音声言語情報処理 19－20
(1997. 12. 12)

わかってうなずくコンピュータの試作

平沢 純一　川端 豪

ＮＴＴ基礎研究所
〒243-01 神奈川県厚木市森の里若宮 3-1
jun@idea.brl.ntt.co.jp

あらまし　人間が違和感なく対話できる音声対話システムの実現を目指している。人間が自由なタイミングで発話でき、あいづちやうなずきなどのシステムからの適切な応答によってシステムへの入力の様子を確認できるシステムにするには、システムを構成するモジュール間で密な通信を行い、ユーザの発話が終らないうちから認識の途中結果を逐次的に処理していく必要がある。その目的のために、モジュール間で通信するプロトコルの設計を行い、逐次的な処理を可能にしたシステムを試作した。さらに試作したシステムを用いて、逐次処理により出力されるふるまいの効果を調べた

キーワード　音声対話システム、対話の調整、逐次処理、モジュール間通信プロトコル

A Computer that Nods

HIRASAWA Jun-ichi　KAWABATA Takeshi

NTT Basic Research Laboratories
3-1 Morinosato Wakamiya, Atsugi-city, Kanagawa 243-01 Japan
jun@idea.brl.ntt.co.jp

Abstract　To realize a spoken dialogue system which enables us to communicate effortlessly in spoken language, we need to make the system process the user's utterances incrementally and respond to them appropriately. We designed the language of communication protocol between modules in the system, which contributes to incremental processing. We also attempted to develop a system using the communication protocol and examined the effects of the behaviors derived from incremental processing.

key words　Spoken Dialogue System,Dialogue Coordination, Incremental Processing, Inter-module Communication Protocol

1 はじめに

近年のメディア技術の発展はめざましく、人間と計算機を結ぶインターフェースとしての**音声対話システム**の実現も現実的な課題として考えられるようになった。コミュニケーションの手段として「音声対話」を用いることは、大半の人間にとって特別な訓練が必要なく、他のメディアにはない「気楽さ」に関するかぎり格段に優れている。従って、普段人間同士で対話するのと同じように**違和感なく** (effortless) 使えるのでなければ、音声対話を手段とすることの意義は半減してしまう。

目指すべきは「違和感なく対話できるシステム」である。ここで違和感のないシステムとは少なくとも

1. ユーザの発話タイミングは自由であり、かつ表現にも制約がない
2. システムからの適切な応答があり、ユーザは自分の発話が伝わったのかどうかを確認できる（システムの内部状態開示）

のふたつの要件を充たしている必要がある。

問題点 音声対話システムでは、ユーザからの入力（発話）を分節化 (segmentation) 構造化 (finding internal structure) して、最終的に発話の意図や意味の内部表現 (representation) を得たり、何らかの行動 (behavior) をする。しかしながら発話（言語表現）は本来的に部分的な情報であり [4]、曖昧性を残さず内部表現やふるまいに変換することができない。究極的には発話の解釈とは世界知識も含めた「状況」の中に置かれてはじめて可能となるものである [5]。

さらに音声対話システムでは実時間の制限が加わる [6]。人間同士の対話では、聞き手の応答（あいづちを含む）は話し手の発話に重なる（割り込む）[10] ことが確実で、自由タイミングのユーザ発話に対して、システムがあいづちを含めた適切な応答をするためには、従来の音声対話システムがしていたようにユーザの発声が完了してから認識結果を得るのではなく、発声中にも逐次的に認識の途中結果を得ていく必要がある。

表 1: 応答のタイミング

応答のタイミング	実例
(1) 間があく	通訳を介する対話、衛星中継対話、従来の対話システム
(2) 間がない	学芸会の演劇（ピンポン対話）
(3) 重なる	あいづち、同時唱和、割り込み

解決 (対話の調整) 従来の自然言語処理では統語、意味、語用などの様々なレベルでの制約や選好を情報源として情報の部分性への対処（曖昧性の解消）を試みてきた。しかし、音声対話システムでは実時間性という制限をも付け加えて対処しなければならなくなるので、「**対話の調整** (coordination)[7]」行動を曖昧性解消への情報源としなければ、違和感のない音声対話システムを実現することは不可能であろう。

情報の部分性のみならず実時間制限のもとでは、曖昧性の解消に関して十分な探索が完了していなくとも、ある時点での最善の解釈 (仮説) をとりあえず出力するしかない。当然その仮説は満足のいく精度ではないかもしれないが、システムが

