検索条件絞り込み

```
<rule_id=0>
<state_id=0 act="start">
<arc_act="" next_state_id=1 next_state_act="S_絞込み要求">
<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_絞込み要求">
<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=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="検索条件作成_AND">
</rule>
<rule_id=4>
<state_id=4 act="L_待機">
<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=5>
<state_id=5 act="検索条件作成_AND">
<arc_act="" next_state_id=7 next_state_act="End">
</rule>
```

論文検索実行

```
<rule_id=0>
<state_id=0 act="start">
<arc_act="" next_state_id=1 next_state_act="C_論文検索実行">
</rule>
<rule_id=1>
<state_id=1 act="C_論文検索実行">
<arc_act="E_論文検索結果取得" next_state_id=2 next_state_act="S_論文検索結果伝
達">
</rule>
<rule_id=2>
<state_id=2 act="S_論文検索結果伝達">
<arc_act="E_ポーズ長超過" next_state_id=3 next_state_act="S_表示確認">
<arc_act="U_表示 OK" next_state_id=4 next_state_act="End">
</rule>
<rule_id=3>
<state_id=3 act="S_表示確認">
<arc_act="E_ポーズ長超過" next_state_id=3 next_state_act="S_表示確認">
<arc_act="U_表示 OK" next_state_id=4 next_state_act="End">
</rule>
```

論文表示実行

```
<rule_id=0>
<state_id=0 act="start">
<arc_act="" next_state_id=1 next_state_act="C_論文表示実行">
</rule>
<rule_id=1>
<state_id=1 act="C_論文リスト実行">
<arc_act="E_ポーズ長超過" next_state_id=2 next_state_act="S_表示結果確認">
<arc_act="表示結果 OK" next_state_id=3 next_state_act="End">
</rule>
<rule_id=2>
<state_id=2 act="S_表示結果確認">
<arc_act="E_ポーズ長超過" next_state_id=2 next_state_act="S_表示結果確認">
<arc_act="リスト表示結果 OK" next_state_id=3 next_state_act="End">
</rule>
```

論文リスト表示実行

```
<rule_id=0>
<state_id=0 act="start">
<arc_act="" next_state_id=1 next_state_act="C_論文リスト表示実行">
</rule>
<rule_id=1>
<state_id=1 act="C_論文リスト表示実行">
<arc_act="E_ポーズ長超過" next_state_id=2 next_state_act="S_リスト表示結果確
認">
<arc_act="リスト表示結果 OK" next_state_id=3 next_state_act="End">
</rule>
<rule_id=2>
<state_id=2 act="S_リスト表示結果確認">
<arc_act="E_ポーズ長超過" next_state_id=2 next_state_act="S_リスト表示結果確
認">
<arc_act="リスト表示結果 OK" next_state_id=3 next_state_act="End">
</rule>
```

(c) 意図解析文法

意図解析文法としては、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－キーフレーズ対応表を表 4.5 に示す。

表 4.3: ユーザ発話意図（発話タイプとパラメータ）とキーフレーズの対応表

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	$author	Start S_依頼了解 S_属性値要求 L_待機	/$author/ and /探す/ /$author/ /$author/ /$value/
U_属性値不明		S_属性値要求	/わかりません/
U_表示 OK		S_論文検索結果伝達 S_表示確認	/表示/ and /して/ /お願いします/
U_リスト表示結果 OK		C_論文リスト表示実行 S_リスト表示結果確認	/リスト/ or /全部/ and /見せて/ /お願いします/
U_表示結果 OK		C_論文表示実行 S_表示結果確認	/いい/ /はい/

表 4.5 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。表の例では、パラメータを取る発話意図は「属性値指定」のみで、この値は後の検索条件を作成するライブラリ「検索条件作成_AND」または「検索条件作成_OR」において利用できるように記録しておく必要がある。

(d) スロット記述

スロット記述としてタスクに依存しない形で、以下の情報を記しておく必要がある。

(i) スロット数 スロットの数。

(ii) スロット名 各スロットの名前。

(iii) 有効値リスト 各スロットが取り得る有効な値のリスト。

(iv) 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。

前述の論文検索タスクの場合の例として、あらかじめ用意すべき上記の情報を以下に示す。なお、論文データベースとしては500件の書誌情報が登録されており、そこに現れる語彙のうち重要語5000語を扱うことを想定する。

表 4.4: スロット記述の例（論文検索タスクの場合の一例）

スロット名	有効値	必須
AUTHOR	1300件の人物名	—
KEYWORD	3700件の重要語	—
SOCIETY	学会名	—
CATEGORY	分野名	—

(e) ライブラリ

上述の論文検索タスクの場合、以下のライブラリを用意する必要がある。

表 4.5: ライブラリの例（論文検索タスクの場合の一例）

ライブラリ名	処理内容
L_必須条件判定	取得条件が検索の必須条件を満たしているかを判定
L_待機	何も行動をせずに現在状態を維持
L_検索条件作成_AND	取得した属性値を検索条件にANDで結合
L_検索条件作成_OR	取得した属性値を検索条件にORで結合

