2B1-3

マルチエージェントインタラクションにおける並行シナリオの協調

Coordination of Concurrent Scenarios in Multi-Agent Interaction

田仲　理恵　　中西　英之　　石田　亨
Rie Tanaka　Hideyuki Nakanishi　Toru Ishida

京都大学情報学研究科社会情報学専攻
Department of Social Informatics, Kyoto University

Though research on agents that interact with humans via voice or text was intensively conducted, agents that can interact with more than two people in parallel have not been studied well. To enable an agent to interact with plural persons, we propose to assign multiple scenarios to one agent; each scenario descrives an interaction protocol between the agent and each person. Obviously, coordination among multiple scenarios are required to avoid conflicts of actions. For example, one agent cannot execute walking and turning actions in parallel. More importantly, however, a coordination policy, the way to specify how to manage conflicts among multiple actions, is needed to be introduced. We propose the way to avoid conflicts of actions and coordinate scenarios accrding to the coordination policy.

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)[Cassell 00] 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[Lester 00] 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[Rickel 99b] 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，エージェントに全体目標のためのプランやルールではなくプロトコルを複数持たせ，それらを使い分けることを目的とする．

本研究でエージェントを制御するプロトコルは，相手の行動や周囲の状況を観測し，それに対するアクションを行うプロセスの繰り返しを記述するものと仮定する．プロトコルを実行可能な書式で記述したものをシナリオと呼ぶ．

エージェントに複数のプロトコルを持たせる場合，一つのシナリオにまとめると複雑になる．対話相手がいかなるタイミングで行動しても対応可能にするためには，考えられうる行動の組み合わせをすべて列挙しなければいけないためである．よって，プロトコルごとに用意したシナリオを複数エージェントに与えて並行実行させ，シナリオ間で矛盾が生じないよう協調を行う方法を提案する．

2. シナリオ並行実行時の問題点

図 1 に示したインタラクションを行うプロトコル A, B があったとする．プロトコルで対応可能な二人の客が同時に店員エージェントのところに来た場合に，起こりうるインタラクションとして図 2 に示したパターンが考えられる．同じタイミングで客が来たとしても，対応の仕方は一通りとは限らない．

連絡先: 〒 606-8501 京都府京都市左京区吉田本町 京都大学大学院情報学研究科 石田研究室，075-753-5396

〈 プロトコル A 〉
客 A：すみません.
店員：いらっしゃいませ.
客 A：この店に傘はありますか.

〈 プロトコル B 〉
客 B：すみません.
店員：いらっしゃいませ.
客 B：東京駅へはどう行けばいいですか.

図 1: プロトコルの例

図 2 に示したようなインタラクションを行うためには，以下の二つの問題を解決しなければならない．

並行実行時を行うとアクションの競合が起こる．図 1 のプロトコル A, B を記述したシナリオ A, B を例にすると，エージェントは二人の客に同時に話しかけられると同時に返答するなど，物理的に不可能な行動をしようとする．

さらに，図 2 において同じ状況への対応方法が複数あるように，アクションの競合をどのように処理するかを指定する協調ポリシーも必要となる．つまり，協調ポリシーの書式を定め，エージェントを制御する人間が書いた協調ポリシーに基づいて実行を制御する方法を考案する必要がある．

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 [Georgeff 83]．[Georgeff 83] ではプランに通信機能を埋め込み，同期プログラムを設けることによって協調を図っている．この手法を利用し，協調を行う協調シナリオを設ける．

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．プロトコルに記述する内容は，ある事象の観測と，それに対する複数のアクション系列によって成り立っているため，制御はアクション系列ごとに行う．

協調シナリオは協調ポリシーに従っており，アクション系列

〈 パターン 1 〉
客 A：すみません.
店員：いらっしゃいませ.
客 B：すみません.
店員：いらっしゃいませ.
客 A：この店に傘はありますか.
客 B：東京駅へはどう行けばいいですか.

〈 パターン 2 〉
客 A：すみません.
店員：いらっしゃいませ.
客 B：すみません.
店員：少々お待ちください.
客 A：この店に傘はありますか.
店員：(返答)
（客 A とのインタラクション終了）
店員：お待たせしました.
客 B：東京駅へはどう行けばいいですか.

