マルチエージェントインタラクションのための シナリオ協調メカニズム

田仲　理恵[†a)]　　中西　英之[††]　　石田　亨[†]

Scenario Coordination Mechanism for Multi-Agent Interaction

Rie TANAKA[†a)], Hideyuki NAKANISHI[††], and Toru ISHIDA[†]

Abstract. 人間と音声やテキストを介して対話を行うエージェントの研究は多数行われてきたが，複数の人間を相手に対話を並行して行うエージェントの研究はあまり行われてこなかった．本研究では，身振り手振りを用いて人間と対話を行うエージェントを対象に，複数の人間との対話を実現するため，複数のインタラクションシナリオをエージェントに与えることを考える．インタラクションシナリオは，エージェントと相手とのインタラクションプロトコルを記述したものである．複数のインタラクションシナリオを並行に実行する際，つまり複数の対話を並行して行う場合には競合が生じ，その調整方法は，相手に関係なく話しかけられるたびに対応する方法や，一方の相手を待たせておく方法などさまざまである．この調整方法を協調ポリシーと呼び，任意に指定した協調ポリシーを用いて複数のインタラクションシナリオを矛盾なく並行に実行する方式を提案する．本研究では，インタラクションシナリオの実行を通信によって制御する協調シナリオを導入し，協調を実現する．本研究の特徴は，1 対 1 に特化したインタラクションシナリオと協調ポリシーにより容易に協調が実現できる点である．

Keywords. エージェントシミュレーション，Human Agent Interaction，交渉と協調，対話エージェント

1. はじめに

人間と音声発話やテキストを介して対話を行うエージェントの研究は様々なところで行われている．その中で，仮想空間内に体を持ち，言語に加えて指差しなどの非言語的動作を行う ECAs(Embodied Conversational Agents) [1] の研究も進められている．例えば，空間内の物体を指差しながらネットワークについて 1 対 1 で教えてくれる擬人化学習エージェント Cosmo [2] や，仮想空間内でユーザの操作するアバタ群と対話を行いながら全体で一つのタスクを達成するエージェント Steve [3] などがある．

しかし，現実世界の対話は常に 1 対 1 ではないし，対話での役割や目的が常に明確であるとは限らない．人間は相手ごとに異なった対応の仕方（インタラクションプロトコル）をいくつも持っており，複数の相手と対話を行うときにはそれらをうまく使い分けて対応している．よって本研究では，身振り手振りを交えて相手と対話を行うエージェントを全体目標のためのプランやルールではなくプロトコルによって制御することを想定し，エージェントが複数の相手と対話を行うことができる方法を提案する．また，そのようなエージェントを，エージェントの対話や振る舞いを設計する設計者が容易に構築できる手法を提案する．

エージェントはインタラクションシナリオによって制御する．インタラクションシナリオはインタラクションプロトコルを実行可能な形式で記述したもので，相手の行動や周囲の環境変化など対話のきっかけとなるイベントと，イベントに対応する複数のアクションの対が記述されている．エージェントはシナリオに従って周囲の観測を行い，イベントが観測されると，歩くアクションや記述された言葉を話すアクションなど対応するアクションを実行する．

複数のプロトコルを一つのシナリオにまとめると複雑になるため，シナリオはプロトコルごとに用意して並行実行する．プロトコルにはそれぞれ複数のイベントが記述されており，対話相手がいかなるタイミング

† 京都大学大学院情報学研究科社会情報学専攻，〒 606-8501 京都市左京区吉田本町
†† 大阪大学大学院工学研究科知能・機能創成工学専攻，〒 565-0871 吹田市山田丘 2-1
a) E-mail: rtanaka@ai.soc.i.kyoto-u.ac.jp

で行動しても対応可能にするためには，各プロトコル内のイベントの組み合わせを全て列挙しなければいけないためである．エージェントの身体は一つしかないため，複数のシナリオを並行実行すると問題が生じることがある．本研究で提案する手法は，並行実行時の問題の解決方法を設計者が指定でき，それに従ってエージェントの振る舞いを制御する手法である．対話相手は二人以上の任意の人数を想定しており，シナリオライターは二人とのみ対話可能なエージェントを設計することも，それ以上の人数を相手にしても対話が可能なエージェントを設計することもできる．

