1K3-OS-17a-3

マルチモーダル対話記述における関数による解釈表現の検討

Examination of semantic representation by function in multimodal interaction description

井手 麻友美　　荒木 雅弘
Mayumi IDE　　Masahiro ARAKI

京都工芸繊維大学
Kyoto Institute of Technology

Because of development of interactive system technologies and diversification of the modalities, Multi-modal Interaction(MMI) system is expected as the next interaction technology. However, a description of the modality integration process is complex in the implementation of the MMI system. In this paper, we propose a new interaction description to eliminate the description that depends on the modality by means of converting the result of understanding input to function. We have made it possible to eliminate complex descriptions corresponding to the combination of modalities by applying the multiple function.

1. はじめに

認識技術やセンサ技術の発展によって，情報機器への入力手段として GUI のみならず多様なモダリティを扱えるようになり，次世代のインタラクション技術としてマルチモーダル対話 (MMI; Multi-modal Interaction) への期待が高まっている．特に音声入出力機能を持つ MMI システムが実現すれば，手入力の操作とハンズフリーの操作を状況によって使い分けたり組み合わせたりすることが可能になり，携帯端末での操作やアクセシビリティの向上などを実現できる．

一方，MMI システムの実現には，各モダリティからの入力に対する対話を制御するための記述の複雑さという問題がある．タスク達成を目的とした対話システムにおいて対話制御を行うためには，入力理解結果と対話の流れに応じた状態遷移を開発者が記述しなければならない．しかし，MMI システムの対話記述は複数のモダリティを組み合わせた入力に対する処理を記述しなければならないため，単一のモダリティを使用したシステムの対話記述と比較すると複雑である．特に入力モダリティの統合処理に対する記述の複雑さは MMI システムの開発者にとって負担となり，手軽に MMI システムを実現することは困難である．この問題を解決するためには，システム開発者が対話制御の記述をより簡潔に記述できる手法が必要である．

このため本研究では，入力理解結果に応じた状態遷移を記述しない新たな対話制御手法を提案する．提案手法では，入力理解結果を関数化し，その関数を現在の対話状態に適用することによって次の対話状態への遷移を行う．また，関数の適用に適した対話制御手法として，情報状態更新モデルを使用する．これにより，情報状態に入力理解結果の関数を適用することで MMI システムの対話制御を行うことができる．

2. 複数モダリティの入力記述手法

MMI システムにおいて，複数あるモダリティの中から一つのモダリティを選択するような入力だけでなく，複数のモダリティによる逐次的な入力や同時的な入力が予想されるが，全ての入力パターンを記述するのは負担が大きい．そのため [菊地 13] では，逐次的・同時的・択一的な 3 種類の入力を行うためのメソッドをあらかじめ用意し，受け付ける入力のパターンの組合せをあらかじめ記述することでマルチモーダル対話制御を実現している．この手法では，モダリティの追加・変更の際に修正量が多くなってしまうという問題点がある．[Johnston 02] では個別のモダリティの理解結果をラティスで表し統合するという手法を取っているが，この手法もモダリティの追加・変更が統合処理に及ぼす影響は大きい．

一方，本研究では，対話状態の遷移記述を関数の連続適用へと置き換え，対話制御の記述をなくすことでこれを解決する手法を提案する．各モダリティへの入力の理解結果を関数として生成し，複数の入力モダリティを組み合わせた入力に対する統合処理を，各入力モダリティの入力理解結果の関数を多重適用することで実現する．これにより，予想される全ての入力パターンの列挙や複雑な統合処理を記述する必要がなくなる．

3. 提案手法

3.1 入力理解結果の関数化

従来の MMI システムでは，ユーザからの入力を解釈して得られた入力理解結果は，入力モダリティが何であったかという情報を保ったまま対話制御部に送られ，記述された条件分岐に従ってタスクの達成に必要な変数を埋めるために使用されていた．しかし本研究の提案手法では，入力理解結果を関数として生成し，関数を適用することによって，選択している変数に自動的に値を埋める．これにより各入力モダリティに対応した処理を記述するのではなく，入力モダリティが何であるかに関わらず関数を適用するため，モダリティに応じた処理を記述する必要がなくなる．しかしこれを実現するには，今までモダリティ毎に異なっていた，変数に代入する値とその規則を統一する必要がある．

