構造解析の入力となる。この表現は、情報伝達における発話の意図を中心に解釈したデータ表現である。発話の意図は、情報伝達行為として抽象的に解釈する。このデータの書式は、以下のようになる:

発話の表現 ::= (情報伝達行為 話し手 聞き手 トピック 命題内容)

情報伝達行為と命題内容は以下で説明する。話し手・聞き手はそれぞれを表すシンボル、またトピックは命題内容に現れるシンボル (語彙記述の見出しあるいは話し手・聞き手のシンボル) が入る。

情報伝達行為 (CAT)

情報伝達行為は、対話における発話行為の一種である。それは、発話表現のいわゆる意図の部分と話題を表す部分の関係から記述される。DIANA で扱っている情報伝達行為タイプの一覧を表 2.1 に示す (表では、これらの接続関係 (発話対) も示している)。システム内では、情報伝達行為は常にシンボルとして扱われる。

命題内容 (PRP)

命題内容は、発話が運んでいる情報内容である。それは、一般的に発話文章内の述部の本動詞とその格要素により表現されることが多い。DIANA では、命題内容を 1 つの述語とその格要素からなるリストで表現する。リストの要素には、日本語解析機構の語彙記述 (辞書) の見出しとなっているシンボルが入る。

表 2.2: 構造体アクションのスロット

スロット名	データタイプ	内容
header	list	行為の見出し
decomposition	list of lists	副行為の系列
precondition	list of lists	行為実行の前提条件
effect	list of lists	行為実行による効果
constraint	-	行為実行の制約条件
type	symbol	行為の階層クラス

2.1.3 インタラクションプラン

インタラクションプランは、発話対を表した知識である。発話対は、協調的情報伝達における談話構造の構成単位となる発話クラスタを構成するための、例えば質問-応答のような、情報伝達行為間の関係を記述したものである。DIANA で設定している発話対の一覧が表 2.1 に示たものである。

インタラクションプランは、階層型プラン認識モデルにおける最も下位の知識であり、スキーマ形式で表現される。プランスキーマは以下に説明するデータ構造アクションで記述する。また、談話構造解析 (プラン認識) における発話もアクションとして扱う。

アクション (入力発話・プランスキーマ)

アクションは、現実世界における行為の計算機上への記述を表すデータ構造である。プラン認識は、基本的に、プランニング (行動計画) の裏返しであるため、その扱うデータは行為を表現していることが期待されている。アクションは、スキーマ形式で記述し、表 2.2のようなスロットを持つ。

アクションの見出し (header) は、ある行為の表現であり、それは述語表現形式のリストになる。decomposition は、その行為を実現するための副行為の系列であり、その副行為の見出しが要素となる。precondition, effect は、それぞれ実行前に充足されていなければならない前提条件と実行後に与えられる効果を記述するスロットであり、状態の記述 (行為の見出しの記述と同型式) が入る。constraint は、プラン適用における制約条件を記述する[2]。type は、行為のタイプであり、階層型プラン認識モデルにおける行為の所属するクラスを与える。

以下にインタラクションプランのアクション記述の例を示す。これは、値の入手を行なう発話対 (GET-VALUE-UNIT) の例である[3]。

---

[2]現在のところ、この制約条件には、唯一 :ORDERED というシンボルのみを与えることができる。これは、プランの decomposition の実行に順序 (リストの先頭から終りの方向) がある場合に指定する。

[3]実際の記述には、LISP マクロを利用する [山岡 92TR2].

表 2.3: 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)

```
#S(ACTION
    TYPE     :INTERACTION-PLAN
    HEADER   (GET-VALUE-UNIT ?SP ?HR ?(OBJ value))
    PRECONDTION    NIL
    DELETE-LIST    NIL
    EFFECT         ((KNOW ?SP (is ?OBJ ?VAL)))
    DECOMPOSITION ((ASK-VALUE     ?SP ?HR ?OBJ (IS ?OBJ ?VAL))
                   (INFORM-VALUE ?HR ?SP ?OBJ (IS ?OBJ ?VAL))
                   (CONFIRMATION ?SP ?HR ?OBJ (IS ?OBJ ?VAL)))
    CONSTRAINT     NIL
    )
```

プラン認識のトップレベルの入力である発話自体もアクションで表現できる。発話を表すアクションは、一般的に precondision, effect, decomposition スロットの値を持たないアクションである。DIANA では、発話の所属する階層クラスとして、:input を与えている。これらの見出しは、入力された発話の表現の内容 (そのもの) である。

2.1.4 ゴールスタック (対話構造 = 理解状態)

ゴールスタックは、談話構造を保持・管理する構造体である。その内容は、表 2.3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 (statements list のみ) が要素としてはいる。

incomplete は、未充足のプランスキーマを格納するプッシュダウンスタックである。complete1 は、すべてのスロットが充足されたプランスキーマを格納するプッシュダウンスタックである。(発話を表すアクションは常にここにはいる。) また、complete2 は、プラン自体は未充足であるが、既に談話セグメントが他へ移ってしまったものを格納しておく。この中の要素は、その各スロットが 1) 暗黙的に了解されている、あるいは 2) その対話の内容あるいは対話参加者にとって重要でない行為・状態である、などの解釈ができよう。さらに、statements には、充足されたプランの効果を格納しておく。これらは、残りの対話において共通理解事項として扱うことができる。

2.2 サブシステム

2.2.1 情報伝達行為認定システム CATE

情報伝達行為認定システム CATE(Communicative Act Type Extractor) [Yamaoka91] [山岡 92TR1] は、日本語解析結果である意味表現素性構造を入力とし、情報伝達行為とその命題内容による発話の表現を出力する。CATE は一文を越えたいわゆる文脈の情報を利用せず、発話における語用論的知識のみからその発話の解釈を行なう。従って、一つの発話の情報伝達行為が一意に決定できるとは限らない。

2.2.2 階層型プラン認識システム LAYLA

階層型プラン認識システム LAYLA(LAYered pLAn recognizor) [飯田 90] [山岡 92TR2] では、発話の表現と前文脈までに構築されている対話構造を入力として、プラン認識を行ない、新たな－入力発話の解釈を踏まえた－対話構造を出力する。プラン認識のための知識は、プランスキーマとして記述される。また、入力となる発話の表現・対話構造は 1 つとは限らない。LAYLA は、ゴールスタックによる対話構造管理機構を併せ持つ。

2.2.3 概念・表現ネットワーク検索システム NP

概念・表現ネットワーク検索システム NP[有田 91] は、名詞句・述語の概念・表層表現の関係を記述した知識であるネットワークの検索機構。例えば、ある表現から関連する概念の集合を求めたり、概念同士の関係を求める時に利用される。