2. シナリオ協調時の問題

一人の店員エージェントが客 1 への対応シナリオ 1 と，客 2 への対応シナリオ 2 を持っている状況を仮定する．シナリオ 1 には例えば，客 1 が『すみません』と話しかけてくるとエージェントが『どうしましたか？』と返答するといった内容が記述されている．エージェントと客 1 のシナリオ 1 を用いたインタラクション，エージェントと客 2 のシナリオ 2 を用いたインタラクションの例を図 1 に示す．二人の客への対応方法は図 2 に示すように何通りも考えられる．

シナリオ 1 と 2 を並行実行すると問題が生じる．例えば，客 1 と客 2 が全く同時に話しかけてくると，店員エージェントは同時に返答しようとする．店員エージェントには身体は一つしかないため，このような行動を行うことは物理的に不可能である．このような問題をアクションの競合と呼ぶ．複数の相手と対話を行うにはアクションの競合を解決しなければならないが，図 2 に示したように，競合の解決方法，つまり独立して処理されるシナリオの協調方法は複数考えられるため，協調のためのポリシーを設定する必要がある．協調ポリシーはシナリオの協調方法を指定するもので，図 2 の例では，並行対応協調ポリシーは，エージェントが相手に関係なく話しかけられるたびに応答することを示し，順次対応協調ポリシーは，同じ相手に対応し続け，別の相手は対応が終わるまで待たせることを示している．このいずれを用いるか，もしくは別のポリシーを用いるかは，エージェントをシナリオを与えて制御するシナリオライターの意図によって指定され，どのような協調ポリシーが設定されるかにより，エージェントが何人の相手と対話を行うことができるかが決まる．以降の章では，協調ポリシーを記述する方式と，記述された協調ポリシーに従い，かつ競合が起きないようにインタラクションを進める方法を述べる．

➢シナリオ 1 によって生成されるインタラクション

客 1：すみません．
店員：どうしましたか？
客 1：この店に傘はありますか？
店員：申し訳ありませんが，取り扱っておりません．
客 1：わかりました．ありがとうございます．

➢シナリオ 2 によって生成されるインタラクション

客 2：すみません．
店員：どうしましたか？
客 2：東京駅はどこですか？
店員：この道をまっすぐ行ってください．
客 2：わかりました．

図 1　一人の相手とのインタラクションの例

➢シナリオ 1, 2 と並行対応協調ポリシーによって生成されるインタラクション

客 1：すみません．
客 2：すみません．
店員：（客 1 に対し）どうしましたか？
店員：（客 2 に対し）どうしましたか？
客 1：この店に傘はありますか？
店員：（客 1 に対し）申し訳ありませんが，取り扱っておりません．
客 2：東京駅はどこですか？
店員：（客 2 に対し）この道をまっすぐ行ってください．
客 1：わかりました．ありがとうございます．
客 2：わかりました．

➢シナリオ 1, 2 と順次対応協調ポリシーによって生成されるインタラクション

客 1：すみません．
客 2：すみません．
店員：（客 1 に対し）どうしましたか？
店員：（客 2 に対し）少々お待ちください．
客 1：この店に傘はありますか？
店員：（客 1 に対し）申し訳ありませんが，取り扱っておりません．
客 1：わかりました．ありがとうございます．
店員：（客 2 に対し）お待たせしました．どうしましたか？
客 2：東京駅はどこですか？
店員：この道をまっすぐ行ってください．
客 2：わかりました．

図 2　複数の相手とのインタラクションの例

3. シナリオ協調アーキテクチャ

3.1 協調シナリオ

本研究では一人のエージェントが複数のシナリオを

持つことを前提とするが，関連研究として，プランを持った複数のエージェントの協調を行うマルチエージェントプラニングの研究がある [6]. [6] では各プラン内の，他のプランと並行実行すると問題が生じる臨界領域を求め，その前後に通信機能を埋め込み，同期プログラムを設けて協調を図っている．プラニングでは複数のアクションの実行順序が決まっているのに対し，シナリオを用いたインタラクションでは実行順序をあらかじめ定めておくことができない．よってこの手法をそのまま適用することは不可能であるが，通信により協調を行うという手法を利用し，協調を行うための協調シナリオを設けることで協調を実現する．

