音声言語情報処理 30－2
（2000．2．4）

音声対話システム汎用プラットフォームの検討

青山一美　平野泉　菊池英明　坪川拓史　白井克彦
早稲田大学理工学部
東京都新宿区大久保3-4-1
kazumi@shirai.info.waseda.ac.jp

あらまし　音声を入出力手段として問題解決を支援・代行する音声対話システムの構築を容易にするために，システムの汎用化が望まれる．システムの汎用化を目指した従来研究をふまえて，対話の多様性と制御規則記述の容易性のトレードオフを考慮した手法を検討する．具体的には，行動レベルの上に問題解決レベルとして対話を一段抽象化することで，対話の多様性と記述の容易性を兼ね揃える．また，多様な制御を実現するために，主なモジュールをエージェントとしたマルチエージェントシステムとして実装する．

キーワード　音声対話，システムの汎用化，対話制御，マルチエージェント

A Study on Domain Independent Platform of Spoken Dialogue System

Kazumi Aoyama　Izumi Hirano　Hideaki Kikuchi　Hiroshi Tsubokawa　Katsuhiko Shirai
School of Science and Engineering, Waseda University
3-4-1, Okubo, Shinjuku-ku, Tokyo, 169-8555, JAPAN
kazumi@shirai.info.waseda.ac.jp

Abstract　It is expected to generalize architecture of spoken dialogue system for making it easy to construct a system which can support and act problem solving for user using speech input and output. In this research, we propose a method of generalization considering trade-off between variety of dialogue and easiness to describe rules. Concretely, we constructed the platform of spoken dialogue system which has problem-solving-level and action-level rules rules. A system designer only should describe problem-solving-level rules manually and action-level rules semi-automatically. Also, the system was implemented as the multi-agent system for a various control of modules.

Key Words　Spoken Dialogue, Domain Independent Platform, Dialogue Management, Multi Agent

1 はじめに

我々は, タスクやアプリケーションに依存しない音声対話システム汎用プラットフォームの構築を目指している. 一般的に音声対話システムを構築する際, 問題解決規則や対話制御規則については, タスクごとにそれぞれ個別に設計され, 知識のモジュール化についてあまり注意が払われていない. これは, 問題解決や対話制御の規則を抽象化することは実際には難しいこと, またそれらの一般的な表現形式として, フレームやプロダクションルールが存在するものの, 記述が容易ではないことによると考えられる. そのため, 他のタスクでの対話制御に関する知識の再利用が難しく, 新しいシステムを構築する際には対話モジュールを最初から設計し直さねばならず, システム開発において大きな負担のかかる要因となっている.

これまでにも, 上述の理由から, 音声対話システムの汎用化についていくつかの機関で研究されている. 秋葉ら [3] は, 対話に関する処理を独立に記述するマルチモーダル対話言語を設計した. この記述言語は, より抽象度の低いレベルでの対話のモデル化を目的とし, 割り込みなどを含む対話中の任意の状態に対して, 局所的なシステムの挙動を細かく指定することができるという特徴をもつ. 田中ら [1] はタスクを情報検索に限定し, データベースからの情報をもとに文法と語彙の設定を半自動的に行う, 汎用的な音声対話プラットフォームを構築している. このプラットフォームでは,GUI を採用することでユーザ発話を誘導し, 限定された発話・対話パターンで音声対話を実現することで, より容易に対話を記述できるという特徴をもつ. また, 荒木ら [2] は音声対話システムの対話管理部を作成する方法として, 一般的なタスクドメインを情報の流れる方向に着目して分類し, 適用できる対話ライブラリを構築し, 汎用的な対話ライブラリを組み合わせるといったアプローチを提案している.

一般的に音声対話システムを構築する際に, 問題解決規則や対話制御規則については, 特定のタスクやアプリケーションに依存した形で実装することが多い. こうした知識を外在化せずに実装すると, 特定の範囲のタスクやアプリケーションにおいては設計者の意図どおりにシステムを動作させることができるが, 他のタスクやアプリケーションでの再利用が難しくなってしまう. そこで, 知識を外在化して実装しようとすると, 知識の記述の容易性と多様性のトレードオフが問題となる. そこで, それらの知識を外在化し, 多様な対話制御をシステムの外部で記述使用とすると, 扱うパラメータが増え, アルゴリズムが複雑になってしまう. また, 外在化した知識の記述を容易にするために扱うパラメータを減らすと, 多様な記述が不可能となってしまう.

