自然言語対話システムにおけるユーザモデルの更新に伴う対話の再プランニングに関する考察

佐川雄二† 大西昇† 杉江昇†

自然言語による対話システムにおいて対話のプランニングを行う際，最も大きな影響をもつのは，ユーザの信念に関するユーザモデルである．ユーザの信念に関する情報をあらかじめすべてもっておくのは不可能であるため，システムはデフォールトに基づくことになるが，対話を進めるうちにそれが誤っていたことが明らかになることがある．また，ユーザが対話の途中で前にいったことを翻すこともある．このような場合，単にユーザが間違った信念を抱いているというだけなら，システムはそれを指摘し修正するだけで良いが，それまでの対話の一部がその間違っていた信念と何らかの関係をもつ場合，その部分の対話は結果としてうまく機能しなかったことになる．本論文では，まず対話のプランニングの見地からユーザモデルの更新がすでに行われた発話に与える影響を分析する．続いてユーザの発話から発話のプランニング過程を推論することでユーザモデルの更新を行い，対話への影響を検出する手順について論ずる．この際本研究ではユーザの発話から得られた情報を仮定として ATMS を用いて管理し，ユーザモデルの更新のチェックを行う．そして，すでに実行した対話に影響するとき，対話を修復し続行するためのメタプランニングの手法を提案する．この手法を導入した対話システムの概要についても触れる．

On Replanning of Natural Language Dialogue Caused by Updating a User Model

Yuji Sagawa,† Noboru Ohnishi† and Noboru Sugie†

In this paper, we investigate how updating a user model influence on dialogue planning. Dialogue planning highly depends on user's belief. But it is always changing during the time course of dialogue. A user may change his mind, or the system's assumption about user's belief may be wrong. So it may be revealed afterwards that some part of dialogue does not work well. We propose some strategies for replanning of dialogue when update of a user model has influence on some part of dialogue carried out so far. And we show the outline of dialogue system incorporating the strategies.

1. はじめに

近年，自然言語対話システムの研究において，話者があるゴールを達成するためにとる行為の一つとして発話をとらえ，発話の生成をプランニングの側面からモデル化しようとする試みが盛んになっている[1]．このような枠組では，文の結束性を保ちながら，しかも聞き手の信念や関心を参照しながら対話の流れを制御できるため，「聞き手がすでに知っていることは述べない」といった柔軟な対話を実現できる．

この際重要な役割を果たすのがユーザの信念や意図を表すユーザモデルであるが，ユーザの信念に関する情報をあらかじめすべて持っておくのは不可能であるため，システムはあるデフォールトに基づくことになる．また，ユーザモデルはユーザの発話から得られる情報を基に対話中更新し続けなければならない．したがって，例えば前言の翻しなどのように，更新された部分が更新する以前の対話で用いられていた場合がでてくる．この場合，対話を修復する必要が出てくる可能性がある．

しかし，従来の対話のプランニングに関する研究では，実行したプランはすべて成功したものとして扱われ，対話を修復する必要はないとの仮定のもとに行われてきた．

本論文では，ユーザモデルの更新によって以前の対話が影響を受ける場合があることを指摘し，その場合に対話を再プランニングし，更新がもたらす影響を解消して，対話を続行する手法について考察する．

ユーザモデルが対話に影響を持つ最も典型的な場合

† 名古屋大学工学部情報工学科
Department of Information Engineering, School of Engineering, Nagoya University

はユーザが誤った信念すなわち誤解を持っていることが検出された場合である．このような場合にシステムがどう行動するかについてはいくつかの研究があり，ユーザの誤解をいくつかのクラスに分け，ユーザの発話から推論されるユーザの信念をチェックし誤解の検出とその後のシステムの対応の決定を行う方法が提案されている．例えば，Kaplan の CO-OP システム[2]では，ユーザの

Who advised projects in area 36?