このため本研究では変数に対してドメインを指定する．ユーザが入力した値がドメインの領域内に存在する値であるときのみ変数に代入する，という仕組みを入力理解結果の関数として生成する．この入力理解結果関数の生成過程を図 1 に示す．この時，入力理解結果が変数のドメインに当てはまらない値 (例えば出発駅という変数の場合，その路線に存在する駅名以外の入力) だった場合は，関数を適用しても変数は埋まらず，ドメイン内の入力理解結果の場合のみ変数を埋めることができる．このため関数を適用した際に変化があった場合のみ，その関数

連絡先: 京都工芸繊維大学 情報工学部門
〒 606-8585 京都市左京区松ヶ崎橋上町
E-mail: mayumi@ii.is.kit.ac.jp

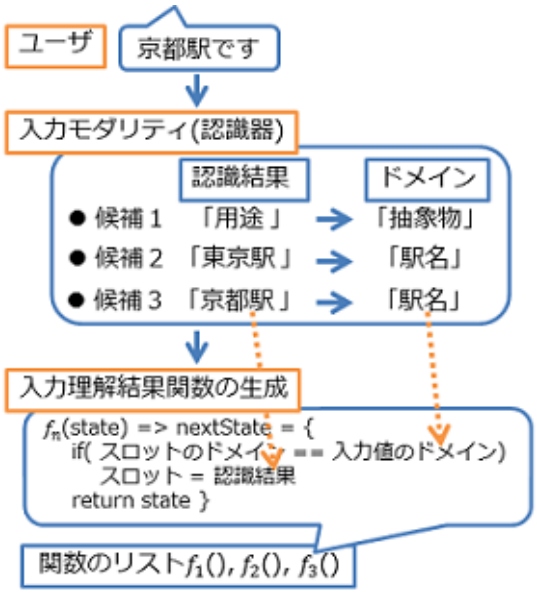

図 1: 入力理解結果関数の生成過程

を有効な入力理解結果であると定める．

3.2 関数のリスト

入力理解結果は，一つの関数ではなく複数の関数のリストとして生成する．これは，入力理解結果に複数の候補が存在する場合を想定しているためである．特に音声認識などでは認識の曖昧性を考慮し，一つの入力に対して複数の認識結果の候補が生成される場合がある．よって認識結果を関数として生成し信頼度が高い順にリスト化を行い，順に適用する手法をとる．例えば，候補の中で最も信頼度の高い入力理解結果を適用した際，状態に変化が起こらなかった場合でも，2 番目に信頼度の高い関数を適用することで状態が変化する可能性がある．また，システムが現在入力理解結果としている関数が，のちのユーザへの確認で誤りであることが示された場合，他の信頼度の低い候補を適用することで状態が変化すれば，ユーザにもう一度入力を促す操作を省略することができる．

3.3 関数の多重適用

入力モダリティが複数のモダリティから選択された単一のモダリティである場合は，そのモダリティの入力理解結果のみを適用すればよいが，逐次的な入力や同時的な入力を解決するには入力の統合処理を行う必要がある．この問題を解決するため，関数は引数と戻り値が同一である場合，多重適用できることを利用する．例えばユーザが 2 つの入力モダリティを組み合わせた入力を行い，各入力モダリティに対する入力理解結果の関数が，入力された順に $f_1()$，$f_2()$ であったとすると，モダリティ統合後の入力理解結果は $f_2(f_1())$ のように，それぞれの関数を順に適用する形にできる．このように，入力に単一のモダリティを利用した場合と同様の形式で，複数のモダリティを組み合わせた場合の入力理解結果の適用が可能となり，従来手法における入力モダリティの統合に関する記述の複雑さを解消することができる．

3.4 情報状態更新モデル

本研究の提案手法を実現するためには，関数の適用によって対話を制御できる仕組みが必要である．これを実現する方法として，本研究では情報状態更新 (Information State Update) モデル [S. Larsson 00] を使用した対話制御を提案する．情報状態更新モデルは TRINDI(Task Oriented Instructional Dialogue) プロジェクト*2と呼ばれる EU のプロジェクトで用いられている対話モデルであり，情報状態とは以下の概念を持つ対話状態の表現方法である．

$$\begin{bmatrix} PLAN & = & \begin{pmatrix} RAISE(RETURN), \\ RAISE(MONTH), \\ RAISE(CLASS), \\ RESPOND(PRICE) \end{pmatrix} \\ TMP & = & \\ BEL & = & \{(TO = PARIS), (HOW = PLANE)\} \\ QUD & = & RAISE(FROM) \end{bmatrix}$$

図 2: 提案手法における情報状態

- informational components:動機や共通の文脈を含んだ対話モデル中の情報要素の表現.
- formal representations:上記のコンポーネントの形式的な表現.
- dialogue moves:情報状態更新のトリガとなる,「対話における行為」の集合.
- update rules:情報状態の更新を管理する更新規則．現在の情報状態が与えられた場合，規則の集合から実行する「対話における行為」を選択する．
- update strategy:与えられた状態に対してどの規則を選択するか決定するための更新戦略.

