特別研究報告書

メタレベル制御による
インタラクションシナリオの協調

指導教員　　石田 亨 教授

京都大学工学部情報学科

澤田 祥一

平成19年2月9日

メタレベル制御によるインタラクションシナリオの協調

澤田 祥一

内容梗概

交通流や経済現象などの社会システムのシミュレーションにおいて，現実世界で生じる現象を再現し，複雑な社会現象の原理を解析するために，マルチエージェントシミュレーションを用いる方法が注目されている．現実世界において人間は，異なる状況に応じて適切なインタラクションを行うための様々なプロトコルを持っている．そして，周囲の環境や相手に合わせてそれらのプロトコルを使い分けることによって適切にコミュニケーションを行うことができる．このような社会的なインタラクションを，マルチエージェントシミュレーションによって再現することが，本研究の目的である．

社会システムのシミュレーションにおいても，現実世界と同様に，個々のエージェントが直面する状況は多岐にわたる．例えば，交通シミュレーションの場合，通勤，買い物，旅行など，目的により走行の仕方が異なる．また，渋滞時や緊急車両発見時などの特殊な場合にも走行の仕方を変更する必要があり，状況に応じた多種多様なプロトコルを与える必要がある．また，人と人との言語を用いたコミュニケーションといった社会的なインタラクションを再現するためには，エージェントと外界とのインタラクションを記述する必要がある．

そこで，エージェントのインタラクションプロトコルをシナリオとして記述することでエージェントの振る舞いを規定するという手法をとる．その際に，これらの多種多様なプロトコルを１つのシナリオとして記述しようとすると，長大で複雑なものとなってしまう．また，周囲の環境は時々刻々と変化するため，状況の変化に対してシナリオの追加や変更を行う必要がある．そのため，特定の状況ごとにプロトコルを分割し，それらを個別のシナリオとして記述して，エージェントに複数のシナリオを割り当てて実行させる．

しかし，上記のようにエージェントに複数のシナリオを割り当てて社会シミュレーションを行うにあたり，以下の２つの課題がある．

1. シナリオを追加・変更する際の競合
   シミュレーションの状況の変化に対して，新たなシナリオの追加や既存のシナリオの変更などの必要性が生じる．しかし，新しいシナリオをそのままエージェントに割り当てて実行させると，他のシナリオと独立して処理

され，互いの動作が考慮されない．そのため，同時に実行できないことをしようとするなどの競合が発生する恐れがある．

2. 状況に応じたプロトコルの切り替え
   状況に応じてプロトコルを使い分けるためには，状況に応じて実行するシナリオを切り替える必要がある．そのためには，特定のシナリオを優先的に実行するための実行制御を行うことはもちろん，そういったシナリオの実行制御の方法を状況に応じて動的に変更できる必要がある．

そこで，本研究ではこの課題を解決するために，メタレベルアーキテクチャを用いてシナリオの実行制御を行うというアプローチをとった．エージェントのシナリオ実行プロセスを，外界とのインタラクションを行う実行レイヤと，シナリオの実行制御を行う制御レイヤの 2 層に分けた．こうすることで，通常時のインタラクションは実行レイヤのみで行い，シナリオの追加や切り替えといった処理が必要なときにだけ制御レイヤに制御を移すということが可能になる．

このアーキテクチャにより，上記の問題を解決するシステムを実現した．以下の 2 点が，本研究の主な貢献である．

1. 協調動作を行うためのシナリオ修正
   シミュレーションの実行中に新たにシナリオを獲得すると，制御レイヤに処理を移し，協調動作を行うための事前・事後処理の記述をそのシナリオに書き加えてからエージェントに割り当てるようにした．こうすることで，シミュレーション中に新たに獲得したシナリオを，既存のシナリオとの間で協調して実行することを可能にした．
2. シナリオの協調方針の切り替え
   協調動作を行う際のシナリオ間のメッセージの受け渡し方法や，シナリオ実行の調整方法などの方針を協調ポリシーとして定めた．そして，状況に応じて協調ポリシーを切り替えることにより，各シナリオの優先順位などを動的に変更することで，エージェントが持つ複数のプロトコルを使い分けて周囲の環境や相手に対して適切な行動をとることを可能にした．

そして，実際にこれらの機能をもつシステムを実装し，交通シミュレーションおよび対話エージェントを用いたシミュレーションにおいて，シナリオの追加・修正や複数のシナリオ間の協調動作が実現されることを示した．それにより，提案したアーキテクチャが，シミュレーションによって社会的なインタラクションを再現するために有効であることを検証した．

Coordination of Interaction Scenarios by Meta Level Control

Shoichi SAWADA

Abstract

Multi-agent simulation is attracting attention to simulate social systems. It is useful to simulate and analyze complex social phenomena such as traffic styles and economic phenomena caused in the real world. In the real world, we have various protocols to interact appropriately depending on various situations. And we can comunicate appropriately with others by using those protocols properly corresponding to the other party and the surrounding environment. It is the purpose of this research to reproduce such social interactions by multi-agent simulation.

In the simulation of social systems, there are various situations that an agent faces as well as the real world. In the case of a traffic simulation, for instance, the method of driving is different according to the purposes such as commuting, shopping, or travel. Moreover, it is necessary to change the method of driving in special situations such as traffic jam or when discovering an emergency vehicle. Thus, it is necessary to give various protocols corresponding to the situation. In addition, it is necessary to describe interactions between agents and the external world to reproduce social interactions using language communication.

Then, I adopt the method of describing agent's interaction protocols as scenarios to provide agent's behavior. In this case, if these various protocols are described in one scenario, it becomes long and complex. Moreover, it is necessary to add or change scenarios according to the change of the situation because the surrounding environment changes every moment. Therefore, I divide the protocol for each specific situation and describe them as individual scenarios, and assign those more than one scenarios to an agent.

However, there are the following two problems to execute social simulation when two or more scenarios are assigned to an agent.

1. Conflict in adding or changing scenarios:

   It is necessary to add a new scenario or change existing scenarios according to the change of the situation in the simulation. However, these scenar-

ios are processed independently of other scenarios without considering the operations of each other, if a new scenario is assigned to an agent as it is. Therefore, conflicts might be occurred such as trying to execute actions which can not do at the same time.

2. Switch of protocols corresponding to the situation:
   It is necessary to switch the scenarios according to the change of situations to use the protocols properly corresponding to the situation. It should be dynamically revokable how to control the execution of scenarios for giving priority to specific scenarios.

To solve these problems, I took the approach of controlling the execution of scenarios by using the meta level architecture. I divided the process of executing scenarios into two layers of the control layer and the execution layer. In this way, usual interactions are processed only in the execution layer, and the process is executed in the control layer only when it is necessary to add or switch scenarios.

I solved the above issues by this architecture. The following two points are the main contributions of this research.

1. Correction of scenarios for operating cooperatively:
   When a new scenario is added, process in the control layer add the description for cooperating with other scenarios into the scenario before assigning it. As a result, it was enabled to execute new scenarios cooperatively with other scenarios.
2. Switch of coordination policies of scenarios:
   I provided the method of handing over the message between scenarios for cooperating with other scenarios as coordination policies.
   It was enabled to use the protocols properly corresponding to the situation by dynamically switching the coordination policies.

I implemented the system and executed a traffic simulation and a conversational simulation. These simulation showed the effectiveness of the system of this research for reproducing social interactions by achieving the cooperation among two or more scenarios.

メタレベル制御によるインタラクションシナリオの協調

目次

第1章　はじめに

交通流や経済現象などの社会システムのシミュレーションにおいて，マルチエージェントシミュレーションを用いる方法が注目されている．これはシステムを構成する各要素をエージェントとしてモデル化し，エージェントの集合によってシステムを構成するものである．例えば，交差点のシミュレーションにおいて生じるデッドロックの問題を分析するために，交差点における車両間のインタラクションのシミュレーションが行われている [1]．また，実際の道路状況やドライバーの振る舞いをモデル化し，そのモデルを用いたシミュレーションによる交通流の分析が行われている [2]．このように，マルチエージェントシミュレーションは，現実世界で生じる現象を再現することで，複雑な社会現象の原理を解析することに役立っている．

現実世界において人間は，異なる状況に応じて適切なインタラクションを行うための様々なプロトコルを持っている．そして，周囲の環境や相手にあわせてそれらのプロトコルを使い分けることによって適切にコミュニケーションを行うことができる．さらに，そういったインタラクションプロトコルもまた，社会的なインタラクションによって学習・獲得される．このような社会的なインタラクションを，マルチエージェントシミュレーションによって再現することが，本研究の目的である．

社会システムのシミュレーションにおいても，現実世界と同様に，個々のエージェントが直面する状況は多岐にわたる．例えば，交通シミュレーションの場合，通勤，買い物，旅行など，目的により走行の仕方が異なる．また，渋滞時や緊急車両発見時などの特殊な場合にも走行の仕方を変更する必要があり，状況に応じた多種多様なプロトコルを与える必要がある．また，エージェントが人間と自然な会話を行うためには，人間と同様に多様な状況に応じたインタラクションプロトコルを持ち，相手や場面に応じてそれらを使い分ける必要がある．

このように，シミュレーションによって，人と人との言語を用いたコミュニケーションといった社会的なインタラクションを再現するためには，エージェントの内部メカニズムではなく，エージェントと外界とのインタラクションを記述する必要がある．そこで，エージェントのインタラクションプロトコルをシナリオとして記述することでエージェントの振る舞いを規定するという手法をとる．その際に，エージェントが直面するあらゆる状況におけるインタラクショ