という発話から，ユーザがエリア 36 が存在していると信じていると推論する．そして仮にそのような事実がシステムの知識に存在しない場合これを誤解として検出し，そのことをユーザに告げる．

CO-OP では，システムはユーザが誤解しているという事実を告げるのみで，誤解を修復する機能は持たなかったが，その後，ユーザの誤解のクラスに応じてそれを修復する発話を行うためのストラテジを備えたシステムが提案された．例えば，McCoy の ROMPER[3]では，オブジェクトに関する誤解，Chin の KNOME[4]では，オブジェクト間の関係に関する誤解，そして Quilich の AQUA[5]では，プランに関する誤解についてそれらを解消するための発話を生成するストラテジについて述べられている．

しかし，いずれの場合もユーザの誤解を修復することと，ユーザモデルを更新することでその後の対話に対する影響を取り除くことのみを扱っており，本研究で扱うような，すでに行われた発話に対する影響は考慮されていない．また，誤解のチェックもユーザの発話に含まれる命題のみをチェックするなど簡単な場合のみを扱っている．また，ユーザの誤解は，単にユーザの対象領域に関する誤解だけでなく，理解していると思っていたものが実はよく理解できてなかった，といったユーザ自身に関する誤解などもあるが，これらを対象にした研究は行われていない．

本論文では，まず対話のプランニングの見地からユーザモデルの更新がすでに行われた発話に与える影響を分析する．続いてユーザの発話から発話のプランニング過程を推論することでユーザモデルの更新を行い，対話への影響を検出する手順について論ずる．この際本研究ではユーザの発話から得られた情報を仮定として ATMS(Assumption-based Truth Maintenance System)[6] を用いて管理し，更新のチェックを行う．そして，すでに実行した対話に影響するとき，対話を修復し続行するためのメタプランニングの手法を提案する．この手法を導入した対話システムの概要についても触れる．

2. ユーザモデルの更新と対話への影響

まず，次の（対話 1）を考える．

(対話 1)

ユ ー ザ：すき焼きの作り方を知りたいのですが．(1)

システム：関東風と関西風の
　　どちらがよろしいですか？(2)

ユ ー ザ：関西風がいいです．(3)

システム：わかりました．(4)
　　まず鍋を熱します．(5)

ユ ー ザ：はい．(6)

システム：次に肉と野菜を鍋にいれます．(7)

ユ ー ザ：あの，
　　やっぱり関東風を知りたいんですが．(8)

システム：わかりました．(9)
　　まず鍋を熱します．(10)

ユ ー ザ：はい．(11)

システム：つぎに肉を焼き，割り下を加えます．(12)

（対話 1）の発話（3）から，システムはユーザが関西風すき焼きの調理法を知りたいのだということを知る．このユーザモデルにしたがってシステムは対話を進めていくが，発話(8)は，このユーザモデルを更新すべき情報を含んでいる．新しいユーザモデル（ユーザは関東風すき焼きの調理法を知りたい）が正しいとすると，それまでの対話（(4)から(7)）は，やり直す必要がある．そこでシステムは新しいユーザモデルに従った対話をやり直している．

本論文では，対話をある目的（ゴール）を持ったシステムとユーザの一まとまりのやりとりと考える．対話は，前提条件および結果を持つ．したがってこれらのどれかが後に更新された場合には，その対話は成功したとはいえない．

一般にプランがゴールとサブゴールの階層構造をなすように，対話のプランニングにおいてもプランは階層的に呼び出され[7]，その結果，対話は**図 1** に示すように，いくつかの副対話に分割され，それぞれの副対話がさらに副対話に分割される階層構造を持つ[8]．

（対話 1）のようにシステムのユーザモデルの更新にともなって対話を再プランニングせねばならない状

況は，実際の対話においてしばしば生じる．これらの原因は，ユーザ側の自己認識の不確かさからだけでなく，システムの仮定したユーザモデルが不適切である場合にも起こり得る．システムは対話に先だってユーザの完全なモデルを得ることは不可能であるため，あるデフォールトのユーザモデルを仮定して対話を始めることになるが，そうすると，このようなシステムのユーザモデルを実際にユーザから得られる情報を用いて更新しなければならなくなる可能性がある．

