

# 特別研究報告書

マルチエージェントシミュレーションにおける  
エージェントモデルとインタラクションモデルの統合

指導教員 石田 亨 教授

京都大学工学部情報学科

小西 信次

平成17年2月10日

# マルチエージェントシミュレーションにおける エージェントモデルとインタラクションモデルの統合

小西 信次

## 内容梗概

今日、問題となっている交通流や避難行動などに対する検証や対策として、マルチエージェントシミュレーションが用いられている。そのようなシミュレーションにおいて、より現実に近い結果を得るためには、より多く、より多様なエージェントが参加していることが望まれる。その際、どのようにエージェントを制御するのが重要になる。従来方法としてはエージェント間のインタラクションを記述し制御する方法がある。この方法では、エージェント、もしくはエージェント群にプロトコルを与えることでその動作を制御している。

既存の方法では、多様なエージェントを再現するにあたり、以下のような問題が存在する。

- エージェントの多様化に伴うプロトコルの可読性の低下  
エージェントは、プロトコルを解釈して、対応したパターンの行動を行う。そのため、多様で複雑な動きを行わせるには、対応する数の動作の記述をしなければならず、プロトコルの記述が長くなりやすい。そのため重要なインタラクションが見えづらくなり、プロトコルの可読性が落ちる。
- プロトコルのシミュレーション環境への依存  
「中央階段の下にいれば上る」「北改札口の中にいれば外に出る」といった、シミュレーションの環境に依存したインタラクションがある場合、環境に依存した記述をプロトコルに書かなければならない。そのため、プロトコルが一般性を持たなくなるので、他のシミュレーションへの応用時にプロトコルの修正が必要となり、さまざまなシミュレーションを設計する際に障害となってくる。

これらの問題を解決するため、本研究では、インタラクションモデルとエージェントモデルの統合モデルによるエージェントの制御という方法を考えた。この統合モデルでは、これまで別々のモデルとして考えてきた、行動の制約もしくは指針となるプロトコルを表現するインタラクションモデルと、エージェントの内部メカニズムである行動ルールを表現するエージェントモデルの2つを統合した。これらを統合したのは、現在単一のプロトコルに対して単一の行動

を起こすエージェントではなく、単一のプロトコルに対して複数の行動を起こすエージェントを設計する必要があることと、修正が難しいエージェントシステムに組み込まずに実現するために必要だからである。例えば、「歩く」というプロトコルに対して、現在ならば「歩く」しか行わないが、統合モデルでは「早足で歩く」「ふらふらしながら歩く」「歩かない」など複数の判断をなす機会を与えられる。さらに、このモデルにおける以下の特徴は前述の問題を解決することができる。

- エージェントモデルで解釈することによる多様な動作の実現

エージェントは、同一のプロトコルに対して、エージェントモデルにある自らの行動ルールやそれまでに得た情報に従ってプロトコルを解釈することにより、多様な解釈を行い行動するようになる。また、これによって、より現実感のあるマルチエージェントシミュレーションを少ないインタラクションで記述することができ、プロトコルの可読性が向上する。

- プロトコルの一般性という観点をもった設計

設計者は、2つのモデルを客観的に区別しながら設計するための基準として、「プロトコルはエージェントシミュレーションの環境によらず、一般性をもった存在である」という観点に従い設計する。シミュレーションの環境に依存したインタラクションは、エージェントモデルとして記述される。また、プロトコルから環境に依存した記述がなくなるため、シミュレーション環境によらない一般性をもったプロトコルの記述が可能となる。

本研究では、上記の特徴が問題を解決していることを確認するため、この統合モデルを次のように実装した。まず、インタラクションモデルの記述には従来のメッセージ通信以外のインタラクションも記述することのできるシナリオ記述言語 Q を用いた。一方、エージェントモデルの記述にはルールベースであることからプロダクションシステムを用いた。そして、エージェントの環境としてはマルチユーザ・マルチエージェントシミュレーションが可能な仮想都市空間 FreeWalk を用いた。またシミュレーションのシナリオは、地下鉄京都駅を舞台とした避難行動のシミュレーションを想定することにした。そして、シナリオのみですべてを記述した場合と、シナリオとプロダクションシステムを用いた場合の2通りを設計した。なお、プロダクションシステムにおける行動ルールは他の研究成果や文献などから得て設計した。

## Integration of Interaction and Agent Model in Multi-Agent Simulation

Shinji KONISHI

### Abstract

The Multi-Agent simulation is used as a method to verify or work out counter-measures for traffic flow problem and action in evacuation, that become problems today. In such simulations, it is hoped that there are more and more various agents to obtain a result more near the reality. To realize it, it is important how to control agents. As a past method, there is a method to control agents with a description of the interaction among agents. In this method, the agent or the agent group are controlled by giving the protocol.

To reproduce various agents, the following problems exist in an usual method.

- Protocol depends on simulation environment

It is necessary to write the description according to the environment in the protocol when there is an interaction that depends on the environment of the simulation, for instance, "Go up in case of being under the central stairs" and "Go out in case of being in the ticket gate". Therefore, when the correction of the protocol is needed when applying it to other simulations, and designing various simulations, it becomes a trouble because the protocol doesn't have generality.

- To be hard to make out protocol because of the agent's diversification

The agent does the action of the corresponding pattern interpreting the protocol. Therefore, it is necessary to describe the action of the corresponding number to make various, complex movements done. So the description of the protocol becomes long easily. Therefore, it becomes difficult to see an important interaction, and to become hard to make out protocol.

To solve these problems, I devised the method of controlling agents by integration the interaction model and the agent model in this research. For this integrated model, I integrated two models, the agent models, which consist of the action rule that was an internal mechanism of the agent, and interaction model, which expresses the protocol that become the restriction or the indicator of the action. There are two reasons to use integrated model. One is to de-

sign the agent to behave two or more judgments for a single protocol. Another is the necessity to achieve various agents without building agent behavior in the agent system with a difficult correction. For instance, the agents do only "Walk" to the protocol now, but with integrated model, two or more judgments such as "Walk fast", "Walk while dizzying", and "Do not walk" will be done. In addition, the following features in this model can solve the above-mentioned problem.

- Design with viewpoint of generality of protocol

The designer designs protocol according to the viewpoint "The protocol is existence with generality it doesn't depend on the environment of the agent simulation" as a standard to design while objectively distinguishing two models. The interaction that depends on the environment of the simulation is described as an agent model. Moreover, because the description that depends on the environment is removed from protocol, the designer becomes able to design the protocol with the generality that doesn't depend on the simulation environment.

- Achievement of various operations by interpreting protocol by agent model
- In integrated model, agents do various action for the same protocol by various interpretation the protocol according to an own action rule that is in their agent model and information obtained till then. Moreover, the multi-agent simulation with a more real feeling can be described by few interactions, and, as a result, it becomes easy to make out protocol.

I mounted this integrated model as follows. First, scenario description language Q that is able to describe interactions other than a past message communication was used for the interaction model's description. Next, the production system that is based the action rule was used for the agent model's description. And, virtual city space FreeWalk that was able to be the multi-user multi-agent simulation was used as agent's environment. Moreover, the shelter simulation in the subway Kyoto Station was done as a scenario. And, I designed two kinds of scenario. One was used the scenario and the production system, and another was described only by the scenario. The action rule in the production system was obtained from other results of the research and documents.

It was confirmed that various agents were able to participate in the simulation by integrating the interaction model and the agent model in the multi agent simulation through the design of this system. Moreover, it came to be able to say the designed protocol being able to lose the description that depended from the protocol to the environment of the simulation, and the possession of generality or more.

# マルチエージェントシミュレーションにおける エージェントモデルとインタラクションモデルの統合

## 目次

第1章	はじめに	1
第2章	研究背景	3
2.1	エージェント制御手法	3
2.2	既存のプロトコル記述手法	3
2.2.1	FIPA-ACL	3
2.2.2	シナリオ記述言語 Q	4
2.3	既存の制御手法における問題	5
第3章	インタラクションモデルとエージェントモデルの統合	5
3.1	統合モデルの必要性	5
3.2	統合モデルの構築方法	8
3.2.1	それぞれのモデルの区別	8
3.2.2	それぞれのモデルの接続方法	9
3.3	統合モデルの実装	10
第4章	シミュレーションシナリオの記述	12
4.1	シミュレーション内容	12
4.2	インタラクションの分類	13
4.3	インタラクションモデルの設計	13
4.3.1	避難者エージェント	14
4.3.2	吸着誘導者エージェント	14
4.3.3	指差誘導者エージェント	15
4.3.4	管理者エージェント	15
4.4	エージェントモデルの設計	15
4.5	モデルの記述	17
第5章	モデルの統合の結果と考察	21
5.1	マルチエージェントシミュレーションの動作	21
第6章	おわりに	21

謝辞	24
参考文献	24
付録：マルチエージェントシミュレーション	A-1
A.1 エージェントモデルを統合した場合のプロトコル . . . . .	A-1
A.2 エージェントモデルの行動ルール . . . . .	A-10
A.3 エージェントモデルを想定していない場合のプロトコル . . .	A-58

## 第1章 はじめに

今日、問題となっている交通流や避難行動などに対する検証や対策として、マルチエージェントシミュレーションが用いられている。これにより、現実空間で再現することが難しい大人数が参加しているシミュレーションや、一時的な占有も難しい公共施設の仮想空間上でのシミュレーションが行われるようになった。そのように用いられているマルチエージェントシミュレーションにおいて、より現実に近い結果を得るためには、より多くの、より多様なエージェントが参加していることが望まれる。これは、現実世界での人間は、そのようにさまざまな個性をもった個体であるということ、現在のシミュレーションの中でのエージェントの数は現実で問題としている規模よりも小さく現実の結果と対応しているとは考えにくいことが挙げられる。

多様なエージェントを再現するにあたり、どのようにエージェントを制御するのが重要になる。従来の方法としてはエージェント間のインタラクションを記述し制御する方法がある。たとえば、エージェントの国際標準化団体 FIPA で規定されている FIPA-ACL[1]、エージェント間でやり取りされるメッセージフローに焦点を当てた AgentUML[2] のプロトコル図、拡張状態遷移機械に基づいた AgenTalk[3] があげられる。いずれも、あるプロトコルを与えることにより、エージェント、もしくはエージェント群の動作の制御をしている。

より現実感を得るために、シミュレーション上における多様なエージェントの再現を実現するにあたり、既存の方法では、以下のような問題が存在する。

- エージェントの多様化に伴うプロトコルの可読性の低下  
エージェントは、プロトコルを解釈して、対応したパターンの行動を行う。そのため、多様で複雑な動きを行わせるには、対応する数の動作の記述をしなければならず、プロトコルの記述が長くなりやすくなる。ため重要なインタラクションが見えづらくなり、プロトコルの可読性が落ちる。
- プロトコルのシミュレーション環境への依存  
「中央階段の下にいれば上る」「北改札口の中にいれば外に出る」といった、シミュレーションの環境に依存したインタラクションがある場合、どうしても環境に依存した記述をプロトコルに書かなければならない。そのため、プロトコルが一般性を持つものではなくなるので、他のシミュレーション環境への応用時にプロトコルの修正が必要となり、さまざまなシミュレー

ションを設計する際に障害となってくる。

これらの問題を解決するためには、現在のようにエージェントがプロトコルをそのままの意味で解釈し行動するのではなく、エージェントがプロトコルを自分が持つ情報に従って解釈し行動する必要がある。しかし、現在のプロトコルとエージェントシステムの対応しか考えていない状況では、プロトコルが複雑となるか、もしくはエージェントシステムが複雑となる。

しかし、エージェントシステムを複雑に設計していく場合、一般的にプロトコルよりも大きなシステムを修正していくことになる。そのような状況では、プロトコルやシミュレーションの条件に応じた柔軟な修正を行うことが困難であると予想できる。また、他のシミュレーションでも用いることを可能とするためにも特定のシミュレーションのために設計することはよくない。そのため、エージェントシステムはプロトコルに対して従順であることが望まれる。また、プロトコルを複雑化することも上記で述べたように問題である。

このため、多様なエージェントを表現するためには、プロトコルとエージェントシステムの対応だけでは不十分であることがわかる。

そこで、本研究では、インタラクションモデルとエージェントモデルの統合によるエージェントの制御という方法を考える。この統合モデルでは、これまで別々のモデルとして考えてきた、行動の制約もしくは指針となるプロトコルを表現しているインタラクションモデルと、そのプロトコルをエージェントがどのように解釈するかを表現しているエージェントモデルの2つを統合した。これらを統合したのは、現在単一のプロトコルに対して単一の判断により行動しているエージェントではなく、複数の判断を下し行動するエージェントを設計する必要があるからである。また、修正が難しいエージェントシステムに組み込まずに実現するために必要であるからである。例えば、「歩く」というプロトコルに対して、現在ならば「歩く」しか行わないが、統合モデルでは、「早足で歩く」「ふらふらしながら歩く」「歩かない」など複数の判断をエージェントが行える。

この統合モデルによって前述の問題を解決することができることを示していく。

## 第2章 研究背景

### 2.1 エージェント制御手法

エージェントシミュレーションにおいて，エージェントを制御するための記述方法には，エージェントの内部メカニズムだけを記述する方法と，各エージェント間のインタラクションなどを記述することでエージェントの行動を記述する方法がある．

前者では，各エージェントは，自分の内部に持つ行動ルールと周りから得た情報をもとにして行動する．このようなモデルとしては，プロダクションシステムや様相論理などが挙げられる．これらのエージェントモデルはエージェント内部のメカニズムを表現するためのモデルである．しかし，多数のエージェントが求められるマルチエージェントシミュレーションにおいて，膨大な内部処理を必要とするエージェントを実装するには現実的な方法ではない．

次に，後者のエージェント間のインタラクションを記述する方法では，エージェント間のメッセージのやり取りや，その他のエージェントが行うことのできるインタラクションやエージェント間のプロトコルを記述しマルチエージェントシミュレーションを設計する．エージェントの内部メカニズムを詳細に記述する手法からエージェント間のインタラクションを記述する手法への移行により，エージェントには依存しないエージェントシミュレーションの設計ができるようになる．そのため，この方法を用いれば，マルチエージェントシミュレーションをより容易に設計することが可能となり，マルチエージェントシミュレーションの設計に有効であると考えられる．

### 2.2 既存のプロトコル記述手法

マルチエージェントシミュレーションにおけるプロトコルを記述するための既存の言語としては，エージェントの国際標準化団体 FIPA で規定されている FIPA-ACL，エージェント間でやり取りされるメッセージフローに焦点を当てた，AgentUML のプロトコル図，拡張状態遷移機械である Q[4] などが挙げられる．

#### 2.2.1 FIPA-ACL

ACL(AgentCommunicationLanguage) とは，エージェント間の会話に用いられる個々のメッセージを記述する言語を用いて，エージェント間のプロトコル

を表現する方法である。FIPA-ACLはFIPA(Foundation for Intelligent Physical Agents)が標準化したもので、その仕様書では、バイナリ形式、ストリング形式、XML形式が定義されている。