1. 適切なタイミングで応答する[1]
2. システムの理解状況を積極的に開示する

ことを実践すれば、対話相手のユーザは必ずや協調的にふるまい、システムは協調的な対話の中で（後続する発話から得られる情報を補助的に用いながら）曖昧性の解消を行える可能性がある。これを「対話の調整」を情報源とする曖昧性の解消と呼ぶ。

具体的に「対話の調整」行動とは「わかったら返事をする」「わからないなら聞き返す」「不安なら確認する」などの人間としてごく当たり前のふるまいのことである。システムがこれらのふるまいを実行すれば、ユーザとシステムの間で「対話の調整」が生じ、曖昧性の解消に必要な情報もユーザが自ずと提示してくれる。実際、人間同士の対話でも、曖昧性を解消できなかったり、解釈を間違える（誤解する）ことは頻発するのであり、要は実際の対話の進行の中でそれを段階的に解決していければよい。人間の対話は、

[1] 応答のタイミングには大雑把に分けて表 1の三段階が考えられるが、違和感のない音声対話システムのためには (3) の段階を実現できなければならない。

独り閉じこもって黙々と曖昧性の解消を試みるのではなく、あいづちを交わしたり、相手の発話を繰り返したり、時には先取りしたりしながら、共同作業的に指示対象の同定や発話行為の認識を進めていくものである。逆に言えば、ユーザとの間で対話の調整を実現できるシステムでなければ、違和感のない音声対話システムとは言えまい。

対話の調整の実現　それではユーザとの間で「対話の調整」が生じるようなシステムを実現するには何が必要であろうか？まず、対話の進行を妨げないように「時間を守る」ことである。そのためにはユーザの発話が完了してから認識結果を得ていたのでは対話（応答）の適切なタイミングを実現できない。ユーザが話すそばから刻々と逐次的に認識の途中結果を得てシステムのふるまいに反映させ、各モジュールは前モジュールの処理が完了するのを待たずに、半ば同時に動作するようにモジュール間で密に通信することになる [3]。

以上の議論より、各モジュール間で通信し合うプロトコルを定めることの重要性が明らかとなる。各モジュールの入口と出口を規定して、どんなタイミング [2] でどんな情報 [11] をやりとりすればよいのかを明確にする。そうすることで各モジュールの仕様を明示し独立した開発を可能にしながら [1]、モジュール間の密な結合が実現され、時間を守って対話の調整を生じさせる違和感のないシステムを実現できることになる。

2　モジュール間通信プロトコル

違和感のない音声対話システムを実現するには各モジュール間での密な通信が必要不可欠になる。システムを設計するには、そのシステムがどんなモジュールから構成され、各モジュールの間でいつ（どんなタイミングで）何（情報内容）を通信するのかをデザインしなければならない。

そこで、モジュール間の通信に必要となる通信プロトコルを記述するための言語 (InterModuleProtocol (IMP) ver.1.0) を設計した（図 1）。この言語によるプロトコル記述では、モジュール間で「トークン」を単位として情報を通信する。トークンにはタイプが定義され、例えば、単語タイプトークンや音素タイプトークン、あるいは内容語トークンなど、モジュール構成に応じて複数の異なるタイプのトークンを定義できる。

各トークンは必須項と optional 項から成り、それぞれの項には変数の値が格納される。変数として他のトークンへのポインタを用いることができるので、トークン間の関係を記述することも可能である。これにより、例えばトークンを単体で個別に送信しても、最後に送信されたトークンからポインタを辿っていくことで構造化情報（単語ラティスなど）を再現することが可能となる。

このプロトコル記述言語を用いた、モジュール間通信プロトコルの記述例を図 2に示す。図 2は、音声認識モジュールが単語仮説を生成した時点で出力する **WORD タイプトークン (認識の途中結果)** と発声の終了を認識した時点で出力する **UTTERANCE タイプトークン (認識の最終結果＝単語列)** の記述例である。音声認識モジュールの後モジュールではこのプロトコル記述に従って出力されるトークン（例えば図 3）をもとに処理を進める。