Alterman[9] は，常識的プランニングにおけるプランニングの失敗をプランのゴール，前提条件，結果の3つのうち，どの部分が失敗したかによって分類している．対話のプランニングにおいても，ユーザモデルの更新による以前の対話への影響をどう修復し対話を続行するかは，ユーザモデルの更新が対話のどの部分に影響するかによって変わる．そこで本論文では，ユーザモデルの更新の対話への影響を以下の四つに分ける．

1 以前の対話の前提条件への影響
2 以前の対話の結果への影響
3 ユーザの発話の目的への影響
4 以前の対話に影響しない更新

以下，これらの簡単な説明を行う．

以前の対話の前提条件への影響

対話はある前提条件の下で行われる．例えば，質問をする場合には，相手が回答能力を持っていると質問者は信じていなければならないし，ペンを取ってもらうよう依頼する場合には，ペンが相手の取れる範囲になければならない．

対話における前提条件は，主に相手の信念や能力などに関わるものが多いため，ユーザモデルの更新によりそれまでに行った対話の前提条件が実は間違っていた，という事態が起こる可能性がある．その場合，その対話はやはり直す必要が生じる．次の（対話2）を考える．

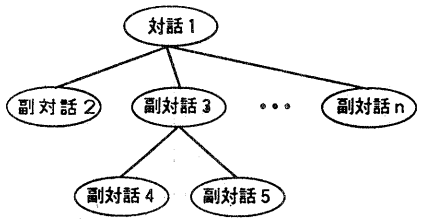

図1 対話の階層構造
Fig. 1 Hierarchical structure of dialogue.

(対話2)

システム：ではお名前と電話番号を
おっしゃって下さい．(1)
ユーザ：引っ越したばかりで
電話がまだないんですが．(2)
システム：じゃ電話番号は結構です．(3)
お名前をおっしゃって下さい．(4)

(対話2)では，(デフォールトのユーザモデルに基づいて)ユーザが電話を持っていることを前提に行われているが，発話(2)によりこの前提条件が間違っていたことが明らかになる．そこでこの対話をやり直し，電話番号は特になくても差し支えない場合，(3)(4)のように対話を続ける．

デフォールトのユーザモデルを持たず，不明なものはユーザに尋ねるという方法をとればこのような事態は生じないが，すべてのユーザに関する情報についてそのようにすると対話が繁雑になり過ぎる．したがって大多数のユーザがそうであるような情報はデフォールトとして持っておくのが自然だと思われる．

なお，前出の（対話1）もこのタイプの更新が起こっている例である．

ここで，「以前の対話」とは，現時点までに行った発話からなる対話を指す．例えば図1において，対話5が現在実行中の対話で，対話5の一部をすでに発話しているとすると，対話1, 2, 3, 4, 5の目的を指す．

以前の対話の結果への影響

対話を行うに従ってユーザの心的状態は変化していく．対話の目的はその変化をひきおこすことであるが，対話の直接の目的だけでなく，副作用としてもたらされる変化もある．これらを対話の結果と呼ぶ．例えば，質問に答えることは，相手にその答を知らせるだけでなく，自分がその答を知っているという情報を相手に知らせることになる．

対話の結果がうまく得られていないことが明らかになった場合，その部分の対話はうまく行われていない可能性があるのでやり直す必要がある．次の（対話3）を考える．

(対話3)

システム：次に醤油をおおさじ2杯入れます．(1)
ユーザ：はい．(2)
システム：次に味噌をおおさじ1杯入れます．(3)
ユーザ：はい．(4)
醤油と同じ量？(5)
システム：いえ，醤油はおおさじ2杯で，

味噌はおおさじ1杯です．(6)

(対話3)では，発話(5)からユーザが醤油と味噌の分量を正しく理解していないことがわかる．これは対話(1)～(4)の結果として得られる「ユーザが醤油と味噌の分量をそれぞれ理解している」というユーザモデルを更新しなければならない．そのため，対話(1)～(4)とおなじ内容の対話をやり直す必要がある．