図 2: 複数人への対応例

の競合検出機能も持っている．競合の解決は最低限行わなければならない処理であり，協調シナリオは定義された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機能を備えている．

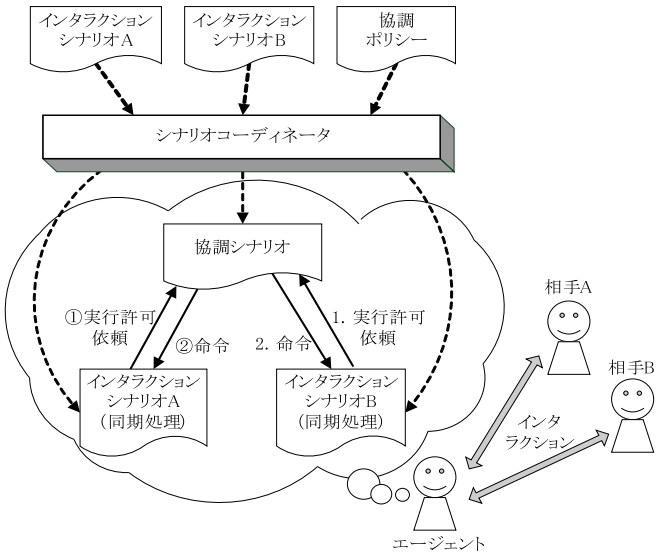

図 3: シナリオの協調の概念図

協調シナリオを用いた制御の概念図を図 3 に示す．図中のシナリオコーディネータは，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元に，協調シナリオと，同期処理が可能なインタラクションシナリオを自動生成する．

生成されたインタラクションシナリオと協調シナリオをエージェントに与えて並行実行させると，何らかの事象が観測されるたびに以下の手順で制御が行われる．

まず，インタラクションシナリオは協調シナリオに実行許可依頼メッセージを送り，命令が送られてくるまでアクション系列を実行せずに待つ．協調シナリオは依頼されたアクション系列が実行可能かを調べ，結果を命令として返答する．変更後シナリオは受け取った命令に従う．

3.2 シナリオコーディネータ

シナリオコーディネータの役割は，協調シナリオを生成し，インタラクションシナリオを同期処理が可能なように変更することである．

インタラクションシナリオの変更は，1 対 1 用に記述された元のシナリオに通信などの機能を付加することで行う．

協調シナリオは，協調ポリシーとインタラクションシナリオの両方から生成する．まずシナリオコーディネータは，インタラクションシナリオのアクション系列のすべての組み合わせについて，あらかじめ定義しておいた競合条件と照合して競合が起こるかどうかを検出し，競合データとして保存する．そして協調ポリシーを元に，保存したデータを参照するように協調シナリオを生成する．

競合条件は，アクション系列の前提条件・事後条件・継続条件・リソースを用いて定義する．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している．リソースは手や足である．

競合条件は次の 5 つである．一つ目の条件はアクション系列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ときには並行実行できないことを示し，二つ目の条件は継続条件が矛盾するアクション系列は並行実行できないことを示す．三つ目の条件は実行中のアクション系列の継続条件と矛盾する前提条件を持つアクション系列は続けて実行開始できないことを示し，四つ目の条件は継続条件と事後条件の矛盾するアクション系列は並行実行できないことを示す．最後に五つ目の条件は，同じリソースを使用するアクション系列は並行実行できないことを示す．競合検出の際には以上の条件をすべて調べ，いずれか一つでも満たされた場合に競合と判断する．

4. シナリオ協調メカニズム

4.1 協調シナリオの生成

協調ポリシーはプロトコルと同じ形式で記述する．これは次のような理由による．

シナリオの協調は，協調シナリオが通信によりアクション系列の実行を制御することにより行う．つまり，シナリオからアクション系列の実行依頼などのメッセージを受信し，対応する命令を送信するプロセスの繰り返しとなる．プロトコルは外界を観測し，観測結果に対応する行動を記述するものであり，協調方法の記述にも適している．

表 1: 通信に用いるメッセージ

依頼メッセージ	意味
実行許可依頼	アクション系列の実行を依頼する
アクション終了報告	アクション系列の終了を伝える
待機報告	待機前処理の終了を伝える
シナリオ終了報告	シナリオの終了を伝える

返答メッセージ	意味
実行命令	アクション系列の実行を命令する
取消命令	アクション系列の実行を取り消す
待機命令	待機処理をするよう伝える

協調シナリオが受信するメッセージ（依頼メッセージ）と送信する命令（返答メッセージ）を表 1 に示す．待機処理は，あ

る相手とのインタラクション中は他の相手を待たせておくなどのポリシーを記述する際，より自然なインタラクションのために用意した処理である．シナリオは待機命令を受け取ると相手に『お待ちください』と言うなどの待機前処理を行う．再開時には『お待たせしました』と言うなどの待機後処理を行ってから再開をする．待機前処理・後処理で具体的に何を話し行動するかは自由に記述できる．