3　システムの試作

前章で述べた逐次処理用の通信プロトコルを用いる音声対話システムを試作した。このシステムは会議室の予約受付を行う。予約受付タスクは予約表を埋めるだけの単純なスロットフィリングタスクである。ユーザからの入力は音声入力のみで行われ、システムからの出力は「画面上のＣＧ顔画像マスコット [8] の

```
プロトコル記述言語     ::= <Token_Description>*
<Token_Description> ::= <Type_Name> ":{" <Items> "}"
        <Type_Name> ::= <英大文字 | 数字 | 記号>*
          (例. WORD, PHONEME1, CONTENT_NOUN, etc.)
            <Items> ::= "OBL" <Item>* "OPT" {Item}*
             <Item> ::= <Var_Type> <Var_Name> ";"
         <Var_Name> ::= <英文字 | 記号>*
 (例. id_num,entity,start_fr_num,left_context,etc.)
         <Var_Type> ::= "int" | "char" | <Ptr_Type>
         <Ptr_Type> ::= "Ptr (" <Type_Name> ")"

<a>*,{a}* はそれぞれ a の 1 回以上，0 回以上のの繰り返し。
```

図 1: プロトコル記述言語の定義 (シンタクス)

```
WORD : {
      OBL
        int   id_num;
        char  entity;
        int   start_fr_num;
      OPT
        Ptr(WORD)       left_context;
        Ptr(UTTERANCE) upper_context;
}
UTTERANCE : {
      OBL
        int  id_num;
        char entity;
      OPT
}
```

図 2: プロトコル記述の例

```
WORD : {
      OBL
        <id_num, 15624>
        <entity, "だいいちかいぎしつ">
        <start_fr_num, 625 >
      OPT
        <left_context,  &xxxx >
        <upper_context, &yyyy >
}
```

図 3: トークンの例

顔(視線)の動き」と「少数の録音音声による発話」の2種類に限定されている。システムは以下の3つのモジュールから構成されている（図4）。(1)**音声認識モジュール**：音声入力から単語候補を出力する。(2)**単語処理モジュール**：音声認識モジュールの途中結果から単語を処理して「あいづち(うなずき)/聞き返し(首かしげ)」を出力する。(3)**予約処理モジュール**：音声認識モジュールの(発声終了後に得られる)最終結果から会議室の予約表を埋めて、予約状況の確認発話を出力する。

3.1 音声認識モジュール

音声認識モジュールは音素ごとに連続分布型の隠れマルコフモデルを用いた。また音素モデルの学習には音響学会の連続音声データベース[9]を用いた。語彙数は助詞・助動詞・接辞・間投詞を含めて56単語で、文法規則は43のネットワーク文法である。

音声認識モジュールでは単語候補が生成される度にプロトコルに従って単語トークンを出力する（単語処理モジュールに送られる）。またユーザ発声が完了した時点で発話トークン（単語列）も出力する（予約処理モジュールで使われる）。

表 2: 単語処理モジュールの出力

行動	視線	音声	開示される意味
よそ見	浮遊	-	入力を受け付けていない
注視	正面固定	-	入力を受け付けている (i.e. 何らかの処理をしている)
あいづち (うなずき)	上下	はい	ある程度のスコアで語を認識した
聞き返し (首かしげ)	横	え？ は？	低いスコアで語を認識した (i.e. 自信がない)

3.2 単語処理モジュール

単語処理モジュールは音声認識の途中結果を用いて「うなずき / 首かしげ」などを出力する。単語処理モジュールでは、認識モジュールからプロトコルに従って出力されてくる単語トークンを受け付け、その単語トークンのスコアに応じて「うなずき / 首かしげ」などのふるまいをする。単語モジュールが反応する単語は会議室予約に関連した名詞（曜日 / 時間 / 部屋名）である。単語処理モジュールは、これらのキーワードを認識したことを以って「理解」としており、現在は発話の意図（発話行為）を理解していない。

単語処理モジュールが出力可能な行動を表2に示した（確認発話の生成は予約処理モジュールが担当している）。また、それらのふるまいのタイミングに応じて、(1)**逐次処理タイプ**(ユーザの発話途中でも応答する)、(2)**一括処理タイプ**(ユーザの発話終了を待ってから応答する)、(3)**ランダムタイプ**(ユーザの発声と無関係に応答する)の3種類のモジュールバリエーションを用意した。

3.3 予約処理モジュール

予約処理モジュールは、ユーザの発話が完了して初めて認識モジュールから出力される発話トークン(認識の最終結果＝単語列)を入力として受け付ける。発話トークン中に、会議室予約のための単語(曜日 / 時間 / 部屋名)があれば、予約表を埋める。予約処理モジュールは予約表の3つのスロット（曜日 / 時間 / 部屋名）すべてが埋まった時点で予約内容の確認発話生成を始める。確認した予約内容に間違いがあってユー