2.3 知識・データベース

2.3.1 情報伝達行為認定のための語用論規則ベース

情報伝達行為認定のための語用論規則ベース (Pragmatic rules) [Yamaoka91] は、CATE で利用される情報伝達行為抽出のための語用論知識ベースである。基本的に、長谷川 RWS 素性構造書換え規則 [長谷川 90TR] により記述する。発話の意図部分に関する知識と話題の属性 (命題部分) に関する知識に区別される。

CATE ver3(1992.2 現在) では、意図部分に関する知識は長谷川 RWS 書式により、また話題の属性に関する知識は NP を利用した Lisp code で実装している。

2.3.2 プランスキーマ

プランスキーマ (Plan schemata)[飯田 90][山岡 90] は、対話運用や話題領域に関する知識 (プラン) ベースである。(現在のところ、DIANA で扱っているのはインタラクションプランのみ)

2.3.3 概念辞書

概念辞書 (シソーラス)(Concept dictionary)[有田 91] は、名詞句や述語 (動詞) に関する概念間・表層表現間およびそれらの間の関係を記述した辞書である。システム内では、概念ネットワークと表現ネットワークに展開される。

(現在のところ、DIANA の構成においては、概念辞書のみ利用する。また辞書のエントリは、サンプル会話 1–10 までの出現表現のみを保持する。)

第 3 章

処理の概要

DIANA の処理には、大きく二つのものがある。入力を解析して談話構造を管理する談話構造解析と、対話翻訳制御部からの要請により、保持している談話構造を参照して、文脈情報を獲得・提供する処理である。

3.1 談話構造解析

DIANA の談話構造解析処理は、図 2.1の流れに従う。以下に、入力発話の解析 (談話構造解析)

1. 対話翻訳制御部から、素性構造意味表現を受けとる[1],
2. CATE により、情報伝達行為解析を行ない、情報伝達行為タイプをキーとする発話の表現を出力し、LAYLA に渡す,
   CATE では、以下の 2 つの処理が行なわれる (処理の詳細と例は、参考文献 [山岡 92TR1] 参照):
   (a) 発話モダリティ解析
   語用論規則と素性構造書換えエンジンにより、入力素性構造意味表現の発話意図 (モダリティ) 表現部分を解析・変換する。(中間) 素性構造を以下に渡す,
   (b) 話題属性解析・発話表現変換
   NP を参照して、話題属性 (命題部分) に関する知識により、情報伝達行為タイプを決定する。さらに、その結果に従い、発話の表現を構成する,
3. LAYLA により、インタラクションプランを利用して、入力発話のプラン認識を行ない、発話クラスタを構成する。現在の LAYLA では、以下のような機能を実現して、柔軟なプラン認識を行なっている (処理の詳細と例は、参考文献 [山岡 92TR2] 参照:

[1]現在のところ、この入力は唯一つであるとする。

(a) 単一化の拡張

インタラクションプランと入力発話のマッチングは、(拡張された) 単一化により行なわれる。ここで、話し手により、あるいは発話時間により、同一概念の表層表現が変化することに対応して、概念の同一性を検査することによる等価性の評価を行なう。これは、NP を参照することにより、入力概念と前出概念が近い概念であれば、等価であると判断することである。

例えば、簡単な例でいうと、発話対を構成する発話“参加料はいくらですか?”,“料金は 4 千円です”の『参加料』と『料金』は厳密な意味で等価ではないが、発話の解釈上は (また技術的には)、同一の概念を指示していると考えたい。この例では、『参加料』と『料金』は、NP の知識ベースの中で上位下位関係になっているので、NP から近似概念として提示される。それに基づきプラン認識機構は、この 2 つの発話の内容について 1 つのインタラクションプランで解釈できるようになる。

(b) 予測情報の利用

次発話の予測から提示された文脈情報を利用して、入力発話表現の優先順位を設定し、処理の効率化を行なう。DIANA への入力素性構造は、唯一であるが、CATE により複数の解釈が行なわれることがある。複数の解釈 (発話の表現) が LAYLA に入力された時は、次発話予測情報 (現在は、情報伝達行為タイプのみを利用) のレベルの浅いものに適応する解釈を優先して、プラン認識を行なうようにする。

例えば、“...して下さい”という表現は、本質的に、要求 REQUEST-ACTION と応答 INFORM-ACTION 双方の情報伝達行為を持ち、一文内の意図解析である CATE の出力においても双方が提示される。しかし、例えば、前文が“参加料はどのようにお支払いすれば良いのですか?”という要求であれば、その応答として INFORM-ACTION (例えば、“銀行に振り込んで下さい”) を優先したい。そうすれば、プラン認識の負担はいくらか軽減される。

次発話の予測により、上の要求発話を認識した時点で、INFORM-ACTION は最優先のレベルで予測されていることになる。従って、プラン認識を行なう前処理として、この情報と入力 (すなわち CATE の出力) をてらし合わせる部分[2]を設けることにより、まず、INFORM-ACTION のみをプラン認識するようにする。

4. プラン認識された発話・発話クラスタは、ゴールスタックにより保持・管理する。入力発話のアクションは、常に完全 (完成したアクション) であるので、completel にプッシュする。発話クラスタ (インタラクションプランのインスタンス) は、そのアクションの decomposition スロットの状態により、それぞれのスタックにプッシュされる。

[2]現在は、予測情報の情報伝達行為タイプのみについてのみしか、照応を行なわない。

以後、文脈情報の提示は、このようにして構成された談話構造内の発話アクション・インタラクションプランインスタンスのアクションを参照することにより可能となる。

3.2 文脈情報の提供

はじめに述べたようにDIANAの提供する文脈情報は、与えられた発話に対して、基本的に以下の2つである (詳しい仕様は、参考文献 [菊井 92] 参照):

- 発話クラス
  発話クラスは、以下の3つのシンボルで表す.

  1. :demand: 要求の発話
  2. :response: 応答の発話
  3. :acknowledge: 確認の発話

  なお、上記のいずれにも属さない発話の発話クラスは、 nil とする。

- 発話クラスタ
  発話クラスタは、システム内ではインタラクションプランそのものに相当する。すなわち、ある発話の属する発話クラスタの情報はそのインタラクションプランから構成される。DIANAでは、発話クラスタの情報を以下のような書式で表現する:

  発話クラスタ ::= (*demand_ID response_ID acknowledge_ID*)

  ここで、各要素は、発話ID(任意のシンボル) あるいは、そのリスト[3]である。対応する発話が、存在しない場合やまだ発話されていない場合は、 nil が入る。

以下で各情報の導出アルゴリズムを説明する前に、前提となる項目を列挙する:

- ある発話には、ユニークなID(シンボル) が付され、談話構造中 (ゴールスタック) で管理されている,
- あるインタラクションプランのインスタンス (概念的に発話クラスタと同等) についても、ユニークなID(シンボル) が付され、談話構造中で管理されている,
- ある発話とインタラクションプランのインスタンス、およびプランのインスタンスとインスタンス間の連鎖の状態は、 ID(*id, cluster_id*) をラベルとするポインタで表す。