そこで本稿では, システム行動の多様性のための対話記述の自由度と記述容易性のトレードオフを考慮し, システムの行動レベルの記述の上に, 問題解決レベルとして対話を一段抽象化することによって対話の多様性と記述の容易性を実現する手法を提案する. また, 多様な対話制御を実現するために, 主なモジュールをエージェントとしたマルチエージェントシステムとして, 対話システムを実装する. また, ここで示した記述方法について実際の対話データを使用した記述力の評価を行った結果についても述べる.

2 音声対話システム汎用プラットフォームの概要

2.1 本プラットフォームの設計方針

先にも述べたように, 本プラットフォームでは音声対話による問題解決を行う上で必要な知識について, 記述多様性, 記述容易性のトレードオフを考慮した形で蓄え, 知識の内容に依存しない形で利用できるように実装する. この双方のトレードオフを実現するために, システムの実際の行動レベルの記述の上に一段階抽象化した問題解決レベルを用意する手法を提案する. 具体的には, 上位レベルとなるシステムの問題解決戦略を記述するプランニングルールをプロダクションルールで表現する. また, 行動レベルの記述, すなわち実際に行う発話や, アプリケーションコマンドを決定する行動決定ルールを状態遷移図によって表現する. その際, 各ルールの内容に特化した最小限のライブラリを用意するだけで, システムが動作できるように, 最大限に抽象化する. これにより, 問題解決及びその手段としての対話の多様性と対話記述の容易性を実現する. また, システムを構成するモジュールを並列に動作させて, 多様な対話制御方法に対応し, ユーザとシステムの自由なコミュニケーション形態を実現する.

2.2 プラットフォームの構成

今回設計, 構築する音声対話システム汎用プラットフォームは, スロットフィリング型タスクを基盤とした, データベース検索や, 会議室の予約など様々なタスクに対応できるものである. また, システム開発者によって対話制御方法を多様にかつ簡単に変化させることのできるプラットフォームを目指している. 本プラットフォームの構成を図 1 に示す.

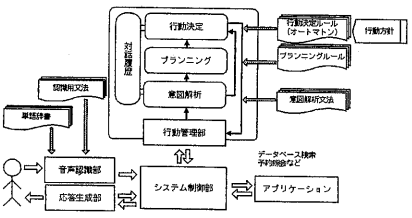

図 1: 音声プラットフォームの構成

この図に示すように, 行動管理部が先に述べた本プラットフォームの設計方針に従い二段構成のルール群 (プランニングルールと行動決定ルール) を読み込み, これらの外在化された知識を用いて問題解決を行う仕組みとなっている. 行動管理部での動作, ルール群の記述については次節で詳しく説明する.

また, 対話システムは音声認識部, システム制御部, 行動管理部, 応答生成部と, タスクに応じたデータベース検索などの具体的なアプリケーション部から構成される. システムは, 各部が並列に動作し, その間でメッセージ通信を行なって状況に応じた適切な処理を行なうマルチプロセス形式で構築される. しかし, 実質的にはメッセージの全てがシステム制御部を介してやりとりされるため, システム制御部を中心とした集中制御となる. 以下, 各プロセスについて簡単に説明する.

音声認識部

IBM ViaVoice[4] を用いて音声認識を行う. 認識結果は, システム制御部を通して, 行動管理部に渡される.

システム制御部

各プロセスからのメッセージを受け取り, メッセージタイプに応じて送信先のプロセスへ渡す. またタイマイベントを常に発生させ, 時間の経過を監視する. メッセージ処理規則の汎用化は今回は行わない.

行動管理部

システム制御部から受け取った認識結果や, アプリケーションから受け取った結果をもとにシステム設計者の記述した外部知識から, システムの動作を決定する. 詳しくは次節で述べる.

応答生成部

発話意図を受け取り, 発話意図に応じた応答生成 (発話生成や CG 出力) を行う. 発話意図は発話タイプとパラメータ群からなりこれらをもとに発話タイプと発話表現の対応表から発話生成を行う. 発話タイプと発話表現の対応表はシステム設計者により与えられる.

アプリケーション部

システム制御部から受け取ったメッセージとパラメータをもとにアプリケーションを実行する. アプリケーション部は, システム制御部との通信用のインタフェースが用意されており, データベース検索コマンドなどの外部コマンドを簡単に組み込むことができるようになっている.