本研究ではこの情報状態を，入力モダリティ統合時に入力理解結果の関数を用いて更新することで対話制御を実現する手法を提案する．ただし情報状態の更新規則はモダリティ統合処理を中心に定義する．そのため本研究で使用する情報状態は図 2 のようになり，以下の性質を持つ 4 つのサブフィールドに分割されている．

- PLAN:長期的な対話の目標となる行動のリスト
- TMP:対話参加者間で明示的に対話によって確立された情報を反映したフィールド
- BEL:システムが対話によって確認した命題の集合
- QUD: 議論中の質問のスタック

達成するタスクは，あらかじめ開発者が PLAN スタック (図 3) に定義するものとする．またタスクは，情報の取得に必要な変数にあたる RAISE クラスのインスタンスと，変数に入力された値が正しいか確認する対話を行うことを指示する，RESPOND クラスのインスタンスを，実行する順に PLAN スタックに入れることで表現する．確認動作は RESPOND クラスのインスタンスだけでなく，変数に値を代入するたびに，その値を毎回確認する対話を行うかどうかを記述できるように，responseMode 変数を用意する．さらに，RAISE クラスには変数名とドメイン名を記述しておく．ドメインに含まれる値の定義の仕方については具体的に定めず，入力理解結果の持つドメイン名と変数が持つドメイン名が一致した場合にのみ入力が可能であるとする．

*2 http://www.ling.gu.se/projekt/trindi/.

```
class Sample extends IS {
    respondMode = true
    plans += Raise("出発駅", "地下鉄駅名")
    plans += Raise("到着駅", "地下鉄駅名")
    plans += Raise("枚数", "数量")
    plans += Respond()
}
```

図 3: PLAN スタックの記述例

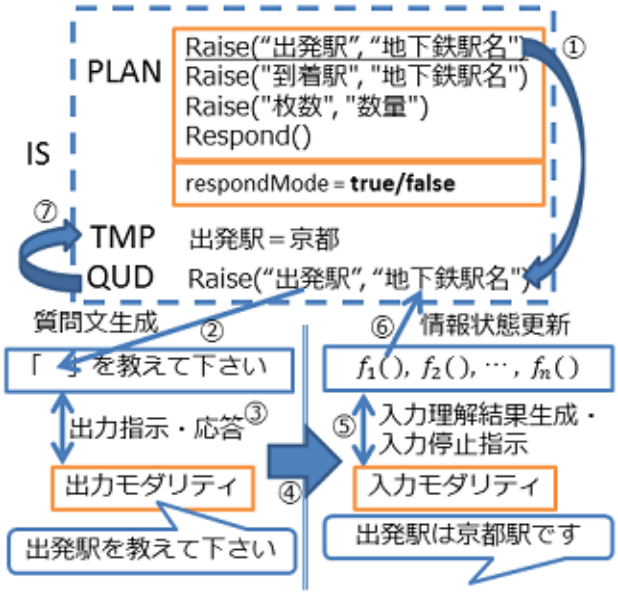

図 4: 情報状態の基本動作

4. 対話処理の過程

提案手法で MMI システムを実現した場合の情報状態の動作について説明する (図 4).

4.1 基本動作

対話開始:各入出力モダリティと接続を開始し，接続が成功すると接続完了の応答を返す.

① システム発話行為の決定:PLAN スタックからアクションを pop して QUD フィールドへ移動

② システム発話生成:アクションクラスのドメイン名フィールドを用いて質問文を生成

③ 出力指示: 生成された出力内容を各出力モダリティへ送信し出力指示を行う．出力モダリティは出力が成功すると出力応答を返す.

④ 入力開始指示: 全ての出力モダリティが出力を完了すると，対応した入力の受付を開始する.

⑤ 入力理解結果の生成: ユーザが入力モダリティへ入力を行うと，認識結果を元に入力理解結果関数のリストを生成する．入力候補が複数存在する場合は全て関数化し，信頼度の高い順にリストにする．もしもユーザの入力を認識できない場合やエラーが生じた場合はエラーメッセージを出力する.