[3]現在のインタラクションプランの設定から、 *response_ID* のところのみにリストが入り得る。また、そのリストが発話クラスタデータそのものである場合もある。たとえば、付録の利用例の最後の部分の発話ID=1 を参照。

例えば、発話 ID=ID_1 とする“住所をお願いします”(発話の表現は、(ASK-VALUE 事務局 質問者 住所 -1 ?PRP)) という発話と連鎖した時のインタラクションプランのインスタンスの状態は、以下のようになる:

```
#S(ACTION
    ID          CLUSTER_ID_1          ;; プランの ID
    TYPE        :INTERACTION-PLAN
    HEADER      (GET-VALUE-UNIT 事務局 質問者 住所 -1 ?PRP )
    PRECONDTION    NIL
    DELETE-LIST    NIL
    EFFECT         ((KNOW 事務局 (is 住所 -1 ?VAL)))
    DECOMPOSITION (ID_1           ;; 入力発話の ID
                   (INFORM-VALUE 質問者 事務局 住所 -1 (IS 住所 -1 ?VAL))
                   (CONFIRMATION 事務局 質問者 住所 -1 (IS 住所 -1 ?VAL)))
    CONSTRAINT     NIL
    )
```

その時の ID_1 に対する発話クラスタの情報は、(ID_1 nil nil) となる。

- インタラクションプランの decomposition スロットは、基本的に、要求 – 応答 – 確認の 3 つのようその並びで構成される[4]。

ある発話 (発話 ID=ID) の発話クラスは、以下のように求める:

注意) 現在の DIANA のインプリメンテーションでは、談話構造 (理解状態) として、1 つ (選択順序は任意) のゴールスタックしか参照しない。(将来は、発話クラスとして複数の可能性を提示できるように拡張されることが期待される。)

1. ID に対応する発話アクションを談話構造中から求める,
   なければ *nil* で終了.
2. 1 の属するインタラクションプランを談話構造中から求める,
   なければ *nil* で終了.
3. 2 の decomposition スロット内の ID の位置が
   (a) 先頭であれば、:demand,
   (b) 2 番目であれば、:response,

[4]現在のインタラクションプランの設定では、いくつかの例外がある。それは、yes/no 疑問文などに対する応答である。例えば、“はい”の後に続く“そうです”などは、一つの応答として認識したい。これに対応するために、特殊なプラン『応答ユニット (response segment)』なるものを設定している。現在のところ AFFIRMATIVE, NEGATIVE に対するもの 4 つのみである。
これらは、厳密には、インタラクションプランではない。また、発話の入力単位に依存する。

(c) 3番目であれば、:acknowledge.

(d) (それ以外は、 *nil*)[5]

として、処理終了.

ある発話 (発話 ID=ID) の発話クラスタは、以下のように求める:

1. ID に対応する発話アクションを談話構造中から求める,
   なければ *nil* で終了.

2. 1 の属するインタラクションプランを談話構造中から求める,
   なければ *nil* で終了.

3. 2 の decomposition スロットの各要素について以下の変換を行なう

   (a) ID を表すもの (シンボル) であれば、その ID に対応するアクションを談話構造中から求め、そのアクションについて

      i. 発話アクションであれば、その ID をかえす,

      ii. プランであれば、3 を繰り返し、その結果を返す.

   (b) 発話の表現あるいは、命題であれば、 *nil* とする.

4. 変換後の decomposition スロットのリストを結果として返す.

[5]なお、ここでは、前提項目の 4 で述べた点を考慮して、行なう必要がある。すなわち、ここで例外のプランが出てきた時は、それぞれに対応した (実際は、これらが出た場合はすべて:response) 処理を行なう。

第 4 章

関数リファレンス

ここでは、DIANA の主な関数について説明する。なお、DIANA のソースコードは、現在のところ、以下のところにある:

- "as22:/home2/nadine91/discourse/Yamaoka/DIANA/diana.lisp"

4.1 主関数 (対話制御部とのインタフェース)

DM_initialize [*Function*]

引数: なし
リターン値: numeric (ロードしたブランスキーマの数)

DIANA を初期化する。

DM_reset *&optional id* [*Function*]

引数:

1. *id (&optional)*: 発話 ID

リターン値: id / nil

DIANA の内部状態 (対話構造と入力発話のポインタ) を *id*で指定された発話の解釈終了時の状態に戻す。引数 *id*はオプショナルであり、指定されなければ、何も解釈していない状態 (初期状態) に戻す。

DM_analysis *id node* [*Function*]

引数:

1. *id* : 発話 ID
2. *node* : 意味表現素性構造 (長谷川 RWS で読み込み可能なもの)

リターン値: :demand / :response / :acknowledge

入力素性構造の ID を id として談話構造解析を行ない、入力発話の属する発話のクラスを返す。

DM_uclass *id* [*Function*]

引数:

1. *id* : 発話 ID

リターン値: :demand / :response / :acknowledge

発話 ID で指定された発話の属する発話のクラスを返す。

DM_cluster *id* [*Function*]

引数:

1. *id* : 発話 ID

リターン値: (list demand-id response-id acknowledge-id)

発話 ID で指定された発話の属する談話クラスタの情報を返す。

リターン値であるリストは 3 要素であり、各々の要素には、id / nil あるいはそのリストが入り得る。

DM_cluster_demand *id1* [*Function*]

引数:

1. *id1* : 発話 ID

リターン値: id / :demand / nil

発話 ID でしてされた発話のクラスが, :demand であれば :demand を返し、:response または :acknowledge であれば、対応する :demand 発話の id を返す。後者の場合で対応する :demand 発話がない場合は NIL を返す。

DM_cluster_response *id1* [*Function*]

引数:

1. *id1* : 発話 ID

リターン値: id / :demand / :response / nil

発話 ID で指定された発話のクラスが, :demand または :response であれば:demand または :response を返し、:acknowledge であれば、対応する :response 発話の id を返す。(後者の場合で対応する :demand 発話がない場合は nil を返す。)

4.2 知っておくと便利な関数

DM-first-goal-stack [*Macro*]

引数: なし
リターン値: その時点 (呼び出された時点) における第 1 番目の対話構造 (理解状態, ゴールスタック) を返す。

(注: DIANA においてゴールスタックの順序は任意であるので、第 1 番といっても最適解を示しているとは限らない。)

DM-get-utterance-action *id* [*Macro*]

引数:

1. *id* : 発話 ID

リターン値: 構造 action

発話 ID で指定された発話のアクションを返す。

(注: ここで返る値は、上記 [DM-first-goal-stack] で得られるゴールスタックに属するものである。その他のゴールスタックから取り出す方法はない。)

DM-get-interaction-plan *id* [*Macro*]

引数:

1. *id* : 発話 ID

リターン値: 構造 action

発話 ID で指定された発話の属するインタラクションプランを返す。

ここでは、直属 (すぐ上) のインタラクションプランが返る。発話の属する談話クラスタを求める時は、関数 [DM-upper-cluster] を用いる。

DM-upper-cluster *id* [*Function*]