3 行動管理部での処理とルールの記述

ここでは問題解決を行う上で, 対話管理やアプリケーションへのコマンド発行などの行動を決定する行動管理部の処理手順と行動管理部で使用するシステム設計者の記述する問題解決に関する記述について説明する.

3.1 行動管理の処理フロー

行動管理の処理フローを図 2 に示す.

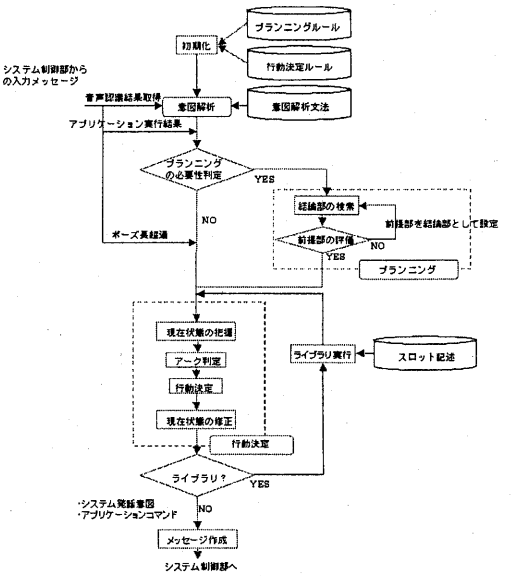

図 2: 行動管理の処理フロー

この図に示すように, 行動管理部はまず起動時に初期化としてプランニングルールと行動決定ルールとを読み込む. 通常時は, システム制御部からメッセージを受けとるごとに, 意図解析以降の処理が開始される.

音声認識処理結果取得メッセージの場合, 意図解析文法を用いて, 音声認識結果から意図解析を行う. さらに, その結果に従ってプランニングの必要性を判定する. アプリケーションコマンド実行結果メッセージの場合には, 意図解析部を通らずにプランニングの必要性判定を行う. ユーザ発話後のポーズ長超過メッセージの場合には直接行動決定に進む. また, プランニングの必要性がないと判定された場合にも直接行動決定に進む.

プランニングを行なう場合は, 初期化の際に読み込んだプランニングルールを用いて後向き推論を行い, 状況に応じて問題解決の目標を設定し, 目標を達成するための前提条件を評価する. 前提条件が満たされればその目標は達成されたことになる. 満たされていなければ, 条件を満たすべく, 条件を結論部に持つルールを検索した後, さらにそのルールの前提部の条件を評価する. これを繰り返して, 最終的に目標が達成されるまで後向きにルールを検索していく. 条件評価の過程で, 問題解決の目標を達成するために行動プランの実行が必要な場合には, その行動プランを選択して行動決定処理に進む. 行動決定処理では, 初期化の際に読み込んだ行動プランに応じた行動決定ルールに基づいて, 状態遷移制御を行なう. 行動決定処理におい

て最終的に行動プランが終了した場合, 次のプランニング処理で, 問題解決の目標を達成するための次の行動プランを決定する.

このようにして, 行動管理部では現在の状態や入力メッセージなどに応じて, プラニングルールや行動決定ルールなどのルール群を用いて, システムが次にとるべき行動を決定し, 最後に出力するメッセージを作成する.

3.2 対話の記述

図2に示すように, 対話管理部でシステム設計者が記述するものは, プランニングルール, 行動決定ルール, 意図解析文法, スロット記述である. プランニングルールは, 目標に応じた方法で問題解決を行なうことを可能にするために, システム設計者によってプロダクションルールを用いて記述される. 行動決定ルールは, プランニングルールで決定された抽象的な行動プランごとに, サブオートマトンとしてシステム設計者によって半自動的に記述される. 意図解析文法は, ひな形に従ってシステム設計者によって半自動的に記述される. スロット記述は, スロットフィリング型のタスクにおいて, システムがあらかじめ用意するスロットとスロットに入る値の制約を記述するものである. これらのルールの記述形式について説明する.

3.2.1 プランニングルール

問題解決の目標を決定するための記述を, プロダクションルールで表現する. 音声での対話による問題解決の場合, 対話の過程で動的に問題解決の目標が変わることがある. たとえば, 論文検索タスクを考えた場合,

1. 属性値を知っている特定の論文を探したい
2. 指定した属性値に関連するより多くの論文を集めたい

という二つのユーザの目標が考えられる. したがって, このような複数の目標が同時に存在する場合や, それらが対話の途中で変化する場合には, 問題解決の目標に応じて, 対話戦略を決定するためのプランニングが必要となる. 問題解決の目標から抽象的な行動プランを後ろ向き推論によって決定するためのルールがプランニングルールである. プランニングルールの記述形式は以下の通りである.