⑥ 関数の適用: リストから順に関数を取り出し情報状態に適用する．関数適用によって状態の更新が行われた場合，関数適用後の情報状態を新たな情報状態とする．また，リスト内の関数を全て適用しても更新できなかった場合は再入力をユーザに促す.

⑦ 状態更新後の動作: QUD フィールドのアクションが RAISE クラスのインスタンスであった場合，TMP フィールドへ移動する．QUD フィールドのアクションが RESPOND クラスのインスタンスである場合，確認が完了した場合は TMP フィールドのクラスを全て BEL フィールドへ移動し，誤解が生じた場合は 4.2 節の誤解の解消動作を行う．動作完了後□へ戻り，PLAN スタックが空になるまで繰り返す.

対話終了: TMP フィールドにインスタンスが存在していなければタスクを達成するための情報がそろった (変数が全て満たされた) とみなし対話を終了する．TMP フィールドにインスタンスが存在していれば RESPOND クラスのインスタンスを QUD フィールドへ置き，確認動作を行ってから終了する.

4.2 誤解の解消動作

ユーザの入力に対する理解結果が正しくないと分かった場合，誤りを修正するため以下の方法を用いる.

- 最初から入力をやり直す: TMP フィールドに存在する全てのインスタンスの保持している入力値を削除し，PLAN スタックへ戻し，一度行った対話を再び行う.
- 毎回確認動作を行う: 一つのインスタンスに対して入力が完了した時点ですぐに確認動作を行う．毎回 RESPOND クラスのインスタンスを置くのは手間がかかるため，responseMode パラメータを真にすることで自動的に確認動作を行う.
- 入力理解結果の他候補の適用: 入力理解結果のリストのうち，まだ適用していない他候補を適用する．これによって更新が行われればユーザの入力を省略できる.

4.3 複数のモダリティによる入力への対応

入力理解結果の関数は一つの変数に対して入力を行うものであるので，一つのモダリティの入力と一つの変数が対応していれば，複数の変数に対して一つずつモダリティを使用し，一度に変数を埋めるような入力が可能になる．QUD フィールドに並んでいるインスタンスの順番とユーザが入力モダリティを使用した順番を対応させ，順に情報状態に適用することによって，複数の変数へ逐次的な入力を行うことができる.

5. システムへの導入

提案手法を使用し，実際に MMI システムの実装を行った．実装したシステムは，開発者が情報状態の PLAN スタックに記述した一つのスロットフィリング型タスクを達成するシステムである.

5.1 ITSCJ MMI システムのアーキテクチャ

アーキテクチャとして情報規格調査会 (ITSCJ) で提案された MMI6 階層モデル [新田 07][*3]を使用した (図 5)．このアーキテクチャでは対話記述から各モダリティに関する記述を分離

*3 情報規格調査会 (ITSCJ) マルチモーダル対話のための記述言語 要求仕様 — http://www.itscj.ipsj.or.jp/ipsj-ts/ts0012/mmi-arch.html.