FIPA エージェント間でやりとりされるメッセージは、CORBA や RMI など他の分散オブジェクト指向アーキテクチャでのメッセージ送信とは少し異なる。CORBA や RMI でのメッセージ送信がリモート・オブジェクトのメソッド呼び出しを強く意識しているのに対し、FIPA エージェント間でやり取りされるメッセージは人間同士がやり取りする電子メールに近い概念で行われている。そのため、ACL メッセージの構造も普段使い慣れている電子メールの構造と良く似た形式になっている。

エージェントは、プロトコルに従順であることが望まれている。そのため、多様なエージェントを表現するためには、複雑なプロトコルが必要となる。

### 2.2.2 シナリオ記述言語 Q

シナリオ記述言語 Q は、エージェント間の社会的インタラクションを記述することが可能な言語であり、拡張状態遷移機械を用いている。この言語は、従来から用いられているメッセージ通信でのインタラクションだけでなく、シミュレーション環境を用いた「Aさんがこっちにむかって歩いている」のようなさまざまな社会的インタラクションも記述することが可能となっている。また、Q では、プロトコルにより制御されているエージェントを、人間が制御することのできるアバターへ簡単に換えることができるという特徴もある。この特徴は、マルチエージェントシミュレーションの中の一部のエージェントを人間が操作するアバターへ換えて参加型シミュレーションを実施することを、容易にする。

また、Q 言語自体は、プロトコルを記述することが定義されているだけでなく、Scheme や JAVA, C++ などさまざまなプラットフォームに対応可能である。また、コネクタを用意することで、従来のエージェントシステムに対しても外付けという形でエージェントの動作を記述することができる。

エージェントに対しては、プロトコルにおけるイベントの観測と行動を「依頼」という形で伝える。Q 言語ではエージェントがその依頼に対して従うことは求められていない。そのため、エージェントがその依頼に対してどのように行動するかは、そのエージェントシステムの実装による。そのため、FIPA-ACL や AgenTalk のようにプロトコルに対して従順なエージェントシステムを必要とはしない。

## 2.3 既存の制御手法における問題

上記で述べたそれぞれの制御手法では，エージェント間のインタラクションに注目することで，エージェントの複雑な内部メカニズムの記述を少なく，もしくはなくそうとしている．そして，効率よく多数のエージェントを制御できるようにしている．

しかし，既存の制御手法には以下のような問題が存在する．

まず，本研究での最終目標としているエージェントの多様化に伴った，プロトコルの煩雑化と可読性の低下である．既存の制御手法においては，エージェントシステムでのエージェントがプロトコルに従順であることを仮定している．そのため，エージェントが多様化するためには，プロトコルも多様化する必要が出てくる．しかし，そのようにした場合，元々表現したかったインタラクションと，エージェントを多様化するために追加されたインタラクションが，混在してしまうため，プロトコルの可読性が低下しやすい．

次に，既存の制御手法の記述にはエージェント間のインタラクションだけでなく，そこで問題としているシミュレーションの環境に強く依存した記述が混入してしまうことである．これにより，一般性を持ったプロトコルが設計できなくなる．そのため，一般的なプロトコルを証明するためにシミュレーションを行ったとしても，その結果得られたプロトコルはそのシミュレーションに依存したものであると結論付けられうる，ということになる．

これらの問題を解決するため，本研究では，次章以降で述べるインタラクションモデルとエージェントモデルの統合によるエージェントの制御という方法を考える．

## 第3章 インタラクションモデルとエージェントモデルの統合

### 3.1 統合モデルの必要性

2.3. の1つ目の問題が起きている原因は，マルチエージェントシミュレーションにおいて，プロトコル記述言語では，エージェントがプロトコルに従順であることを仮定していることが多いからである．このように従順であるエージェントシステムを用いれば，プロトコルによってその動作を記述しやすく，柔軟な設計が可能となる．また，エージェントシステムがプロトコルに従順でなかつ

た場合、プロトコルに対応するエージェントの動きはエージェントシステムに依存し、それがどうなるのかはプロトコル設計者が把握できないため、マルチエージェントシミュレーションの設計には向いていないと考える。しかし、エージェントが従順であるためには、表現したいインタラクション以外の細かな動作をエージェントにさせるために相応の記述が必要となっていた。

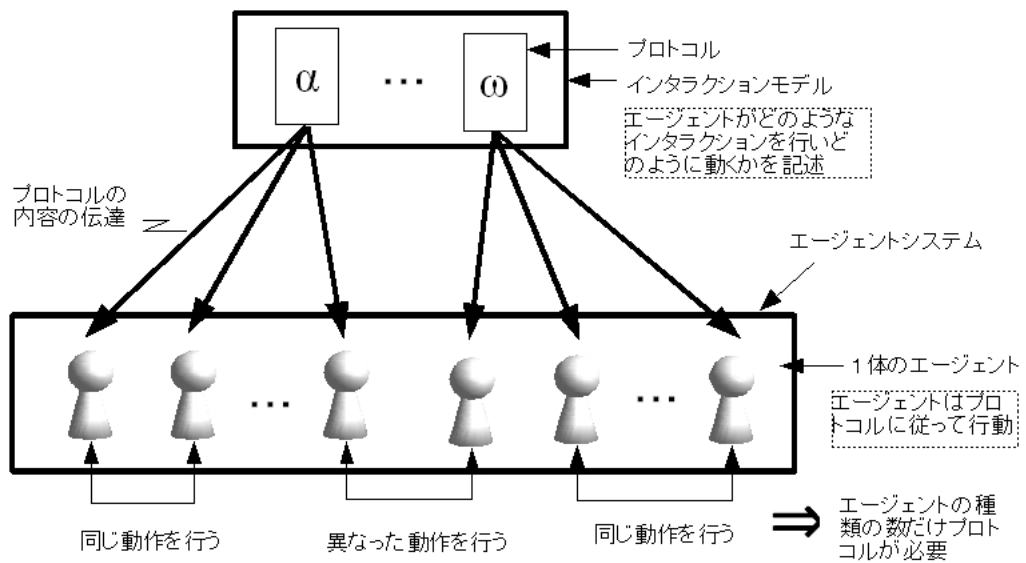
また、2つ目の問題は、既存の制御のモデルでは、プロトコルとエージェントシステムしか考えていないことにある。2つのモデルしか考えていないため「中央階段の下に来れば中央階段を上る」や「出口 A の近くに来たら出て行く」といったシミュレーションの環境に依存した記述は2つのモデルのどちらかに記述されることになる。もし、エージェントシステムの中に記述する場合、そのような行動を起こすように設計されることになる。しかし、一般的にエージェントを実装するシステムは、インタラクションモデルを実装するシステムよりも巨大なシステムである。そのようなシステムの中に1つのシミュレーションでしか利用できないインタラクションを記述するというのは明らかに効率が悪い。仮に1つのシミュレーションに特化してシステムを作るとしても、細かく柔軟な修正をすることが難しく、労力が必要となる。また、エージェントシステムの設計者とプロトコルの設計者が異なっている場合、プロトコル設計者はエージェントシステムの内部には関知していない。そのため、エージェントシステムの内部を設計することは、プロトコル設計者にとっては困難であり、大変な労力が必要となる。

そこで、従来のインタラクションモデルとエージェントシステムの関係に、エージェントモデルを統合する方法を考える。

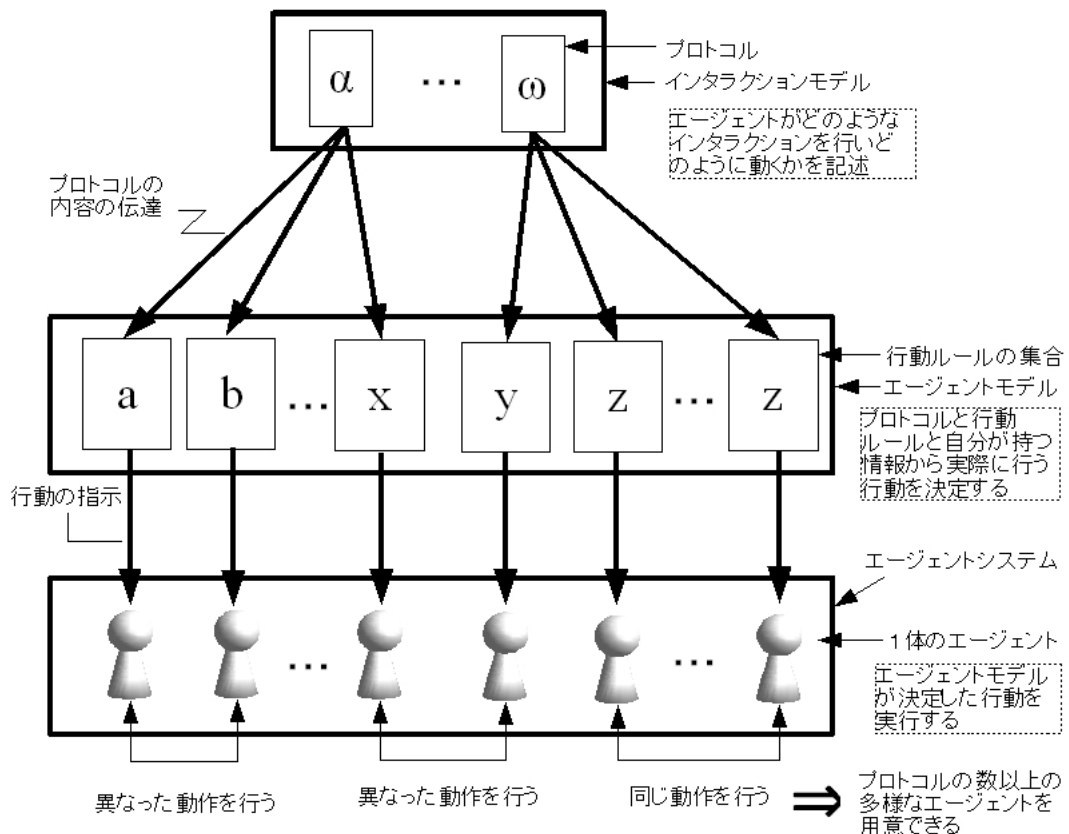
ここで述べたインタラクションモデルとは、それぞれのエージェントとエージェントもしくはエージェント群との間におけるインタラクションを記述し、エージェントがどのように振舞うかというプロトコルを表現するモデルである。また、エージェントモデルとは、与えられたプロトコルと行動ルール、周囲から得た情報をもとにして、実際にどのように振舞うかの判断・決定を表現したモデルである。

2つのモデルを統合して用いる事により、先の問題に対して、シミュレーションの環境に依存したインタラクションを記述する場を与えることができる。

また、このエージェントモデルを統合することによって、プロトコルを現在より柔軟に解釈し行動することが可能となる。そのため、エージェントシステ



(a) インタクションモデルとエージェントシステムを用いる既存のモデル



(b) エージェントモデルを考慮した統合モデル

図 1: マルチエージェントシステムのモデル図

ムでのエージェントがプロトコルに従順であると仮定しているとき、次ページの図1のように、既存のプロトコルに対して、エージェントシステムのみを接続していたときにはプロトコルに従った行動しかしていなかったエージェントが、エージェントモデルを統合することでプロトコルとは異なった行動を起こすことが可能となる。このことは、本研究の最終目標である多様なエージェントの再現に対して有効な手段であることを示している。

また、この統合モデルを、現実空間における被験者を用いたシミュレーションで考えた場合、インタラクションモデル、エージェントモデル、エージェントシステムの関係は、被験者に与える制約、被験者の意思決定、被験者の身体やシミュレーションの空間などの環境と対応付けができる。よって、この統合モデルは既存のインタラクションモデルとエージェントシステムのみを考えるモデルよりも現実世界に近く優れていると考えられる。

## 3.2 統合モデルの構築方法

この統合モデルの構築方法について考える。

### 3.2.1 それぞれのモデルの区別

統合モデルを設計していくにあたり、インタラクションモデルとエージェントモデルそれぞれをどこまで設計するかという問題がある。

例として、シミュレーションの内容に「中央階段の下まで来たら中央階段を上る」「前方に人がいれば歩行速度を落とす」「避難誘導灯が見えたらそこまで移動する」があった場合を考える。まずインタラクションモデルではそれらすべてを記述し、エージェントモデルでは「プロトコルに従い行動する」とだけ記述した場合、シミュレーションの結果としては正しいものが得られる。また、インタラクションモデルでは「歩く」とだけ記述し、エージェントモデルでは「歩く」というプロトコルに対して実際にどのように行動するのかという行動ルールを記述した場合も、シミュレーションの結果としては正しいものが得られる。

上記のような状態が生じるため、この統合モデルを設計するにあたり、インタラクションモデルとエージェントモデルを設計していくための客観的な基準が重要となる。そこで「インタラクションがシミュレーションの環境に依存したものであるかどうか」という判断を、それぞれのモデルの区別の基準の1つとすることにした。この基準を用いる理由は、この判断には主観が入らない、と

いう利点に加え，この基準をもって設計されたインタラクションモデルにはシミュレーションの環境に依存したインタラクションが存在しないことが言えるようになる，という利点が挙げられる．特に後者の利点は，プロトコルが一般性を持っている事の証明にもなるため，シミュレーションの有効性を示す判断材料を提供することができると思う．

次に，もう1つの基準として，表現したいインタラクションの粒度が挙げられる．これは，それぞれのモデルで表現しようとしているインタラクションの粒度が異なることを用いたものである．エージェント間のインタラクションなどの記述と解釈・決定に関する記述はそれぞれエージェントシステムを構成するための記述ではある．しかし，前者は多数のエージェントについて考えているため，粒度が大きいものとなる．それに対して，後者はエージェントの内部でのどのようにするかというメカニズムを考えているため，粒度は小さいものとなる．このことから，インタラクションモデルとエージェントモデルは粒度の異なったモデルであることがわかる．この粒度の差と，表現したいインタラクションの粒度を考え，プロトコルとするか行動ルールとするかを決定し設計をすることになる．

### 3.2.2 それぞれのモデルの接続方法

それぞれのモデルは3.2.1で述べたように，基準により区別することとした．そこでこれらを接続する手段が必要となる．本研究における基本アイデアでは，インタラクションモデルとエージェントシステムとの接続はすでに存在していると考えており，エージェントモデルはその中間に介入する形式になっている．そのため，エージェントモデルに求められるのは，内部メカニズムの表現能力に加えて，インタラクションモデルとエージェントシステム間のやり取りに介入できるシステムが必要となる．

インタラクションモデルとエージェントシステム間の接続には，共有メモリ，TCP/IP，オブジェクトのメソッド呼び出しなど，さまざまな方法が考えられる．どの方法においてもほぼ同じ対応とできる統合方法のアイデアとしては，どちらかのシステムにおける，システム間の接続に関する既存のコードを変更し，送出と読み出しを行っている部分のエイリアスを作成し，エージェントモデルがそのエイリアスを用いる方法がある．この方法では既存の接続を用いるため，容易に統合を行うことができると考える．