引数:

1. *id* : 発話 ID

リターン値: 構造 action

発話 ID で指定された発話の属する談話クラスタのプランスキーマを返す。

これは、DIANA では、いずれにせよインタラクションプランになる。

第 5 章

おわりに

本稿では、対話翻訳システム *ASURA*において、談話理解を行ない文脈情報を提供するための談話構造解析モジュール DIANA について述べた。DIANA は、談話構造解析のために、語用論的知識に基づく情報伝達行為の認定、発話対の知識を利用したプラン認識による発話クラスタの同定と発話の解釈を行なう。さらに、文脈情報提供のために、スタックによる談話構造の管理を行ない、対話翻訳制御部の要求に対して、文脈情報の提供を行なう。

DIANA の提供する文脈情報は、発話対という局所的文脈範囲における

1. その発話のクラス,

2. 属する発話クラスタの内容,

である。この文脈情報を利用することにより、従来一文単位では困難であった翻訳が行なえるようになる。これにより、より高品質の翻訳が可能となり、対話翻訳システム全体の性能向上が望める。

今後は、より広い範囲の文脈情報の提供を企図することに加え、対話翻訳において文脈情報の適用範囲を広げていくことを期待する。

参考文献

[菊井 92] 菊井玄一郎: 『対話翻訳における談話情報の管理と利用 – *AUSRA*における談話処理機能 – 』, ATR 自動翻訳研究所 TR-1-0259, (1992.3)

[長谷川 90TR] 長谷川敏郎: 『素性構造書換えシステムマニュアル (改定版)』, ATR 自動翻訳研究所 TR-1-0187, (1990.10)

[定延 90TR] 定延利之, 山岡孝行, 飯田仁: 『協調的な目標指向型対話における文形式と発話者の意図との対応 – 文形式を重視した情報伝達行為の分類 –』, ATR 自動翻訳研究所 TR-1-0220, (1991.3)

[山岡 90] 山岡孝行, 飯田仁: 『文脈を考慮した音声認識結果絞り込み手法』, 情報処理学会自然言語処理研究会資料 78-16, (1990.7)

[久米 90TR] 久米 雅子, 永田 昌明: 『日本語解析文法の意味表現について』, ATR テクニカルレポート, TR-I-0155, (1990)

[山岡 92TR1] 山岡 孝行: 『情報伝達行為解析システム CATE』, ATR テクニカルレポート, TR-I-0254, (1992)

[山岡 92TR2] 山岡 孝行, 西村 仁志, 飯田 仁: 『階層型プラン認識システム LAYLA』, ATR テクニカルレポート, TR-I-0255, (1992)

[有田 89TR] 有田英一, 飯田仁 1989 『対話翻訳のための階層型プラン認識モデル』 ATR 自動翻訳研究所 TR-1-0067, (1989.2)

[有田 91] 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』 情報処理学会自然言語処理研究会資料 81-13, (1991.1)

[飯田 90] 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』 情報処理学会論文誌, 第 31 巻 第 6 号, pp. 810-21, (1990.6)

[Yamaoka90] Yamaoka, T. and Iida, H. 1990 "A Method to Predict the Next Utterance Using a Four-layered Plan Recognition Model", ECAI'90.

[Yamaoka91] Yamaoka, T. and Iida, H. 1990 "Dialogue interpretation model and its application to next utterance prediction for spoken language processing", Eurospeech'91.

付録 A

DIANA 単体での実験

ここでは、対話構造解析モジュール単体で実験を行なう環境を提供するプログラムについて説明する。

A.1 実験プログラムの機能

本実験プログラムの提供する機能は以下のものである:

1. 処理対象入力会話データの設定と管理、
2. 談話構造解析処理の実行、
3. 処理結果並びに処理過程の表示。

A.1.1 入力データについて

実験プログラムで処理の対象となる入力会話データは、1つのファイルに書かれていることを前提としている。その内容 (とフォーマット) は、現在のところ、以下の条件を満たすものに限定する:

- 日本語解析結果の意味表現素性構造はただ1つとする、
- 意味表現素性構造の直前に、発話文字列を表すデータが書かれている、
  (従って、発話文字列・素性構造の並びが整然と交互に書かれていることになる。)
- 発話文字列並びに意味表現素性構造は、長谷川 RWS の素性構造入出力関数 [rws::push-fs-from-file] により読み込み可能なものでなければならない。

また、発話 ID は実験システムが自動的に付加する。(現在のところ、これは正数であるが、その順序に意味はない。)

A.1.2 利用方法

現在のところ、本実験システムは、CommonLisp 上で動作し、DIANA がロードされると同時にロードされる。[1]

A.2 関数リファレンス

---

temp-path [*Variable*]

タイプ: string
初期値: (マシンに依存)

DIANA が存在しているディレクトリを指定する。以後、このディレクトリの下の DIANA が作動する。

A.2.1 入力会話データに関するもの

---

kaiwa-data-path [*Variable*]

タイプ: string
初期値: (マシンに依存)

実験対象となる入力会話データファイルの存在するディレクトリを指定する。

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

タイプ: string
初期値: (concatenate 'string *kaiwa-data-path* "kaiwa1.data")

実験対象となる入力会話データファイルのファイル名。

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

---

[1]現在、ソースコードは、"as22:/home2/nadine91/discourse/Yamaoka/DIANA/diana-test.lisp" にある。

タイプ: list
初期値: nil

入力会話の発話 ID と素性構造 (内部データ rws::node) および発話文字列の関連を保持するリスト。関数 [DM-load-kaiwa-input] により設定される。

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

引数:

1. *file (&optional)* : ファイル名 (ストリング)

リターン値: numeric (読み込んだ発話の数)

*file*の内容を実験対象入力会話としてシステム内に読み込む。ここで、*file*の内容は、現在のところ、発話文字列 (シンボル) と解析結果素性構造が順番 (交互に) にならんでいることが期待される。このとき、発話 ID は任意に付加される。

DM-load-kaiwa *id* [*Function*]

引数:

1. *id* : 会話番号 (1–10)

リターン値: numeric (DM-load-kaiwa-input-file に同じ)

会話番号に対応する会話 (ATR サンプル会話) の入力ファイルを読み込み、話し手と発話 ID の関連をシステム内に設定する。[2]

DM-kaiwa-input-id-list [*Function*]

引数: なし
リターン値: (list $id_1 id_2 ... id_n$)

現在システム内に設定されている対象発話の ID のリストを返す。要素の順序は、読み込まれた順序 (= 入力ファイルに書かれている順序) になる。

[2] 1992 年 2 月現在、会話 1,2 のみ。

DM-get-input-fs *id* [*Function*]

引数:

1. *id* : 発話 ID

リターン値: FS

*id*で指定された発話の、意味表現素性構造を返す。

対象となる意味表現素性構造は、*DM-kaiwa-input-list* 中から *id*をキーに検索される。従って、処理対象が DM-load-kaiwa などによりシステム中に読み込まれている必要がある。

DM-get-input-string *id* [*Function*]

引数:

1. *id* : 発話 ID

リターン値: string

*id*で指定された発話の、発話文字列を返す。

DM-show-kaiwa [*Function*]

引数: なし
リターン値: nil

現在入力対象となっている会話の文字列を端末出力に表示する。

表示の順序は、読み込まれた順序 (= 入力ファイルに書かれている順序) になる。

A.2.2 談話構造解析実行に関するもの

DM-analysis-test *id* [*Function*]

引数:

1. *id* : 発話 ID

リターン値: 関数 (DM-IF-functions-test *id*) の値

*id*で指定された発話について、対話構造解析を行ない、処理過程の様々な情報を表示する。関数 DM-analysis-main の独立利用試験関数。

対象となる意味表現素性構造は、 *DM-kaiwa-input-list* 中から *id*をキーに検索される。従って、処理対象が DM-load-kaiwa などによりシステム中に読み込まれている必要がある。

処理過程表示の内容として、現在のところ以下のものを表示する:

1. 対象発話の文字列、
2. 情報伝達行為認定処理の
   (a) 処理時間、
   (b) (表層発話行為) 結果素性構造,
   (c) および (情報伝達行為認定後) 発話の表現,
3. プラン認識処理の
   (a) 連鎖処理に要した処理時間,
   (b) 対話構造管理に要した処理時間,
4. および対話翻訳制御機構とのインタフェース関数適用結果。 ((DM-IF-functions-test *id*) の結果に同様)

詳しくは、利用例 (付録 B) を参照されたい。

A.2.3 処理結果表示に関するもの

DM-IF-functions-test *id* [*Function*]

引数:

1. *id* : 発話 ID

リターン値: nil

*id*で指定された発話について、その時点での対話翻訳制御機構とのインタフェース関数の内、文脈情報提示に関連する以下の 4 つの関数の適用の結果を表示する。表示は端末出力へなされる。

- DM_uclass, DM_cluster, DM_cluster_demand, DM_cluster_response

---

DM-number-of-structures [*Macro*]

引数: なし
リターン値: numeric

その時点でシステムが保持している対話構造 (= 理解状態) の数を返す。

DM-show-structure *num* [*Function*]

引数:

1. *num* : 正数 ($0 \leq num \leq$ (DM-number-of-structures))

リターン値: nil

*num*番目の対話構造解析結果 (構造ゴールスタック) を端末出力に表示する。

結果の順序は任意であるので、その数字 (num) にあまり意味はない。詳細は、利用例 (付録 B) を参照されたい。

DM-show-all-structures [*Function*]

引数: なし
リターン値: nil

すべての対話構造解析結果 (構造ゴールスタック) を端末出力に表示する。

結果の順序は任意であるので、その表示順序にあまり意味はない。詳細は、利用例 (付録 B) を参照されたい。

情報伝達行為認定単独実験用

DM-cate-test *id* [*Function*]

引数:

1. *id* : 発話 ID

リターン値: 素性構造のリスト

*id*で指定された発話について、情報伝達行為認定処理を行ない、その結果の発話の表現のリストを返す。

DM-print-cate-results *results* [*Function*]

引数:

1. *results* : 素性構造のリスト (のリスト)

リターン値: nil

与えられた情報伝達行為認定結果 (*results*)(正確には、表層発話行為認定結果) をプリティプリント (rws::pprint-fs) する。

---

urep-list-with-string *id results* [*Function*]

引数:

1. *id* : 発話 ID
2. *results* : 素性構造 (rws::node) のリスト (のリスト)

リターン値: (list $list_1 list_2 ... list_n$)

表層発話行為抽出後の素性構造を発話の表現に変換し、そのリスト返す。この時、(DM-get-input-string *id*) によりえられる文字列を発話の表現に付加する。(内部仕様)

付録 B

利用例

DIANA の処理のログの詳細は、以下のディレクトリの下にあるファイルを参考されたい:

- "as22:/home2/nadine91/discourse/Yamaoka/Log/"

以下に、サンプル会話 1 についての行なった処理出力を掲載する。

```
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.

>
;;
;; 変換モジュールと一緒にロードする
;;
> (load "demo-load-with-transfer")
;;; Loading source file "demo-load-with-transfer.lisp"

(省略: 長い)

#P"/mnt/as22-home/nadine91/discourse/Yamaoka/demo-load-with-transfer.lisp"
;;
;; 会話 1 の素性構造意味表現のロード
```

```
;;
> (dm-load-kaiwa 1)
Loading kaiwa file "/mnt/as22-home/nadine91/transfer/fs/kaiwa1.data"
and Making FS-nodes ....
Load and make FS-node end.
20
;;
;; DIANA の初期化
;;
> (dm_reset)
NIL
;;
;; 読み込んだ入力すべてについて、
;; 談話構造解析
;;
> (dolist (id (dm-kaiwa-input-id-list)) (dm-analysis-test id))
;;
;; 発話文字列
;;
"もしもし"

--- CATE-transfer (1) ---
;;
;; CATE による情報伝達行為解析結果
;;
  CATE              1    0.394
;;
;; 発話モダリティ解析結果(中間素性構造)
;;
 [[CAT GREETING-OPEN]
  [PRP [[RELN OPEN-DIALOGUE]]]
  [TPC OPENING]]
;;
;; 発話の表現出力
;;
((GREETING-OPEN SP1 SP2 OPENING (OPEN-DIALOGUE) "もしもし"))
---(1)-------- Plan-inference with input 1
;;
;; LAYLA 入力
;;
(GREETING-OPEN SP1 SP2 OPENING (OPEN-DIALOGUE))
Input order:
  1: (GREETING-OPEN)
;;
```

```
;;                  3 つの談話構造を構築
;;                  |
  CHAIN             3    0.360
--------------- Result Number is 3

---- Prediction at the end of (1) ----
;;
;; 次発話の予測
;;
  PREDICTION        3    0.003
 Predicted CATs::
  Next SP1: (INFORM-VALUE CONFIRM-VALUE-UNIT)
  Next SP2: (GREETING-OPEN)
;;
;; (ここまでの処理については、他の参考文献(TR-I-0254, 0255) 参照
;;  以下の発話では省略する)
;;
;; 文脈情報の提示
;;
DM_uclass:  DEMAND     ;; 発話クラス
DM_cluster: (1 NIL)    ;; 発話の属する発話クラスタの内容(要素は発話 ID)
Demand:     DEMAND     ;; 発話のデマンド
Response:   NIL        ;; 発話のレスポンス
;;
;; 次へ
;;
"そちらは会議事務局ですか"
DM_uclass:  DEMAND
DM_cluster: (2 NIL NIL)
Demand:     DEMAND
Response:   NIL

"はい"
DM_uclass:  RESPONSE
DM_cluster: (2 (3) NIL)
Demand:     2
Response:   RESPONSE

"そうです"
DM_uclass:  RESPONSE
DM_cluster: (2 (3 4) NIL)
Demand:     2
Response:   RESPONSE
```