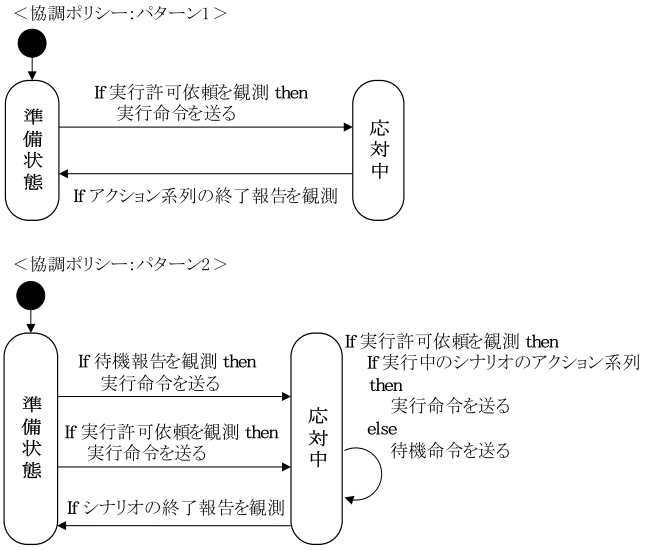

図 4: 協調ポリシーの例

表 1 のメッセージを用い，図 2 の対話パターンに対応する協調シナリオを状態遷移モデルで表したものが図 4 である．

パターン 1 では，準備状態で実行許可依頼を受け取ると実行命令を出し，応対中に遷移してアクションの終了を待つ．終了報告を受け取ると準備状態に戻る．これは相手に関係なく順次実行命令を出すポリシーである．

パターン 2 では，準備状態で実行許可依頼を受け取ると実行命令を出し応対中に遷移する．応対中に実行許可依頼を受け取ると，実行中のシナリオであれば実行命令を出し，そうでなければ待機命令を出す．実行中のシナリオが終了すると準備状態に遷移する．待機報告に対し実行命令を出す枝は，応対中に待機命令を出し待機させていたシナリオの実行を再開することを意味している．つまり，最初に実行命令を出した相手に対応し続け，対応中に来た客には待機処理を行い，対応が終わると待たせていた客への対応を再開するポリシーである．

協調シナリオは，協調ポリシーで使用されている観測対象とアクションを実行可能な形式に変換することで生成する．ただし協調ポリシーには競合に関する記述は含まれないので，実行命令を出す部分に機能を追加する．

実行命令を出す状況は二通りある．一つは実行許可依頼に対して出す場合で，実行するアクション系列が実行中のアクション系列と競合するかを定義した条件と照合して調べ，いずれとも競合しなければ実行命令を出し，いずれかと競合すれば取消命令を出す．もう一つは再開を待っているシナリオに出す場合で，待機しているシナリオの実行を取り消すことはできないため，再開するアクション系列が実行中のアクション系列と競合しなくなるまで待ってから実行命令を送る．

4.2 インタラクションシナリオの同期処理

インタラクションシナリオには観測と行動からなるルールが複数あり，図 5 のアルゴリズムに従って各ルールを変更することにより同期処理が可能となる．

シナリオはルールの集合である．各ルールは以下の形式とする．
　If *event* を観測する **then** *actions* を実行する
このうち，“ **If** *event* を観測する **then** ”をルールの条件部，“ *actions* を実行する ”を動作部と呼ぶ．

S ← ファイルを読み込み，シナリオを示す文字列を取得
D ← 空リスト
for each *rule* **in** S **do**
　L, C ← 空リスト
　C の末尾に以下の項目を示す文字列を順に追加
　　“ 協調シナリオに実行許可依頼を送る ”コマンド
　　“ 返答を待つ ”コマンド
　　条件部“ **If** 返答が実行命令 **then** ”
　　rule の動作部
　　“ 協調シナリオにアクション終了報告を送る ”コマンド
　　条件部“ **If** 返答が待機命令 **then** ”
　　“ 待機前処理をする ”コマンド
　　“ 協調シナリオに待機報告を送る ”コマンド
　　“ 実行命令が送られてくるのを待つ ”コマンド
　　“ 待機後処理をする ”コマンド
　　rule の動作部
　　“ 協調シナリオにアクション終了報告を送る ”コマンド
　　条件部“ **If** 返答が取消命令 **then** ”
　　“ *event* が観測されたことを記憶する ”コマンド
　　“ 協調シナリオにアクション終了報告を送る ”コマンド
　　“ 観測を続ける ”コマンド
　L の末尾に *rule* の条件部を示す文字列を追加
　L の末尾にリスト C を追加
　L の末尾に条件部“ **else if** *event* を観測したと記憶されている **then** ”を示す文字列を追加
　L の末尾にリスト C のコピーを追加
　L を D の末尾に追加
D の要素を順に取り出してファイルに書き込む