ユーザの発話の目的への影響

ユーザが対話の目的を正しく認識している限り，対話の各時点でのユーザの発話はほぼ予測できる．細かい内容まではわからなくても，質問をすればそれに対する答を行うであろうことは予測できる．

ところが料理の説明をしている最中に時刻を尋ねるなど，そうした予測を越えた対応をユーザがとる場合がある．

この場合，新しい対話を行った後，現在の対話に戻ればよい．

これは，Litman ら[10]や Cawsey[11] によって扱われた subdialogue であり，現在実行中の対話の状態を保存した後，新しい対話を行い，終了した時点で中断した対話を再開すればよい．対話の重要度の判断を一般的に行うのは難しいので，本研究では，新しくユーザの出した対話の目的が常に重要であると仮定する．

また，対話の内容によって現在の対話に戻らない場合もあるが，これも対話の重要度の決定と密接に関連するので，本研究の対象外とする．

以前の対話に影響しない更新

これが従来のユーザの誤解として扱われてきたものである．主にユーザが対話の対象となっている一般世界に関する誤解を持っている場合が多い．

この場合はその誤解をとるための説明を行った上で，元の対話に戻ればよい．

3. 対話への影響の検出と対話の再プランニングを行う試作システム

前章で述べた対話に影響するユーザモデルの更新を検出し，対話の再プランニングを行う機能を対話システム RECIPE[12] 上にインプリメントした．**図2**にその概要を示す．RECIPE のタスクは料理の調理法など一連のある手順をユーザに教えることである．しかし，以下で述べる対話のプランニング，ユーザモデルの矛盾の検出，対話の再プランニングの手法は，料理の調理法でも機械の使い方でも同様であり，特に料理の調理法というタスクに依存するものではないことを付け加えておく．

システムは対話プランと呼ぶ対話のプランニング知識に基づいて対話を進める．ユーザモデルは ATMS を用いて管理し，更新が行われたときは対話プランナに報告される．報告を受けた対話プランナは更新が対話に影響するかどうかを対話の履歴を調べて決定し，その種類に応じて対話を再プランニングする．その際用いる知識がメタ対話プランである．

以下，対話のプランニング，ユーザモデルの更新の対話への影響の検出，対話の再プランニングについて詳細を述べる．文解析部，文生成部の詳細については，筆者らが以前報告したシステム[8]とほぼ同様であるので，そちらを参照されたい．

対話のプランニング

料理の調理法の説明もビデオの操作法の説明も，ある程度共通した構造を持っている．これは，人が対話を行う際にその目的に応じた対話の方法すなわち何をいつ発話し，相手の発話をどう処理すればいいか，に関する知識を持っていることを示している．これが対話のプランニング知識であり，対話の状況に応じた柔軟な対応が可能であるため，対話システムのみならず，このような言語使用に関するプランニング知識は，広く一般のテキスト生成の議論の中で注目を集めている[13)~16)]．

これらのプランニング知識はさまざまな名前で呼ば

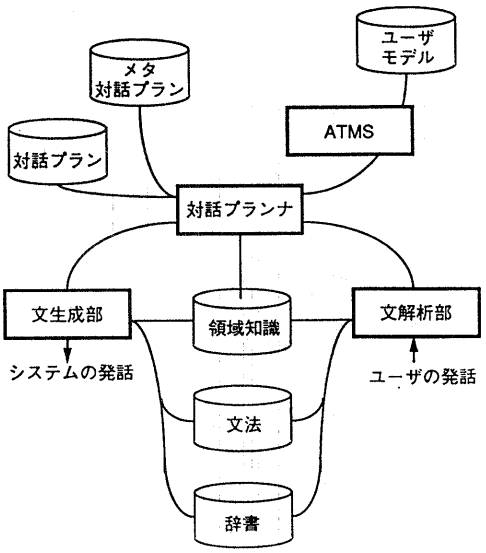

図2 RECIPE の概要
Fig. 2 The architecture of RECIPE.