"どのようなご用件でしょうか"
DM_uclass: DEMAND
DM_cluster: (5 NIL NIL)
Demand: DEMAND
Response: NIL

"会議に申し込みたいのですが"
DM_uclass: RESPONSE
DM_cluster: (5 6 NIL)
Demand: 5
Response: RESPONSE

"どのような手続きをすればよろしいのでしょうか"
DM_uclass: DEMAND
DM_cluster: (7 NIL NIL)
Demand: DEMAND
Response: NIL

"登録用紙で手続きをして下さい"
DM_uclass: RESPONSE
DM_cluster: (7 8 NIL)
Demand: 7
Response: RESPONSE

"登録用紙は既にお持ちでしょうか"
DM_uclass: DEMAND
DM_cluster: (9 NIL NIL)
Demand: DEMAND
Response: NIL

"いいえ"
DM_uclass: RESPONSE
DM_cluster: (9 (10 NIL) NIL)
Demand: 9
Response: RESPONSE

"まだです"
DM_uclass: RESPONSE
DM_cluster: (9 (10 11) NIL)
Demand: 9
Response: RESPONSE

"分かりました"
DM_uclass: ACKNOWLEDGE

```
DM_cluster: (9 (10 11) 12)
Demand:     9
Response:   (10 11)

"それでは登録用紙をお送り致します"
DM_uclass:  RESPONSE
DM_cluster: (NIL 13 NIL)
Demand:     NIL
Response:   RESPONSE

"ご住所とお名前をお願いします"
DM_uclass:  DEMAND
DM_cluster: (14 NIL NIL)
Demand:     DEMAND
Response:   NIL

"住所は大阪市北区茶屋町二十三です"
DM_uclass:  RESPONSE
DM_cluster: (14 15 NIL)
Demand:     14
Response:   RESPONSE

"名前は鈴木真弓です"
DM_uclass:  RESPONSE
DM_cluster: (14 16 NIL)
Demand:     14
Response:   RESPONSE

"分かりました"
DM_uclass:  ACKNOWLEDGE
DM_cluster: (14 16 17)
Demand:     14
Response:   16

"登録用紙を至急送らせて頂きます"
DM_uclass:  DEMAND
DM_cluster: (18 NIL NIL)
Demand:     DEMAND
Response:   NIL

"よろしくお願いします"
DM_uclass:  RESPONSE
DM_cluster: (18 19 NIL)
Demand:     18
```

```
Response:   RESPONSE

"それでは失礼します"
DM_uclass:  DEMAND
DM_cluster: (20 NIL)
Demand:     DEMAND
Response:   NIL
;;
;; 会話1 終了
;;
NIL
```

```
;;
;; 最終的に保持している談話構造の表示
;;
> (dm-show-all-structures)
GS-ID = 32
INCOMPLETE --------------------
  118:GREETING-CLOSE-UNIT  : 92    (NIL NIL (90 *))
  NIL:DIALOGUE             : NIL   (NIL NIL (5 28 33 42 62 64 66 85 92))
COMPLETE1 --------------------
  116:GREETING-CLOSE       : 90    それでは失礼します
  112:ACCEPT-OFFER         : 87    よろしくお願いします
  106:OFFER-ACTION         : 83    登録用紙を至急送らせて頂きます
  82:GET-VALUE-UNIT        : 64    (NIL (*) (63 70 73))
  95:CONFIRMATION          : 73    分かりました
  90:INFORM-VALUE          : 70    名前は鈴木真弓です
  86:INFORM-VALUE          : 67    住所は大阪市北区茶屋町二十三です
  83:ASK-VALUE             : 65    ご住所とお名前をお願いします
  81:ASK-VALUE             : 63    ご住所とお名前をお願いします
  76:INFORM-ACTION         : 60    それでは登録用紙をお送り致します
  53:CONFIRM-STATEMENT-UNIT  : 42  (NIL NIL (39 46 50))
  65:CONFIRMATION          : 50    分かりました
  58:NEGATIVE-U            : 46    (NIL NIL (44 47))
  60:NEGATIVE              : 47    まだです
  56:NEGATIVE              : 44    いいえ
  50:CONFIRM-STATEMENT     : 39    登録用紙は既にお持ちでしょうか
  44:INFORM-ACTION         : 34    登録用紙で手続きをして下さい
  40:ASK-ACTION            : 32    どのような手続きをすればよろしいのでしょうか
  37:INFORM-WANT           : 30    会議に申し込みたいのですが
  32:ASK-STATEMENT         : 26    どのようなご用件でしょうか
  25:AFFIRMATIVE-U         : 21    (NIL NIL (19 23))
  28:AFFIRMATIVE           : 23    そうです
  23:AFFIRMATIVE           : 19    はい
  9:CONFIRM-VALUE          : 7     そちらは会議事務局ですか
  3:GREETING-OPEN          : 2     もしもし
COMPLETE2 --------------------
  108:OFFER-ACTION-UNIT    : 85    (NIL NIL (83 87 *))
  78:ASK-ACTION-UNIT       : 62    (NIL (*) (* 60 *))
  41:ASK-ACTION-UNIT       : 33    (NIL (*) (32 34 *))
  34:INFORM-WANT-UNIT      : 28    (NIL (*) (26 30 *))
  12:CONFIRM-VALUE-UNIT    : 10    (NIL NIL (7 21 *))
  6:GREETING-OPEN-UNIT     : 5     (NIL NIL (2 10))
  84:GET-VALUE-UNIT        : 66    (NIL (*) (65 67 *))
unrelate gs number = 0
```

```
GS-ID = 33
INCOMPLETE --------------------
  117:GREETING-CLOSE-UNIT  : 91    (NIL NIL (89 *))
  NIL:DIALOGUE             : NIL   (NIL NIL (5 28 33 42 62 64 66 85 91))
COMPLETE1 --------------------
  115:GREETING-CLOSE       : 89    それでは失礼します
  112:ACCEPT-OFFER         : 87    よろしくお願いします
  106:OFFER-ACTION         : 83    登録用紙を至急送らせて頂きます
  82:GET-VALUE-UNIT        : 64    (NIL (*) (63 70 73))
  95:CONFIRMATION          : 73    分かりました
  90:INFORM-VALUE          : 70    名前は鈴木真弓です
  86:INFORM-VALUE          : 67    住所は大阪市北区茶屋町二十三です
  83:ASK-VALUE             : 65    ご住所とお名前をお願いします
  81:ASK-VALUE             : 63    ご住所とお名前をお願いします
  76:INFORM-ACTION         : 60    それでは登録用紙をお送り致します
  53:CONFIRM-STATEMENT-UNIT  : 42  (NIL NIL (39 46 50))
  65:CONFIRMATION          : 50    分かりました
  58:NEGATIVE-U            : 46    (NIL NIL (44 47))
  60:NEGATIVE              : 47    まだです
  56:NEGATIVE              : 44    いいえ
  50:CONFIRM-STATEMENT     : 39    登録用紙は既にお持ちでしょうか
  44:INFORM-ACTION         : 34    登録用紙で手続きをして下さい
  40:ASK-ACTION            : 32    どのような手続きをすればよろしいのでしょうか
  37:INFORM-WANT           : 30    会議に申し込みたいのですが
  32:ASK-STATEMENT         : 26    どのようなご用件でしょうか
  25:AFFIRMATIVE-U         : 21    (NIL NIL (19 23))
  28:AFFIRMATIVE           : 23    そうです
  23:AFFIRMATIVE           : 19    はい
  9:CONFIRM-VALUE          : 7     そちらは会議事務局ですか
  3:GREETING-OPEN          : 2     もしもし
COMPLETE2 --------------------
  108:OFFER-ACTION-UNIT    : 85    (NIL NIL (83 87 *))
  78:ASK-ACTION-UNIT       : 62    (NIL (*) (* 60 *))
  41:ASK-ACTION-UNIT       : 33    (NIL (*) (32 34 *))
  34:INFORM-WANT-UNIT      : 28    (NIL (*) (26 30 *))
  12:CONFIRM-VALUE-UNIT    : 10    (NIL NIL (7 21 *))
  6:GREETING-OPEN-UNIT     : 5     (NIL NIL (2 10))
  84:GET-VALUE-UNIT        : 66    (NIL (*) (65 67 *))
unrelate gs number = 0
2
```

```
;;
;; 木構造で表示
;;
> (dm-show-all-structures :type :tree)
+--DIALOGUE
    |--[D]: GREETING-OPEN-UNIT