プラニングルール記述形式

```
<rule name='' ルール名''
<cond ''前提部''>
<conc ''結論部''>
</rule>
```

前提部と結論部は手続き名と引数からなり,or /and により複数並ぶこともある. 前提部で and で複数並んだルールが記述された場合, それぞれの手続きは記述されている順に実行されることになる. 論文検索タスクを例にしたプランニングルールの記述例を以下に示す.

プラニングルール (論文検索タスクの場合の一例)

```
<rule id=1>
<cond ''SubGoal 検索条件取得''
and ''Act 論文表示実行''>
<conc ''Goal 特定論文表示''>
</rule>

<rule id=2>
<cond ''Act 検索条件取得_AND''
and ''Act 論文検索実行''
and ''Equal 検索結果件数 1''>
<conc ''SubGoal 検索条件取得''>
</rule>

<rule id=3>
<cond ''Act 検索条件絞り込み''>
<conc ''Equal 検索結果件数 1''>
</rule>

<rule id=4>
<cond ''Act 検索条件取得_OR''>
and ''Act 論文検索実行 (検索条件)''
and ''AlmostEqual 検索結果件数 10''
and ''Act 論文リスト表示実行''
<conc ''Goal 関連論文リスト表示''>
</rule>

<rule id=5>
<cond ''Act 検索条件追加_OR''>
<conc ''AlmostEqual 検索結果件数 10''>
</rule>
```

上記のプラニングルール群のうち最初の3つは, 「ユーザが属性値を知っている特定の (1件の) 論文を表示する」という目標に関するルールであり, 後の2つは「ユーザが指定する属性値に関係する (10件の) 論文リストを表示する」という目標に関するルールである. 目標をどのタイミングでどのように設定するかは, システム設計者によるが, 現段階ではメニュー選択により設定するものとする.

3.2.2 行動決定ルール

行動決定ルールは, 基本的にシステムの行動を状態, ユーザの行動や, アプリケーションコマンドの結果, ライブラリ実行結果などの外部イベントをアークとした状態遷移の規則を示したものである. 先に示したプランニングルールの例と同じ論文検索タスクの場合を想定して具体化したサブオートマトンどうしの関係を図3に示す.

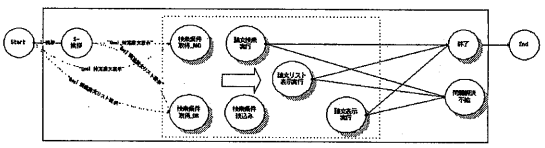

図 3: サブオートマトンの関係 (論文検索タスクでの一例)

図3において, 最初のユーザ発話の意図解析結果で「挨拶」の発話タイプが得られた場合のみ, 挨拶サブオートマトンに遷移する. 挨拶サブオートマトンへの遷移に関わらず, 問題解決の目標として「特定論文表示」や「関連論文リスト表示」が得られた場合には, プランニングによって最終的にそれぞれ"検索条件取得_AND"あるいは"検索条件取得_OR"の行動プランが選択され, その結果それぞれの行動プランに対応したサブオートマトンの初期状態に現在の状態が遷移する. サブオートマトンが終了状態となったとき, サブオートマトンに対応する行動プランが達

成されたことになる. 終了状態に至る前に問題解決の目標が変わりプランニングで異なるサブオートマトンが選択された場合には, サブオートマトンを切替えその先の初期状態に現在の状態が移る.

サブオートマトンにおいて, 状態として表現するシステムの行動には,(1) システムの発話 (応答),(2) アプリケーションコマンドの実行,(3) ライブラリの実行がある.(1),(2) に関しては決定された行動としてメッセージを作成し, システム制御部へ送るが, ライブラリを実行する場合には, ライブラリを実行した上で, その行動を現在の状態とし, ライブラリの実行結果をアークとした行動決定処理に再度進む必要がある. その際, ライブラリの種類によってはシステム設計者によるスロット記述 (後述) を参照して処理を行う. 図 3 中のサブオートマトンのひとつを詳細にしたものを図 4 に示す.