協調シナリオは，インタラクションシナリオとの通信によってアクションの同期制御を行う．インタラクションシナリオに記述する内容は，あるイベントの観測と，それに対応する複数のアクションの系列によって成り立っているため，制御はその系列ごとに行う．簡単のため，これ以降ではアクションの系列を単にアクションと記述する．協調シナリオはシナリオライターの記述した協調ポリシーに従っており，アクションの競合を検出し解決する機能も持っている．協調シナリオは後述する競合条件を逐一調べ，競合が起こる場合には協調ポリシーに実行すると書かれていても実行しないよう命令を出す．

制御は次の手順で行われる．まず，相手の行動や環境の変化など，インタラクションのきっかけとなる何らかの事象が観測されると，インタラクションシナリオは対応するアクションは実行せず，協調シナリオに報告メッセージを送って返信を待つ．報告メッセージには，実行可能な状態になったアクションなどの情報が含まれる．協調シナリオはメッセージを受け取ると，実行命令を待っているアクションが実行中のアクションと矛盾するかどうかを調べ，矛盾する場合には実行しないよう命令メッセージを送り返す．インタラクションシナリオは受け取った命令メッセージに従う．

3.2 アーキテクチャ

協調シナリオを用いた制御アーキテクチャを図 3 に示す．シナリオ生成器は，1 対 1 用のインタラクションシナリオと協調ポリシーをもとに協調シナリオを自動生成する．また，インタラクションシナリオを，協調シナリオとの通信による同期処理が可能となるように変更する．実行時には，シナリオ生成器の出力として得られる協調シナリオと，同期処理が可能な変更後のインタラクションシナリオをエージェントに与え，

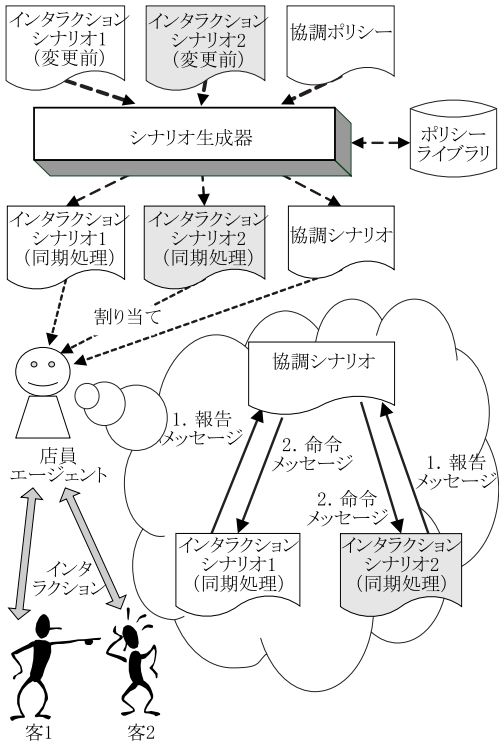

図 3　シナリオ協調アーキテクチャ

並行実行させる．競合解決など必要な機能は組み込まれており，自動的に協調が行われる．エージェント内では，協調シナリオは次にどのアクションが実行できるか決める役割，インタラクションシナリオは命令に従ってアクションを実行する役割となる．実際にアクションを実行するのはエージェントであるが，エージェントはインタラクションシナリオの通り実行しているだけなので，これ以降はインタラクションシナリオがアクションを実行するという書き方をする．シナリオライターは，1 対 1 の対応用に記述されたインタラクションシナリオと協調ポリシーをシナリオ生成器に与え，出力されるファイルをそのままエージェントに与えるだけで，容易に協調を実現できる．

ポリシーライブラリには，図 2 に示した並行対応協調ポリシーや順次対応協調ポリシーのように，シナリオの内容に依存せず作成することが可能な典型的な協調ポリシーをあらかじめ定義して蓄積しておく．シナリオライターは，シナリオの内容に依存した協調ポリシーを新たに作成してシナリオ生成器に与えることもできるし，ポリシーライブラリのポリシーを指定して

表 1　通信に用いるメッセージ

報告メッセージ（インタラクションシナリオ → 協調シナリオ）

メッセージ	意味
準備完了報告	アクションが実行可能な状態になったことを伝える
アクション終了報告	アクションの実行終了を伝える
待機報告	待機状態に入ったことを伝える
シナリオ終了報告	シナリオの実行終了を伝える