### 3.3 統合モデルの実装

本研究における統合モデルの実装について説明する．

インタラクションモデルの記述には，インタラクション記述言語である Q 言語を用いている．

エージェントモデルの記述には，ルールベースであるプロダクションシステムを用いる．

エージェントの環境としては，マルチユーザマルチエージェントシミュレーションが可能な仮想都市空間 FreeWalk[5] を用いて，上記のことを確かめることにした．

これらの実装は，京都大学石田研究室に所属する村上が実装 [6] しており，このエージェントアーキテクチャを，本研究では利用することにする．

このエージェントアーキテクチャでは，Q インタプリタ，プロダクションシステム，FreeWalk それぞれの接続は，Q と FreeWalk 間の既存のメッセージハンドラを利用し，この通信にプロダクションシステムが介入するようになっている．

まず，Q インタプリタからの依頼は，メッセージハンドラを經由してワーキングメモリ (WM) にワーキングメモリエレメント (WME) として追加され蓄えられる．またイベントの観測により得た外界の情報も，同様に WME として WM に蓄積される．このようにプロトコルによる行為の制約と外界の情報が WM に追加されることで，刻々と変化するエージェントの状況は WM で管理される．

エージェントは，状況に適した行為を決定するために，発火可能な行動ルールがプロダクションメモリ (PM) 内に存在しないか，ワーキングメモリ内で条件照合を行う．発火可能な行動ルールが複数存在すれば，競合解決を行い一つの行動ルールを選択する．なお，各行動ルールには優先度を割り当てることができる．競合解決では，その優先度も考慮する MEA (means-end analysis) 戦略が用いられる．この方法では行動ルールに優先度を割り当てることによって，プロトコルを優先するエージェント，またはその逆に，プロトコルを重視しないエージェントを意図的に作成することが可能になる．

最後に競合解決によって選択されたルールに従い，WM の更新やセンサ，アクチュエータの起動を行う．プロダクションシステムには存在しないセンサとアクチュエータの起動を行うために，プロダクションルールの R H S には以下

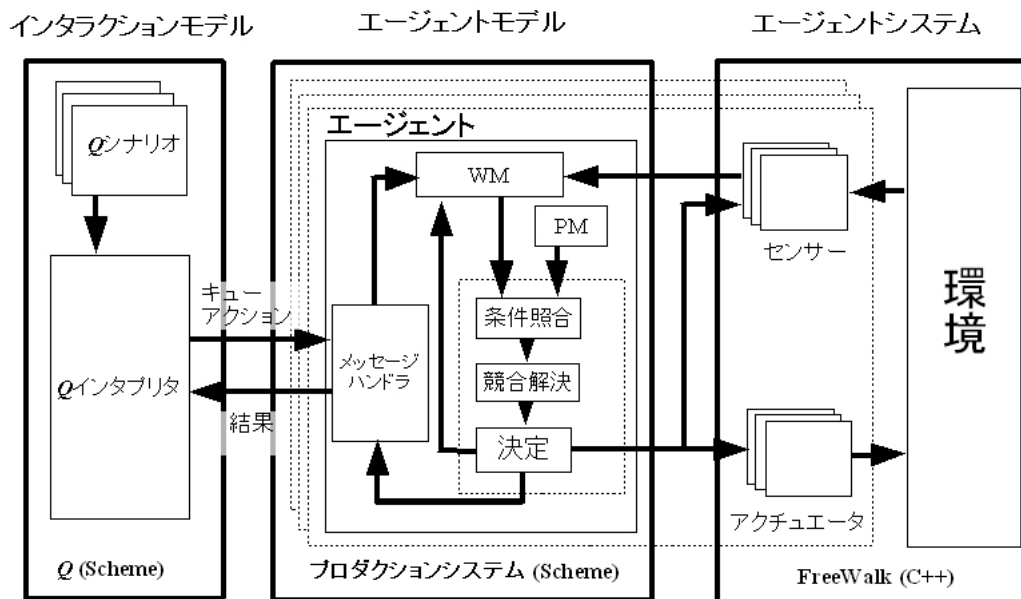


図 2: 統合モデルの実装 [6]

の 4 点の記法が追加されている .

- (!activate :cue ...)

cue パラメータで指定されたイベントの観測を開始する .

- (!test-func :cue ...)

cue パラメータで指定されたイベントが起こっているかどうかを確認する . 起こっていれば (true :cue ...), 起こっていなければ (false :cue ...) という WME を WM に追加する .

- (!follow :request ...)

request パラメータで指定された, *Q* インタプリタからの依頼に従って, センサやアクチュエータを起動する . 具体的に, request パラメータには LHS でバインドされた依頼の WME が指定される .

- (!起動するアクチュエータ名)

指定されたアクチュエータを起動する .

これらのセンサやアクチュエータの起動では, 既存の *Q*/FreeWalk 間で用いられていたコールバック関数が呼び出されている .

## 第4章 シミュレーションシナリオの記述

ここまで述べたことを確認するために、実際にマルチエージェントシミュレーションを設計することにする。

### 4.1 シミュレーション内容

近年、災害や事故に対する事前評価や分析が大いに必要とされている。しかし、災害や事故は、人間の数や現実空間での制約など、通常時での再現が難しく、実験が困難な問題である。そのため、仮想空間を用いたマルチエージェントシミュレーションを用いることで、そのような再現することが困難な問題領域に対しての貢献が期待できる。

そこで、今回はすでに仮想空間が用意されている地下鉄京都駅での避難訓練シミュレーションを設計の対象とする。図3に地下鉄京都駅の改札階、およびプラットフォーム階の地図を示す。問題を簡略化するため、図3の網掛け部分に相当する空間での、エージェントの移動のみを考慮することとする。この空間は、現実空間において人間がどこにいるかという情報を得るためのカメラが設置されている空間であり、今回は試していないが、今後参加型シミュレーションとしての評価を行う際にその有効性を示しやすいためである。

シナリオとしては、地下鉄において地下鉄京都駅のプラットフォームにいるエージェントが避難完了とする出口Aまでいかに避難するかをシミュレートする。ただし、出口BとCは誤った出口であると仮定しておく。このシミュレーションでは避難者全員を正しい出口Aに誘導させることが目標となる。

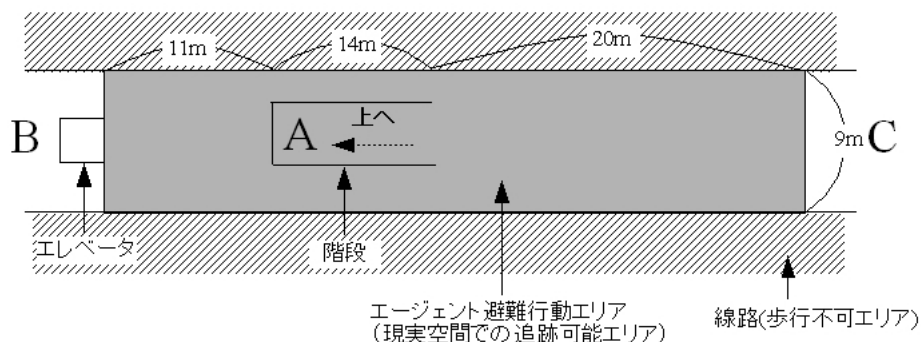


図3: 仮想地下鉄京都駅空間

エージェントは避難者エージェントと誘導者エージェントを用意する。

誘導者エージェントは避難者エージェントを誘導し避難させる。誘導法としては、杉万教授により実施された統制実験 [7] でも用いられている指差誘導法、吸着誘導法を行う。指差誘導法は誘導者が大声で誘導法の指示を行いながら出口を指差し、最終的に出口に向かう方法である。吸着誘導法は、誘導者が身近にいる避難者だけに自分についてくるように指示し、出口に向かう方法である。

避難者エージェントは、避難訓練開始の合図があれば、近くにいる誘導者の指示に従って避難する。しかし、周りに誘導者がいない、または指示が何もなければ、近くの出口に向かう。

## 4.2 インタラクションの分類

インタラクションモデルとエージェントモデルの設計では 3.2.1. で述べたインタラクションの分類基準を考慮しながら設計した。

シミュレーション環境に依存したインタラクションの例としては、

- 追従をしていなくて、出口が見えていればそこへ向かう。
- 誘導対象者の姿が見えなければ、その誘導対象者の誘導を諦め、避難経路を進む。

が挙げられる。たとえば、1つ目で観測を行っている出口の座標点は絶対座標であり、今回のシミュレーション特有である。また、避難経路も今回のシミュレーションだけでのみ有効である。よって、このインタラクションはエージェントモデル内で記述する。

次に、インタラクションの粒度による分類の実例としては表 1 が挙げられる。

表 1 で挙げた例で分かるように、プロトコルで用いる語彙の粒度と、行動ルールで用いる語彙の粒度は異なっており、行動ルールで用いる語彙の方が、粒度が細かい。このように表現しようとするインタラクションの粒度によって、インタラクションモデルとエージェントモデルのどちらのモデルに分類するかを決定した。

## 4.3 インタラクションモデルの設計

エージェントのインタラクションモデルを設計する。

インタラクションモデルは、エージェント間のインタラクションがどのようなプロトコルに従って行われるかを表現するモデルなので、この設計ではプロ

表 1: インタラクションの粒度の差

粒度の大きい語彙	粒度の小さい語彙
誘導者に追従する .	追従の依頼が来たとき,自分がまだ誰の後にも追従していなければ,その追従の対象者の姿が見えるかどうかを観測する.追従すべき対象者の姿が見えていればその後ろに追従する .
付いてくるように指示をする .	「付いてきてください」と指示するときに,指示の対象を誘導中として覚えておき,誘導対象者との距離が離れていないかを観測し始めておく .
避難する方向を指示する .	「あちらへ避難してください」と指示をすると同時に,その指示の対象者の方向を見て指示する .
避難完了を確認したというメッセージを伝える	避難完了のエリアにおり,まだ避難完了が確認されていない人がいれば,その人に避難完了を確認したということを伝える .

トコルを示す状態遷移図を設計することとなる .

今回の研究では,避難者エージェント,誘導者エージェント,および避難シミュレーションをコントロールするための管理者エージェントがいる .

それぞれの詳細なシナリオ,およびそのプロトコルを示す状態遷移図は以下のとおりである .ただし,今回のプロトコルでは開始と終了を明確にするために,冗長ではあるが,2つの状態を組み込んでいる .

#### 4.3.1 避難者エージェント

誘導者は,避難訓練の開始の合図とともに行動を開始する .避難者は,誘導者からついてくるように指示があれば,その誘導者についていく .また,進むべき方向を指示された場合もその方向へ移動する .この動作を自分の避難完了が確認されるまで行う .

#### 4.3.2 吸着誘導者エージェント

誘導者は,避難訓練の開始の合図とともに行動を開始する .もし,付近に誘導の対象者がいれば,その対象者に対してついてくるように指示をし,出口まで避難を行う .この動作を自分の避難完了が確認されるまで行う .

また,誘導中に誘導の対象者との距離が開けば,対象者のほうを向き,対象者の姿がまだ見えているのであれば,再びついて来るように指示を行う .もし対象者の姿が見えなければあきらめることとする .

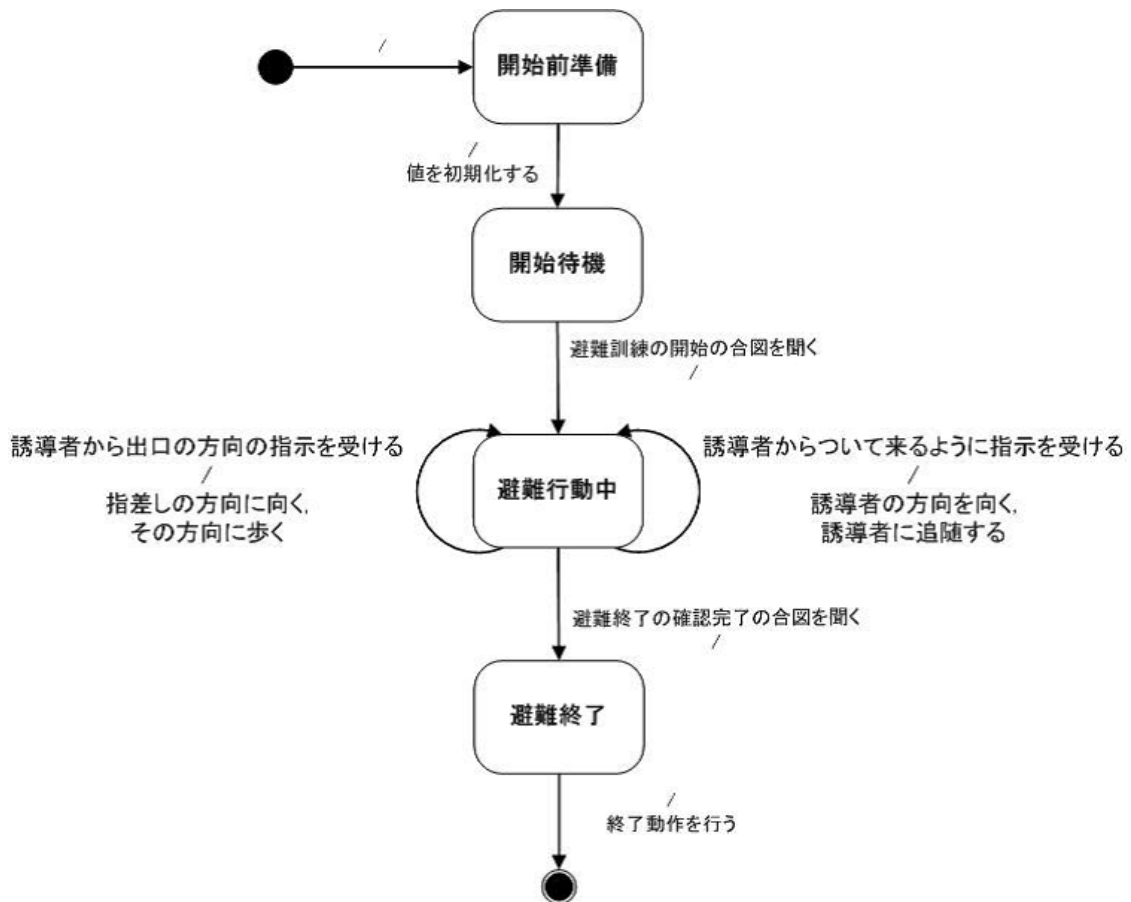


図 4: 避難者エージェントの状態遷移図

そして、誘導対象者との距離がある程度近づけば、避難経路を歩く。

#### 4.3.3 指差誘導者エージェント

誘導者は、避難訓練の開始の合図とともに行動を開始する。もし、付近に誘導の対象者がいれば、その対象者に対して避難すべき方向を指差す。付近にそのような誘導の対象者がいなければ、自分も避難する。この動作を自分の避難完了が確認されるまで行う。