図 5: インタラクションシナリオの変更アルゴリズム

同期処理が可能なルールが図 6 である．協調シナリオにメッセージを送る機能と，受信した命令に従う機能が付加されている．実行命令に対しては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して観測を続ける．新たな事象を観測した場合には新たなアクション系列の実行許可依頼を送り，観測しなかった場合には，記憶された事象に対する依頼を再度送る．

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る．実行許可を待っている間の状況変化にも対応できるということである．

5. 実装

本研究で提案した手法を，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*[Nakanishi 04] を用いて実現した．FreeWalk/*Q* では，シナリオは *Q*[Ishida 02] 言語を用いて状態遷移機械モデルで記述される．*Q* シナリオはキュー（インタラクションのきっかけ）とアクションにより記述する．よって協調ポリシーも同じく状態遷移機械モデルを採

If *event* を観測する **then**
　協調シナリオに実行許可依頼を送る
　返答を待つ
　If 返答が実行命令 **then**
　　actions 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**
　　待機前処理をする
　　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
　　待機後処理をする
　　actions 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 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
　　観測を続ける
else if *event* を観測したと記憶されている **then**
　event を観測したときと同じ処理を行う

図 6: インタラクションシナリオの同期処理

用し，使用するキューとアクションを定義した．そして，協調ポリシーと Q シナリオを与えると協調シナリオと変更された Q シナリオを出力するシナリオコーディネータを実装した．

図 4 のパターン 2 に対応する協調ポリシーを用い，生成された協調シナリオと同期可能な Q シナリオをエージェントに与えて実行させた（図 7）．中央の女性が店員エージェントで，右の男性客との対話中に来た左の女性客に『お待ちください』と言っている．

図 7: FreeWalk/Q での実行時のスクリーンショット

6. おわりに

本研究では複数人での対話を目標とし，エージェントに相手ごとの対応プロトコルを記述したシナリオを複数持たせて並行実行する方法を提案した．並行実行時にはアクションの競合が起こり，また，シナリオの協調方法の指定が必要になる．これらの問題を，シナリオとの通信によってアクションの実行を制御する協調シナリオを導入して解決した．

シナリオの協調は，シナリオがアクションを実行する前に協調シナリオへメッセージを送り，協調シナリオが実行可能かを判断して返信し，返信されたメッセージにシナリオが従うプロセスの繰り返しにより行う．アクションの競合は，協調シナリオが判断を行う部分で，実行中のアクションと競合するかを調べ，競合しなければ実行するようなメッセージを，競合するときは実行しないようなメッセージを送ることで解決する．

シナリオの協調方法の指定については，受信したメッセージに対しどのように返信するかをプロトコルと同様の形式を用いて記述し，それをもとに協調シナリオを作成することで解決する．協調方法を記述したものを協調ポリシーと呼ぶ．

協調シナリオは協調ポリシーにアクションの競合検出機能を追加することで生成する．競合検出は，これまでの関連研究をもとに，インタラクションで使用するアクションに必要な条件を補った競合条件を使用して行う．

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．

謝辞　本研究は科学研究費補助金萌芽研究（課題番号 17650041）ならびに財団法人国際コミュニケーション基金の助成を受けています．

参考文献

[Cassell 00] Cassell, J., Sullivan, J., Prevost, S. and Churchill, E.: *Embodied Conversational Agents*, MIT Press (2000).

[Lester 00] 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).

[Rickel 99b] 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).

[Georgeff 83] Georgeff, M.: Communication and interaction in multiagent planning, In *Proceedings of the 3th National Conference on Artificial Intelligence*, pp. 125-129 (1983).

[Nakanishi 04] 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).

[Ishida 02] Ishida, T.: Q: A Scenario Description Language for Interactive Agents, *IEEE Computer*, Vol. 35, No. 11, pp. 54–59 (2002).