命令メッセージ（協調シナリオ → インタラクションシナリオ）

メッセージ	意味
実行命令	アクションを実行するよう命令する
取消命令	アクションを実行しないよう命令する
待機命令	アクションを実行せず待機処理をするよう命令する

使用することもできる．協調ポリシーについて詳しくは 4.1 節に述べる．

4.　シナリオ協調メカニズム

4.1　協調ポリシーの定義

協調ポリシーは状態遷移機械モデルで記述する．これは次のような理由による．協調シナリオは，通信によりアクションの実行制御を行うことでインタラクションシナリオの協調を行う．つまり，インタラクションシナリオから報告メッセージを受け取り，それに対する命令メッセージを送り返すプロセスを繰り返し行う．報告メッセージとは，たとえばアクションが実行可能な状態になったことを示すメッセージである．状態遷移機械モデルは状態と，状態間の遷移，アクションから構成されており，遷移の条件がインタラクションシナリオからの報告メッセージの受信，アクションが命令メッセージの送信に当たる．このモデルを用いることで，例えば全てのアクションに実行するよう命令する状態と，あるアクションにのみ実行するよう命令する状態を書き分けることができる．詳しくは次節で説明するが，協調ポリシーの状態や状態遷移はそのまま協調シナリオに反映される．

協調シナリオが受信する報告メッセージと送信する命令メッセージの典型的な例を表 1 に示す．待機報告と待機命令は，エージェントが相手を待たせておくときに行われる待機処理に使用されるメッセージである．待機処理は次の手順で行われる．まず協調シナリオはインタラクションシナリオに待機命令を送る．インタラクションシナリオは相手に待たせる意を伝えるため，『少々お待ちください』と言うなどの待機前アクションを行い，協調シナリオに待機報告を送って再開を待つ．協調シナリオが対話を再開させるために実行命令を送ると，インタラクションシナリオは『お待たせしました』と言うなどの待機後アクションを行って対話を再開する．待機前アクションと待機後アクションで具体的に何を話し行動するかはシナリオライターが自由に記述でき，インタラクションシナリオや協調ポリシーとともにシナリオ生成器に与えられる．

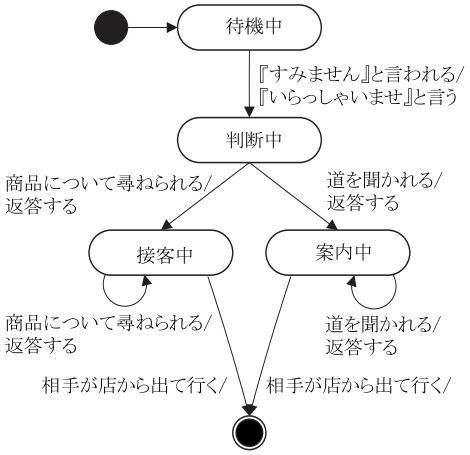

図 4　店で客に応対するインタラクションシナリオ

図 2 では典型的なポリシーの例を示したが，ここではシナリオの内容に依存するようなポリシーの例を示す．図 5 は，店にいるエージェントを想定したとき，買い物客に道を聞きに来た観光客などよりも優先して対応するポリシーである．エージェントに与えるインタラクションシナリオを示す状態遷移図を図 4 に示す．インタラクションシナリオは最初は待機中にあり，相手に声をかけられると判断中に遷移して対応を開始する．判断中において商品に関する質問を受け取ると，相手が買い物客であることが分かり接客状態に遷移する．道を聞かれた場合には相手が観光客であることが分かり案内中に遷移する．接客状態と案内中ではそれぞれ相手と対話を行い，相手が店から出て行くと対話が終了する．実行時には，店に入ってきた相手ごとにこのシナリオを一つずつ割り当てる．なお，簡単のため，実際には具体的な質問や返答内容を記述するところを単純化して表記している．

図 5 に示すポリシーは準備状態，優先状態，非優先状態の三状態からなり，準備状態は相手が何かアクションを起こすのを待っている状態，優先状態は買い物客に優先的に対応している状態，非優先状態はそれ以外