ンプロトコルを 1 つのシナリオとして記述しようとすると，長大で複雑なものとなってしまう．また，周囲の環境は時々刻々と変化するため，状況の変化に対してシナリオの追加や変更を行う必要がある．そのため，特定の状況ごとにプロトコルを分割し，それらを個別のシナリオとして記述する．そして，エージェントに複数のシナリオを割り当てて，それらのシナリオを独立して実行するようにする．

シナリオを分割することで，一つ一つのプロトコルを簡潔にすることが可能となり，容易にエージェントのシナリオを作成できるようになる．また，プロトコルの追加・変更は，個々のシナリオを追加・修正することで行うことが可能になる．さらに，多くのプロトコルから状況に応じて適切なものを選択して実行するということは，複数のシナリオの中から優先して実行するシナリオを状況に応じて切り替えるということに帰着する．

しかし，上記のようにエージェントに複数のシナリオを割り当てて社会シミュレーションを行うにあたり，以下の 2 つの課題がある．

1. シナリオを追加・変更する際の競合
   シミュレーションの状況の変化に対して，新たなシナリオの追加や既存のシナリオの変更などの必要性が生じる．しかし，新しいシナリオをそのままエージェントに割り当てて実行させると，他のシナリオと独立して処理され，互いの動作が考慮されない．そのため，同時に実行できないことをしようとするなどの競合が発生する恐れがある．
2. 状況に応じたプロトコルの切り替え
   状況に応じてプロトコルを使い分けるためには，状況に応じて優先して実行するシナリオを切り替える必要がある．そのためには，特定のシナリオを優先的に実行するための実行制御を行うことはもちろん，そういったシナリオの実行制御の方法を状況に応じて動的に変更できる必要がある．

そこで，本研究ではこの 2 つの課題を解決するために，メタレベルアーキテクチャを用いてシナリオの実行制御を行うというアプローチをとる．こうすることで，柔軟なシナリオの実行制御を行い，状況に応じた動的な処理をすることを目指す．

以下，本稿の構成は次のとおりである．第 2 章では，マルチエージェントシミュレーションのインタラクション記述における課題を示す．第 3 章では，本研究の基盤となる関連研究について述べる．第 4 章では，本研究で実現するシ

ステムのアーキテクチャについて説明し，第 5 章で，システムの実装とその適用例を示す．そして，第 6 章で本研究のまとめを行う．

第 2 章　マルチエージェントシミュレーションにおけるインタラクション記述

本研究では，マルチエージェントシミュレーションを用いて社会的なインタラクションを再現するために，エージェントと外界とのインタラクションを記述することでエージェントの動作を規定するというアプローチをとる．以下，本章ではエージェントのインタラクションを記述するための言語について，およびインタラクションの記述によってエージェントに社会的なインタラクションを行わせる際の課題について述べる．

2.1　シナリオ記述言語

エージェントのインタラクションを記述するための言語に，シナリオ記述言語 *Q*[3] がある．*Q* は，エージェントの内部メカニズムには言及せず，エージェントと外界とのインタラクションプロトコルを，拡張状態遷移機械モデルを用いてシナリオとして記述するための言語である．人間行動のモデル化において，*Q* のように人間と外部世界とのインタラクションプロトコルをそのままシナリオとして記述する方式のほうが，人間行動の中身を記述する旧来のエージェント記述方式よりも効果的であることが示されている [4]．*Q* の特徴的な言語機能としては，以下の 3 点が挙げられる．

1. キューとアクション

   キューは，インタラクションのきっかけとなるイベントの観測を行うために用いる．そのため外界への副作用はなく，キューによる観測は，きっかけとなるイベントが観測されるまで続けられる．一方のアクションは，外界に作用するエージェントの振る舞いを規定するために用いられる．なお，キューは?から始まるコマンドで，アクションは!から始まるコマンドで記述される．

   図 1 の例は，誰かから“ おはよう ”と話しかけられると，その人に対して“ おはようございます ”と返事するということを意味する．
2. ガード付きコマンドとシナリオ

```
(?hear :word "おはよう" :from $x)
(!speak :word "おはようございます" :to $x)
```

図 1: キューとアクションの例

ガード付きコマンドは，複数のイベントを並列に観測するために導入されている．観測中のイベントのうちひとつでも観測されれば，後続するアクションを実行する．シナリオは状態遷移表現を用いて記述され，各状態はこのガード付きコマンドとして定義される．

3. エージェントとアバター
   エージェントとアバターを定義することが可能である．エージェントには何をすべきかが記述された行動シナリオが与えることが必要であり，エージェントはそれに基づいて行動する．一方，人間によって制御されるアバターは，基本的にはシナリオを必要としないが，その振る舞いを制限する必要があればシナリオを与えることも可能である．

また，*Q* にはシナリオ記述のサポートのために，インタラクションパターンカード (IPC) と呼ばれるツールが導入されている．これは，カード基本文法と呼ばれるシンタックスに従ってシナリオを設計するためのツールである．これにより，計算機の非専門家でも容易にシナリオの記述ができるようになっている．

2.2 インタラクション記述における課題

社会システムのシミュレーションにおいて，エージェントの直面する状況は多岐にわたる．そのため，あらゆる状況におけるインタラクションプロトコルをひとまとめに記述しようとすると，長大で複雑なものとなってしまう．そこで，特定の状況ごとにプロトコルを分割しそれらを個別のシナリオをとして記述するという手法をとる．こうすることで，一つ一つのプロトコルを簡潔に記述することが可能となり，容易にエージェントのシナリオを作成できるようになる．

しかし，シナリオを複数に分割することにより１つのエージェントに対して複数のシナリオを割り当てることになる．そのため，それらのシナリオを並列にあるいは状況に応じて切り替えて実行する必要があり，その際に問題が生じる．

ここで例として，身体を持ち，音声やテキストを介して対話を行うエージェ

ントが，他のエージェントと会話をするというシミュレーションを考える．会話の仕方や内容といったプロトコルは，親しい友人，目上の人，他人など，相手によって異なる．また，日常会話，公式な場での発言などの場面の違いや，家，学校，電車の中などの場所の違いによっても異なる．このように，エージェントは状況に応じて多種多様なプロトコルを持ち，それらを使い分ける必要がある．そのため，それらのプロトコルを特定の状況ごとに分割し，複数のシナリオとして記述する．そして，エージェントはその複数のシナリオを，並列にあるいは状況に応じて切り替えて実行する必要がある．

このシミュレーションにおいて，複数のエージェントから同時に話しかけられたとする．すると，相手に応じて異なるシナリオを並列に実行することになる．しかし，エージェントの身体は１つしかないため，複数の相手に対して同時に話しかけることは不可能である．このように，シナリオを並列に実行する場合，各シナリオは独立に処理され互いの内容は考慮されない．そのため，同時に実行できないことをしようとしたり，互いに矛盾する動作をしようとしたりするといった，競合する動作が同時に発生することがある．この問題を解決するためには，各シナリオの状態を把握し，競合する動作の実行を待機もしくは中止させるといった同期処理を行う必要がある．

また，シミュレーションにおいて，休み時間中に友人と会話をしていたとする．このときに始業のチャイムが鳴ると，それまでの会話をやめて授業を聞く必要がある．また，授業中に何かを言うときには，形式的に発言しなくてはいけない．そのため，授業が始まると日常会話を行うシナリオの実行を抑制し，発言する際には公式な場での対話シナリオを実行する必要がある．このように，状況に応じてシナリオを切り替えて実行するためには，シミュレーションの実行状況を観測し，状況に応じた柔軟なシナリオの実行制御を行うことが必要となる．また，状況に応じて特定のシナリオの動作を優先したり，あるいは抑制したりするためには，特定のシナリオを優先的に実行するための動作制御を行う必要があるだけでなく，そういったシナリオの動作制御や同期処理の方法を動的に変更する必要がある．

さらに，相手や場面の変化や新たな状況に対応するために，プロトコルを新たに獲得したり修正したりすることも必要になる．このように，シミュレーション環境の変化に対応するためや，他のエージェントとのインタラクションによって，新たなシナリオの獲得や既存のシナリオの修正などを行う必要が生じる．そ

のためには，シミュレーション中にシナリオを動的に制御し，シナリオの修正や割り当てを行うといった機能が必要となる．

第3章　関連研究

本研究で提案するシステムを実現するために，既存の研究で提案されている2つの手法を取り入れた．本章では，それらの手法について説明する．

3.1　メタレベル制御機構

マルチエージェントシミュレーションにおけるシナリオ実行制御を行うための手法として，大規模マルチエージェントシステムのためのメタレベル制御機構 [5] がある．これは，エージェントの動作制御を行うために，エージェント記述言語に対してメタプログラミングを行うという手法を提案したものである．この中で，メタプログラミングによるシナリオ実行制御と，エージェントシステムに対する制御管理機構を統合して行うための記述方式として，メタシナリオが提案されている．

メタシナリオは，エージェントの動作を規定するエージェント記述に対して直接アクセスすることが可能であり，これにより柔軟なエージェントの動作制御が実現されている．メタシナリオの主な機能としては，エージェントのシナリオ実行制御とシミュレーション環境の制御の2つがある．

メタシナリオはQと同様に，拡張状態遷移機械モデルで記述される．これにより，シミュレーションの状況を状態遷移機械の状態，制御を状態遷移機械の遷移として表現することができる．メタシナリオの基本的な言語仕様はQと同様であり，キュー，アクション，ガード付きコマンドを用いたシナリオとして記述する．メタシナリオの各状態において観測の対象となるものは大きく分けて2つあり，エージェントシステムの環境とエージェントのシナリオ実行状況である．ここでエージェントのシナリオ実行状況とは，シナリオの状態遷移及びそれに伴う外部への作用を指す．メタシナリオの状態遷移に伴う作用としては，エージェントシステムの環境設定，エージェントのキュー・アクションの起動，シナリオの割り当て・開放，シナリオの実行開始・停止などがある．