    |   |--[D]: GREETING-OPEN (SP1) もしもし
    |   +--[D]: CONFIRM-VALUE-UNIT
    |       |--[D]: CONFIRM-VALUE (SP1) そちらは会議事務局ですか
    |       |--[D]: AFFIRMATIVE-U
    |       |   |--[D]: AFFIRMATIVE (SP2) はい
    |       |   +--[D]: AFFIRMATIVE (SP2) そうです
    |       +--[D]: (CONFIRMATION SP1 SP2 そちら -1 (IS そちら -1 会議事務局 -1))
    |--[D]: INFORM-WANT-UNIT
    |   |--[E]: (WANT SP1 (申し込む -1 ?AGEN6-62 ?RECP6-62 会議 -1))
    |   |--[D]: ASK-STATEMENT (SP2) どのようなご用件でしょうか
    |   |--[D]: INFORM-WANT (SP1) 会議に申し込みたいのですが
    |   +--[D]: (CONFIRMATION SP2 SP1 用件 -1 (申し込む -1 ?AGEN6-62 ?RECP6-62 会議 -1))
    |--[D]: ASK-ACTION-UNIT
    |   |--[E]: (KNOW SP1 (する -1 SP1 手続き -1))
    |   |--[D]: ASK-ACTION (SP1) どのような手続きをすればよろしいのでしょうか
    |   |--[D]: INFORM-ACTION (SP2) 登録用紙で手続きをして下さい
    |   +--[D]: (CONFIRMATION SP1 SP2 ?TPC8-76 (する -1 SP1 手続き -1))
    |--[D]: CONFIRM-STATEMENT-UNIT
    |   |--[D]: CONFIRM-STATEMENT (SP2) 登録用紙は既にお持ちでしょうか
    |   |--[D]: NEGATIVE-U
    |   |   |--[D]: NEGATIVE (SP1) いいえ
    |   |   +--[D]: NEGATIVE (SP1) まだです
    |   +--[D]: CONFIRMATION (SP2) 分かりました
    |--[D]: ASK-ACTION-UNIT
    |   |--[E]: (KNOW SP1 (送る -1 ?AGEN13-107 ?RECP13-107 登録用紙 -1))
    |   |--[D]: (ASK-ACTION SP1 SP2 ?TPC13-106 (送る -1 ?AGEN13-107 ?RECP13-107 登録用紙 -1))
    |   |--[D]: INFORM-ACTION (SP2) それでは登録用紙をお送り致します
    |   +--[D]: (CONFIRMATION SP1 SP2 ?TPC13-106 (送る -1 ?AGEN13-107 ?RECP13-107 登録用紙 -1))
    |--[D]: GET-VALUE-UNIT
    |   |--[E]: (KNOW SP2 (IS 名前 -1 NAME))
    |   |--[D]: ASK-VALUE (SP2) ご住所とお名前をお願いします
    |   |--[D]: INFORM-VALUE (SP1) 名前は鈴木真弓です
    |   +--[D]: CONFIRMATION (SP2) 分かりました
    |--[D]: GET-VALUE-UNIT
    |   |--[E]: (KNOW SP2 (IS 住所 -1 ADDRESS))
    |   |--[D]: ASK-VALUE (SP2) ご住所とお名前をお願いします
    |   |--[D]: INFORM-VALUE (SP1) 住所は大阪市北区茶屋町二十三です
    |   +--[D]: (CONFIRMATION SP2 SP1 住所 -1 (IS 住所 -1 ADDRESS))
    |--[D]: OFFER-ACTION-UNIT
    |   |--[D]: OFFER-ACTION (SP2) 登録用紙を至急送らせて頂きます
    |   |--[D]: ACCEPT-OFFER (SP1) よろしくお願いします
    |   +--[D]: (CONFIRMATION SP2 SP1 ?TPC19-151 (送る -1 ?AGEN18-142 ?RECP18-142 登録用紙 -1))
    +--[D]: GREETING-CLOSE-UNIT
        |--[D]: GREETING-CLOSE (SP1) それでは失礼します
        +--[D]: (GREETING-CLOSE SP1 SP2 CLOSING (CLOSE-DIALOGUE))
```

```
+--DIALOGUE
    |--[D]: GREETING-OPEN-UNIT
    |   |--[D]: GREETING-OPEN (SP1) もしもし
    |   +--[D]: CONFIRM-VALUE-UNIT
    |       |--[D]: CONFIRM-VALUE (SP1) そちらは会議事務局ですか
    |       |--[D]: AFFIRMATIVE-U
    |       |   |--[D]: AFFIRMATIVE (SP2) はい
    |       |   +--[D]: AFFIRMATIVE (SP2) そうです
    |       +--[D]: (CONFIRMATION SP1 SP2 そちら -1 (IS そちら -1 会議事務局 -1))
    |--[D]: INFORM-WANT-UNIT
    |   |--[E]: (WANT SP1 (申し込む -1 ?AGEN6-62 ?RECP6-62 会議 -1))
    |   |--[D]: ASK-STATEMENT (SP2) どのようなご用件でしょうか
    |   |--[D]: INFORM-WANT (SP1) 会議に申し込みたいのですが
    |   +--[D]: (CONFIRMATION SP2 SP1 用件 -1 (申し込む -1 ?AGEN6-62 ?RECP6-62 会議 -1))
    |--[D]: ASK-ACTION-UNIT
    |   |--[E]: (KNOW SP1 (する -1 SP1 手続き -1))
    |   |--[D]: ASK-ACTION (SP1) どのような手続きをすればよろしいのでしょうか
    |   |--[D]: INFORM-ACTION (SP2) 登録用紙で手続きをして下さい
    |   +--[D]: (CONFIRMATION SP1 SP2 ?TPC8-76 (する -1 SP1 手続き -1))
    |--[D]: CONFIRM-STATEMENT-UNIT
    |   |--[D]: CONFIRM-STATEMENT (SP2) 登録用紙は既にお持ちでしょうか
    |   |--[D]: NEGATIVE-U
    |   |   |--[D]: NEGATIVE (SP1) いいえ
    |   |   +--[D]: NEGATIVE (SP1) まだです
    |   +--[D]: CONFIRMATION (SP2) 分かりました
    |--[D]: ASK-ACTION-UNIT
    |   |--[E]: (KNOW SP1 (送る -1 ?AGEN13-107 ?RECP13-107 登録用紙 -1))
    |   |--[D]: (ASK-ACTION SP1 SP2 ?TPC13-106 (送る -1 ?AGEN13-107 ?RECP13-107 登録用紙 -1))
    |   |--[D]: INFORM-ACTION (SP2) それでは登録用紙をお送り致します
    |   +--[D]: (CONFIRMATION SP1 SP2 ?TPC13-106 (送る -1 ?AGEN13-107 ?RECP13-107 登録用紙 -1))
    |--[D]: GET-VALUE-UNIT
    |   |--[E]: (KNOW SP2 (IS 名前 -1 NAME))
    |   |--[D]: ASK-VALUE (SP2) ご住所とお名前をお願いします
    |   |--[D]: INFORM-VALUE (SP1) 名前は鈴木真弓です
    |   +--[D]: CONFIRMATION (SP2) 分かりました
    |--[D]: GET-VALUE-UNIT
    |   |--[E]: (KNOW SP2 (IS 住所 -1 ADDRESS))
    |   |--[D]: ASK-VALUE (SP2) ご住所とお名前をお願いします
    |   |--[D]: INFORM-VALUE (SP1) 住所は大阪市北区茶屋町二十三です
    |   +--[D]: (CONFIRMATION SP2 SP1 住所 -1 (IS 住所 -1 ADDRESS))
    |--[D]: OFFER-ACTION-UNIT
    |   |--[D]: OFFER-ACTION (SP2) 登録用紙を至急送らせて頂きます
    |   |--[D]: ACCEPT-OFFER (SP1) よろしくお願いします
    |   +--[D]: (CONFIRMATION SP2 SP1 ?TPC19-151 (送る -1 ?AGEN18-142 ?RECP18-142 登録用紙 -1))
    +--[D]: GREETING-CLOSE-UNIT
        |--[D]: GREETING-CLOSE (SP1) それでは失礼します
        +--[D]: (GREETING-CLOSE SP2 SP1 CLOSING (CLOSE-DIALOGUE))
2
```

```
;;
;; 最終的な会話の解釈(1番目の談話構造による)
;;
> (dolist (x (dm-kaiwa-input-id-list))
    (format t "~%~a:~a" x (dm-get-input-string x))
    (dm-if-functions-test x))
1: もしもし
DM_uclass:  DEMAND
DM_cluster: (1 (2 (3 4) NIL))
Demand:     DEMAND
Response:   (2 (3 4) NIL)
2: そちらは会議事務局ですか
DM_uclass:  DEMAND
DM_cluster: (2 (3 4) NIL)
Demand:     DEMAND
Response:   (3 4)
3: はい
DM_uclass:  RESPONSE
DM_cluster: (2 (3 4) NIL)
Demand:     2
Response:   RESPONSE
4: そうです
DM_uclass:  RESPONSE
DM_cluster: (2 (3 4) NIL)
Demand:     2
Response:   RESPONSE
5: どのようなご用件でしょうか
DM_uclass:  DEMAND
DM_cluster: (5 6 NIL)
Demand:     DEMAND
Response:   6
6: 会議に申し込みたいのですが
DM_uclass:  RESPONSE
DM_cluster: (5 6 NIL)
Demand:     5
Response:   RESPONSE
7: どのような手続きをすればよろしいのでしょうか
DM_uclass:  DEMAND
DM_cluster: (7 8 NIL)
Demand:     DEMAND
Response:   8
8: 登録用紙で手続きをして下さい
DM_uclass:  RESPONSE
DM_cluster: (7 8 NIL)
```