#### 4.3.4 管理者エージェント

管理者は、避難訓練の開始と終了の宣言を行う。出口に来た避難者もしくは誘導者に対して、そこが出口であり避難が完了したことを知らせる。

### 4.4 エージェントモデルの設計

エージェントモデルは、エージェントがプロトコルや自分が手にしている情報からどのように考え行動を決定するか、をモデルにしたものなので、このエー

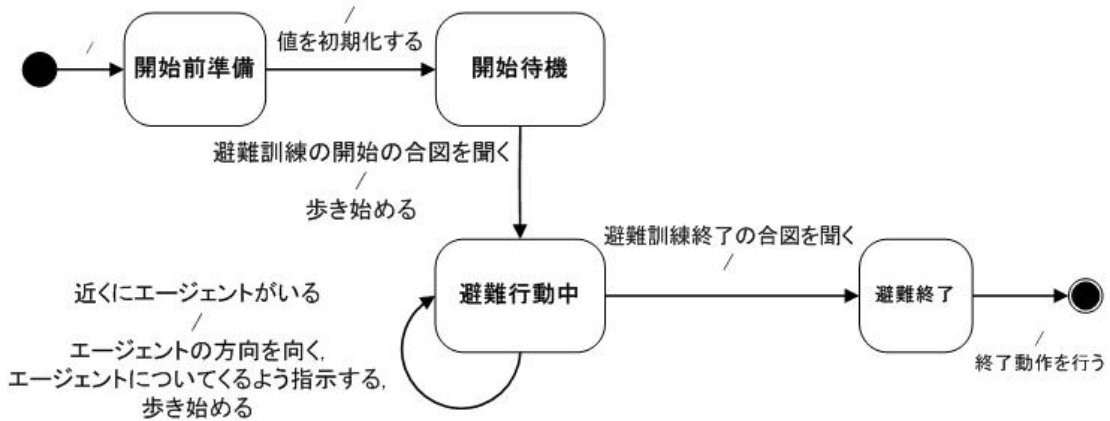


図 5: 吸着誘導者エージェントの状態遷移図

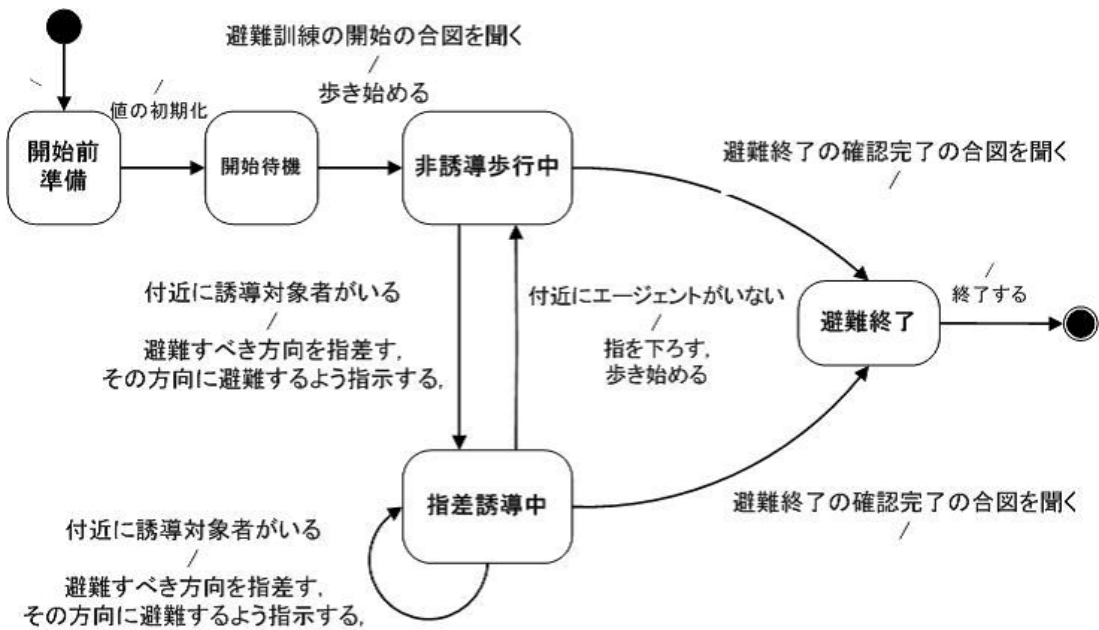


図 6: 指差誘導者エージェントの状態遷移図

エージェントモデルを設計するためには行動ルールを作成していくことになる。

今回のシミュレーションでのエージェントの避難行動および誘導行動における行動ルールは、本来ならば現実空間での京都駅での避難訓練から、分析・評価をして行動ルールを抽出することが好ましい。しかし、現実空間での京都駅でそのような実験を行うことは、大量の人の流れのある公共施設では難しく、現在までにそのようなデータは得られていなかったため、文献などから、基本的な行動ルールを作成した。

表 2 に行動ルールを示す。

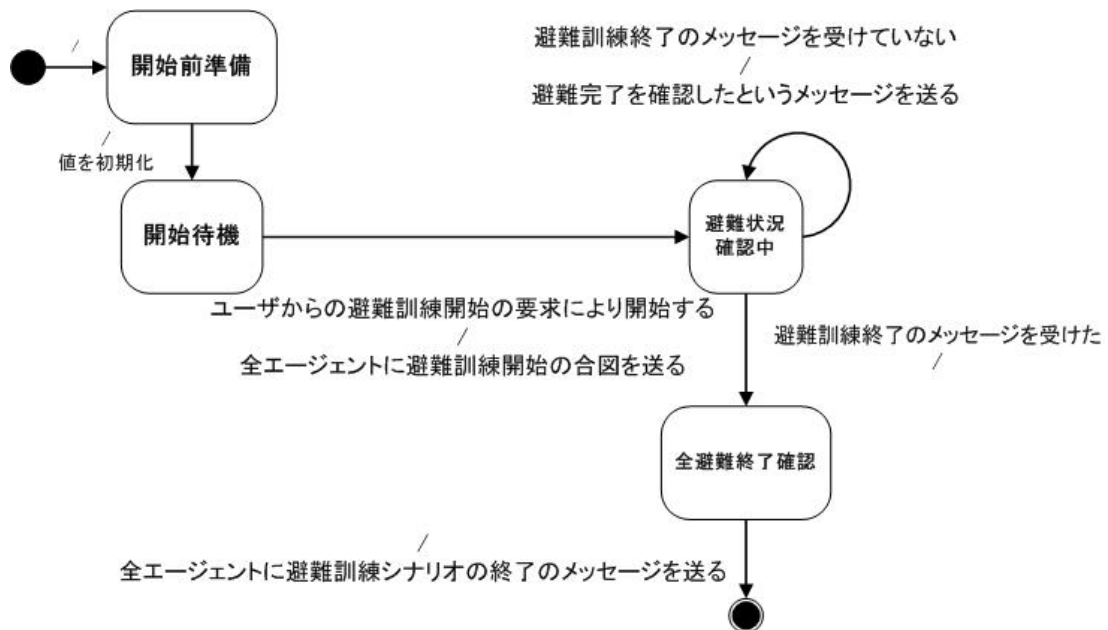


図 7: 管理者エージェントの状態遷移図

ただし、ここでの依頼とは Q インタプリタから渡されるプロトコルにおけるエージェントの次の行動についての依頼メッセージである。

表 2 において、分類基準とは、3.2.1. で述べたプロトコルと区別するための基準である。表 2 に示す行動ルールは、その基準に従ってエージェントモデル内の記述として分類されている。また、プロトコルをそのままの意味で解釈する「プロトコルに従う」行動ルールはどちらの基準でも分類できない特殊なルールのため、分類基準は「その他」とした。

#### 4.5 モデルの記述

ここまでで設計したプロトコルと行動ルールから実際の記述を行った。

プロトコルは、Q の記述法と FreeWalk の仕様に従って記述した。行動ルールは、プロダクションルールの従来の記述法と 3.3. で述べた拡張した記述法を用いて記述した。記述内容の詳細は付録に示す。

簡単な対応を以下に示す。ただし、ここではコードではなく、その内容だけを記述しておく。

- ・避難行動中の状態で、「私についてきてください」という指示を聞いたら、その指示をした人の方向を向き、その人についていく。
- という、プロトコルに対して以下の行動ルールが対応している。

表 2: 避難者の行動ルール

分類基準	行動ルール
シミュレーションの環境に依存したインタラクション	追隨をしていなくて、出口が見えていればそこへ向かう。
	指差しをしていることを観測したら、その指差しの方向を向き、同時にその前方に通路がないかどうかを観測する。
表現するインタラクションの粒度	歩行中、前方にいる人数に応じて歩行速度を変更する。
	歩行中、他のエージェントに接近しすぎていれば、体の向きを変えて移動する。
	追隨の依頼が来たとき、自分がまだ誰の後にも追隨していなければ、その追隨の対象者の姿が見えるかどうかを観測する。
	追隨すべき対象者の姿が見えていればその後ろに追隨する。
	追隨の対象との距離が離れば、その姿が見えていれば、歩行速度を上げて追いかける。
	追隨の対象との距離が離れ、その姿が見えなくなれば、追隨を諦め、あたりを見渡し、追隨する対象となる人がいないかを観測する。
	追隨する対象者の姿が見えなければ、あたりを見渡し、追隨する対象となる人がいないかを観測する。
	追隨の依頼を受けたとき、自分が誰かの後を追隨していれば、現在の追隨をやめ、その追隨の対象者の姿が見えるかどうかを観測する。
	追隨の依頼を受けたとき、自分が誰かの後を追隨していれば、依頼の内容を無視し、現在の追隨を継続する。
	追隨の依頼を受けたとき、その依頼は無視し、あたりを見渡し、追隨する対象となる人がいないかを観測する。
	誘導者が指示している方向を向く。
	誰かの方向を向くときには、向くと同時にその人が指差しをしているかどうかを観測する。
	指差しをしていることを観測したら、その指差しの方向を向き、同時にその前方に他のエージェントがいないかどうかを観測する。
	指を指している方向にエージェントがいれば、そのエージェントの後ろについていく。
その他	プロトコルに従う。

表 3: 超越者の行動ルール

分類基準	行動ルール
シミュレーションの環境に依存したインタラクション	避難完了のエリアに入れば，その人に避難完了を確認したということを伝える．
その他	プロトコルに従う

表 4: 吸着誘導者の行動ルール

分類基準	行動ルール
シミュレーションの環境に依存したインタラクション	歩行中，あらかじめ決められた避難経路の上を歩いていく．
	誘導対象者の姿が見えなければ，その誘導対象者の誘導を諦め，避難経路を進む．
	誘導中に，「付いてきてください」と指示するように依頼がきたら，その依頼は無視して，現在の誘導を続けて避難経路を進む．
表現するインタラクションの粒度	歩行中，前方にいる人数に応じて歩行速度を変更する．
	歩行中，他のエージェントに接近しすぎていれば，体の向きを変えて移動する．
	「付いてきてください」と指示するときに，指示の対象を誘導中として覚えておき，誘導対象者との距離が離れていないかを観測し始めておく．
	誘導対象者との距離が離れれば，立ち止まって誘導対象者の方向を向き，誘導対象者の姿が見えるかどうかを調べる．
	誘導者の姿が見えていれば，その誘導対象者がこちらに近づいているかどうかを観測する．
	誘導対象者が近づいていれば，誘導対象者を待つ．
	誘導対象者が近づいていなければ，付いて来るように指示する．
	誘導者の姿が見えていれば，その誘導対象者に近づき，ついてくるように指示する．
	「付いてきてください」と指示するときに，指示の対象を誘導中として覚えておく．
その他	プロトコルに従う

- 行動ルール 1 - 1 :  
プロトコルに追従の記述があるとき，自分がまだ誰の後にも追従していなければ，その追従の対象者の姿が見えるかどうかを観測する．
- 行動ルール 1 - 2 :  
追従する対象者の姿が見えれば，ついていく．
- 行動ルール 1 - 3 :  
追従する対象者の姿が見えていなければ，あたりを見渡し，周りに追従する対象となる人がいないかを観測する．

表 5: 指差誘導者の行動ルール

分類基準	行動ルール
シミュレーションの環境に依存したインタラクション	歩行中，あらかじめ決められた避難経路の上を歩いていく．
	指を指す依頼が来れば，自分が次に向かおうとしていた避難経路の方向を指差し，その方向を指差していることを覚えておく．
	「あちらへ避難してください」と指示をする依頼が来れば，その指示の対象者の方向を見て指示する．その後，すぐに避難経路を進みだす．
	指差誘導中に，自分から見て指の方向とは反対の位置で立ち止まっているエージェントがいれば，避難する方向を指示する
表現するインタラクションの粒度	歩行中，前方にいる人数に応じて歩行速度を変更する．
	歩行中，他のエージェントに接近しすぎていれば，体の向きを変えて移動する．
	「あちらへ避難してください」と指示をする依頼が来れば，その指示の対象者の方向を見て指示する．
	「あちらへ避難してください」と指示をする依頼が来れば，その指示の対象者の方向を見て指示する．同時に他にも指示をする対象者がいないかを調べる．
	誘導対象者が近づいていなければ，付いて来るように指示する．
その他	プロトコルに従う

- 行動ルール 1 - 4 :  
プロトコルに追隨の記述があるとき，自分が誰かの後を追隨していれば，現在の追隨をやめ，その追隨の対象者の姿が見えるかどうかを観測する．
- 行動ルール 2 - 1 :  
プロトコルに追隨の記述があるとき，自分がまだ誰の後にも追隨していなければ，その追隨の対象者の姿が見えるかどうかを観測する．
- 行動ルール 2 - 2 :  
プロトコルに追隨の記述があるとき，自分が誰かの後を追隨していれば，プロトコルの記述を無視する．
- 行動ルール 3 - 1 :  
プロトコルに追隨の記述があるとき，プロトコルの記述を無視する．  
このプロトコルと行動ルールのセットから，以下のことが読み取れる．
- 行動ルール 1 のセットを用いるエージェントは，「ついてきてください」と指示されたとき，すでに他の誘導者に追隨していても，誘導されるたびに追隨の対象が変わりうる．

- 行動ルール2のセットを用いるエージェントは、「ついてきてください」と指示されても、すでに他の誘導者に追従していれば、他の誘導は無視する。
- 行動ルール3を用いるエージェントは、「ついてきてください」と誘導されても追従しない。

以上のように、単一のプロトコルに対して、複数の動作を行うエージェントが考えられる。

## 第5章 モデルの統合の結果と考察

### 5.1 マルチエージェントシミュレーションの動作

第4章で設計したシミュレーションを実際に動作させた。

そのシミュレーション画面の中から、確認された行動を以下に示す。

以上、3種類のプロトコルに対し、12種類の行動を行うエージェントが確認できた。また、避難者については、吸着誘導と指差誘導に対する行動は別々に起きるため、 $3 \times 3$ で9通りの場合が考えられ、合計で15種類のエージェントを実現できたと考える。