既存研究では，大規模マルチエージェントシステムのシミュレーション環境の制御やエージェントの動作制御を行うための基盤の提供を目的としている．そ

のため，エージェントに複数のシナリオを割り当てて並列実行する際に，どのようにして状況に応じて適切なシナリオを実行するかという具体的な手法には着目していない．したがって，シミュレーションの実行中にシナリオを追加・修正する機能や，状況に応じて特定のシナリオを優先して実行するための機能が定義されていない．

本研究では，メタシナリオの持つ機能のなかで，特にエージェントのシナリオ実行制御の部分に焦点を当てる．そして，このメタシナリオに，新しく獲得したシナリオを修正するための機能と，シナリオを協調実行するための方針を定め，その方針を状況に応じて変更するための機能を追加することを考える．こうすることで，シミュレーション実行中の動的なシナリオの追加や修正，および状況に応じた協調方針の切り替えを行うことを目指す．

3.2 シナリオ協調メカニズム

複数のシナリオを並列に実行するための手法として，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 [6] がある．これは，マルチエージェントシミュレーションにおいて複数のシナリオを並列に実行する際に，競合を回避しエージェントが矛盾なく動作を行うための方式を提案するものである．

エージェントには，相手や場面に応じて異なる複数のシナリオを与え，周囲の状況の変化や他のエージェントやユーザなどからの呼びかけに対してそれぞれのシナリオが独自に動作をする．そのため，複数のシナリオでキューが同時に発火してシナリオが動き出すと，同時に行うことが不可能なことをしようとしたり，互いに矛盾する行動をとろうとしたりなど，シナリオの間で競合が発生することがある．

このような競合を回避し，シナリオ間の協調動作を実現するために，協調ポリシーとコーディネーションシナリオが提供されている．協調ポリシーは，シナリオ間で協調動作を行うための方針を状態遷移モデルで記述したものである．また，コーディネーションシナリオは，他のシナリオとの通信によって競合の調査とシナリオ実行の同期処理を行うという役割を持つシナリオである．競合の調査は，各アクションの実行前後の状態の制約や，アクションで使用されるリソースの制約の情報を用いて行われる．

協調ポリシーでは，各状態において，シナリオからのメッセージに対してど

のような指示を返すかという方針を規定する．シナリオに対しての指示は，動作許可，動作待機，動作中止および動作再開などである．また，同期処理を行う際の事前・事後処理の仕方などを定める．

コーディネーションシナリオは，シナリオからメッセージを受け取り，シナリオの状態遷移や動作状態を把握して，他のシナリオと競合しないように動作の指示を出すという役割を持つ１つのシナリオである．シナリオから受け取るメッセージは，動作の実行確認，動作の完了報告，状態遷移の報告などである．実行確認のメッセージを受け取ると，他のシナリオの動作と競合が起こらないかを調べる．そして，競合が発生するならば動作の中止を指示し，発生しないならば協調ポリシーで規定された方針に従って指示を出す．

このコーディネーションシナリオは，協調ポリシーをもとにエージェントの持つ複数のシナリオに対応して，シナリオ生成器よって自動的に作成される．それと同時に，各シナリオに対して，コーディネーションシナリオにメッセージを送る機能と，そのメッセージに従って動作を行うための機能が，シナリオ生成器によって自動的に付加される．このように，エージェントの持つ複数のシナリオに対して協調ポリシーを指定することで，協調動作を行うようにシナリオを自動的に修正することが可能である．

しかしながら，既存の手法では，シミュレーションを行う前に事前に処理を完了しておく必要がある．そのため，シミュレーション中に新しいシナリオを獲得したとしてもそのシナリオをすぐに実行することはできない．また，協調ポリシーを動的に切り替えることもできない．そこで本研究では，これらの処理をシミュレーションの実行中でも行えるようにし，シナリオの割り当てやポリシーの切り替えを動的に行えるようにすることを目指す．

第４章　メタシナリオを用いたシナリオ実行制御

本章では，提案するシステムのアーキテクチャについて説明する．まず全体の構成について説明した後，実現する機能の詳細について述べる．

4.1　システムの構成

シミュレーションにおける動的なシナリオの実行制御を行うため，システムを実行レイヤと制御レイヤの２つの層で構成する．これは，シミュレーション

の実行状況の観測や実行制御をメタレベルで行うことにより，柔軟なシナリオの制御を行うためである．ここで，通常のインタラクションはすべて実行レイヤで実行し，シナリオの追加や修正，実行の開始・停止といった特別な制御を行う必要があるときだけ制御レイヤで処理を行う．

メタシナリオは，シナリオの実行制御の仕方を記述したものであり，観測結果に対してシナリオの修正や割り当て，実行開始・停止などの制御を規定する．協調ポリシーは，シナリオの協調動作を行う際の方針を記述したものであり，シナリオからのメッセージの観測と，状況ごとのメッセージに対する指示の仕方を規定する．コーディネーションシナリオは，シナリオの同期処理の仕方を記述したものであり，シナリオからのメッセージに対して競合の調査を行い，競合しないようにかつ協調ポリシーに従ってメッセージに対する指示を出すことを規定する．シナリオは，外界とのインタラクションを行うためのプロトコルを記述したものであり，キューによるイベントの観測と観測結果に対するアクションを規定する．

以下，2つのレイヤに分けてシステムの構成を説明する．このシステムの構成図を図2に示す．

4.1.1　実行レイヤ

実行レイヤでは，周囲の状況の観測と，動作や発話などの行動によって，エージェントと外界とのインタラクションを行う．

実行レイヤは，シナリオインタプリタとコーディネーションシナリオインタプリタの2つで構成される．シナリオインタプリタはインタラクションを記述したシナリオの解釈・実行，コーディネーションシナリオインタプリタはコーディネーションシナリオの解釈・実行を行う役割を持つ．

シナリオは，エージェントと外界とのインタラクションプロトコルを記述したものであり，1体のエージェントに対して複数割り当てて並列に実行することが可能である．1つのシナリオに対して1つのシナリオインタプリタが対応し，シナリオの記述を解釈し，それに従って動作を実行する．シナリオが複数割り当てられているときは，その数だけシナリオインタプリタが起動し，独立して動作を実行する．

コーディネーションシナリオは，シナリオが協調して動作を行うようにシナリオの実行を調整するという役割を持ち，1体のエージェントに対して1つ割り当てられる．コーディネーションシナリオインタプリタは，シナリオインタ

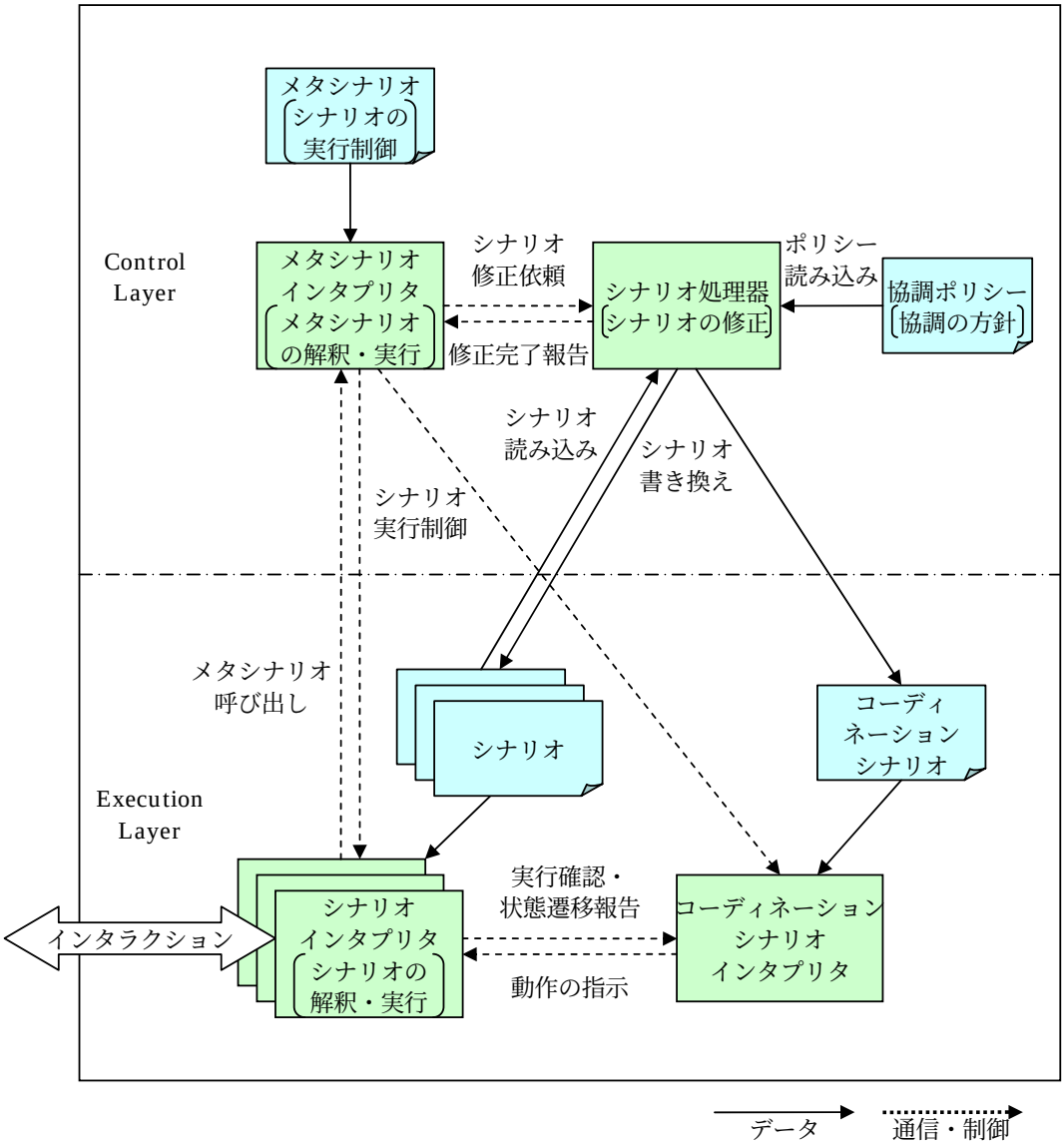

図 2: システムの構成図

プリタと通信することで，シナリオの状態を把握し，協調動作を行うための指示を出す．また，コーディネーションシナリオインタプリタは，外界とのインタラクションは行わない．