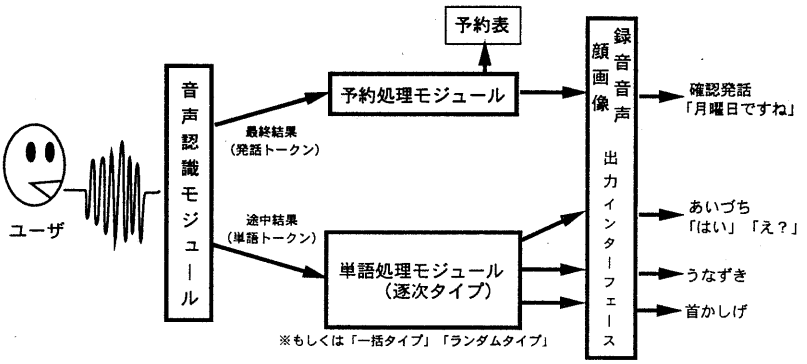

図 4: システムの構成

ザが訂正した時は指摘された予約内容を修正して再度確認する。対話は、あらかじめ決めてある表現[2]をユーザが発話するか、制限時間(90 秒)を超えて強制終了するか、のいずれかで終る。

3.4 理解とふるまい

我々の音声対話システムにおいては、各種のふるまいがシステムの理解状況・内部処理状態の開示として出力されることで「対話の調整」を生じさせようとしている。内部状態とふるまいとの対応関係[3]は現在のところ設計者が恣意的に決めている。

また現在は、認識の途中結果の単語トークンをうなずき行動の生成に用い、最終結果の発話トークンを予約に用いている。これは認識途中結果の単語トークンでは認識の信頼性が低く、ユーザが発話した覚えのない単語を誤認識してしまう湧き出し誤り(空耳現象)が頻発するためである。

4 実験

前章で述べた対話システムを用いて、逐次処理プロトコルによるふるまいの効果を調べた。実験は音声認識モジュールと予約処理モジュールを共通にして、単語処理モジュールの出力(うなずき / 首かしげ)のタイミングによって異なる 3 種類のシステム(逐次タイプ・一括タイプ・ランダムタイプ)を用意し、その効果を調べた。

4.1 設定

被験者 音声言語研究に関連する 5 人(男 4 女 1)

対話数 練習を除き、ひとり 3 対話を収録した(図 5)。

タスク 被験者には「曜日(月曜〜金曜)」「時間(午前 9 時〜午後 5 時, 1 時間毎)」「部屋名(第 1 〜 3 会議室)」が指定され、WS 画面中のマスコットに向かって予約を取るよう指示された。システムが予約内容を把握すると確認発話をしてくることは事前に明かされた。

実験条件 システムのうなずき応答のタイミングに応じて 3 つの実験条件(逐次・一括・ランダム)が設定された。

実験後 1 対話が終了するたびに、被験者はシステムへの印象(職務忠実度、快適度)を五段階評定し、簡単なコメントも記入した。

装置 SGI O2 ワークステーション 1 台。被験者はシステムからの出力音声をイヤホンで聞いた。

4.2 結果と分析

ふたつの観点から実験結果の分析を試みた。ひとつは「会議室を予約する」というシステムの本来の使命がどの程度達成されたのか?(**課題の達成度**)という観点から、もうひとつは「違和感のない音声対話インターフェース」という目的がどの程度実現されたのか?(**対話の快適性**)という観点からである。

[2]例えば現在は「ありがとうございました」としてある。

[3]ふるまいのセマンティクスと考えることができる。問題が厄介なのは、あいづちの「はい」ひとつ取っても、必ずしもシステム設計者の思惑通りに機能するとは限らないところである。