よって、エージェントモデルの統合によって、プロトコルどおり動くエージェントに加えて、12種類のエージェントを表現することができたことを確認できた。

## 第6章 おわりに

本研究では、インタラクションモデルとエージェントモデルの統合モデルを提案し、Q、プロダクションシステム、FreeWalkを用いて実装した。まず、インタラクションモデルの記述には従来のメッセージ通信以外のインタラクションも記述することのできるインタラクション記述言語Qを用いた。次にエージェントモデルの記述にはルールベースであることからプロダクションシステムを用いた。そして、エージェントの環境としてマルチユーザ・マルチエージェントシミュレーションが可能な仮想都市空間FreeWalkを用いた。またシナリオとしては、地下鉄京都駅を舞台とした避難シミュレーションを設計することとした。そして、シナリオのみですべてを記述した場合と、シナリオとプロダクションシステムを用いた場合の2通りを設計した。なお、プロダクションシステムにおける行動ルールは他の研究成果や文献などから得て設計した。

表 6: シミュレーションで確認されたエージェントの行動

エージェント	確認された行動
吸着誘導者	プロトコルどおり、近くのエージェントに対して吸着誘導を行い、そのエージェントが離れていれば、そのエージェントが来るまで立ち止まって待っている。
	誘導中のエージェントが離れたとき、立ち止まらずにその誘導中のエージェントに近づく。
	一度誘導を開始したらそれ以外のエージェントの誘導は行わず、避難経路を進む。
指差誘導者	プロトコルどおり、付近にエージェントがいれば出口までの方向を指示し、いなくなるまでその場で立ち止まって、指示をしている。
	付近にエージェントがいれば出口までの方向を指示する。指示を行ったあとは、付近に他のエージェントがいても避難経路を少し進もうとする。
	付近のエージェントに出口までの方向を指示した後、他の方向を見て、自分から見て出口の方向とは反対方向の位置で立ち止まっているエージェントがいれば、そのエージェントにも出口の方向を指示する。
避難者	プロトコルどおり、「付いてきてください」という指示を受けるたびにについていく対象を変更する。
	「付いてきてください」という指示を受けたときにすでに誰かの後ろを付いて行っていれば、その他の誘導者の指示は聞かない。
	「付いてきてください」という指示を受けても、その誘導者には従わずに、周りを見渡してから、近くにいるエージェントの後をついていく。
	プロトコルどおり、出口までの方向を指示されたとき、その方向に向かう。
	出口までの方向を指示されたとき、その方向に通路が見えれば、その通路を歩く。
	出口までの方向が指示されたときは、その方向にエージェントがいるかどうかを確認し、いればそのエージェントに近づく。いなければプロトコルどおり指示された方向へ歩いていく。

この統合モデルの設計において、以下の2つの結果を得た。

- プロトコルの一般性という観点をもった設計  
設計者は、2つのモデルを客観的に区別しながら設計するための基準として、「プロトコルはエージェントシミュレーションの環境によらず、一般性をもった存在である」という観点に従い設計する。シミュレーションの環境に依存したインタラクションは、エージェントモデルとして記述される。また、プロトコルから環境に依存した記述がなくなるため、シミュレーション環境によらない一般性をもったプロトコルの記述が可能となる。
- エージェントモデルで解釈することによる多様な動作の実現  
同一のプロトコルに対して、エージェントモデルにある自らの行動ルールやそれまでに得た情報に従ってプロトコルを解釈することにより、多様な解釈が行い行動を行うようになる。また、これによって、より現実感のあるマルチエージェントシミュレーションを少ないインタラクションで記述することができ、プロトコルの可読性が向上する。

これらの結果は、第1章で述べた「プロトコルのシミュレーション環境への依存」「プロトコルの可読性の低下」などの問題に対して、対処ができることを示している。

以上から、本システムの設計を通じて、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モデルとエージェントモデルの統合により、多様なエージェントをシミュレーションに参加させることができることを確認した。また、プロトコルからシミュレーションの環境に依存した記述を無くすことができ、設計されたプロトコルが、より一般性を持っているということが示された。また、インタラクション記述が簡潔に表現できたことによるプロトコルの可読性の向上も確認した。

この設計方法は、プロトコルを用い、エージェントがそのプロトコルにより行動しているマルチエージェントシステム全体に適用可能な手法であり、多様なエージェントを実装する必要のあるマルチエージェントシミュレーションの分野において有効な手法であると考えられる。

## 謝辞

本研究について多大なるご指導，ご助言を賜りました石田亨教授に厚く御礼申し上げます。

また，研究全般にわたり多くの有益なご助言をいただきました中西英之助手に深謝致します。

また，日頃から多くのご協力を頂きました石田研究室の皆様にご心より感謝致します。

## 参考文献

- [1] Foundation for Intelligent Physical Agents: FIPA ACL Message Structure Specification, *FIPA XC00061*, (AUG. 2000).
- [2] Foundation for Intelligent Physical Agents: Representing Agent Interaction Protocols in UML, *Agent-Oriented Software Engineering*, pp. 121–140 (2001).
- [3] Foundation for Intelligent Physical Agents: AgenTalk: Coordination Protocol Description for Multiagent Systems, *International Conference on Multi-Agent Systems (ICMAS-95)*, p. 455 (1995).
- [4] Toru Ishida: Q: A Scenario Description Language for Interactive Agents, *IEEE Computer*, Vol. 35, pp. 54–59 (2002).
- [5] Hideyuki Nakanishi: FreeWalk: A Social Interaction Platform for Group Behavior in a Virtual Space, *International Journal of Human Computer Studies*, Vol. 60, pp. 421–454 (2004).
- [6] 村上陽平 石田 亨: マルチエージェントシミュレーションによるプロトコル設計, 合同エージェントワークショップ&シンポジウム (JAWS2004), pp. 66–73 (2004).
- [7] T.Sugiman and J.Misumi: Development of a New Evacuation Method for Emergencies: Control of Collective Behavior by Emergent Small Groups, *Journal of Applied Psychology*, Vol. 73, pp. 3–10 (1988).

# 付録：マルチエージェントシミュレーション

## A.1 エージェントモデルを統合した場合のプロトコル

```
;;
;; *-- scheme --*
;;

;; 地下鉄京都駅避難訓練シナリオファイル
;; file   : skyoto-scenario-20050204.q
;; creator: Konishi Shinji
;; date   : 2005/02/04

(define
  (wait-for x)
  x
  (do ((cs (+ x (current-seconds)) cs))(> (current-seconds) cs)))
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; エージェントの基本モデル。
;; インタラクションは開始と終了のみでその他のインタラクションはない。
;; 基本の骨組みを理解しておくためだけに設計。
;;
(display "(defscenario scenario-skyoto-base)")(newline)
(defscenario
  scenario-skyoto-base
  ($area $scenario-type $emailaddress)
  (let
    (($route (list (list(list 0.0 0.0)) $area $area))
     ($agents '())
     ($loop-end #f)
     ($agent-rp agent-rp)
     ($agent-rr agent-rr)
     ($agent-slr agent-slr)
     ($agent-str agent-str)
     ($walk-w walk-w)
     ($walk-v walk-v)
     ($walk-a walk-a)
     ($turn-v turn-v)
     ($turn-a turn-a)
     ($tmp 0))
    (scene_init
     (#t
      (!!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a
```

```

                :angle_velocity $turn-v :angle_acceleration $turn-a)
      (go scene_normal)))
(scene_normal
  ((?receive :from_s $agents :word "避難訓練を開始")
    (!finish :action "walk")
    (set! $agents '())
    (set! $walk-w *evac-walk-w*)
    (set! $walk-v *evac-walk-v*)
    (set! $turn-v *evac-turn-v*)
    (set! $turn-a *evac-turn-a*)
    (!!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a :angle_v
    (go scene_evac)))
(scene_evac
  ((?finish :action "walk")
    (!!send :to_s *special-agents* :sentence *finish-keyword*)
    (go scene_end)))
(scene_end
  (#t
    (!disappear))))))

;;
;; 特殊エージェント 1
;; シナリオの進行管理を行う
;; また、避難完了エージェントの管理も行っている。
;; 他者とのインタラクションは
;; ・避難訓練開始、終了メッセージの送信
;; ・避難完了エリアへのエージェントの移動を観測 避難完了確認メッセージ送信
;; である
;;
(display "(defscenario scenario-skyoto-special1)")(newline)
(defscenario
  scenario-skyoto-special1
  ($area $scenario-type $emailaddress)
  (let
    (($route (list (list(list 0.0 0.0)) $area $area))
     ($agents '())
     ($loop-end #f)
     (scenario-time 0)
     ($tmp 0))
    (scene_init
      (#t
        (go scene_normal)))
    (scene_normal
      ((?input :key "return" :mode "temporarily")
        (?position :name_s $agents :from_coordinate '(0.0 0.0 0.0 0.0) :rectangle '((-1000.0 -1000.0

```

```

        (display "(!speak :to_s *all-agents* :sentence *evacuation-exam-start-sentence-1* :loudness
        (!speak :to_s $agents :sentence "ただいまから避難訓練を開始します。みなさん
階段を使って速やかに上の階まで避難してください。" :loudness 10.0); 人間型エージェント
トにしか話せない
        (!!send :to_s self :sentence "避難訓練を開始")
        (set! *scenario-run* #t)
        (set! $agents '())
        (set! scenario-time (current-seconds))
        (go scene_evac)))
(scene_evac
  ((?receive :from_s self :word "避難訓練を終了")
    (go scene_end))
  ((?position :name_s $agents
    :from_coordinate '(0.0 0.0 0.0 0.0)
    :rectangle (cadr (assoc 'exit-d-c *area-rectangle-list*))
    :height '(-8.0 -6.0))
    (!!speak :to_s $agents :sentence "あなたの避難完了を確認しました。" :loudness 10.0)
    (set! $agents '())
    (wait-for 2)
    (go scene_evac))
  ((?position :name_s $agents
    :from_coordinate '(0.0 0.0 0.0 0.0)
    :rectangle (cadr (assoc 'exit-s-n *area-rectangle-list*))
    :height '(-12.5 -10.5))
    (!!speak :to_s $agents :sentence "あなたの避難完了を確認しました。" :loudness 10.0)
    (set! $agents '())
    (wait-for 2)
    (go scene_evac))
  )
(scene_end
  (#t
    (display "全員の避難が完了しました")
    (!!speak :to_s *all-agents* :sentence "全員の避難が完了しました。これで避難
訓練を終了いたします。" :loudness 10.0))))))

;;
;; アバターシナリオ
;; シナリオの進行に応じて、状態を遷移させるが、その行動は操作者に従う。
;; 他者とのインタラクションは
;; ・避難訓練開始、終了メッセージの受信
;; である
;;
(display "(defscenario scenario-skyoto-avatar1)")(newline)
(defscenario
  scenario-skyoto-avatar1

```

```

($area $scenario-type $emailaddress)
(let
  (($route (list (list(list 0.0 0.0)) $area $area))
   ($agents '())
   ($loop-end #f)
   ($tmp 0))
  (scene_init
   (#t
    (go scene_walk)))
  (scene_walk
   ((?receive :from_s $agents :word "避難訓練を開始")
    (!!send :to_s self :sentence "避難訓練を開始")
    (set! $agents '())
    (go scene_evac)))
  (scene_evac
   ((?receive :from_s $agents :word "避難完了を確認")
    (!!send :to_s self :sentence "避難完了を確認")
    (set! $agents '())
    (go scene_end)))
  (scene_end
   (#t
    (display '(,self avatar finished))(newline)))
  ))

```

;; 誘導者エージェント1

;; ・基本的に現在地から出口まで避難。

;; ・開始と同時に近くにいる避難者エージェントについてくるように指示をし、避難を開始する。

;; ・誘導対象者が自分に近づく様子がなく、立ち止まっていれば、再度指示する。

;; ・誘導対象者が自分以外のエージェントについていているようならば、

;; そのエージェントの誘導をやめる。

;; ・避難経路誘導中、誘導対象者との距離により、適切な動作（減速、停止+振り向き+指示、停止+接近+指示）をし、避難する。

;;

```
(display "(defscenario scenario-skyoto-guide1)")(newline)
```

```
(defscenario
```

```
  scenario-skyoto-guide1
```

```
  ($area $scenario-type $emailaddress)
```

```
  (let
```

```
    (($route (list (list(list 0.0 0.0)) $area $area))
```

```
     ($agents '())
```

```
     ($loop-end #f)
```

```
     ($agent-rp agent-rp)
```

```

($agent-rr agent-rr)
($agent-slr agent-slr)
($agent-str agent-str)
($walk-w *evac-walk-w*)
($walk-v *evac-walk-v*)
($walk-a *evac-walk-a*)
($turn-v *evac-turn-v*)
($turn-a *evac-turn-a*)
($tmp 0))
(scene_init
(#t
(go scene_normal)))
(scene_normal
((?receive :from_s $agents :word "避難訓練を開始")
(display '(,self start))(newline)
(!send :to_s self :sentence "避難訓練を開始")
(set! $agents '())
(wait-for 5)
(go scene_evac_walk_without_guide)))
(scene_evac_not_walk_without_guide
((?receive :from_s $agents :word "避難完了を確認")
(!send :to_s self :sentence "避難完了を確認")
(go scene_end))
((?position :from self :name_s $agents :distance_range '(0.0 3.5) :height '(-1.0 1.0))
(!speak :to_s $agents :sentence "これから避難します。私について来て下さい。")
(wait-for 2)
(!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a
:angle_velocity $turn-v :angle_acceleration $turn-a)
(set! $agents '())
(go scene_evac_walk_with_guide))
(otherwise
(!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a
:angle_velocity $turn-v :angle_acceleration $turn-a)
(go scene_evac_walk_without_guide)))
(scene_evac_walk_without_guide
((?receive :from_s $agents :word "避難完了を確認")
(!send :to_s self :sentence "避難完了を確認")
(go scene_end))
((?position :from self :name_s $agents :distance_range '(0.0 3.5) :height '(-1.0 1.0))
(!speak :to_s $agents :sentence "これから避難します。私について来て下さい。")
(wait-for 2)
(!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a
:angle_velocity $turn-v :angle_acceleration $turn-a)
(set! $agents '())
(go scene_evac_walk_with_guide))

```

```

(otherwise
  (!!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a
    :angle_velocity $turn-v :angle_acceleration $turn-a)
  (set! $agents '())
  (go scene_evac_walk_without_guide))
)
(scene_evac_walk_with_guide
  ((?receive :from_s $agents :word "避難完了を確認")
    (!!send :to_s self :sentence "避難完了を確認")
    (go scene_end))
  (otherwise
    (go scene_evac_walk_without_guide))
  )
(scene_end
  (#t
    (display '(,self end))(newline)
    (!finish :action "walk")
    (!finish :action "approach")
  )))