4.1.2 制御レイヤ

制御レイヤでは，エージェントへのシナリオの割り当てや開放，シナリオの実行開始や停止といった，メタレベルでのシナリオの実行制御を行う．また，シナリオ処理器によってシナリオの変更や追加などの処理を行う．

制御レイヤは，メタシナリオインタプリタとシナリオ処理器の 2 つで構成さ

れる．メタシナリオインタプリタは，メタシナリオの解釈・実行を行う．シナリオ処理器は，シナリオの修正とコーディネーションシナリオの作成を行う．

メタシナリオは，各状態における観測対象と，観測した状況に対するシナリオ実行処理の仕方を記述したものである．メタシナリオインタプリタは，メタシナリオの記述を解釈し，それに従ってシナリオの動作や状態遷移，環境の変化などのシミュレーションの実行状況を観測する．そして，観測結果に応じて，シナリオの追加や修正，シナリオの実行の開始・停止，協調ポリシーの切り替えといったシナリオの実行制御を行う．メタシナリオの持つ具体的な機能は表1に示す．

表1: メタシナリオの機能

エージェントの制御	
?observeAction	アクションの実行の観測
?observeCue	キューの実行の観測
?observeTransition	状態遷移の観測
?observeScenario	シナリオ実行の観測
!runCue	キューの実行を依頼
!runAction	アクションの実行を依頼
!releaseScenario	シナリオの解放
!assignScenario	シナリオの割り当て
!modifyScenario	シナリオの修正
シミュレーションの制御	
?observeEnvironment	シミュレーション環境の観測
!getEnvironment	シミュレーション環境情報の取得
!setEnvironment	シミュレーション環境の設定
!createAgent	エージェントの作成
!deleteAgent	エージェントの削除
!createCrowd	群集の作成
!createAvatar	アバタの作成
!startSimulation	シミュレーションの開始
!stopSimulation	シミュレーションの停止

シナリオ処理器は，協調ポリシーをもとにコーディネーションシナリオを作成し，同期処理を行うための記述をシナリオに書き加えるという処理を行う．まず，処理を行う対象のシナリオを読み込み，指定された協調ポリシーを読み込む．次に，協調ポリシーに従ってシナリオに協調動作を行うように指示を出すためのメッセージのやり取りの方法を記述したコーディネーションシナリオを作成する．そして，動作を実行する前の確認方法とコーディネーションシナリオからの指示に対する応じ方といった事前・事後処理を行う部分をシナリオに追加する．

4.2 動的なシナリオの追加・修正

シミュレーション中に，インタラクションによって新たにシナリオを獲得したり，シミュレーション環境の変化に対してシナリオを変更する必要性が生じたりしたときに，シナリオを修正してから割り当てることにより他のシナリオとの協調動作が行えるようにする機能について説明する．ここで重要となるのは，メタレベルで制御を行うことにより，シミュレーション中に獲得したシナリオを修正して割り当てることができ，その場で他のシナリオと協調して実行することが可能になるということである．

シミュレーション中に，他のエージェントとのインタラクションなどにより新しいシナリオを獲得すると，新しいシナリオを受け取ったシナリオがメタシナリオを呼び出し，シナリオを獲得したことを報告する．シナリオからの新しいシナリオを獲得したという報告をメタシナリオが観測すると，シナリオの処理を開始する．また，既存のシナリオを変更するという場合はそのシナリオの実行を停止してシナリオを開放してから処理を行う．この処理は制御レイヤで行われ，通常のインタラクションは実行レイヤのみで実行されるため，処理中も他のシナリオの実行を継続することが可能である．

シナリオ処理器で行われるシナリオ修正の処理フローを図 3 に示す．シナリオ修正の処理は，メタシナリオが修正するシナリオと用いる協調ポリシーを指定して，シナリオの修正をシナリオ処理器に依頼することで開始される．依頼を受けたシナリオ処理器は，まず指定されたシナリオと協調ポリシーを読み込む．次に，協調ポリシーをもとに，シナリオとメッセージをやり取りしてシナリオの状態を把握し，シナリオ間で協調動作を行うようにシナリオに動作の指示を出すためのコーディネーションシナリオを作成する．そして，コーディネー

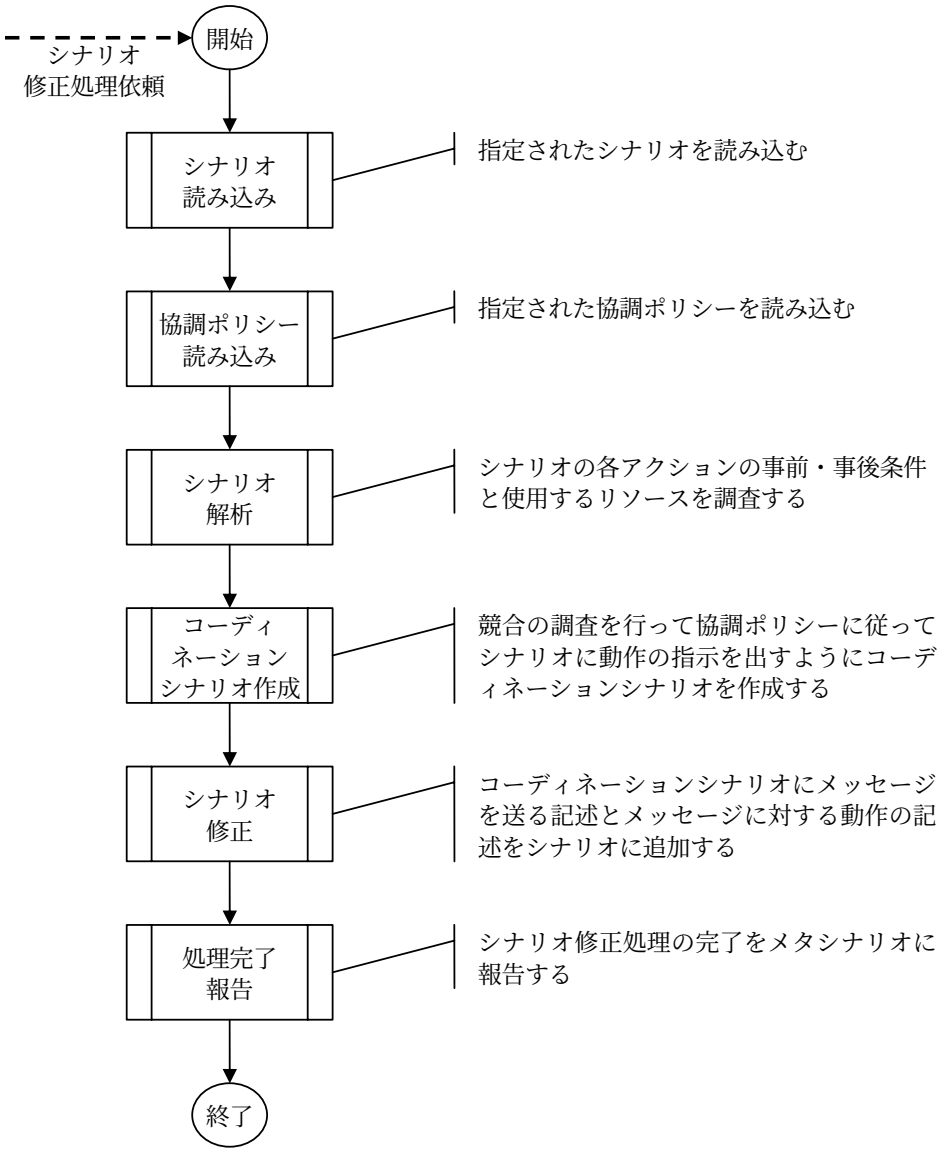

図 3: シナリオ処理器の処理フロー

ションシナリオに実行確認をするためのメッセージの受け渡しの仕方や命令に従って動作を変更する際の処理の仕方といった，事前・事後処理を行うための記述をシナリオに追加することで，シナリオの修正が完了する．シナリオの修正が完了すると，そのことをメタシナリオに報告し，シナリオ処理器は実行を終える．最後に，修正完了報告を受けたメタシナリオが，修正されたシナリオとコーディネーションシナリオを割り当てることにより，新しいシナリオの実行が開始される．