れているが，本研究で用いるものは「対話プラン (dialogue plan, 以下 DP)」と呼ぶことにする．**図3**に対話プランの例を示す．また，**表1**にプランオペレータの働きを示す．なお，図中，know などの概念を表すのに簡単のため述語表現を用いているが，実際のシステムではフレームとして実現している．

%consult は，RECIPE のトップレベルのプランであり，ユーザの要求を調べ (%clarify-slots)，それに応じた DP を呼び出す．現在のインプリメンテーションでは，手順の説明 (%explain-procedure) のみが可能である．手順の説明は，まずそのゴールを明確にし (%clarify-class，(対話1) で関西風か関東風か尋ねる部分)，各処理をその順に説明する (%explain-step) ことによって行われる．%explain-step では，ユーザがその説明を理解したことがはっきりすれば終了し，次のステップの説明に移るが，そうでない場合には，さらに詳細を説明するため，%explain-procedure を実行する．

各 DP は，ゴール (:goal)，前提条件 (:precond)，結果 (:effect)，本体 (:body) からなる．プランは前提条件が満たされるときのみ実行可能で，これにより「既知の情報について質問しない」，「相手の知っていることを説明しない」などが実現できる．

```
(defdp %consult (?speaker ?hearer)
      :precond   (able ?speaker (help ?speaker ?hearer))
      :effect    nil
      :body      (optional (%clarify-slots ?speaker ?hearer
                           `(want ?hearer (know ,?hearer unknown))))
                 (select (%explain-procedure ?speaker ?hearer ?plan)
                      ;;; other types of dialogues
                 ))

(defdp %clarify-slots (?speaker ?hearer ?con)
      :precond   (know ?hearer ?con)
      :effect    (know ?hearer (know ?spreaker ?con))
      :body      (do-actions (?slot? (unknown-slot ?con))
                      (%%ask-wh-q ?speaker ?hearer ?con ?slot?)
                      (%recog ?speaker ?hearer
                           `(%%inform-filler
                                ,?hearer ,?speaker
                                ,?con ,?slot? ?filler))))

(defdp %explain-procedure (?speaker ?hearer ?goal)
      :precond   (want ?hearer (know ?hearer (plans-of ?goal)))
                 (know ?speaker (plans-of ?goal))
                 (not (know ?hearer (plans-of ?goal)))
      :effect    (know ?hearer (plans-of ?goal))
      :body      (optional (%clarify-class ?speaker ?hearer ?goal))
                 (do-actions (?plan? (plans-of ?goal))
                      (%explain-step ?plan?)))

(defdp %explain-step (?speaker ?hearer ?step)
      :precond   (not (know ?hearer ?step))
                 (know ?speaker ?step)
      :effect    (know ?hearer ?step)
      :body      (%inform-prop ?speaker ?hearer ?step)
                 (select (%recog ?speaker ?hearer
                           `(%%inform-recog ,?hearer ,?speaker))
                      (%explain-procedure ?speaker ?hearer ?step)))
```

図3 対話プランの例
Fig. 3 Example dialogue plans.

表1 プランオペレータの機能
Table 1 Functions of example plan operators.

オペレータ名	引　数	機　能
do-actions	(変数式) プラン1…プランn	式の返すリストの各要素を順に変数の値としてプラン1からプランnを繰り返す
optional	プラン1…プランn	プラン1からプランnのいずれも実行不能な場合スキップできる
select	プラン1…プランn	プラン1からプランnのうち実行可能なものを1つ実行する

表2 P-DP の例
Table 2 Example P-DPs.