```

;; 誘導者エージェント2

;; ・基本的に現在地から出口まで避難。

;; ・避難中に誘導対象者がいれば、出口の方向を指示する。

;; ・誘導対象者が誘導方向に移動し、付近に誰もいなければ自らも避難を開始する。

;;

```
(display "(defscenario scenario-skyoto-guide2)"(newline))
```

```
(defscenario
```

```
scenario-skyoto-guide2
```

```
($area $scenario-type $emailaddress)
```

```
(let
```

```
  (($route (list (list(list 0.0 0.0)) $area $area))
```

```
  ($agents '())
```

```
  ($loop-end #f)
```

```
  ($agent-rp agent-rp)
```

```
  ($agent-rr agent-rr)
```

```
  ($agent-slr agent-slr)
```

```
  ($agent-str agent-str)
```

```
  ($walk-w walk-w)
```

```
  ($walk-v walk-v)
```

```
  ($walk-a walk-a)
```

```
  ($turn-v turn-v)
```

```
  ($turn-a turn-a)
```

```
  ($tmp 0))
```

```
(scene_init
```

```

(#t
  (go scene_normal)))
(scene_normal
  ((?receive :from_s $agents :word "避難訓練を開始")
    (display '(,self start))(newline)
    (!!send :to_s self :sentence "避難訓練を開始")
    (set! $agents '())
    (set! $walk-w *evac-walk-w*)
    (set! $walk-v *evac-walk-v*)
    (set! $turn-v *evac-turn-v*)
    (set! $turn-a *evac-turn-a*)
    (wait-for 8)
    (!!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a
      :angle_velocity $turn-v :angle_acceleration $turn-a)
    (go scene_evac_walk_without_guide)))
(scene_evac_walk_without_guide
  ((?receive :from_s $agents :word "避難訓練を終了")
    (!!send :to_s self :sentence "避難訓練を終了")
    (go scene_end))
  ((?receive :from_s $agents :word "避難完了を確認")
    (!!send :to_s self :sentence "避難完了を確認")
    (set! $agents '())
    (go scene_end))
  ((?position :name_s $agents :distance_range '(0.0 3.5)
    :angle_range '(30.0 330.0))
    (!finish :action "walk")
    (!!point :to_coordinate (car $route) :limit_angle_to_body 90.0)
    (!!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a
      :angle_velocity $turn-v :angle_acceleration $turn-a)
    (!speak :to_s $agents :sentence "あちらへ避難してください")
    (!finish :action "walk")
    (wait-for 3)
    (set! $agents '())
    (go scene_evac_guide))
  ((?position :name_s $agents :distance_range '(0.0 1.5)
    :angle_range '(-40.0 40.0))
    (!finish :action "walk")
    (!!point :to_coordinate (car $route) :limit_angle_to_body 90.0)
    (!!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a
      :angle_velocity $turn-v :angle_acceleration $turn-a)
    (!speak :to_s $agents :sentence "あちらへ避難してください")
    (!finish :action "walk")
    (wait-for 3)
    (set! $agents '())
    (go scene_evac_guide))

```

```

)
(scene_evac_guide
  ((?receive :from_s $agents :word "避難訓練を終了")
    (!!send :to_s self :sentence "避難訓練を終了")
    (set! $agents '())
    (go scene_end))
  ((?receive :from_s $agents :word "避難完了を確認")
    (!!send :to_s self :sentence "避難完了を確認")
    (set! $agents '())
    (go scene_end))
  ((?position :from self :name_s $agents :distance_range '(0.0 2.0)
    :angle_range '(30.0 330.0))
    (!!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a
      :angle_velocity $turn-v :angle_acceleration $turn-a)
    (!speak :to_s $agents :sentence "あちらへ避難してください")
    (!finish :action "walk")
    (wait-for 3)
    (set! $agents '())
    (go scene_evac_guide))
  ((?position :from self :name_s $agents :distance_range '(0.0 1.0)
    :angle_range '(320.0 400.0))
    (!!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a
      :angle_velocity $turn-v :angle_acceleration $turn-a)
    (!speak :to_s $agents :sentence "あちらへ避難してください")
    (!finish :action "walk")
    (wait-for 3)
    (set! $agents '())
    (go scene_evac_guide))
  (otherwise
    (!finish :action "point")
    (!!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a
      :angle_velocity $turn-v :angle_acceleration $turn-a)
    (set! $agents '())
    (go scene_evac_walk_without_guide)))
(scene_end
  (#t
    (display '(,self end))(newline)
    (!finish :action "walk")
    (!finish :action "approach")
  )))

```

```

;;
;; 避難者シナリオ 4
;; 避難訓練開始とともに、避難を開始する。
;; 他者とのインタラクションは

```

```

;;・避難訓練開始のメッセージを受信する 避難行動開始
;;・誘導者の誘導に従う
;;・避難完了確認メッセージを受信する 姿を消す
;;である。
(display "(defscenario scenario-skyoto-evacuee4)")(newline)
(defscenario
scenario-skyoto-evacuee4
($area $scenario-type $emailaddress)
(let
  (($route '(((0.0 0.0)) , $area (, $area)))
   ($agents '())
   ($agent-rp agent-rp)
   ($agent-rr agent-rr)
   ($agent-slr agent-slr)
   ($agent-str agent-str)
   ($walk-w walk-w)
   ($walk-v walk-v)
   ($walk-a walk-a)
   ($turn-v turn-v)
   ($turn-a turn-a)
   ($follow-to '())
   ($tmp 0)
   ($tmp2 0)
   ($tmp3 0))
(scene_init
  (#t
   (display '(,self ready))(newline)
   (go scene_normal)))
(scene_normal
  ((?receive :from_s $agents :word "避難訓練を開始")
   (!!send :to_s self :sentence "避難訓練を開始")
   (display '(,self start))(newline)
   (set! $agents '())
   (set! $walk-w *evac-walk-w*)
   (set! $walk-v *evac-walk-v*)
   (set! $turn-v *evac-turn-v*)
   (set! $turn-a *evac-turn-a*)
   (wait-for 2)
   (go scene_evac)))
(scene_evac
  ((?receive :from_s $agents :word "避難完了を確認")
   (!!send :to_s self :sentence "避難完了を確認")
   (go scene_end))
  ((?receive :from_s $agents :word "ついて来て下さい")
   (!turn :to $agents :angle_velocity $turn-v :angle_acceleration $turn-a)

```

```

    (!!approach :to_s $agents
      :distance 1.0 :velocity $walk-v :acceleration $walk-a
      :angle_velocity $turn-v :angle_acceleration $turn-a)
    (set! $agents '())
    (go scene_evac))
  ((?receive :from_s $agents :word "あちらへ避難してください")
    (!turn :to $agents :angle_velocity $turn-v :angle_acceleration $turn-a)
    (!!approach :to_s $agents
      :distance 1.0 :velocity $walk-v :acceleration $walk-a
      :angle_velocity $turn-v :angle_acceleration $turn-a)
    (set! $agents '())
    (wait-for 2)
    (go scene_evac))
  )
(scene_end
  (#t
    (display '(,self end))(newline)
    (!finish :action "walk")
    (!finish :action "approach")
  )))

```

## A.2 エージェントモデルの行動ルール

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; -----
;; defrule
;; -----
;; ((rule-name [:ruleset ruleset-name :priority priority-number] LHS... --> RHS...)... )
;;   LHS
;;     (match pattern)
;;     (not pattern)
;;     (bind pattern-variable pattern)
;;     (test scheme-exp)
;;     (test (call scheme-exp))
;;     (test (expand scheme-exp))
;;     pattern
;;   RHS
;;     (make pattern)
;;     (remove pattern-variable)
;;     (expand scheme-exp)

```

```
;;      (bind  pattern-variable scheme-exp)
;;      (call  scheme-exp)
;;      (focus  [:append] ruleset-name...)
;;      (halt)
;;      scheme-exp
```

```
(literalize agent
:id
:name)
```

```
(literalize avatar
:id
:name)
```

```
(literalize
test
:id
:name)
```

```
;;
;;Q 依頼のための述語
;;
(literalize request
:id
:type
:request-name
:option-list
:cont
:cont-error)
```

```
(literalize true
:id
:cue)
```

```
(literalize false
:id
:cue)
```

```
(literalize
?observe
:id
:name_s
:gesture
:action)
```

:shape)

(literalize  
?position  
:id  
:name\_s  
:from  
:from\_coordinate  
:distance\_range  
:relative\_angle  
:angle\_range  
:point\_angle  
:rectangle  
:height)

(literalize  
?receive  
:id  
:from\_s  
:word)

(literalize  
?hear  
:id  
:from\_s  
:word)

(literalize  
?finish  
:id  
:action)

(literalize  
?see  
:name)

(literalize  
?input  
:id  
:key  
:mode)

(literalize  
guide-action  
:id)

```
:to
:message)

;;
;; エージェントに関する述語
;;
(literalize
 is-pointing
 :id
 :to_coordinate
 :time)

(literalize
 is-following
 :id
 :to
 :time)

(literalize
 is-guiding
 :id
 :to
 )

(literalize
 is-turning
 :id
 :to
 :to_coordinate
 :angle
 :relative_angle
 :time)

(literalize
 is-facing
 :id
 :angle_to_body
 :to)

(literalize
 is-looking
 :id
 :name
```

```

:shape
:cont)

(literalize
route
:id
:route
:mode
)

(literalize
in-evac
:id)

;; エージェントの情報
(literalize
agent-type
:id
:name
:type
:options)

;; アバターの情報
(literalize
avatar-type
:id
:name
:type
:options)

;; アバターの位置を観測中
(literalize
check-avatars-position
:id
:name
:rectangle
:height
:area
:num
:flag)

;; (エージェントが) 自分が避難を完了しているを知っている
(literalize
has-reached-goal
:id)

```

```
;;rectangle を監視中
(literalize
 is_check_rectangle
 :id
 :from_co
 :rectangle
 :height
 :relative_angle
 :threshold
 :rectangle_to_speak
 :height_to_speak
 :relative_angle_to_speak
 :state
 :sentence
 :time_stamp)
```

;;id は name\_s(他人) が指差し行動をしていることを知っている。

```
;;
(literalize
 know-pointing
 :id
 :name_s
 :point_angle
 :relative_angle
 :angle_range
 :time)
```

;;(超越者が)evacuee が避難を完了していることを知っている

```
(literalize
 know_as_evacuee_finished
 :id
 :evacuee
 )
```

;;メッセージ送信の依頼がある

```
(literalize
 send-request
 :id
 :to_s
 :sentence)
```

;;自分の視野の中に(どの方向を向きながら)いる

```
(literalize
```



```

(dbg "defagents ID: ")(dbg ?id)
(cond ((equal? (get-type-with-scenario (get-scenario-type ?id)) 'evacuee)
      (if (null? *evacuee-agents*)
          (set! *evacuee-agents* '(,?id))
          (set! *evacuee-agents* (append *evacuee-agents* '(,?id))))
      (if (null? *evacuee-agents-and-avatars*)
          (set! *evacuee-agents-and-avatars* '(,?id))
          (set! *evacuee-agents-and-avatars* (append *evacuee-agents-and-avatars* (list ?id))))
      (if (null? *movable-agents*)
          (set! *movable-agents* (list ?id))
          (set! *movable-agents* (append *movable-agents* (list ?id))))
      (if (null? *all-agents*)
          (set! *all-agents* (list ?id))
          (set! *all-agents* (append *all-agents* (list ?id)))))
      ((equal? (get-type-with-scenario (get-scenario-type ?id)) 'keitai)
      (if (null? *evacuee-agents*)
          (set! *evacuee-agents* (list ?id))
          (set! *evacuee-agents* (append *evacuee-agents* (list ?id))))
      (if (null? *keitai-agents*)
          (set! *keitai-agents* (list ?id))
          (set! *keitai-agents* (append *keitai-agents* (list ?id))))
      (if (null? *evacuee-agents-and-avatars*)
          (set! *evacuee-agents-and-avatars* (list ?id))
          (set! *evacuee-agents-and-avatars* (append *evacuee-agents-and-avatars* (list ?id))))
      (if (null? *movable-agents*)
          (set! *movable-agents* (list ?id))
          (set! *movable-agents* (append *movable-agents* (list ?id))))
      (if (null? *all-agents*)
          (set! *all-agents* (list ?id))
          (set! *all-agents* (append *all-agents* (list ?id)))))
      ((equal? (get-type-with-scenario (get-scenario-type ?id)) 'guide)
      (if (null? *guide-agents*)
          (set! *guide-agents* (list ?id))
          (set! *guide-agents* (append *guide-agents* (list ?id))))
      (if (null? *movable-agents*)
          (set! *movable-agents* (list ?id))
          (set! *movable-agents* (append *movable-agents* (list ?id))))
      (if (null? *all-agents*)
          (set! *all-agents* (list ?id))
          (set! *all-agents* (append *all-agents* (list ?id)))))
      ((equal? (get-type-with-scenario (get-scenario-type ?id)) 'special1)
      (if (null? *special-agents*)
          (set! *special-agents* (list ?id))
          (set! *special-agents* (append *special-agents* (list ?id))))
      (if (null? *all-agents*)
          (set! *all-agents* (list ?id))
          (set! *all-agents* (append *all-agents* (list ?id)))))

```

```

        (set! *all-agents* (list ?id))
        (set! *all-agents* (append *all-agents* (list ?id))))
((equal? (get-type-with-scenario (get-scenario-type ?id)) 'special2)
 (if (null? *special-agents*)
     (set! *special-agents* (list ?id))
     (set! *special-agents* (append *special-agents* (list ?id))))
 (if (null? *all-agents*)
     (set! *all-agents* (list ?id))
     (set! *all-agents* (append *all-agents* (list ?id))))
 )
((equal? (get-type-with-scenario (get-scenario-type ?id)) 'special3)
 (if (null? *special-agents*)
     (set! *special-agents* (list ?id))
     (set! *special-agents* (append *special-agents* (list ?id))))
 (if (null? *all-agents*)
     (set! *all-agents* (list ?id))
     (set! *all-agents* (append *all-agents* (list ?id))))
 )
((equal? (get-type-with-scenario (get-scenario-type ?id)) 'avatar)
 (if (null? *avatars*)
     (set! *avatars* (list ?id))
     (set! *avatars* (append *avatars* (list ?id))))
 (if (null? *evacuee-agents-and-avatars*)
     (set! *evacuee-agents-and-avatars* (list ?id))
     (set! *evacuee-agents-and-avatars* (append *evacuee-agents-and-avatars* (list ?id))))
 (if (null? *movable-agents*)
     (set! *movable-agents* (list ?id))
     (set! *movable-agents* (append *movable-agents* (list ?id))))
 (if (null? *all-agents*)
     (set! *all-agents* (list ?id))
     (set! *all-agents* (append *all-agents* (list ?id))))))
((equal? (get-type-with-scenario (get-scenario-type ?id)) 'etc)
 (if (null? *etc-agents*)
     (set! *etc-agents* (list ?id))
     (set! *etc-agents* (append *etc-agents* (list ?id))))
 (if (null? *all-agents*)
     (set! *all-agents* (list ?id))
     (set! *all-agents* (append *all-agents* (list ?id))))))
(else
 (dbg "some error")
 (newline)))
(make (route :id ?id
            :route (!value-of! (list (list(list 1.0 0.0))
                                     (eval (get-init-areaid-from-arg ?id))
                                     (list (eval (get-init-areaid-from-arg ?id)))))

```