新しく割り当てられたシナリオは，実行時にコーディネーションシナリオに動作確認を行うことにより，既存のシナリオと競合しないように動作を行うことが可能となる．

4.3 シナリオの協調方針の切り替え

周囲の状況の変化に応じて協調ポリシーを切り替えることで，相手や場面に応じて適切なシナリオを実行できるようにする機能について説明する．

協調ポリシーは，シナリオ間で協調動作を行うための方針を定めるものであり，状態ごとにシナリオからの動作確認のメッセージに対しての指示の仕方を指定する．例えば，あるシナリオが動作を実行しているときには他のシナリオには動作の待機を指示し，動作を完了すると待機しているシナリオに対して順に動作の許可を与えるというような協調ポリシーを与えれば，動作確認を行った順に 1 つずつシナリオを実行するということができる．

こういった協調ポリシーを複数用意しておき，状況の変化に応じて用いる協調ポリシーを切り替えることにより，優先して実行するシナリオを場合ごとに変化させることができ，相手や場面に応じて適切なシナリオを優先して実行することが可能になる．

協調ポリシーの切り替えは，シナリオからのメタシナリオ呼び出しや，メタシナリオのシミュレーション環境の観測結果をトリガーとして行う．メタシナリオは，シナリオからの状況の変化を報告する呼び出しや，シミュレーション環境の観測により，状況が変化したという情報を得る．その状況の変化に応じて，メタシナリオは適切な協調ポリシーを 1 つ選択し，そのポリシーに従ってシナリオ間の協調処理を行うように処理をする．

まず，メタシナリオは協調ポリシーを指定して，シナリオ処理器に対してコーディネーションシナリオを作成するように依頼する．依頼を受けたシナリオ処理器は，指定された協調ポリシーを読み込み，その協調ポリシーをもとにシナリオ間で協調動作を行うためのコーディネーションシナリオを作成する．作成が完了するとメタシナリオに報告し，メタシナリオは以前のコーディネーションシナリオを開放して新しいコーディネーションシナリオを割り当てる．

こうして，特定シナリオの動作を他のシナリオよりも優先するという協調ポリシーをもとにしたコーディネーションシナリオを割り当てれば，他のシナリオの実行を抑制して，あるシナリオの動作を優先的に実行するといったことも

可能となる．

第5章　システムの実装と評価

本研究では，シナリオ記述言語 Q と，その母体言語である Scheme によってシステムの実装を行った．本章では，システムの実装と，そのシステムを用いて行ったシミュレーションについて説明し，それにより本研究で提案するシステムの評価を行う．

5.1　システムの実装

システムの実装にあたっては，第3章で紹介した研究で提供されているメタシナリオとシナリオ生成器を用いた．本研究で提案するシステムにあわせて，シナリオ生成器を一部改変したものをシナリオ処理器と呼ぶ．そして，メタシナリオに，シナリオ処理器を呼び出す機能と協調ポリシーを切り替える機能を追加することで，提案する2つの機能をもつシステムを実現した．

- メタシナリオがシナリオ処理器を呼び出し，シナリオと協調ポリシーを指定してシナリオの修正やコーディネーションシナリオの作成を依頼するための機能を定義した．
- シナリオからメタシナリオを呼び出すためのアクションと，メタシナリオがシナリオからの呼び出しを観測するための機能を定義した．
- インタラクションによってシナリオの受け渡しを行うためのアクションとキューを定義した．

このシステムを用いて，交通シミュレーションと対話エージェントを用いたシミュレーションを行った．以下では，その2つのシミュレーションについて説明したうえで，提案した機能が実現されることを示す．

5.2　交通シミュレーション

道路上を移動する自動車が，緊急車両が接近してきたときに道を譲る際の動作をシミュレーションすることを考える．このシミュレーションでは，協調ポリシーを動的に切り替えることにより，状況に応じて協調処理の方針を切り替えて優先するシナリオを変更し，周囲の環境や相手に対して適切な行動を行うことができることを示す．

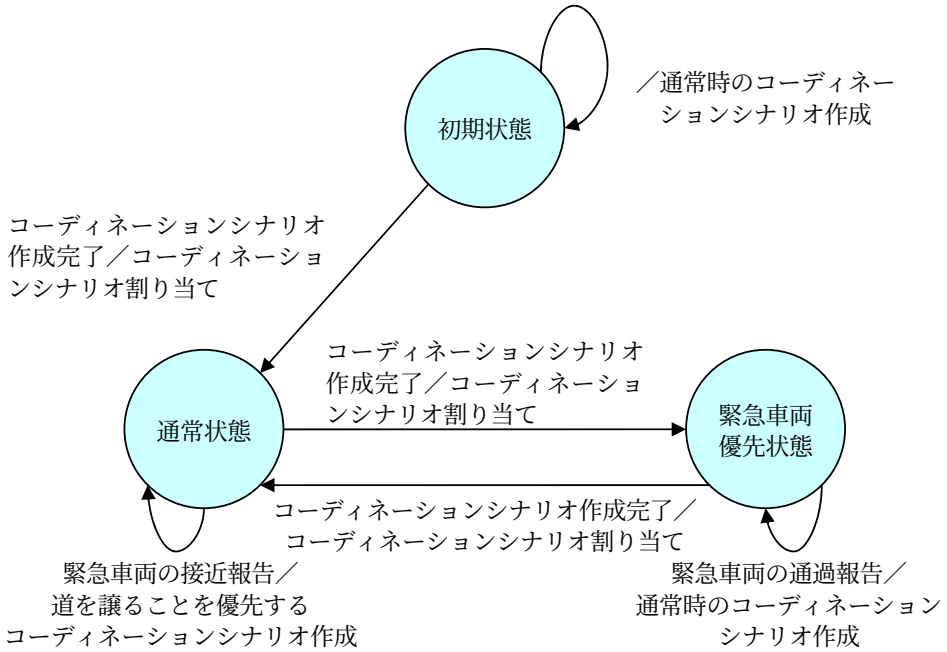

図 4: 交通シミュレーションにおけるメタシナリオ

道路上を移動する自動車は，通常時は目的地に向かって走行しているが，緊急車両を観測すると，道を譲るために減速し車線を空ける．これをマルチエージェントシミュレーションに当てはめて考え，目的地に向かって通常走行するシナリオと他車に道を譲るシナリオ，周囲を観測するシナリオの 3 つのシナリオをエージェントに割り当てる．そして，状況に応じて協調ポリシーを切り替えることにより，緊急車両が近づくと道を譲るという動作を実現する．

このシミュレーションで用いたメタシナリオのモデルを表す状態遷移図を図 4 に示す．このメタシナリオは，シナリオからの報告に対して，状況に応じた協調ポリシーに従ってコーディネーションシナリオを作成して割り当てるという動作を規定する．また，道を譲ることを優先する協調ポリシーの状態遷移図を図 5 に示す．これは，通常走行シナリオと道を譲るシナリオを区別し，道を譲るシナリオからの動作確認を優先的に許可するという動作を規定する．

シミュレーションにおいて通常時には，通常走行シナリオが目的地に向かって道路を直進するという動作を主に実行する．もちろん，緊急車両以外でも後ろから速い車両が近づいて来ることを観測すれば道を譲るということもありえるが，道を譲ることを優先するわけではないので必ずしも道を譲るということはない．

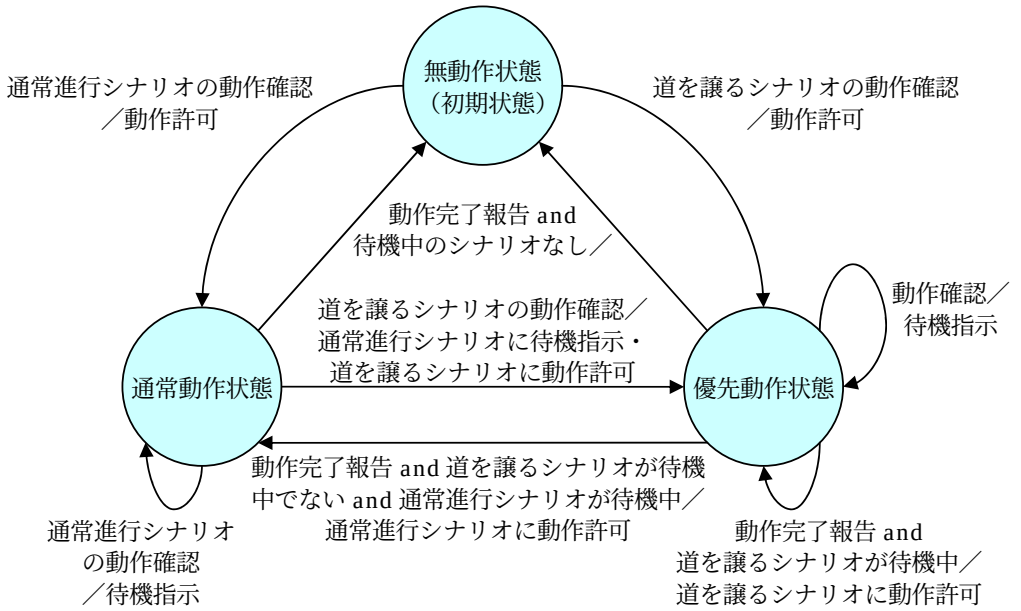

図 5: 道を譲ることを優先する協調ポリシー

そして，周囲を観測するシナリオが緊急車両を観測すると，メタシナリオに対して緊急車両の接近を報告する．報告を受けたメタシナリオは，道を譲ることを優先する協調ポリシーに従ってコーディネーションシナリオを作成し，割り当てる．こうすることで，道を譲るシナリオの動作が優先され，通常走行シナリオの動作が抑制されるようになる．そのため，通常走行シナリオが直進しようとしても，道を譲るシナリオが車を減速して車線を変更するという動作をしようとしていれば，後者のほうが優先され前者の動作は待機するように指示が出されることになる．この，緊急車両を観測してから協調ポリシーを切り替えて道を譲るシナリオの実行を優先するまでの，一連の流れを表すシーケンス図を図 6 に示す.