P-DP 名	機　能	対応する発話例
%%ask-wh-q	フレームの不明なスロット値を尋ねる	wh-疑問文
%%inform-filler	スロットのフィラーの値を知らせる	ダ文，分裂文
%%inform-recog	肯定する	「はい」，「ええ」

%%で始まる DP は，プリミティブ DP (primitive DP, P-DP) と呼ばれ，実際の発話と直接対応する．**表2**に P-DP の例を示す．対話プランニングの際に P-DP を実行すると，対応する発話のフレームが文生成部に送られ，実際のシステムの発話が生成される．

また，ユーザの発話を理解する際には，ユーザの発話に対応する P-DP がユーザの発話の目的として記憶され，対話の目的に合っているかどうかがチェックされる．例えば，「今何時ですか」という発話からは，

(%%ask-wh-q ユーザシステム (time-of now))

が，発話の目的として抽出される．ユーザの発話を受けとるための対話プラン %recog は，

(%recog システムユーザ P-DP…P-DP)

という形をしている．P-DP…P-DP は，予測されるユーザの発話の目的を表す．ユーザがこの中にない P-DP を目的とした発話を行った場合，発話の目的に関する矛盾が検出される．

対話への影響の検出

ユーザの発話からは，発話の目的の他に，ユーザの信念に関するさまざまな情報が推論される．例えば，「花子はいつ太郎と会ったの？」という発話からは，ユーザが花子と太郎が会ったと信じていることがわかる．これは，wh 疑問文を行う前提条件として，その文に含まれる述語を信じていなければならないことから推論される．すなわち，この発話からユーザが

(%%ask-wh-q ユーザシステム (会った　花子　太

郎) attime)
を実行したと判断する．%%ask-wh-q の前提条件とマッチングすることにより，
(know ユーザ (会った　花子　太郎))
を得る．

本研究ではこのような，システムの持っている，ユーザの信念などのユーザモデルに更新の必要性を生じる場合を扱うことを目標としているので，このように発話から得られたユーザの信念のモデルは一種の仮説として扱い，矛盾を検出できるようにしなければならない．

このため，本研究では，発話から得られるものも含め，ユーザモデルを Assumption-based Truth Maintenance System (ATMS) を用いて管理する．したがって，ユーザモデルは反証のない限り真であると仮定され，矛盾するノードが追加されたときに ATMS はこれを検出する．ATMS を用いることで，対話プランナ自体はユーザモデルを管理する必要がなくなり，ATMS が矛盾を報告してきたときに，それが対話に与える影響を判断するのみでよくなる．

なお，ユーザの発話の目的に対する影響の検出については，前項で述べた．

対話の再プランニング

ユーザモデルの更新が対話に影響するかどうかを判断するため，対話プランナは，現在実行中のものも含め，過去の対話プランの実行の履歴を管理している．具体的には対話プランの呼び出し状況と各対話プランに関係する変数の束縛情報，前提条件や結果が記憶されている．

対話プランナは，ユーザモデルの更新が ATMS より報告されると，否定された述語と同じ述語が，記憶した履歴内の対話プランの前提条件および結果として存在しないか調べる．その結果，どのように対話を続けるかは，**表 3** に示す四つのメタ対話プランに従って決定される．以下，それらについて説明する．

表 3 メタ対話プラン
Table 3 Meta dialogue plans.

No.	否定された命題	アクション
1	以前の対話の前提条件	その対話プランを呼び出した対話プランを再実行
2	以前の対話の結果	その対話プランを再実行
3	ユーザの発話の目的	新しい目的に対応する対話プランを実行
4	履歴中には存在しない	正しい命題を知らせる

(メタ対話プラン 1)

否定された述語が履歴中のある対話プランの前提条件と一致するときは，その対話プランを呼び出した対話プランを再実行する．

その対話プランそのものではなく，一つ親の対話プランをやり直す理由は，子の対話プランの呼び出し自体が誤った前提に基づいて行われているので，正しい前提のもとで呼び出されるべき対話プランを選択するためである．

(メタ対話プラン 2)

否定された述語が履歴中のある対話プランの結果と一致するときは，その対話プランを再実行する．

(メタ対話プラン 3)

ユーザの発話の目的に影響を生じたときは，新しい目的に対応する対話プランを実行する．

対話プランの決定は，現在のシステムでは，%%ask-wh-q にはその答を知らせる対話プランといったように，1 対 1 の発話対を用いて行う．

(メタ対話プラン 4)