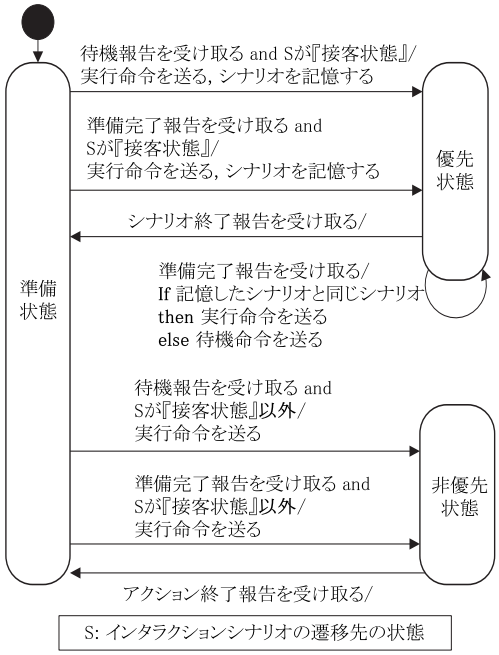

図 5　買い物客を優先する協調ポリシー

の場合に何らかのアクションを実行している状態である．このポリシーから作られる協調シナリオは，インタラクションシナリオから受け取った報告メッセージを元に，そのシナリオの遷移先が接客状態であるかどうかを判断することで，そのシナリオに対応する客が買い物客であるかどうかを判断する．協調シナリオの動きを，実際の対話を想定して説明する．まず，相手が声をかけてくると，図 4 のシナリオから準備完了報告が送られてくる．このときの遷移先は判断中であり接客中ではないので，協調シナリオは実行命令を送って非優先状態に遷移し，アクションの実行が終了し完了報告を受け取ると準備状態に戻る．

相手が質問をすると新たに準備完了報告が送られてくる．協調シナリオは遷移先を調べ，遷移先が買い物客を示す接客中であれば，実行命令を送って優先状態に遷移する．このとき，どのシナリオをこれから優先するのかを記憶しておく．接客中でなければ，同様に非優先状態に遷移してアクションを実行させる．優先状態において準備完了報告を受け取った場合，優先すると記憶したシナリオと同じシナリオであれば実行命令を送るが，別のシナリオからの準備完了報告であれば，それは相手が観光客であるか新たに来た客であることを示しているので，その客は待たせるよう待機報告を送る．買い物客が店から出て行って対応が終ると，協調シナリオはシナリオ終了報告を受け取って準備状態に戻る．優先状態においていずれかのシナリオに待機命令を送っていた場合は，待機前処理を終えたシナリオから待機報告が送られてきており，協調シナリオは準備状態に戻ったときにそれを受け取る．そして再び遷移先を調べて優先するかどうか判断し，実行命令を送って該当する状態へ遷移することで対話を再開する．このプロセスの繰り返しとなる．

図 2 に示した並行対応協調ポリシーと順次対応協調ポリシーも同様に定義できる．前者は準備完了報告を受け取るたびに実行命令を送るようにすればよい．後者は，図 5 と同様に考えて準備状態と応対中を用意し，準備状態を相手の行動を待つ状態，応対中を誰かに対応している状態になるように定義すればよい．

ポリシーにおいては，対話相手の管理をする必要はない．図 5 に示したポリシーでは，優先状態は誰かに優先的に対応している状態，非優先状態は誰も優先せず何らかのアクションを実行している状態を示すだけである．ある相手に対しては優先状態にあり，別の相手に対しては非優先状態にあるわけではない．このポリシーを用いると三人以上の相手とも対話が可能である．優先中に来る相手が増えると，話しかけてきた順に待つよう言って準備状態に戻るたびに一人ずつ対話を再開していく．ただし相手の人数が増えると対話が機械的になるため，自然に振舞うにはもっと複雑なポリシーの方が適していると言える．

4.2　協調シナリオの生成

協調シナリオは，協調ポリシーの状態はそのままに，条件部とアクションを実行可能な形式に変換することで作成できる．協調ポリシーには図 5 に示した例のように，競合の解決については記載しないため，アクションの実行に関係する部分，つまり実行命令を送る部分と待機命令を送る部分については，実行可能な形式に変換するだけではなく，図 6 に示す条件を調べ，条件に対応する処理を行うように変換する．

まず図 6 で用いられている『競合』とは何かを定義する．競合の判断は，各アクションの前提条件，事後条件，継続条件，リソースを調べることにより判断する．継続条件は，言葉を話すアクションのように一定時間継続するアクションが実行中に満たすべき条件を表す．リソースは手や足である．各条件とリソースは実行系やシナリオの記述に使用するアクションによって異なるので，シナリオライター，もしくはシナリオ