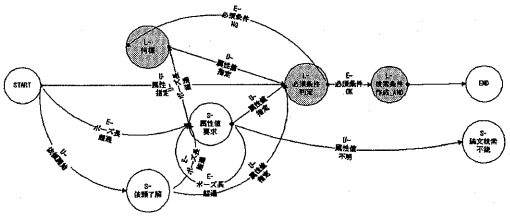

図 4: サブオートマトンの例 (検索条件取得_AND)

サブオートマトンは, 行動決定ルール記述言語により記述される. 例えば, 図 4 に示したサブオートマトンの状態は以下のように記述される.

検索条件取得_AND

```
<rule id=0>
<state id=0 act=''Start''>
<arc act=''U_依頼開始'' next_state_id=1 next_state_act=''S_依頼了解''>
<arc act=''E_ポーズ長超過'' next_state_id=2 next_state_act=''S_属性値要求''>
<arc act=''U_属性値指定'' next_state_id=3 next_state_act=''L_必須条件判定''>
</rule>

<rule id=1>
<state id=1 act=''S_依頼了解''>
<arc act=''U_属性値指定'' next_state_id=3 next_state_act=''L_必須条件判定''>
<arc act=''E_ポーズ長超過'' next_state_id=2 next_state_act=''S_属性値要求''>
</rule>

<rule id=2>
<state id=2 act=''S_属性値要求''>
<arc act=''U_属性値指定'' next_state_id=3 next_state_act=''L_必須条件判定''>
<arc act=''E_ポーズ長超過'' next_state_id=2 next_state_act=''S_属性値要求''>
<arc act=''U_属性値不明'' next_state_id=6 next_state_act=''S_問題解決不能''>
</rule>

<rule id=3>
<state id=3 act=''L_必須条件判定''>
<arc act=''E_必須条件 NG'' next_state_id=4 next_state_act=''L_待機''>
<arc act=''E_必須条件 OK'' next_state_id=5 next_state_act=''L_検索条件作成_AND''>
</rule>

<rule id=4>
<state id=4 act=''L-待機''>
<arc act=''U_属性値指定'' next_state_id=3 next_state_act=''L_必須条件判定''>
</rule>

<rule id=5>
<state id=5 act=''L_検索条件作成_AND''>
<arc act='''' next_state_id=6 next_state_act=''End''>
</rule>
```

図中の''E_'' はライブラリ実行結果などのイベント, ''U_'' はユーザ発話意図における発話タイプ, ''S_'' はシステム発話意図における発話タイプ, ''L_'' はライブラリ, ''C_'' はアプリケーションコマンドをそれぞれ意味する.

スロットフィリング型タスクで汎用的に扱うことのできる確認の手段や, スロット値の取得方法などに関する記述は, あらかじめプラットフォームで用意されている. 現段階では, システム設計者はそれらを選択し, 自分で用意した行動の定義とあわせて, 行動決定ルールを記述することを想定しているが, 将来的にはオートマトンの合成や拡張を自動的に行うことを予定している.

3.2.3 意図解析文法

意図解析文法とには, 上述の行動決定ルールにより現されるサブオートマトン中の現在の状態と, その状態におけるキーフレーズと発話意図の対応表を用いる. 意図解析処理では, システム設計者によって記述されるこの対応表を読み込んでおき, 音声認識結果にそれぞれの状態に応じたキーフレーズが含まれているか否かによって発話意図を推定する. 上述の論文検索タスクを例にした意図ーキーフレーズ対応表の一部を表 1 に示す.

表 1: ユーザ発話意図とキーフレーズの対応表の例

発話タイプ	パラメータ	現在の状態	キーフレーズ
U_依頼開始		Start	/探す/ and /論文/
U_属性値指定	$value	Start	/$value/ and /探す/
		S_依頼了解	/$value/
		S_属性値要求	/$value/
		L_待機	/$value/

表 1 に示すように, 発話意図は発話タイプとパラメータにより構成される.

3.2.4 スロット記述

スロット記述はタスクに依存しない形で, 以下の情報をシステム設計者が記述する. この記述は, ライブラリで用意された「必須条件判定」で使用される.

1. スロット数 スロットの数.

2. スロット名 各スロットの名前.

3. 有効値リスト 各スロットが取り得る有効な値のリスト.

4. 必須条件 アプリケーションコマンド毎に必要なスロット.