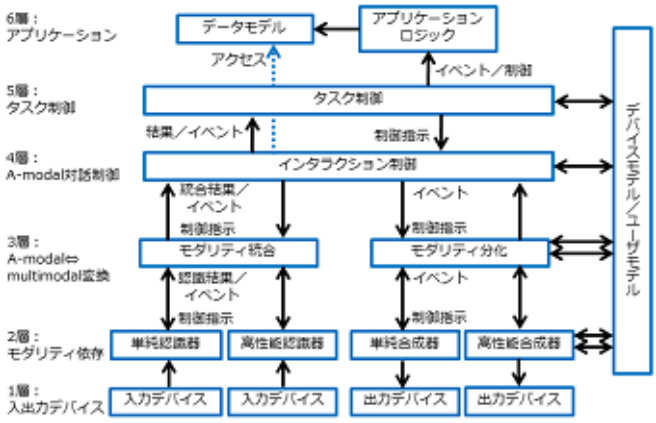

図 5: ITSCJ MMI システムのアーキテクチャ

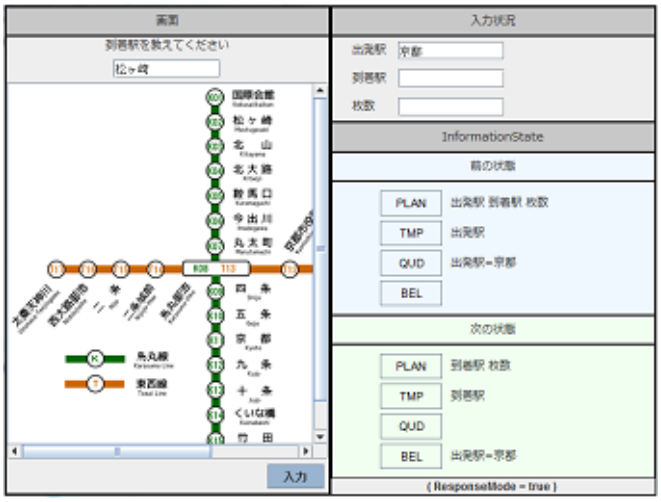

図 6: 実装した MMI システム

しており，独立性を実現させるために必要な階層同士の通信のためのイベントが定義されている．このモデルに準拠することでモダリティ依存の記述を廃止し，拡張性・保守性を高める．本研究ではタスクがあらかじめ定められているため，第 4 層以下の実装を行った．また各階層同士を独立させ通信を非同期に行うため，イベント処理をプログラミング言語 Scala における Actor モデルのメッセージパッシングによって実装した．

5.2 入出力モダリティ

実装した対話システムは，入力モダリティとして音声認識と GUI の画面操作，出力モダリティとして合成音声と画面遷移を用いて，PLAN スタックに記述されたタスクを達成するシステムである (図 6)．音声認識器として Julius[*4] を使用し，音声合成器として Open JTalk[*5] を使用した．

5.3 入力理解結果関数の生成

入力理解結果の関数を生成するため，日本語形態素解析システム JUMAN[*6] を使用した．JUMAN は日本語文を入力することによって形態素解析結果を生成できるため，入力文中に存在する単語を抽出し，その単語の持つ意味情報 (意味カテゴリ，ドメイン，固有名詞辞書，Wikipedia から抽出した辞書) を取得できる．これらの意味情報を元に，単語名とドメイン名 (意味情報) の組を生成する．JUMAN で取得したドメイン名と，スロットのドメイン名を比較し，一致すればスロットの持つ変数に単語名 (認識結果) を代入するような関数を生成する．ただし，単語が意味情報を持たない場合も，単語名を利用してドメインに含まれているかを判断することができるので，ドメイン名を空白とした入力理解結果を生成する．また，1 つの単語に対して意味情報が複数存在する単語は，複数の入力理解結果を生成する．

5.4 実行結果

PLAN スタックにスロット名とドメイン名を持つインスタンスを記述し，Julius などの入出力モダリティと接続することによって，音声認識によるスロット入力と GUI 操作によるスロット入力を行うことができる MMI システムを作成することができた．

6. 課題

関数リストの再適用による誤解の解消は，同一のドメインに対して複数候補を生成することができなかったため今回の実装では検証できなかった．また，音声でドメインを入力しながらタッチ入力で入力値を入力するような，ドメインと値を同時に指定する入力の統合処理の実装を行うことができなかった．さらに，現在の実装では GUI の一部 (例えば駅名を入力するときの地図や，枚数を入力するときのテンキーなど) をあらかじめ準備する必要があり，ドメインも JUMAN の生成する意味情報に合わせているため，対応可能なタスクが限定されている．今後はドメインと値を同時入力するための関数の生成について検討する．また，現在用いているタスクはスロットフィリング型タスクのみであるので，検索を行うタスクなど他のタスクについても関数適用を行えるように検討する必要がある．

参考文献

[菊地 13] 菊地泰己，桂田浩一，入部百合絵，新田恒雄: Web ブラウザ上で MMI システムを実行可能にする JavaScript ライブラリ MMI.js の提案, 情報処理学会研究報告. SLP, 音声言語情報処理, pp.1-2(2013).

[Johnston 02] M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor: MATCH: An architecture for multimodal dialog systems, Proceedings of ACL-02, pp.376-383(2002).

[S. Larsson 00] S. Larsson and David R. Traum: Information state and dialogue management in the TRINDI dialogue move engine toolkit, Natural language engineering, pp.323-340(2000).

[新田 07] 新田恒雄, 桂田浩一, 荒木雅弘, 西本卓也, 甘粕哲郎, 川本真一: マルチモーダル対話システムのための階層的アーキテクチャの提案, 情報処理学会研究報告. SLP, 音声言語情報処理, pp.7-12(2007).

*4 http://julius.sourceforge.jp/.
*5 http://www.sp.nitech.ac.jp/demo/open_ jtalk/.
*6 http://nlp.ist.i.kyoto-u.ac.jp/index.php?cmd=read&page=JUMAN& alias% 5B% 5D=日本語形態素解析システム JUMAN.