本研究で行ったシミュレーションにおいて，緊急車両には，前に車がない限り直進を続けるというシナリオを与えた．また，一般車両には，通常走行シナリオと道を譲るシナリオおよび周囲を観測するシナリオを与えた．ここで，通常進行シナリオは，前方の道が空いていれば直進し，前方に車がいて右車線が空いていれば右に車線を変更する．また一定ステップ進むごとに左車線が空いていれば左に車線を変更するという動作を行う．道を譲るシナリオは，後方に車を観測すると，左車線を確認して空いていれば左に車線を変更する，空いて

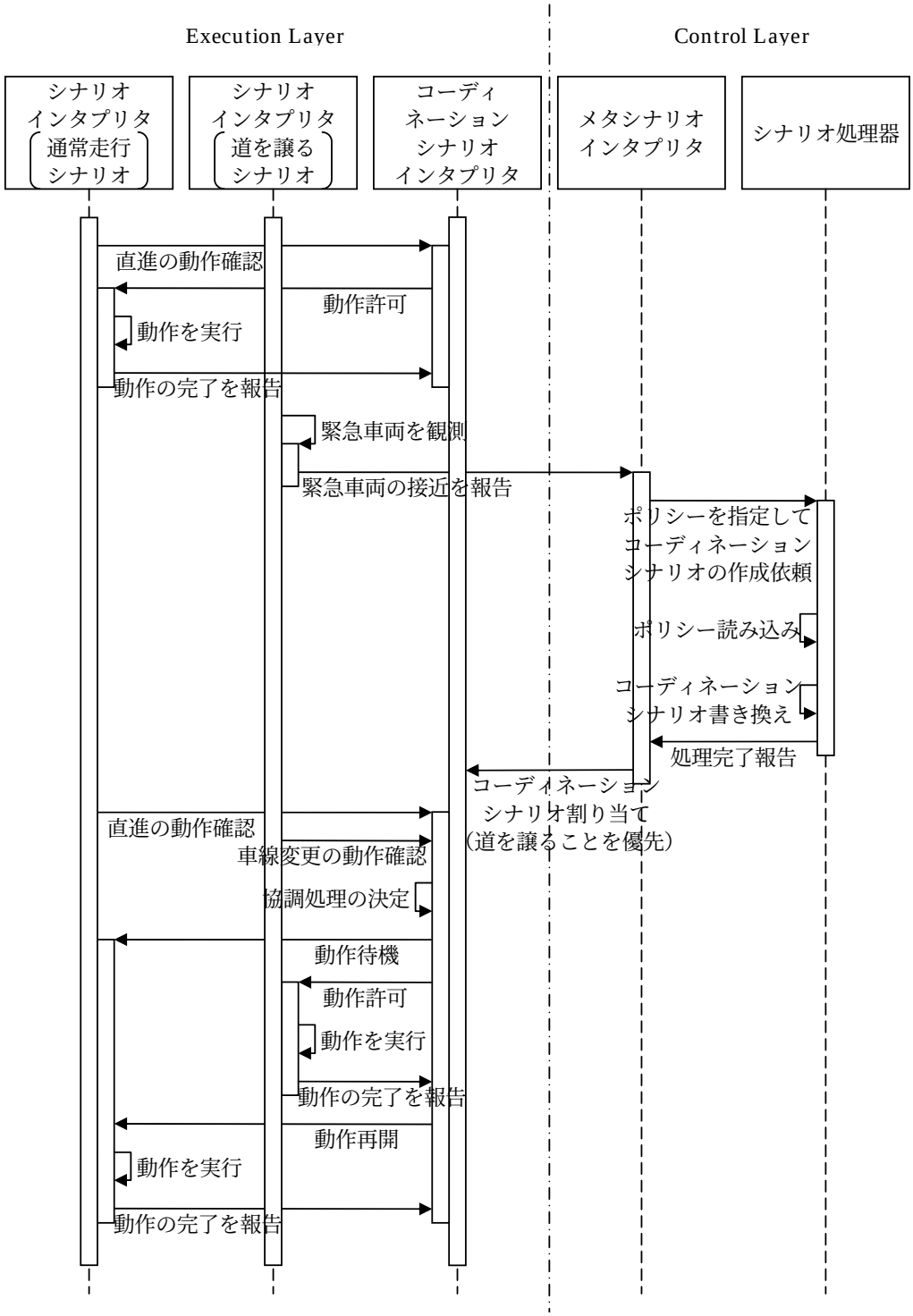

図 6: 協調ポリシーの切り替え

いなければ右車線を確認して空いていれば右に車線を変更するという動作を行う．そして，周囲を観測するシナリオは，緊急車両の接近や通過を観測し，その結果をメタシナリオに報告する．この，一般車両に与えた3つのシナリオを，図7，図8および図9に示す．

```
(defscenario scenario-car (&aux (step 0))
  (move
    ((?exist :direction front)
     (set! step 0)
     (go overtake))
    ((?test :form '(> ,step 8))
     (guard
      ((?movable :direction left)
       (!changeLane :direction left)
       (!move)
       (set! step 0)
       (go move))
      ((?movable :direction front)
       (!move)
       (set! step (+ step 1))
       (go move))))
    ((?movable :direction front)
     (!move)
     (set! step (+ step 1))
     (go move)))
  (overtake
    ((?movable :direction right)
     (!changeLane :direction right)
     (!move)
     (go move))
    ((?movable :direction front)
     (!move)
     (go move))))
```

図7: 通常走行シナリオ

```
(defscenario scenario-yieldway ()
  (wait
    ((?observe :name Cars :direction back)
     (go yield))
    ((?observe :name Ambulance :direction back)
     (go yield)))
  (yield
    ((?movable :direction left)
     (!winker :direction left)
     (!changeLane :direction left)
     (go wait))
    ((?movable :direction right)
     (!winker :direction right)
     (!changeLane :direction right)
     (go wait))
    (otherwise
     (go wait))))
```

図 8: 道を譲るシナリオ

```
(defscenario scenario-observe ()
  (wait
    ((?near :name Ambulance :distance 10)
     (!callMeta :arg 'AmbulanceApproaching)
     (go emergency)))
  (emergency
    ((?far :name Ambulance :distance 10)
     (!callMeta :arg 'AmbulancePassed)
     (go wait))))
```