否定された述語が履歴中に存在しない場合，その正しい内容を知らせる．

例えば，ユーザが関西風のすき焼きに割り下を使うと思っていることがわかった場合には，「割り下は関西風のすき焼きには使いません」と正しい内容を伝える．

対話プランナは以上のメタ対話プランの実行に先だってその時点の対話プランの実行状況を退避させておき，メタ対話プランの実行後，元の対話に戻る．

なお，現在のところ本システムでは，更新の必要性が生じた場合，新しい述語が正しく，古い述語が否定されたものとして扱っている．

4. 例

システムの動作例として次の (対話 4) を考える．

(対話 4)

ユ ー ザ：すき焼きの作り方を知りたいのですが．(1)
システム：関東風と関西風のどちらがよろしいですか？(2)
ユ ー ザ：関西風がいいです．(3)
システム：わかりました．(4)
まず鍋を熱します．(5)
ユ ー ザ：はい．(6)
システム：次に肉と野菜を鍋にいれます．(7)

ユ ー ザ：割り下はいつ入れるんですか？(8)
システム：割り下は，
関西風のすき焼きには入れません．(9)
関東風と関西風の
どちらがよろしいですか？(10)
ユ ー ザ：関西風でいいです．(11)
システム：わかりました．(12)
では，醤油と砂糖を加えて煮ます．(13)
ユ ー ザ：はい．(14)

システムは，当初，ユーザが関東風と関西風の違いについて知っていると仮定している．すなわちノード1 (know ユーザ difference (kanto-sukiyaki, kansai-sukiyaki)) が仮定されている．したがって，関東風と関西風の違いの一つである，ノード2 (know ユーザ not (use (warisita, cook-kansai-sukiyaki))) がその前提として仮定されている．

ところが，発話(8)は，前提条件として，ノード3 (know ユーザ (use (warisita, cook-kansai-sukiyaki))) をとる．まずノード2と3の矛盾が検出される．この更新は以前の対話には特に影響しないため，メタ対話プラン4が実行され，発話(9)が行われる．

しかし，同時にノード2がnogoodとなったことから，ノード1も矛盾する．このノードは，発話(2)，(3)，(4)からなる対話プラン (%clarify-class) の前提条件となっているため，メタ対話プラン1により，この対話プランを呼び出した親の対話プラン (%explain-procedure) が再実行される．

この際，ユーザが再び関西風を希望したため，すでに告げてある手順((5)，(7))には言及していない．これは対話プランナの文脈依存性の一例である．

5. おわりに

本研究では，ユーザモデルの更新が対話に与える影響について考察した．ユーザモデルの更新によって以前の対話が影響を受ける場合があることを指摘し，その場合に対話を再プランニングし，更新がもたらす影響を解消して，対話を続行する手法を提案し，対話システムにインプリメントした．

今後の課題としては，以下のものがあげられる．

1. ユーザモデルに更新の必要性が生じた場合の正しいノードの決定法が単純である．したがって必ずしも正しい方を選べないこともある．

2. ユーザの対話の目的に影響する更新の場合，場合によっては以前の対話は捨ててしまってよい．これを判断するには，対話の目的の重要度をユーザの意図と合わせて決定する必要がある．