4 行動決定ルールの評価

実際の論文検索タスクの対話データから, 上記の行動決定ルールを記述したものと, 他の論文検索タスクの対話データを比較し, このルールで作成した行動決定ルールがどの程度一般性があるかを評価した. ここでは, 評価方法とその結果について述べる.

4.1 評価方法

評価に利用したデータは, 早大の論文検索データ 2 対話, 理科大のデータ 2 対話, 電通大のデータ 1 対話の計 5 対話である. これらのデータに対し, 1 発話ごとに行動決定ルールの状態とアークに対応させてタグ付けを行った. 評価方法としては, ひとつのデータに対して状態系列 $S = \{s_1, s_2, \ldots, s_n\}$ がタグ付けされたときに, 正解系列 $S_c = \{S_1, S_2, \ldots, S_n\}$ と比較して, 一致する者を正解数, 記述したオートマトンではタグ付けのできないものを欠落数, 正解系列に含まれる余分な遷移を挿入数, 結果伝達方法や確認方法が状態系列では間接であるが, 正解系では直接確認となっているものを置換数とした. 正解率は, 状態 (システムの行動) のみ, アークのみ (ユーザ発話などの外部イベント), 状態とアークの合計の 3 種類を求めた. 計算式は次の通りである.

正解率 = (正解数 − 欠落数 − 置換数)/全体数

的確率 = (正解数 − 欠落数 − 置換数 − 挿入数)/全体数

4.2 評価結果と考察

上記の評価方法を用いて整合率を求めた結果を表 2 に示す. 整合率は正解率 (的確率) の順に示す.

表 2: 行動決定ルールの評価

対象データ	状態整合率	アーク整合率	全体の整合率
早大データ 1	60%(60%)	100%(100%)	0%(-16%)
早大データ 2	33%(-13%)	70%(70%)	-55%(-100%)
理科大データ 1	33%(-66%)	20%(20%)	-60%(-160%)
理科大データ 2	100%(0%)	33%(33%)	-20%(-80%)
電通大データ 1	0%(-60%)	50%(50%)	-20%(-120%)

早大データは, システム主導の対話であり, システムの発話に誘導されてユーザが発話するため, ある程度決まった発話が多く, ユーザ発話意図の記述力 (アーク整合率) が高い傾向がみられた. 理科大データではユーザ主導の発話が多くみられ, 挿入数が多く, 全体的に整合率が低くなっていることがわかる. 電通大データにおいては, ひとつの対話の中で, 直接確認や間接確認などが状況に応じて使用されており, システムの行動の整合率が低くなっている. この評価結果は, 今回評価に用いた対話データに対する行動決定ルールの記述力にすぎないが, このようにプラットフォームであらかじめ用意する行動決定ルール記述の一般性を評価し, 結果をフィードバックさせていくことで, よりシステム設計者の要求を満足させることができると考えている.

5 おわりに

本稿では, スロットフィリング型の対話を基盤とした, 音声対話汎用プラットフォームについて, 特に対話戦略の知識の記述方針と記述形式について論文検索タスクを具体例にとりながら述べた. さらに, 説明した記述形式で, 論文検索タスクを実際の対話データから記述し, 他の論文検索タスク対話データと比較して, その記述力を評価した結果について述べた. 今後は他のタスクもこの記述方針に従って記述し, さらに記述力の評価を行う予定である. また, 行動決定ルールの自動生成についての可能性を検討していく.

謝辞 本研究は日本学術振興会未来開拓学術振興研究推進事業による研究プロジェクト「音声言語による人間-機械対話システムの研究」の支援を受けている. 本研究では, プロジェクトで収集した対話データを評価データとして使用させていただきました. また, NTTコミュニケーション科学基礎研究所のご協力に感謝いたします.

参考文献

[1] 田中克明, 河原達也, 堂下修司, " 汎用的な音声対話プラットフォーム", 信学技報, NLC98-45/SP98-109, pp.9 - 16, 1998.

[2] 荒木雅弘, 駒谷和範, 平田大志, 堂下修司, " 音声対話システム構築のための対話ライブラリ", 人工知能学会研究会資料, SIG-SLUD-9901-1, pp.1 - 6, 1999.

[3] 秋葉友良, 神嶌敏弘, 伊藤克亘, " 時間関係と対話性を考慮したマルチモーダル対話記述用スクリプト", 情処研資, SLP-19-19, pp.123 - 130,1997.

[4] http://www.ibm.co.jp/voiceland/