Demand: 7
Response: RESPONSE
9: 登録用紙は既にお持ちでしょうか
DM_uclass: DEMAND
DM_cluster: (9 (10 11) 12)
Demand: DEMAND
Response: (10 11)
10: いいえ
DM_uclass: RESPONSE
DM_cluster: (9 (10 11) 12)
Demand: 9
Response: RESPONSE
11: まだです
DM_uclass: RESPONSE
DM_cluster: (9 (10 11) 12)
Demand: 9
Response: RESPONSE
12: 分かりました
DM_uclass: ACKNOWLEDGE
DM_cluster: (9 (10 11) 12)
Demand: 9
Response: (10 11)
13: それでは登録用紙をお送り致します
DM_uclass: RESPONSE
DM_cluster: (NIL 13 NIL)
Demand: NIL
Response: RESPONSE
14: ご住所とお名前をお願いします
DM_uclass: DEMAND
DM_cluster: (14 15 NIL)
Demand: DEMAND
Response: 15
15: 住所は大阪市北区茶屋町二十三です
DM_uclass: RESPONSE
DM_cluster: (14 15 NIL)
Demand: 14
Response: RESPONSE
16: 名前は鈴木真弓です
DM_uclass: RESPONSE
DM_cluster: (14 16 17)
Demand: 14
Response: RESPONSE
17: 分かりました
DM_uclass: ACKNOWLEDGE

```
DM_cluster: (14 16 17)
Demand:     14
Response:   16
18: 登録用紙を至急送らせて頂きます
DM_uclass:  DEMAND
DM_cluster: (18 19 NIL)
Demand:     DEMAND
Response:   19
19: よろしくお願いします
DM_uclass:  RESPONSE
DM_cluster: (18 19 NIL)
Demand:     18
Response:   RESPONSE
20: それでは失礼します
DM_uclass:  DEMAND
DM_cluster: (20 NIL)
Demand:     DEMAND
Response:   NIL
NIL
```