	開始 - 終了	発話	
S	0.00 - 0.40	beep	(対話開始)
U	1.64 - 2.36	あのー	[予約フェーズ]
U	3.28 - 5.28	会議室の [1] 予約をしたいんですけど	↓
S	3.95 - 4.19	[1] はい	↓
S	5.52 - 5.72	[2] はい (○)	↓
U	6.71 - 7.65	金曜日の [3]	
S	7.65 - 7.81	[3] はい	
U	8.40 - 9.45	午後四時か [4] ら	
S	9.25 - 9.47	[4] はい (○)	
U	9.94 - 11.27	第二会議室 [5] を [6]	
S	10.71 - 10.93	[5] はい	
S	11.27 - 11.51	[6] はい (○)	
U	11.66 - 12.48	お願 [7] いします	
S	11.70 - 12.03	[7] え？	
S	12.80 - 13.46	[8] はーい (○)	
S	13.48 - 15.18	わかりました えーとー	(スロット埋まる)
S	15.20 - 18.23	水曜日午後四時から第二会議室	(最初の確認発話)
S	18.25 - 19.73	ということでよろしいでしょうか	[確認フェーズ]
U	20.37 - 21.40	金曜日の	↓
U	21.62 - 22.69	午後四時か [9] ら [10]	↓
S	22.16 - 22.37	[9] はい	↓
S	22.47 - 22.66	[10] はい (○)	
U	23.01 - 24.16	第二かい [11] ぎしつ	
S	23.48 - 23.69	[11] はい	
U	24.22 - 25.38	を [12] お願いします [13]	
S	24.24 - 24.43	[12] はい (○)	
S	25.06 - 25.24	[13] はい	
S	25.66 - 26.34	[14] はーい (○)	
S	26.39 -	金曜日午後四時から	(確認発話
	- 30.08	第二会議室ですね	2回目)
U	30.05 - 30.38	はい	
U	30.42 - 31.03	そうです	
S	31.16 - 31.38	[15] はい	
U	31.81 - 32.96	ありがとうございました	(終了表現)
S	33.32 - 33.44	[16] は？	
S	33.46 - 34.39	どういたしまして	(対話終了)

図 5: 逐次条件での対話の実例（○は適切な「はい」）

4.2.1 課題の達成度

収録した対話の一覧を表 3に示した。全 15 対話のうち制限時間 (90 秒) 内に対話を完了できなかった対話が 4 つあった（全対話の 16%）。4 つの対話が制限時間内に対話を終えられなかった原因は、いずれも予約内容の確認フェーズ（予約スロットが埋まって以降）において予約内容の修正と確認を円滑に行えなかったためであり、そもそも予約スロット自体に情報を入力できなかったためではない。

つまり、時間超過対話は確認フェーズ (システムの確認発話生成以降) において生じているのであり、確認フェーズは各実験条件で共通の「予約処理モジュール」の性能に負うところが大きい。従って、時間超過対話の数や総所要時間（最短 31.12 秒、最長 90 秒制限超過）で実験条件間の違いを論じるべきでない。

そこでまず最初に各条件共通の「予約処理モジュール」自体の性能を調べた。つまり、発話トークンを用いた予約処理モジュールがどの程度の認識性能でどの程度の課題達成性能があるかを調べた。確認フェーズに入って（全スロットがうまり 1 回目の確認発話が生成されて）から対話が完了するまでの所要時間は平均 34.67 秒で、最短 9.63 秒（確認発話 1 回）、最長 65.54 秒（制限時間超過。確認発話 17 回）だった。生成された確認発話の回数は 1 対話あたり平均 5.6 回で、そのうちユーザの発話を正しくとらえた確認発話は 2.2 回であり、全確認発話のうち約 60% は的外れな（不適切な）確認をしていたことになる。語彙や文法のデザインの不適切さも原因であるが、システムとしてこの程度の課題遂行能力しかない段階では、インタフェースとしての快適さの条件の違いを比較するのに不十分である。

表 3: 収録対話

条件	収録対話数	時間超過対話数	完了対話数	完了対話の平均所要時間 (sec)
逐次	5	1	4	42.11
一括	5	3	2	34.05
ランダム	5	0	5	53.70
全体	15	4	11	45.91

表 4: 予約フェーズでの対話のようす

条件	平均所要時間 (sec)	はい数	え？数	うなずき等の平均間隔 (sec)
逐次	18.73	6.8	3.0	1.91
一括	25.04	4.2	5.6	2.55
ランダム	23.41	6.8	4.6	2.05
全体	22.39	5.9	4.4	2.17

1 対話あたりの平均値

4.2.2 対話の快適性

確認フェーズに入ってからでは予約処理モジュールによる「確認発話生成」が対話進行の成否を握ってしまうので、対話の快適性の違いを調べるために、分析の対象を単語処理モジュールの種類の違いが現れやすい「予約フェーズ（確認発話が生成される以前）」に限定した。