図 9: 周囲を観測するシナリオ

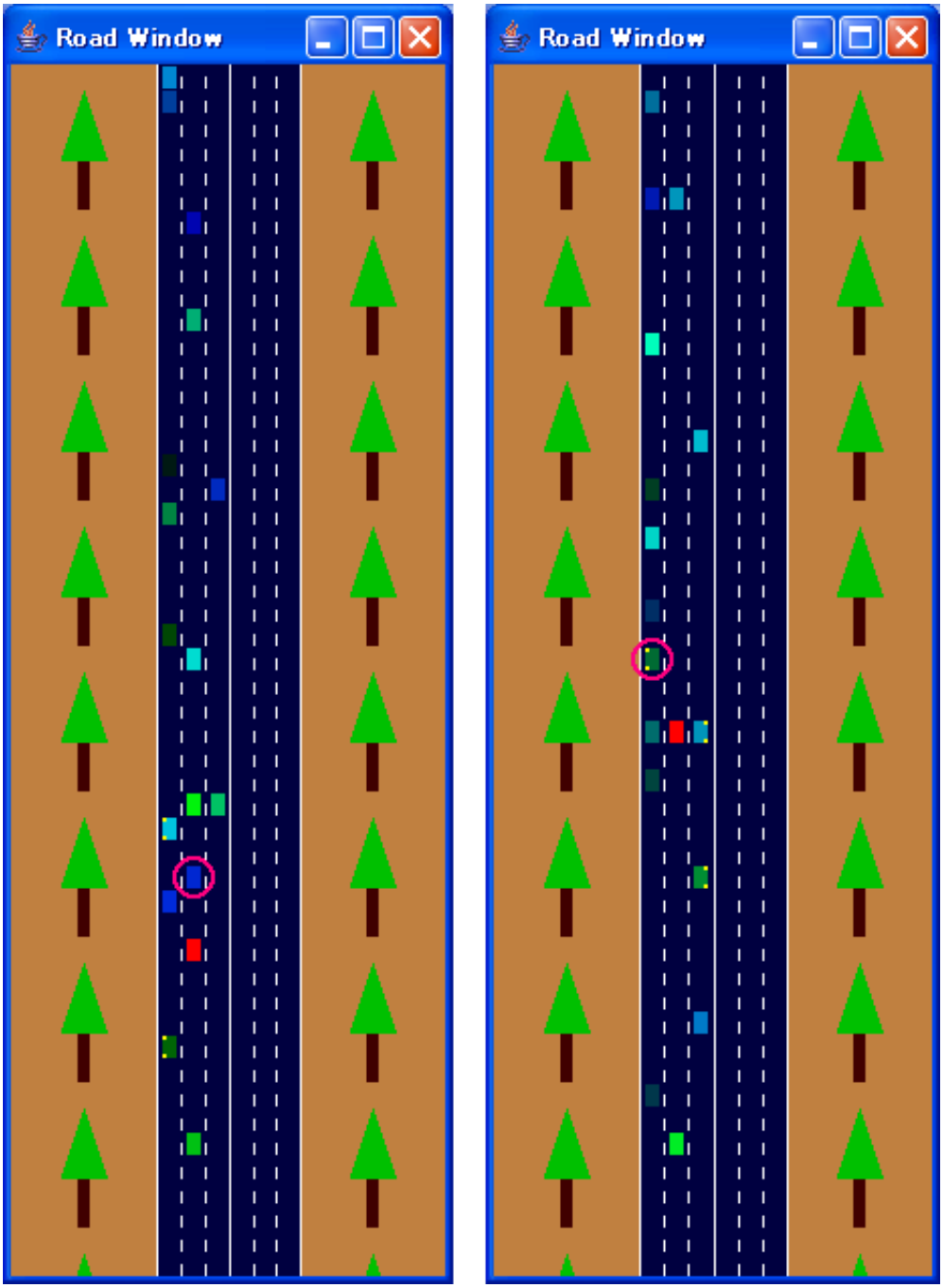

図 10: 交通シミュレーションのスクリーンショット

上記のシナリオを与えた緊急車両 1 台と一般車両 15 台を，片側 3 車線の直線道路上で走らせてシミュレーションを行った．実際にシミュレーションを行った際のスクリーンショットを図 10 に示す．左の図は道を譲るシナリオを優先しないときの様子，つまり協調ポリシーの切り替えを行わないときの動作である．それに対して，右の図は緊急車両が接近すると道を譲るシナリオを優先するときの様子であり，緊急車両を観測すると協調ポリシーを切り替えるようにしたときの動作である．図中で特徴的な動作を示している車両に印を付けた．

左の，協調ポリシーの切り替えを行わない場合では，緊急車両が接近しているにもかかわらず道を譲ろうとしていない．これは，他の車両を追い越そうとする動作が行われているため，道を譲る動作が実行されないためである．それに対して，右の，協調ポリシーの切り替えを行った場合には，車線変更先の前方に車両があるにもかかわらず，緊急車両に道を譲っている．これは，協調ポリシーによって道を譲る動作を優先することにより，追い越しをする動作が抑制されているからである．このように，協調ポリシーを切り替えることにより，緊急車両が接近してきたときに優先して道を譲るという動作が実現された．

以上により，本研究で提案したシステムを用いることにより，周囲の状況にあわせてプロトコルを切り替えて適切な行動をとることが可能となることが示された．

5.3 対話エージェント

仮想空間に身体を持った店員エージェントと店主エージェントが存在し，店主が店員に対して，客に話しかけられたときの対応の仕方を指示する場面を考える．そして，その指示された動作を店員エージェントがその場で実行できるようにするというシミュレーションを行う．このシミュレーションでは，新しく獲得したシナリオを，既存のシナリオと競合しないように協調実行できるようにするという例を示す．

このシミュレーションにおいて，身体を持った各アバターに 1 つのエージェントを対応させる．そして，各エージェントは状況に応じた動作や会話を行うためのインタラクションプロトコルを記述した複数のシナリオを持つ．また，それらのシナリオ間の動作の調整を行うためのコーディネーションシナリオ，およびシナリオの実行制御を行うメタシナリオを持つとする．

このシミュレーションで用いたメタシナリオのモデルを表す状態遷移図を図

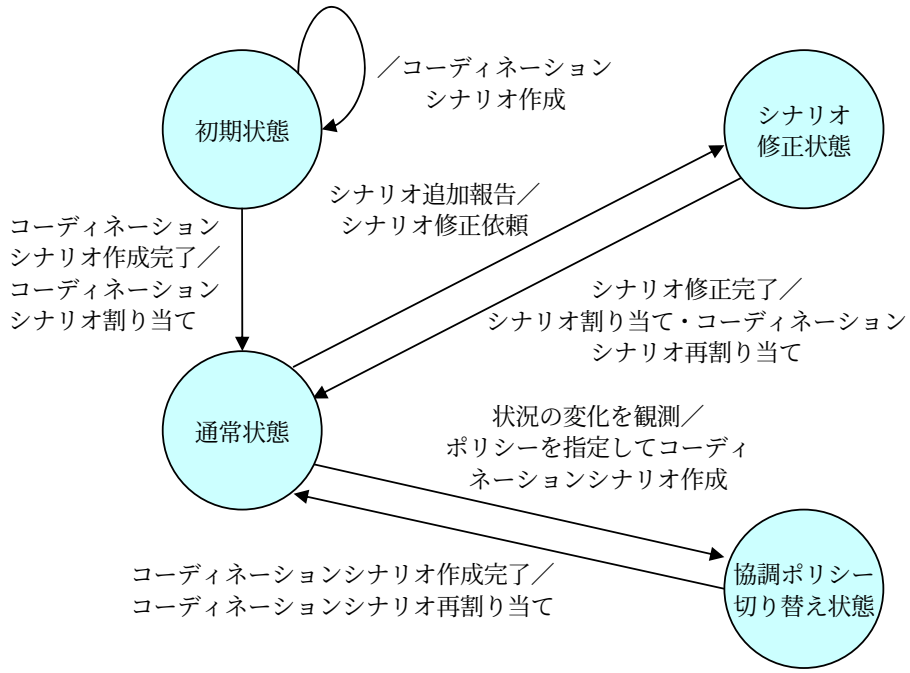

図 11: 対話エージェントのメタシナリオ

11 に示す．このメタシナリオは，シナリオを獲得したという報告に対して，シナリオを処理して協調動作を行うための記述を書き加えてから割り当てるという動作を規定する．また，状況の変化を観測すると，協調ポリシーの切り替えを行ってコーディネーションシナリオを割り当てなおす．

シミュレーションにおいて，店主と店員が会話によってインタラクションを行う．対話の中で店主は，店の品揃えを聞かれたときの対応，商品の詳細について聞かれたときの対応，道を聞かれたときの対応などを指示する．これらの指示は，相手によって対応の仕方が異なるインタラクションプロトコルを定めた，複数のシナリオとして店員に渡されるものとする．

店員の対話シナリオがインタラクションプロトコルを記述したシナリオを受け取るが，対話シナリオにとって受け取ったシナリオはただの文字列であり，そのままでは実行することはできない．そこで，その記述をシナリオとしてインタプリタに割り当て，実行できるようにする処理が必要となる．そのため，対話シナリオは新しくシナリオを受け取ると，メタシナリオに対してそのことを報告するようにする．そしてメタシナリオが受け取ったシナリオを処理し，インタプリタに割り当てて実行を開始させることにより，そのシナリオに従って

動作を行うことが可能となる．

ここで店員は，新しくシナリオを受け取ったときに，そのまま割り当てて実行を開始したとしても，シナリオに定められた動作を行うことは可能である．しかしそのまま実行したのでは，各シナリオは独立して実行されるため他のシナリオとの協調動作が行えず，同時に行うことのできない動作や他のシナリオと矛盾するような動作をしようとするといった競合が発生する可能性がある．そこで，受け取ったシナリオを修正して，協調動作を行うためのメッセージのやり取りや動作を変更する際の処理の方法などを定めてからシナリオを割り当てるようにする．こうすることで，シナリオは動作を行う前にコーディネーションシナリオに確認のメッセージを送り，メッセージに対する指示に従って動作を実行するようにする．

上記のように，新たに獲得したシナリオを処理してから割り当てることによって，新しいシナリオを他のシナリオと協調して実行できるようにするという処理を行う際のシーケンス図を図 12 に示す．

また店主は，道を聞く人よりも商品を購入する人を優先するといった，対話相手に対する対応の優先順位についても指示を出す．そのような，複数の相手に応対する際の行動のとり方や相手に応じた行動の優先度を定める方針は，協調ポリシーとして店員に与えられる．この協調ポリシーの状態遷移図を図 13 に示す．この例では，道を尋ねる人は一般人，商品のことを尋ねる人は客とみなし，一般人との対話よりも客との対話を優先するという方針を定める．

協調ポリシーを受け取った店員の対話シナリオは，そのことをメタシナリオに報告する．報告を受けたメタシナリオは，受け取った協調ポリシーを蓄える．そして，状況に応じて，その受け取ったポリシーを適用する際には，そのポリシーを用いてコーディネーションシナリオを作成し割り当てることで，そのポリシーに従ってシナリオ間の協調動作を行うようにすることができる．

以上のようにして，他のエージェントとのインタラクションなどによって，新しくシナリオを獲得した際に，既存のシナリオとの競合を回避し，協調して実行できるようにするという動作が実現される．これにより，本研究で提案したシステムを用いることで，シミュレーションの状況の変化に対してシナリオの追加や変更の必要性が生じたときに，他のシナリオと協調して実行できるようにするという機能が実現されることが示された．