に使用するアクションを定義する者が記述し，どの条件の組を矛盾とみなすかの定義とともにシナリオ生成器に与えておく．競合とは，以下の 5 つの条件のうちいずれかが満たされることを示す．

1) 実行命令を待つアクションと実行中のアクションを並行実行した結果が，いかなる順で逐次実行した結果とも異なる（事後条件同士が矛盾する，または実行命令を待つアクションの前提条件と実行中のアクションの事後条件が矛盾する）
2) 実行命令を待つアクションと実行中のアクションの継続条件が矛盾する
3) 実行命令を待つアクションの前提条件が，実行中のアクションの継続条件と矛盾する
4) 一方の継続条件と他方の事後条件が矛盾する
5) 実行命令を待つアクションと実行中のアクションが同じリソースを使用する

実行前にシナリオを生成するシナリオ生成器は，全てのアクションについて競合の有無を総当りで調べ，協調シナリオの中に競合データとして記載する．調べるアクションの組には 3 種類あり，1 種類目はインタラクションシナリオで使用されているアクションの全ての組み合わせである．2 種類目は待機前アクションと使用されているアクションの全ての組み合わせであり，待機状態に入る前に行う行動とすでに実行中の行動との競合の検出に用いる．3 種類目は，待機後アクションに続けてアクションを実行するアクション系列とアクションの組み合わせであり，対話の再開時に行う行動とすでに実行中の行動との競合の検出に用いる．各組は，一方が実行中で他方が実行命令待ちとした場合とその逆の場合の両方について調べる．実行時にはこの競合データを参照して競合を判断する．

次に図 6 に示した条件について説明する．実行命令を送る状況は，準備完了報告に対してと待機報告に対しての二通りある．前者はこれからアクションを実行する状況であり，実行中のアクションと競合が起きなければ実行命令を，競合が起きれば取消命令を送る．後者は待機中のシナリオに再開するよう命令する状況で，実行中のアクションと再開後に実行するアクション系列が競合すれば，競合が起きなくなるまで待ってから実行命令を送る．取消命令を送らないのは，待機報告を送ったシナリオが再開を待ち続けており，実行を取り消すことができないためである．待機命令を送る場合には，待機前アクションと実行中のアクション

➢協調ポリシーの “実行命令を送る” 記述は，以下の条件を調べて対応するアクションを実行するように変更される．受信した報告メッセージを M，報告されたアクションを x，実行中のアクションの集合を A とおく

Case M が準備完了報告で，A と x が競合する:
　取消命令を送る
　元の状態に遷移する
Case M が待機報告で，待機後アクションの後に x を実行した系列と A が競合する:
　競合が起こらなくなるまで待つ
　実行命令を送る
Otherwise:
　実行命令を送る

a) 実行命令を送る場合に調べる条件

➢協調ポリシーの “待機命令を送る” 記述は，以下の条件を調べて対応するアクションを実行するように変更される．実行中のアクションの集合を A とおく

Case 待機前アクションと A が競合する:
　取消命令を送る
　元の状態に遷移する
Otherwise:
　待機命令を送る

b) 待機命令を送る場合に調べる条件

図 6　実行命令および待機命令を送る場合に調べる条件

が競合すれば取消命令を送る．

協調ポリシー中の実行命令を送る部分と待機命令を送る部分を以上のように変更することで，競合は自動的に防がれ，どのような協調ポリシーが与えられても最低限問題なく並行実行を行うことができる．

4.3　インタラクションシナリオの変更

シナリオ生成器は同期処理のため，協調シナリオに報告メッセージを送る機能と送られてきた命令メッセージに従う機能をインタラクションシナリオに付加する．インタラクションシナリオは，図 7 に示す例のように複数のルールからなる．図 7 では，エージェントははじめに *event-1* が起きたか調べ，*event-1* を観測した場合は後続する *actions-1* を実行する．観測しなかった場合は次に *event-2* を調べる，というプロセスを繰り返す．いずれのイベントも観測しなければ，再び最初に戻って *event-1* から観測を始める．