予約フェーズにおける各実験条件ごとの平均所要時間、うなずき等（「はい」「え？」「(首かしげ)」）

の平均出力数、及びその出力間隔を表4に示した。所要時間の短さを以って対話が順調に行われたと考えれば逐次処理システムが優れていると言えるかもしれないが、ことはそれほど単純ではない。

最大の誤算は、システム内部での「処理メカニズムによる違い」が、そのまま「うなずき等の機能の違い」に直結するとは限らない点である。例えば、ユーザの発声がない無音区間にシステムが唐突に「え？」と出力しても対話の進行に大きな影響は与えないが、ユーザがやりとりの中から予約を確認できたと感じた直後にシステムが「え？」を出力するとそれまでに築いた確認はご破算になる（少なくともユーザを不安にする）。いずれの「え？」も同じメカニズムで出力されていたとしても、その「え？」が対話に与える影響力には大きな差が出る。

他にも例えばユーザの「もしもし」という発話に対してシステムが「水曜日」と誤認識したために「はい」とあいづちを打つと、システムの内部的には「誤認識によるあいづち出力」とみなせるが、対話の流れ的にはむしろ「ちゃんと呼び掛けに応えた」という安心感をユーザに与えている。また、ユーザの発話途中に応答する逐次条件のあいづちも、ユーザの発話が短ければ一括条件と違いがなくなる。逆に一括条件によるあいづちもユーザの発話が連続すれば、偶然に後続発話の途中に出力されて逐次あいづちと受け取られる場合もある。

各条件におけるうなずき等の効果を調べるには、それぞれのうなずきやあいづちについて、それがシステム内部のどのようなメカニズムで生成されたのか？を考えるだけでなく、そのうなずきが結果として対話の流れとユーザにどんな影響を与えたかを考慮しなければ厳密な調査とは言えない[4]。

また対話の快適性については、対話終了後、被験者に「受付係としてのシステムの愛想の良さ」を「好感が持てる」を最高5点として5段階評価してもらった。その点数を単純に平均すると、逐次条件(3.4)、一括条件(2.8)、ランダム条件(3.6)となるが課題の達成が思わしくなかった（誤認識が多かった）対話には低い快適性をスコアリングする被験者もいるので、課題遂行との相関を調べたり、課題達成と独立させた印象を尋ねるよう質問を工夫しないと快適性の印象評定に課題達成度の要因が絡んでしまう。

4.3 考察

「モジュール間通信プロトコル」の効用を示すため、「逐次処理によるうなずき等」が「音声対話システムの快適性」に与える効果を調べたが、他の実験条件と比較評価する以前の不十分な結果しか得られなかった。

理由のひとつは、「課題達成度の未熟さ」が快適性調査の障害となってしまったことである。タスク指向対話を対象とする限り、そもそもの課題が達成できたのかどうかが対話の進行にも印象にも大きな影響を与える。一定レベルの課題達成性能が実現しないうちに、他の要因（対話の快適性など）について独立して議論するのは難しい。

ふたつめは、システムが出力する「ふるまいの機能を十分に統制できていなかった」ことである。システム内部の「処理方式の違い」をそのまま出力に反映させても、それがそのまま対話における「ふるまいの機能の違い」として現れるとは限らない。被験者の中には、ランダム条件によるあいづちの出力タイミングを、ランダムと感じず「処理の遅延によるタイミングのずれ」と感じた例もある。人間は表出されるふるまいに過剰に意味を読まずにはいられない存在であることを肝に命じるべきである。

5 今後の課題

実験の中から明らかになった今後の課題（システムの改造すべき点）を3つ指摘する。ひとつは、とにかく「システムの課題遂行能力を高める」ことである。そのためにはまずシステムが「対話の履歴情報」を利用することである。「はい」を「火曜日」と誤認識してしまうことから生じる確認発話の半無限循環[5]などを避けるには、単に以前に言及されたかどうかを参照するだけでも大きな効果があるだろう。また音声認識モジュールの劇的な性能向上が望めないならば、

[4] しかしそれぞれのうなずき等をユーザがどう受け止めたかを調べて把握するのは、現実的には難しい。

[5] システム「火曜日ですね？」　ユーザ「はい」　システム「火曜日ですね？」　ユーザ「はい」... を9回繰り返した対話例。しかしこの場合、システムはともかく、ユーザも頑固である。