```

                :walk-options (!value-of! (list ))
                :mode (!value-of! 'evac-stop)))
(make (agent-type :id ?id :name ?name
                :type (!value-of! (get-scenario-type ?id))
                :options (!value-of! '())))
(remove ?1))

(defrule
avatar-assigned
:priority 10
(bind ?1 (agent-type :id (!with-test! ?id (member ?id *avatars*)) :name ?name))
-->
(make (avatar-type :id ?id :name ?name
                :type (!value-of! (get-scenario-type ?id))
                :options (!value-of! '())))
(!activate :cue (?input :key "i" :mode "temporarily"))
(!activate :cue (?input :key "m" :mode "temporarily"))
(remove ?1))

(defrule
delete-route
:priority 300
(bind ?1 (route :id (!with-test! ?id (not (member ?id *movable-agents*)))))
-->
(remove ?1))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; 避難訓練開始時の動作
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; 移動可能なエージェント(アバター+避難者エージェント+誘導者エージェント)が
;; 避難開始の合図を聞けば、自分は避難訓練中という述語を作成し、
;; 避難完了エリアに到着しているかどうかの観測を開始する。
;;
(defrule
start-evacuee-exam1-1-1
:priority 10000
(agent-type :id ?id :type (!with-test! ?type (member ?type
                '(scenario-skyoto-evacuee1
                scenario-skyoto-evacuee3
                scenario-skyoto-keitai1))))
(bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name ('(!send" "!!send"
                :option-list (!with-test! ?options (and(equal-opt-value? ?options "to_s" ?id
                (equal-opt-value? ?options "sentence"
避難訓練を開始"))))))))

```



```

:priority 10000
(agent-type :id ?id :type (!with-test! ?type (equal? ?type 'scenario-skyoto-evacuee2)))
(bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name '("!send" "!!send"
:option-list (!with-test! ?options (and(equal-opt-value? ?options "to_s" ?id
(equal-opt-value? ?options "sentence"
避難訓練を開始")))))
; (bind ?1 (?receive :id ?id :word (!with-test! ?word (equal? ?word *start-keyword*)))
(bind ?1 (route :id ?id :route ?route :mode ?mode))
-->
(display '(***** OK ,?id *****)) (newline)
(make (in-evac :id ?id))
(!activate :cue (?position :name_s ?name :from_coordinate '(0.0 0.0 0.0 0.0)
:rectangle '((376.0 -70.0)(390.0 -55.0)) :height '(-8.0 -7.0)))
(!activate :cue (?position :name_s ?name :from_coordinate '(0.0 0.0 0.0 0.0)
:rectangle '((376.0 -100.0)(390.0 -57.0)) :height '(-12.5 -11.5)))
(!activate :cue (?position :name_s ?id
:from_coordinate '(0.0 0.0 0.0 0.0)
:rectangle '((370.0 -55.0)(378.0 -70.0))
:height '(-8.0 -7.0)))
(!activate :cue (?position :name_s ?id
:from_coordinate '(0.0 0.0 0.0 0.0)
:height '(-11. -7.6)))
(modify ?1 (route :id ?id :route (!value-of! '((375.5 -42.5)) st-sd-2-1 (s-1-c-e st-sd-2-1)))
:mode (!value-of! 'evac-stop)))
(!activate :cue (?position :name_s '() :distance_range '(0.0 0.6)
:angle_range '(0.0 90.0)))
(!activate :cue (?position :name_s '() :distance_range '(0.0 0.6)
:angle_range '(270.0 360.0)))
(!activate :cue (?position :name_s ?id
:from_coordinate '(0.0 0.0 0.0 0.0)
:rectangle '((370.0 -55.0)(378.0 -70.0))
:height '(-8.0 -7.0)))
(bind ?route (make-evac-route-one ?id ?route 1 #f))
(!follow :request ?0)
(remove ?0)
)
(defrule
start-evacuee-exam1-1-3
:priority 10000
(agent-type :id ?id :type (!with-test! ?type (member ?type
(scenario-skyoto-evacuee4))))
(bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name '("!send" "!!send"
:option-list (!with-test! ?options (and(equal-opt-value? ?options "to_s" ?id
(equal-opt-value? ?options "sentence"
避難訓練を開始")))))

```

```

(bind ?1 (route :id ?id :route ?route :mode ?mode))
-->
(display '(***** OK ,?id *****)) (newline)
(make (in-evac :id ?id))
(!activate :cue (?position :name_s ?name :from_coordinate '(0.0 0.0 0.0 0.0)
                    :rectangle '((376.0 -70.0)(390.0 -55.0)) :height '(-8.0 -7.0)))
(!activate :cue (?position :name_s ?name :from_coordinate '(0.0 0.0 0.0 0.0)
                    :rectangle '((376.0 -100.0)(390.0 -57.0)) :height '(-12.5 -11.5)))
(!activate :cue (?position :name_s ?id :from_coordinate '(0.0 -12.0 0.0 0.0)
                    :rectangle '((377.5 -55.0)(380.5 -41.9)) :height '(-0.3 0.3)))
(!activate :cue (?position :name_s ?id :from_coordinate '(0.0 -12.0 0.0 0.0)
                    :rectangle '((368.0 -55.0)(372.1 -41.9)) :height '(-0.3 0.3)))
(!activate :cue (?position :name_s ?id
                    :from_coordinate '(0.0 0.0 0.0 0.0)
                    :rectangle '((370.0 -55.0)(378.0 -70.0))
                    :height '(-8.0 -7.0)))
(!activate :cue (?position :name_s ?id
                    :from_coordinate '(0.0 0.0 0.0 0.0)
                    :height '(-11.9 -7.6)))
(!activate :cue (?position :name_s '() :distance_range '(0.0 0.6)
                    :angle_range '(0.0 90.0)))
(!activate :cue (?position :name_s '() :distance_range '(0.0 0.6)
                    :angle_range '(270.0 360.0)))
(bind ?route (make-evac-route-one ?id ?route 1 #f))
(modify ?1 (route :id ?id :route ?route :mode (!value-of! 'evac-stop)))
(!follow :request ?0)
(remove ?0)
)

(defrule
start-evacuee-exam1-2
:priority 10000
(agent-type :id ?id :type (!with-test! ?type (member ?type
                                                    '(scenario-skyoto-guide1
                                                      scenario-skyoto-guide2))))
(bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name '(!send "!!send"
:option-list (!with-test! ?options (and(equal-opt-value? ?options "to_s" ?id
(equal-opt-value? ?options "sentence"
避難訓練を開始")))))
(bind ?1 (route :id ?id :route ?route :mode ?mode))
-->
(display '(***** OK ,?id *****)) (newline)
(make (in-evac :id ?id))
(bind ?route (make-evac-route-one ?id ?route 1 #f))
(modify ?1 (route :id ?id :route ?route :mode (!value-of! 'evac-stop)))

```

```

(!activate :cue (?position :name_s ?id
                  :from_coordinate '(0.0 0.0 0.0 0.0)
                  :rectangle '((370.0 -55.0)(378.0 -70.0))
                  :height '(-8.0 -7.0)))
(!activate :cue (?position :name_s ?id
                  :from_coordinate ?from_coordinate
                  :distance_range *EVAC_DISTANCE_RANGE_CHECK_WALK*))
(!activate :cue (?position :name_s ?id
                  :from_coordinate ?from_coordinate
                  :distance_range *EVAC_DISTANCE_RANGE_CHECK_WALK_2*))
(!activate :cue (?position :name_s '() :distance_range '(0.0 0.6)
                  :angle_range '(0.0 90.0)))
(!activate :cue (?position :name_s '() :distance_range '(0.0 0.6)
                  :angle_range '(270.0 360.0)))

(!follow :request ?0)
(remove ?0)
)

(defrule
start-evacuee-exam2
:priority 10000
(bind ?1 (avatar-type :id ?id :options ?option))
(bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name '(!send" "!!send"
:option-list (!with-test! ?options (and(equal-opt-value? ?options "to_s" ?id
                                          (equal-opt-value? ?options "sentence"
避難訓練を開始")))))
-->
(display '(***** OK ,?id *****))(newline)
(make (in-evac :id ?id))
(modify ?1 (avatar-type :options (!value-of! (+ 15 (current-seconds))))))
(!follow :request ?0)
(remove ?0)
)

;;
;; 移動不可能なエージェント(特殊エージェント)が
;; 避難開始の合図を聞けば、自分は避難訓練中という述語を作成する
;;
(defrule
start-evacuee-exam3
:priority 10000
(agent-type :id ?id :type (!with-test! ?type (member ?type
(list 'scenario-kyoto-special1

```

```

'scenario-skyoto-special2
'scenario-skyoto-special3))))
(bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name '("!send" "!!send"
:option-list (!with-test! ?options (and(equal-opt-value? ?options "to_s" ?id
(equal-opt-value? ?options "sentence"
避難訓練を開始")))))
-->
(display '(***** OK ,?id *****) (newline)
(make (in-evac :id ?id))
(make (know_as_evacuee_finished :id ?id :evacuee (!value-of! '())))
(!follow :request ?0)
(remove ?0)
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; 避難経路を歩く
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;
;; 避難中の状態で歩くために歩き出すルートを考える
;;
(defrule
  evac-walk-1-1
  :priority 140
  (bind ?0 (request :id ?id
    :request-name (!with-test! ?r-name (member ?r-name '("!walk" "!!walk")))
    :option-list ?options))
  (bind ?1 (route :id ?id :route ?route :mode ?mode))
  (in-evac :id ?id)
  -->
  (dbg '(,?id evac-walk-1-1))
  (modify ?1 (route :id ?id :route ?route :mode (!value-of! 'evac-walk)))
  (bind ?velocity
    (cond ((member (cadr ?route) '(st-sd-1-u st-sd-2-u st-sd-3-u))
      (* *EVAC_WALK_VELOCITY* 0.5))
      ((member (cadr ?route) '(st-sd-1-c st-sd-2-c st-sd-3-c))
      (* *EVAC_WALK_VELOCITY* 0.6))
      (else *EVAC_WALK_VELOCITY*)))
  (bind ?width
    (cond ((member (cadr ?route) '(st-sd-1-u st-sd-2-u))
      (* *EVAC_WALK_WIDTH* 1.1))
      ((member (cadr ?route) '(st-sd-1-c st-sd-2-c))
      (* *EVAC_WALK_WIDTH* 1.2))
      (else *EVAC_WALK_WIDTH*)))
  (bind ?0 (modify ?0

```

```

      (request :id ?id :request-name ?r-name
              :option-list (!value-of! (set-opt-value! ?options
                                          "in" "route" (car ?route)))

(!follow :request ?0)
(bind ?from_coordinate (list (car (list-ref (car ?route) (- (length (car ?route)) 1)))
                             (get-height-with-areaid (cadr ?route))
                             (cadr (list-ref (car ?route) (- (length (car ?route)) 1)))
                             0.0))

(!activate :cue (?position :name_s ?id
                        :from_coordinate ?from_coordinate
                        :distance_range *EVAC_DISTANCE_RANGE_CHECK_WALK*))

(!activate :cue (?position :name_s ?id
                        :from_coordinate ?from_coordinate
                        :distance_range *EVAC_DISTANCE_RANGE_CHECK_WALK_2*))

(!activate :cue (?position :name_s '() :from ?id
                        :angle_range *ANGLE_RANGE_VISIBLE*
                        :relative_angle '(0.0 360.0)
                        :distance_range *DISTANCE_RANGE_NEAR* :height *HEIGHT_EQ*))

(!activate :cue (?position :name_s ?id
                        :rectangle '((371.9 -44.0)(374.0 -41.2))))

(!activate :cue (?position :name_s ?id
                        :rectangle '((374.0 -42.0)(374.2 -40.5))))

(!activate :cue (?position :name_s ?id
                        :rectangle '((377.0 -42.0)(378.5 -40.5))))

(!activate :cue (?position :name_s ?id
                        :rectangle '((370.0 -50.0)(372.0 -42.0))))

(!activate :cue (?position :name_s ?id
                        :rectangle '((377.3 -50.0)(380.0 -41.8))))

(remove ?0)
)

;;
;; 避難中の状態で歩き終わったらまた新しく歩き出す。
;;
(defrule
  evac-walk-1-2
  :priority 100
  (bind ?1 (?position :id ?id :name_s ?id :from_coordinate ?from
                    :distance_range (!with-test! ?dis (equal? ?dis *EVAC_DISTANCE_RANGE_CHECK_WALK*))
                    (bind ?2 (route :id ?id :route ?route :mode (!with-test! ?mode (equal? ?mode 'evac-walk))))
                    (in-evac :id ?id)
                    -->
                    (dbg '(,?id evac-walk-1-2))
                    (bind ?route (make-evac-route-one ?id ?route 1 #f))
                    (modify ?2 (route :id ?id :route (!value-of! ?route))))

```

```

(bind ?velocity
  (cond ((member (cadr ?route) '(st-sd-1-u st-sd-2-u st-sd-3-u))
        (* *EVAC_WALK_VELOCITY* 0.5))
        ((member (cadr ?route) '(st-sd-1-c st-sd-2-c st-sd-3-c))
        (* *EVAC_WALK_VELOCITY* 0.6))
        (else *EVAC_WALK_VELOCITY*)))

(bind ?width
  (cond ((member (cadr ?route) '(st-sd-1-u st-sd-2-u))
        (* *EVAC_WALK_WIDTH* 1.1))
        ((member (cadr ?route) '(st-sd-1-c st-sd-2-c))
        (* *EVAC_WALK_WIDTH* 1.2))
        (else *EVAC_WALK_WIDTH*)))

(!walk :route (car ?route) :width ?width :velocity ?velocity :acceleration *EVAC_WALK_ACCELERATION*
       :angle_velocity *EVAC_TURN_VELOCITY* :angle_acceleration *EVAC_TURN_ACCELERATION*)

(bind ?from_coordinate (list (car (list-ref (car ?route) (- (length (car ?route)) 1)))
                             (get-height-with-areaid (cadr ?route))
                             (cadr (list-ref (car ?route) (- (length (car ?route)) 1)))
                             0.0))

(!activate :cue (?position :name_s ?id
                        :from_coordinate ?from_coordinate
                        :distance_range *EVAC_DISTANCE_RANGE_CHECK_WALK*))

(!activate :cue (?position :name_s ?id
                        :from_coordinate ?from_coordinate
                        :distance_range *EVAC_DISTANCE_RANGE_CHECK_WALK_2*))

(!activate :cue (?position :name_s '())
            :from ?id :angle_range *ANGLE_RANGE_VISIBLE*
            :relative_angle '(0.0 360.0)
            :distance_range *DISTANCE_RANGE_NEAR* :height *HEIGHT_EQ*))