シナリオ生成器は図 8 に示すアルゴリズムに従って，インタラクションシナリオに同期処理を埋め込む．アルゴリズムによって変更されたシナリオが図 9 である．インタラクションシナリオは実行命令を受け取った場

➢“**If** *event-x* を観測する **then** *action-x* を実行する” をルールと呼ぶ

If *event-1* を観測する **then** *action-1* を実行する
If *event-2* を観測する **then** . . . . . .
If *event-n* を観測する **then** *action-n* を実行する

図 7　元のインタラクションシナリオ

S ← インタラクションシナリオ
R1, R2 ← 空
for each *rule* **in** S **do**
　event ← *rule* で観測されるイベント
　action ← *rule* で実行されるアクション
　以下のルールを R1 の末尾に追加:
　　“**If** *event* を観測する
　　then changed-action(*event*, *action*)”
　以下のルールを R2 の末尾に追加:
　　“**If** *event* を観測したと記憶している
　　then changed-action(*event*, *action*)”
end
S ← R1 の後に R2 を連結

changed-action(*event*, *action*) は，以下の *event* と *action* を実際の引数で置き換えたアクション系列を示す.
　準備完了報告を送る
　返答を待つ
　If 返答が実行命令 **then**
　　記憶したイベントを破棄する
　　action を実行する
　　アクション終了報告を送る
　else if 返答が待機命令 **then**
　　記憶したイベントを破棄する
　　相手に “少々お待ちください” と言う
　　待機報告を送る
　　実行命令が送られてくるのを待つ
　　相手に “お待たせしました” と言う
　　action を実行する
　　アクション終了報告を送る
　else if 返答が取消命令 **then**
　　event を観測したことを記憶する
　　アクション終了報告を送る

図 8　シナリオ生成器がインタラクションシナリオに同期処理を埋め込むアルゴリズム

合にはそのまま実行するだけであるが，取消命令に対しては実行せず，観測した事象を記憶する．エージェントはインタラクションが終了しない限りは観測を続けるので，再び条件文 “*event-1* を観測する” から評価を開始する．記憶した *event* は観測済みであるため次に進む．*event-1* から *event-n* までのいずれも観測されなければ，つまり相手の行動や環境に変化がなければ，記憶した *event* に対する準備完了報告を再び送る．新たな事象を観測した場合には，相手が新たな行動を

If *event-1* を観測する **then**
　協調シナリオに準備完了報告を送る
　返答を待つ
　If 返答が実行命令 **then**
　　記憶したイベントを破棄する
　　action-1 を実行する
　　協調シナリオにアクション終了報告を送る
　else if 返答が待機命令 **then**
　　記憶したイベントを破棄する
　　相手に “少々お待ちください” と言う
　　協調シナリオに待機報告を送る
　　実行命令が送られてくるのを待つ
　　相手に “お待たせしました” と言う
　　action-1 を実行する
　　協調シナリオにアクション終了報告を送る
　else if 返答が取消命令 **then**
　　event-1 を観測したことを記憶する
　　協調シナリオにアクション終了報告を送る
else if *event-2* を観測する **then** . . .（同様）
else if *event-n* を観測する **then** . . .（同様）
else if *event-1* を観測したと記憶している **then**
　event-1 を観測したときと同じ処理を行う
else if *event-2* を観測したと記憶している **then**
　. . .（同様）

図 9　同期処理が可能なインタラクションシナリオ

するか環境が変化したことを示しているので，古い状況を示す記憶した *event* は取り消して新たなアクションの準備完了報告を送る．以上のように変更することによって，状況が変化しない限りは実行できるまで依頼を送り続け，状況が変化したときには以前の記憶を破棄し新たな依頼を送ることができ，実行命令を待っている間の状況変化にも対応できる．

5. 実　　装

本研究で提案したシステムを，インタラクションプロトコルによってエージェントを制御する FreeWalk/*Q* [4] を用いて実現した．FreeWalk/*Q* では，シナリオは *Q* 言語 [5] を用いて状態遷移機械モデルで記述され，*Q* シナリオと呼ばれる．*Q* シナリオには，インタラクションのきっかけとなるキューと，キューに対応する振る舞いを示すアクションを記述し，エージェントは *Q* シナリオに従って行動する．よって，表 1 に示したような報告メッセージを受け取るキュー，命令メッセージを送信するアクションを定義し，協調ポリシーを *Q* シナリオと同様の形式で記述できるようにした．シナリオ生成器は，*Q* シナリオと協調ポリシーを受け取り，*Q* シナリオと同様の形式で変更後の *Q* シナリオと協調シナリオを生成するように実装した．生成