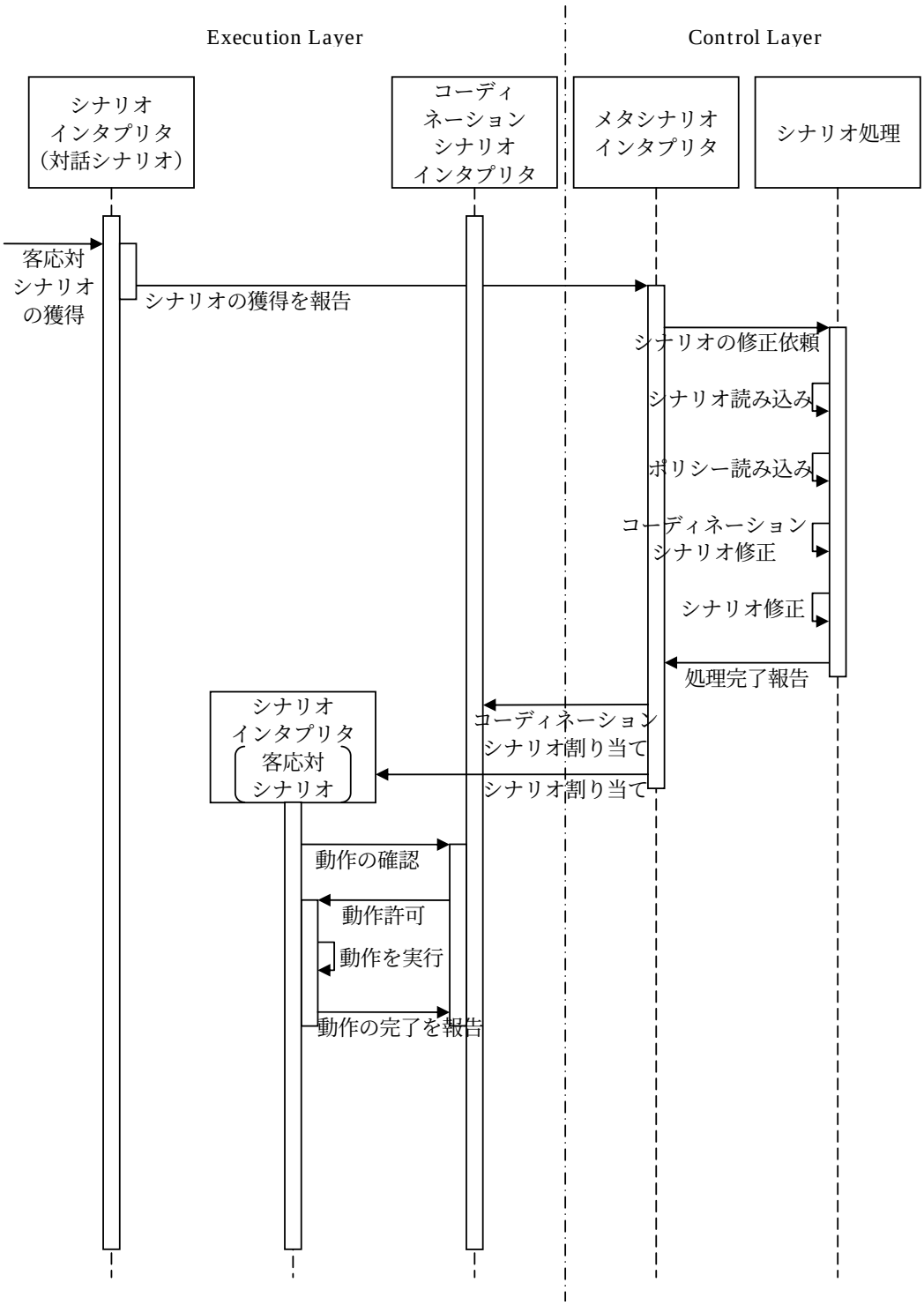

図 12: 新しいシナリオの獲得

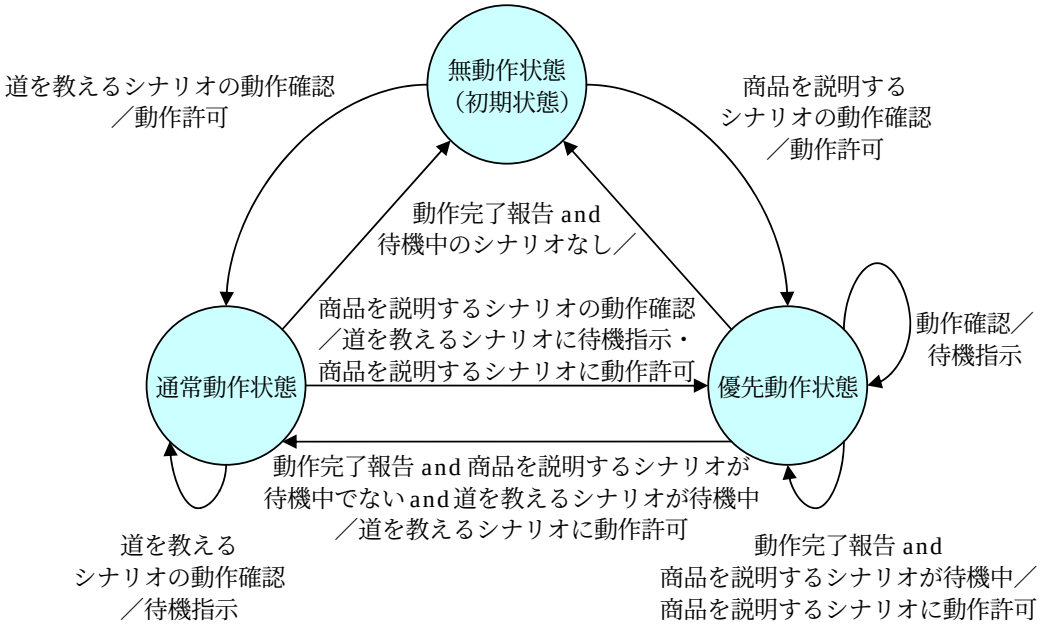

図 13: 対話相手に応じた優先の仕方を定める協調ポリシー

第 6 章　おわりに

本研究では，エージェントのインタラクションプロトコルをシナリオとして記述するマルチエージェントシミュレーションに着目した．そして，シミュレーションによって，現実世界で生じる社会的なインタラクションを再現することを目的とした．そのために，特定の状況ごとにインタラクションプロトコルを分割して複数のシナリオに分けて記述し，それらを並列にあるいは状況に応じて切り替えて実行するという手法をとった．そして，エージェントが状況に応じて複数のプロトコルを使い分けて，相手や場面に対して適切な行動を行うための課題に取り組んだ．

この課題を解決するために，メタレベルアーキテクチャを用いてシナリオの実行制御を行うというアプローチをとった．エージェントのシナリオ実行プロセスを，外界とのインタラクションを行う実行レイヤと，シナリオの実行制御を行う制御レイヤの 2 つの層に分けた．こうすることで，通常時のインタラクションは実行レイヤのみで行い，シナリオの追加や切り替えといった処理が必要なときにだけ制御レイヤに制御を移すということを可能にした．このアーキテクチャにより，以下の 2 つの機能を実現した．

1. 協調動作を行うためのシナリオ修正
   シミュレーションの実行中に新たにシナリオを獲得すると，制御レイヤに処理を移し，協調シナリオとのメッセージの受け渡しの仕方や命令に対する対応の仕方といった事前・事後処理を行う記述をそのシナリオに書き加えてからエージェントに割り当てるようにした．こうすることで，シミュレーション中に新たにシナリオを獲得し，既存のシナリオとの間で協調して実行することを可能にした．
2. シナリオの協調方針の切り替え
   協調動作を行う際のシナリオ間のメッセージの受け渡し方法や，シナリオ実行の調整方法などの方針を協調ポリシーとして定めた．そして，状況に応じて協調ポリシーを切り替えることにより，各シナリオの動作の優先順位などを動的に変更することを可能にした．こうすることで，エージェントが持つ複数のプロトコルを使い分けて，周囲の環境や相手に対して適切な行動をとることを可能にした．

最後に，このシステムをシナリオ記述言語 Q とその母体言語である Scheme を用いて実装し，交通シミュレーションおよび対話エージェントを用いたシミュレーションを行った．そのなかで，シナリオの修正や協調ポリシーの切り替えを行い，複数のシナリオ間の協調動作が実現されることを確認した．これにより，提案したアーキテクチャによって，シミュレーション中のシナリオの追加や変更，状況に応じたプロトコルの使い分けが実現されることが示された．

しかしながら，本研究では，1 体のエージェントに複数のシナリオを与えたときの，エージェント内のシナリオの協調実行に着目したため，エージェント間の協調動作については考慮していない．したがって，多数のエージェントが動作する際に，他のエージェントとの競合を回避して協調して動作を行うための手法について，今後考えていきたい．

謝辞

本研究を行うにあたり，熱心なご指導を頂きました石田亨教授に深謝致します．また，日頃より多くの助言や議論をしてくださいました，松原繁夫助教授をはじめとする石田研究室の皆様に感謝致します．

参考文献

[1] Doniec, A., Mandiau, R., Espie, S. and Piechowiak, S.: Dealing with multi-agent coordination by anticipation: application to the traffic simulation at junctions, *European Modeling Simulation Symposium (EMSS'2005)* (2005).

[2] Espie, S., Saad, F. and Schnetler, B.: Microscopic traffic simulation and driver behavior modeling: the ARCHISIM, *Proceedings of the Strategic Highway Research Program and Traffic Safety on Two Continents* (1994).

[3] Ishida, T.: Q: A Scenario Description Language for Interactive Agents, *IEEE Computer*, Vol. 35, No. 11, pp. 42–47 (2002).

[4] Murakami, Y., Sugimoto, Y. and Ishida, T.: Modeling Human Behavior for Virtual Training Systems, *The 20th National Conference on Artificial Intelligence (AAAI-05)*, pp. 127–132 (2005).

[5] 山根昇平, 石田亨: 大規模マルチエージェントシステムのためのメタレベル制御機構, 情報処理学会論文誌, Vol. 47, No. 5, pp. 1363–1370 (2006).

[6] 田仲理恵, 中西英之, 石田亨: 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, 合同エージェントワークショップ&シンポジウム 2006 (JAWS-2006) (2006).