以下に各ライブラリの処理の仕様を示す。

(1)L_必須条件判定

内　　容 取得した検索条件が、前述のスロット記述における必須条件を満たしているかを判定する。

引　　数 Condition *acquired_condition* 取得した検索条件。条件を格納するための構造体型。

戻 り 値 E_必須条件 NG
E_必須条件 OK

(2)L_待機

内　　容 何も行動をせずに現在の状態を維持する。

引　　数

戻 り 値

(3)L_検索条件作成_AND

内　容 取得した属性値を、取得した検索条件として AND で結合する。

引　数 Condition *acquired_value* 取得した属性値。条件を格納するための構造体型。

戻 り 値 *search_condition*

(4)L_検索条件作成_OR

内　容 取得した属性値を、取得した検索条件として OR で結合する。

引　数 Condition *acquired_value* 取得した属性値。条件を格納するための構造体型。

戻 り 値 *search_condition*

(f) イベント

ユーザ発話意図以外に、状態遷移制御に関わるイベントとして、アプリケーションコマンドの実行結果、ユーザ発話後のポーズ長超過、ライブラリの実行結果などがある。表 4.6 に、上述の論文検索タスクの場合の例を示す。

表 4.6: イベントの例（論文検索タスクの場合の一例）

イベント	内容
検索結果取得	検索コマンドの実行結果
ポーズ長超過	ユーザ発話後のポーズ長が一定値を超過（SYS_ACT_PauseOver）
必須条件 OK	取得条件が検索の必須条件を満たしている
必須条件 NG	取得条件が検索の必須条件を満たしていない

(3) サブルーチンの仕様

行動管理部のサブルーチンである意図解析、プラニング、行動決定、ライブラリ実行、メッセージ作成の各処理の仕様を以下に示す。

(1) 意図解析

使 用 法 Action **intention_analysis**(*recognition_result*)

引　数 RecogResult *recognition_result* 音声認識結果。音声認識結果の単語列格納用の構造体型。

戻 り 値 意図解析結果。発話意図のタイプとパラメータ格納用の構造体型。

内　容 意図解析文法（前述）を参照して、グローバル変数として定義される現在の状態に応じて、音声認識結果に対するキーフレーズマッチングを行なう。戻り値としては、意図解析結果である発話タイプとそのパラメータを返す。

(2) プラニング

使 用 法 int **do_planning**(*goal, action*)

引　　数 char * *goal* 問題解決の目標。文字列型。

Action *action* ユーザ発話意図もしくはイベントなどの行動。行動のタイプとパラメータ格納用の構造体型。

戻 り 値 選択された行動プランのＩＤ。整数型。

内　　容 問題解決の目標およびユーザ発話意図もしくはイベントの入力にともない、プラニングルールを用いて後向き推論によるプラニングを行なう。最終的に行動プランが選択されればそのＩＤを返し、目標が達成されれば 1 を返す。

(3) 行動決定

使 用 法 Action **action_decision**(*plan_id, action*)

引　　数 int *plan_id* 選択された行動プランＩＤ。整数型。

Action *action* ユーザの発話意図やシステム内部のイベントなどの行動。行動のタイプとパラメータ格納用の構造体型。

戻 り 値 システムの発話意図やアプリケーションコマンド、ライブラリなどのシステムの行動。行動のタイプとパラメータ格納用の構造体型。

内　　容 ユーザ発話意図もしくはイベントの入力にともない、行動決定ルールで現されるオートマトンを用いて、現在の状態に応じたシステムの行動を決定する。最終的に行動が決定されればそのタイプとパラメータを返す。

(4) ライブラリ実行

使 用 法 int **do_library**(*library*)

引　　数 Action *library* ライブラリ。ライブラリのタイプとパラメータ格納用の構造体型。

戻 り 値 ライブラリ実行結果としてのイベントＩＤ。整数型。

内　　容 入力されたライブラリのタイプとパラメータを用いて、ライブラリを実行し、実行結果としてイベントのＩＤを返す。

(5) メッセージ作成

使 用 法 char * **construct_message**(*action*)

引　数 Action *action* 発話意図あるいはアプリケーションコマンドなどの行動。行動のタイプとパラメータ格納用の構造体型。

戻り値 作成されたメッセージ。文字列型。

内　容 入力された行動のタイプとパラメータを用いて、システム制御部に送信するメッセージを作成し、それを返す。メッセージの形式は基本的には、メッセージ名と行動タイプとパラメータをスペースで区切った形とする。

4.3.3 応答生成部

応答生成部は表 4.7 に示すテンプレートを用いて発話生成を行う。

表 4.7: システム発話のテンプレート

発話タイプ	パラメータ	発話表現
S_挨拶		「こんにちは」
S_依頼了解		「どのような論文をお探しですか」
S_属性値要求	$SLOT	「$SLOT は何ですか」
S_論文検索結果伝達	$NUMBER	「該当する論文が$NUMBER 件あります」
S_表示確認		「これを表示しますか」
S_リスト表示結果確認		「これでいいですか」
S_表示結果確認		「これでいいですか」

4.4 まとめ

本章では、音声対話制御をアーキテクチャレベルで汎用化するための音声対話インタフェース汎用プラットフォームの実装について述べた。本プラットフォームは、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て、具体的な行為レベルの上に抽象的なプランレベルを設けて制御の単位とする。これによって、従来法よりも少ない記述量で多様な対話制御を実現可能にする。このプラットフォームを用いて、学術論文検索タスク、チケット予約タスクなど異なるタスクについて複数の音声対話システムを実際に構築し、多様な対話制御が実現できることを確認した。記述力および記述量の評価については [41] を参照されたい。