図 10　二人の客と対話するエージェント

器には Q シナリオで使用されているアクションの各条件とリソース，矛盾の条件を与え，また図 2 に示した並行対応協調ポリシーと順次対応協調ポリシーを図 5 のように定義してポリシーライブラリに用意した．

作成したシナリオ生成器に，1 対 1 のインタラクションのための Q シナリオ，Q シナリオと同様の形式の協調ポリシー，待機前アクションと待機後アクションの指定を与え，得られたシナリオを FreeWalk/Q に読み込ませて動作を確認した．図 10 は図 5 に示した買い物客優先ポリシーを用いたときのスクリーンショットである．中央にいる女性が店員エージェントであり，協調シナリオと変更済みの Q シナリオを持っている．図 10 は買い物客である男性客と商品について話しているときに声をかけてきた女性客に『少々お待ちください』と言っている場面で，男性客との対話が終わると，店員エージェントは女性客の方を向き,『お待たせしました』と言って対話を再開した．

6. おわりに

本研究は，複数のインタラクションプロトコルを持ち，複数人と対話を行うエージェントを構築することを目標とした．各プロトコルを一つずつインタラクションシナリオに記述し，複数のシナリオをエージェントに割り当てた上で，問題なく並行実行ができる方法を提案した．複数のシナリオの並行実行時にはアクションの競合が起こり，競合の解決方法を指定した協調ポリシーが必要となる．本研究では，協調のためにインタラクションシナリオと通信を行いアクションの実行を制御する協調シナリオを導入し，協調シナリオを用いたアーキテクチャと協調メカニズムを提案した．

協調シナリオは，協調ポリシーに競合を検出し解決する機能を付加することで作成する．本研究で提案するアーキテクチャは，協調ポリシーから協調シナリオを生成し，インタラクションシナリオを協調シナリオとの同期処理が可能となるように変更する．生成される協調シナリオとインタラクションシナリオはエージェントに与えられ，協調シナリオは通信によって実行時にインタラクションシナリオを制御する．我々はこのアーキテクチャを実装し，設定した協調ポリシーに従ってエージェントが動作することを確認した．

本研究で提案する協調メカニズムは，インタラクションシナリオが協調シナリオにアクションの実行を要求するメッセージを送信し，協調シナリオが競合を調べて実行してよいかを返答し，インタラクションシナリオが返答結果に従うというものである．協調シナリオが競合を調べる際には，あらかじめ定義した競合条件と照らし合わせることで競合を判断する．

本研究で提案する方法では，シナリオは一人の相手に対応できるよう記述すればよく，協調ポリシーは，一人ずつ対応するなどの典型的な協調方法であれば，シナリオの内容の詳細を知らなくても記述できる．別々の人が記述したシナリオを用いて協調を行うことができる点が本研究の特徴である．

文　献

[1] Cassell, J., Sullivan, J., Prevost, S. and Churchill, E.: *Embodied Conversational Agents*, MIT Press (2000).

[2] Lester, J. C., Towns, S. G., Callaway, C. B., Voerman, J. L. and FitzGerald, P. J.: Deictic and Emotive Communication in Animated Pedagogical Agents, In Cassell, J., Sullivan, J., Prevost, S. and Churchill, E.: *Embodied Conversational Agents*, MIT Press, pp. 123–154 (2000).

[3] Rickel, J. and Johnson, W. L..: Virtual Humans for Team Training in Virtual Reality, In *Proceedings of the Ninth International Conference on AI in Education*, pp. 578–585, IOS Press (1999b).

[4] Nakanishi, H. and Ishida, T.: FreeWalk/Q: Social Interaction Platform in Virtual Space, *ACM Symposium on Virtual Reality Software and Technology (VRST2004)*, pp. 97–104 (2004).

[5] Ishida, T.: Q: A Scenario Description Language for Interactive Agents, *IEEE Computer*, Vol. 35, No. 11, pp. 54–59 (2002).

[6] Georgeff, M.: Communication and interaction in multiagent planning, In *Proceedings of the 3th National Conference on Artificial Intelligence*, pp. 125-129 (1983).