参考文献

1) 徳永，乾：1980年代の自然言語生成—2—，人工知能学会誌，Vol. 6, No. 4, pp. 510-519 (1991).
2) Kaplan, S. J.: Cooperative Responses from a Portable Natural Language Query System, *Artif. Intell.*, Vol. 19, No. 2, pp. 165-187 (1982).
3) McCoy, K. F.: Highlighting a User Model to Respond to Misconceptions, Kobsa, A. and Wahlster, W. (Eds.), *User Models in Dialog Systems*, pp. 233-254, Springer-Verlag, Berlin (1989).
4) Chin, D. N.: KNOME: Modeling What the User Knows in UC, Kobsa, A. and Wahlster, W. (Eds.), *User Models in Dialog Systems*, pp. 74-107, Springer-Verlag, Berlin (1989).
5) Quilich, A.: Detecting and Responding to Plan-Oriented Misconceptions, Kobsa, A. and Wahlster, W. (Eds.), *User Models in Dialog Systems*, pp. 108-132, Springer-Verlag, Berlin (1989).
6) de Kleer, J.: An Assumption-based TMS, *Atrif. Intell.*, Vol. 28, No. 1, pp. 127-162 (1986).
7) Sacerdoti, E.: Planning in a Hierarchy of Abstraction Spaces, *Artif. Intell.*, Vol. 5, No. 2, pp. 115-135 (1974).
8) 佐川，杉原，杉江：柔軟な対話制御機構を持ったコンサルテーションシステム，情報処理学会論文誌，Vol. 29, No. 4, pp. 350-358 (1988).
9) Alterman, R.: Adaptive Planning, *Cognitive Science*, Vol. 12, No. 3, pp. 393-421 (1988).
10) Litman, D. L. and Allen, J. F.: A Plan Recognition Model for Subdialogues in Conversations, *Cognitive Science*, Vol. 11, No. 2, pp. 163-200 (1987).
11) Cawsey, A.: Generating Interactive Explanations, *Proc. AAAI-91*, pp. 86-91 (1991).
12) Sagawa, Y., Ohnishi, N. and Sugie, N.: Techniques to Recover User's Utterance Failure in a Plan-Driven Dialogue System, *Proc. Natural Language Processing Pacific Rim Symposium*, pp. 131-138 (1991).
13) Hovy, E. H.: Approaches to the Planning of Coherent Text, Paris, C. L., Swartout, W. R. and Mann, W. C. (Eds.), *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pp. 81-102, Kluwer Academic Publishers, Norwell, MA (1991).
14) McKeown, K. R.: Discourse Strategies for Generating Natural-Language Text, *Artif.*

Intell., Vol. 27, No. 1, pp. 1-41 (1985).
15) Appelt, D. E.: Planning Natural-Language Referring Expressions, McDonald, D. D. and Bolc, L. (Eds.), *Natural Language Generation Systems*, pp. 69-97, Springer-Verlag, New York (1988).
16) Reithinger, N.: POPEL—A Parallel and Incremental Natural Language Generation System, Paris, C. L., Swartout, W. R. and Mann, W. C. (Eds.), *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pp. 179-199, Kluwer Academic Publishers, Norwell, MA (1991).

佐川 雄二（正会員）

昭和 60 年名古屋大学工学電電子卒業．昭和 62 年同大学院情報工学専攻修士課程修了．同年(株)日立製作所入社．平成元年名古屋大学大学院情報工学専攻博士課程入学．平成 4 年単位取得退学．同年同大学情報工学科助手，現在に至る．自然言語処理の研究に従事．工学博士．計量国語学会会員．

大西 昇（正会員）

昭和 48 年名古屋大学工学部電気卒業．昭和 50 年同大学院電気工学専攻修士課程修了．同年労働福祉事業団労災リハビリテーション工学センター研究員．昭和 59 年主席研究員．昭和 61 年名古屋大学工学部電気工学第二学科講師．平成元年同助教授．平成 5 年同情報工学科助教授，現在に至る．生体工学，福祉工学，人工知能，ロボティクスなどの研究・教育に従事．工学博士．電子情報通信学会，計測自動制御学会，ロボット学会，バイオメカニズム学会，IEEE 等各会員．

杉江 昇（正会員）

昭和 32 年名古屋大学工学部電気卒業．同年通商産業省電子技術総合研究所入所．昭和37～39年カナダ・マギル大学客員研究員．昭和 45 年バイオニクス研究室長．昭和 53 年視覚情報研究室長．昭和 54 年名古屋大学大学院工学研究科情報工学専攻教授．昭和 60 年同大学工学部電気工学第二学科教授．平成 2 年同大学工学部情報工学科教授および同大学大型計算機センター長併任，現在に至る．バイオニクス，医用工学，コンピュータビジョン，自然言語処理などの研究・教育に従事．工学博士．電気学会，電子情報通信学会，計測自動制御学会，ロボット学会，エム・イー学会，テレビジョン学会，バイオメカニズム学会，日本神経回路学会，IEEE 等各会員．