システムの側が積極的に対話の主導権を取り巧妙に対話を誘導しながら、動的に文法や辞書を交換して、システム自身の認識の弱点を補償するようにふるまうのも一案である（被験者の感想の中に「システムに対話を誘導される方が話しやすい」というものがあった）。

ふたつめの課題は、不適切なふるまいを減らすため「時間管理を一層強化する」ことである。システムの理解状況の開示としてのふるまいが不適切に機能するのは、そもそも理解状況（認識結果）の精度が十分でない場合も多いが、「開示するタイミング」次第で開示が適切にも不適切にもなる。たとえ適切な開示でも、それが機を逸していたら開示しない方がマシである。各モジュールが逐次的に動作していも、ふるまいとして出力するタイミングの判断を行う機構（謂わば時間管理モジュール）が必要である [6]。違和感のない対話を実現するため逐次的な処理が必要なことに変りはないが、システム側の都合で「処理結果が出たタイミング＝ふるまい出力のタイミング」とするのではなく、あくまでも「対話の流れ」の都合に合わせてふるまいのタイミングを定めるべきである。

3つめの課題はモジュールが複数存在してもシステムはひとつであると考え「モジュール間の一貫性を保つ」ことである。今回の試作システムは、認識の途中結果を用いてうなずき等の逐次性を狙いつつ、認識の最終結果を用いて課題の達成を実現しようとした。しかしふたつのモジュールの間で一貫性を保持する機構がなかった。その結果、単語モジュールが「はい」と答えておきながら予約を入れずにいたり、逆に単語モジュールが「え？」と認識スコアの不十分を開示しておきながら予約モジュールが予約を入れてしまうなどの不適切な挙動をする場合があった。内部状態の正しい開示のためにもこの課題はさらなる検討を要する。

6 まとめ

違和感のない音声対話システムを実現するため (1) モジュール間の通信プロトコルを設計して音声認識の途中結果の利用を可能にし (2) 逐次的な処理を可能にするシステムを試作した。そのシステムを用いて (3) 逐次処理によるうなずき等が対話の快適性に供する効果を調べたが、課題の達成能力が不十分だったため、及び、ふるまいによるシステムの内部状態開示がタイミングも含めて十分に統制しきれていなかったため、十分な比較評価が行えなかった。単に逐次処理を行うだけでなく、ふるまいのタイミングを統制する機構の必要性が示唆された。

謝辞 日頃よりご指導いただく NTT 基礎研究所情報科学部 石井健一郎部長、有益な示唆をいただく対話理解研究グループの諸氏、実験の準備にご協力いただいた NTT-AT 社の木間良子氏、久保田哲也氏に感謝いたします。

参考文献

[1] Amtrup,J.W.,Berna,J. :Communication in large distributed AI Systems for Natural Language Processing. *COLING96*, pp.35-40 (1996)

[2] Görz,G., Kesseler,M. :Anytime Algorithms for Speech Parsing? *COLING94*, pp.997-1001 (1994)

[3] Görz,G., Kesseler,M.,Spilker,J.,Weber,H. :Research on Architectures for Integrated Speech/Language Systems in Verbmobil. *COLING96*, pp.484-489 (1996)

[4] 橋田浩一: 人工知能における基本的問題. 人工知能学会誌, Vol.10, No.3, pp.340-346 (1995)

[5] 平沢純一, 松本裕治: 関連性理論を用いた発話の解釈. 第50回情報処理学会全国大会論文集 (3), pp.139-140 (1995)

[6] 伊藤克亘, 速水悟, 田中和世: 対話システム制御における時間の扱い. 平成8年春季 音講論 1-5-10, pp.19-20 (1996)

[7] 片桐恭弘: 対話調整の分散処理モデル. 情処研報 SLP 94-2, 情報処理学会 (1994)

[8] 川端豪: 音声理解システム JUNO における対話マスコット. 平成9年春季 音講論 2-Q-2, pp.143-144 (1997)

[9] 小林哲則, 板橋秀一, 速水悟, 竹沢寿幸: 日本音響学会研究用連続音声データベース. 日本音響学会誌 48 巻 12 号, pp.888-893 (1992)

[10] 小坂直敏: あいづちを中心とした会話音声の呼応関係の分析. 信学技報, SP87-107, 電子情報通信学会 (1987)

[11] 島津明, 小暮潔, 川森雅仁, 堂坂浩二, 中野幹生: 対話処理システムにおける内的コミュニケーション. 言語処理学会第二回大会, pp.26-27 (1996)