(remove ?1))

;;
;; 歩行の経由地点からある程度の距離までくれば、経由地点を改めて確認して歩行を続行する。

;;
(defrule
  evac-walk-1-3
  :priority 10
  (bind ?1 (?position :id ?id :name_s ?id :from_coordinate ?from
                    :distance_range (!with-test! ?dis (or (equal? ?dis *EVAC_DISTANCE_RANGE_CHECK_WALK*)
                                                           (equal? ?dis *EVAC_DISTANCE_RANGE_CHECK_WALK_2*)))
                    :route :id ?id :route ?route :mode (!with-test! ?mode (equal? ?mode 'evac-walk)))
  (in-evac :id ?id)
  (test (let ((tmp (list-ref (car ?route) (- (length (car ?route)) 1))))
        (and (equal? (car ?from)(car tmp))
              (equal? (caddr ?from)(cadr tmp)))))

```

```

-->
(dbg '(,?id evac-walk-1-3))
(bind ?velocity
  (cond ((member (cadr ?route) '(st-sd-1-u st-sd-2-u st-sd-3-u))
        (* *EVAC_WALK_VELOCITY* 0.5))
        ((member (cadr ?route) '(st-sd-1-c st-sd-2-c st-sd-3-c))
        (* *EVAC_WALK_VELOCITY* 0.6))
        (else *EVAC_WALK_VELOCITY*)))
(bind ?width
  (cond ((member (cadr ?route) '(st-sd-1-u st-sd-2-u))
        (* *EVAC_WALK_WIDTH* 1.1))
        ((member (cadr ?route) '(st-sd-1-c st-sd-2-c))
        (* *EVAC_WALK_WIDTH* 1.2))
        (else *EVAC_WALK_WIDTH*)))
(!!walk :route (list (list(car ?from)(caddr ?from))) :width ?width :velocity ?velocity :acceleration
        :angle_velocity *EVAC_TURN_VELOCITY* :angle_acceleration *EVAC_TURN_ACCELERATION*)
(remove ?1))

;;
;; 視界の中かつ近距離のエージェントの人数に応じて歩行速度や横への移動幅を変更する。
;;
(defrule
  evac-walk-1-4
  :priority 10
  (bind ?1 (see-in-view :id ?id :name_s ?name_s
                        :relative_angle (!with-test! ?relative_angle
                                             (member ?relative_angle
                                                       '((0.0 360.0))))))
  (route :id ?id :route ?route :mode (!with-test! ?mode (equal? ?mode 'evac-walk)))
  (in-evac :id ?id)
  -->
  (dbg '(,?id evac-walk-1-4))
  (bind ?velocity
    (cond ((> (length (return-list ?name_s)) 10)
          (* *EVAC_WALK_VELOCITY* 0.5))
          ((> (length (return-list ?name_s)) 10)
          (* *EVAC_WALK_VELOCITY* 0.7))
          (else
           *EVAC_WALK_VELOCITY*)))
  (!!walk :route (car ?route) :width ?width :velocity ?velocity :acceleration *EVAC_WALK_ACCELERATION*
          :angle_velocity *EVAC_TURN_VELOCITY* :angle_acceleration *EVAC_TURN_ACCELERATION*)
  (!activate :cue (?position :name_s '()) :from ?id
             :angle_range *ANGLE_RANGE_VISIBLE*
             :relative_angle '(0.0 360.0)
             :distance_range *DISTANCE_RANGE_NEAR* :height *HEIGHT_EQ*))

```

```

(remove ?1))

;;
;; 冗長なルール
;; プログラムの観測の遅延に対応するためのもの
;;
(defrule
  evac-walk-1-5
  :priority 50
  (in-evac :id ?id)
  (bind ?1 (?position :id ?id :name_s ?id :from_coordinate ?from
                    :distance_range (!with-test! ?dis (or (equal? ?dis *EVAC_DISTANCE_RANGE_CHI
                                                            (equal? ?dis *EVAC_DISTANCE_RANGE_CHI
                                                            (route :id ?id :route ?route :mode (!with-test! ?mode (equal? ?mode 'evac-walk)))
                                                            (test (let ((tmp (list-ref (car ?route) (- (length (car ?route)) 1))))
                                                                (not (and (equal? (car ?from)(car tmp))
                                                                    (equal? (caddr ?from)(cadr tmp)))))))
-->
  (remove ?1))

;;
;; 入ってはいけないエリアに入ったので、引き返す。
;;
(defrule
  evac-walk-1-7-1
  :priority 20
  (bind ?1 (route :id ?id :route (!with-test! ?route (member (cadr ?route) '(st-sd-1-l st-sd-2-l
                                                                              st-sd-1-c st-sd-2-c
                                                                              st-sd-1-u st-sd-2-u
                                                                              :mode (!with-test! ?mode (equal? ?mode 'evac-walk))))))
  (bind ?2 (?position :id ?id :name ?id
                :rectangle (!with-test! ?rectangle
                              (member ?rectangle '((371.9 -44.0)(374.0 -41.2))
                                                ((374.0 -42.0)(374.2 -40.5))
                                                ((377.0 -42.0)(378.5 -40.5))
                                                ((370.0 -50.0)(372.0 -42.0))
                                                ((377.3 -50.0)(380.0 -41.8))
                                                )))))
-->
  (remove ?2)
  (cond ((equal? ?rectangle '((371.9 -44.0)(374.0 -41.2)))
        (set-car! ?route '((0.1 -0.3))))
        ((equal? ?rectangle '((374.0 -42.0)(374.2 -40.5)))
        (set-car! ?route '((0.5 -0.2))))
        ((equal? ?rectangle '((377.2 -42.0)(378.5 -40.5)))

```

```

      (set-car! ?route '((-0.5 -0.2)))
      ((equal? ?rectangle '((370.0 -50.0)(372.0 -42.0)))
       (set-car! ?route '((-0.5 -0.2)))
       ((equal? ?rectangle '((377.3 -50.0)(380.0 -41.8)))
        (set-car! ?route '((0.5 -0.2))))))
  (!!walk :route (car ?route) :width *EVAC_WALK_WIDTH* :relative #t
         :velocity *EVAC_WALK_VELOCITY* :acceleration (/ *EVAC_WALK_ACCELERATION* 2.0)
         :angle_velocity *EVAC_TURN_VELOCITY* :angle_acceleration *EVAC_TURN_ACCELERATION*)
  (cond ((equal? ?rectangle '((371.9 -44.0)(374.0 -41.2)))
         (set-car! ?route '((374.3 -41.3)(374.8 -43.0)(375.0 -49.0)))
         (set-cdr! ?route '(st-sd-2-c (s-1-c-w st-sd-2-1 st-sd-2-c))))
        ((equal? ?rectangle '((374.0 -42.0)(374.2 -40.5)))
         (set-car! ?route '((375.2 -41.5)(375.0 -49.0)))
         (set-cdr! ?route '(st-sd-2-c (s-1-c-w st-sd-2-1 st-sd-2-c))))
        ((equal? ?rectangle '((377.2 -42.0)(378.5 -40.5)))
         (set-car! ?route '((376.0 -41.5)(376.1 -49.0)))
         (set-cdr! ?route '(st-sd-1-c (s-1-c-w st-sd-1-1 st-sd-1-c))))
        ((equal? ?rectangle '((370.0 -50.0)(372.0 -42.0)))
         (set-car! ?route '((370.0 -58.0)))
         (set-cdr! ?route '(s-7-w (s-1-w s-6-w s-7-w))))
        ((equal? ?rectangle '((377.3 -50.0)(380.0 -41.8)))
         (set-car! ?route '((378.0 -58.0)))
         (set-cdr! ?route '(s-7-e (s-1-e s-6-e s-7-e))))))
  (!!walk :route (car ?route) :width *EVAC_WALK_WIDTH*
         :velocity *EVAC_WALK_VELOCITY* :acceleration *EVAC_WALK_ACCELERATION*
         :angle_velocity *EVAC_TURN_VELOCITY* :angle_acceleration *EVAC_TURN_ACCELERATION*)
  (bind ?from_coordinate (list (car (list-ref (car ?route) (- (length (car ?route)) 1)))
                               (get-height-with-areaid (cadr ?route))
                               (cadr (list-ref (car ?route) (- (length (car ?route)) 1)))
                               0.0))
  (!activate :cue (?position :name_s ?id
                          :from_coordinate ?from_coordinate
                          :distance_range *EVAC_DISTANCE_RANGE_CHECK_WALK*))
  (modify ?1 (route :id ?id :route (!value-of! ?route)))
  (!activate :cue (?position :name_s ?id
                          :rectangle ?rectangle))
  ;(display "evac-walk-1-7-1 : ")(display ?id)(newline)
  )

;;
;; 入ってはいけないエリアに入っており、
;;
(defrule
  evac-walk-1-7-2
  :priority 20

```

```

(bind ?1 (route :id ?id :route (!with-test! ?route (not (member (cadr ?route)
                                                                    '(st-sd-1-l st-sd-2-l st-sd-3-l
                                                                    st-sd-1-c st-sd-2-c
                                                                    st-sd-1-u st-sd-2-u)
                                                                    :mode (!with-test! ?mode (equal? ?mode 'evac-walk))))))
(bind ?2 (?position :id ?id
           :name ?id
           :rectangle (!with-test! ?rectangle
                        (member ?rectangle '((371.9 -44.0)(374.0 -41.2))
                                         ((374.0 -42.0)(374.2 -40.5))
                                         ((377.0 -42.0)(378.5 -40.5))
                                         ((370.0 -50.0)(372.0 -42.0))
                                         ((376.8 -50.0)(379.0 -42.0))))))
-->
;(display "evac-walk-1-7-2")(newline)
(!activate :cue (?position :name_s ?id
                  :rectangle ?rectangle))
(remove ?2))

;;
;; 歩行の終了の依頼が来れば、依頼通り歩行をやめ、歩行をやめているという述語に変更する
;;
(defrule
  evac-walk-1-8
  :priority 1000
  (bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name '(!"finish" "!!fi
                                                                    :option-list (!with-test! ?options (equal-opt-value? ?options "action" "walk
  (bind ?1 (route :id ?id :route ?route :mode ?mode))
  -->
  ;(display "evac-walk-1-8 : ")(display ?id)(newline)
  (!follow :request ?0)
  (modify ?1 (route :id ?id :route ?route :mode (!value-of! 'evac-stop)))
  (remove ?0))

;; 歩行中、前方の人とかなり接近したら、向きを少し変えて歩く。
(defrule
  evac-walk-1-9-1
  :priority 1000
  (bind ?0 (?position :id ?id :name_s ?name_s
                 :distance_range (!with-test! ?dis (equal? ?dis '(0.0 0.6)))
                 :angle_range (!with-test! ?angle_range (member ?angle_range '((0.0 90.0)(2
  (route :id ?id :route ?route )
  (not (is-following :id ?id :to ?to))
  -->
  (dbg '(,?id evac-walk-1-9-1))

```

```

(!!turn :right (if (equal? ?angle_range '(0.0 90.0)) 45.0 -45.0)
  :angle_velocity 360.0 :angle_acceleration 360.0)
(!!walk :route (car ?route) :width *EVAC_WALK_WIDTH*
  :velocity *EVAC_WALK_VELOCITY* :acceleration *EVAC_WALK_ACCELERATION*
  :angle_velocity *EVAC_TURN_VELOCITY* :angle_acceleration *EVAC_TURN_ACCELERATION*)
(!activate :cue (?position :name_s '() :distance_range ?dis :angle_range ?angle_range))
(remove ?0))

```

; 追隨中、前方の人とかなり接近したら、向きを少し変えて追隨を続ける。

```

(defrule
  evac-walk-1-9-2
  :priority 1000
  (bind ?0 (?position :id ?id :name_s ?name_s
    :distance_range (!with-test! ?dis (equal? ?dis '(0.0 0.6)))
    :angle_range (!with-test! ?angle_range (member ?angle_range '((0.0 90.0)(2
  (route :id ?id :route ?route)
  (is-following :id ?id :to (!with-test! ?to (number? ?to)))
  -->
  (dbg '(,?id evac-walk-1-9-2))
  (!!turn :right (if (equal? ?angle_range '(0.0 90.0)) 45.0 -45.0)
    :angle_velocity 360.0 :angle_acceleration 360.0)
  (!!approach :to_s ?to
    :velocity *EVAC_WALK_VELOCITY* :acceleration *EVAC_WALK_ACCELERATION*
    :angle_velocity *EVAC_TURN_VELOCITY* :angle_acceleration *EVAC_TURN_ACCELERATION*)
  (!activate :cue (?position :name_s '() :distance_range ?dis :angle_range ?angle_range))
  (remove ?0))

```

```

(defrule
  evac-walk-1-10-1
  :priority 1000
  (bind ?1 (?position :id ?id :name_s ?id
    :from_coordinate (!with-test! ?from_co (equal? ?from_co '(0.0 -12.0 0.0 0.0)
    :rectangle ?rectangle
    :height (!with-test! ?height (equal? ?height '(-0.3 0.3))))))
  (bind ?2 (route :id ?id :route ?old_route))
  -->
  (dbg '(,?id evac-walk-1-10-1))
  (bind ?old_route
    (cond ((equal? ?rectangle '((377.5 -55.0)(380.5 -41.9)))
      '(((378.8 -56.0)(378.0 -65.0)(377.5 -75.0)(377.0 -100.0)) s-7-e (s-1-e s-6-e s-7-
      ((equal? ?rectangle '((368.0 -55.0)(372.1 -41.9)))
        '(((371.2 -56.0)(370.5 -65.0)(370.0 -75.0)(369.0 -100.0)) s-7-w (s-1-w s-6-w s-7-
      (else
        ?old_route)))
  (!!walk :route (car ?old_route) :velocity 2.0 :acceleration 3.0

```

```

        :angle_velocity 360.0 :angle_acceleration 360.0)
(modify ?2 (route :id ?id :route (!value-of! ?old_route)))
(remove ?1))

```

;; 階段のところまできたら、階段を上る

```

(defrule
  evac-walk-4-1
  :priority 10000
  (bind ?1 (?position :id ?id :name_s ?id
                    :from_coordinate (!with-test! ?from_co (equal? ?from_co '(0.0 0.0 0.0 0.0)
                    :height (!with-test! ?height (equal? ?height '(-11.9 -7.6))))))
  (bind ?2 (route :id ?id :route ?route))
  -->
  (dbg '(,?id evac-walk-4-1))
  (!activate :cue (?position :name_s ?id :from_coordinate '(0.0 0.0 0.0 0.0) :height '(-11.9 -7.6)
  (!!walk :route '((373.0 -65.0)) :velocity 2.0 :acceleration 3.0
          :angle_velocity 360.0 :angle_acceleration 360.0)
  (modify ?2 (route :id ?id :route (!value-of! '(((373.0 -65.0)) d-2-e (d-1-e d-2-e))))))
  (remove ?1))

```

;;

;; 吸着誘導を行いながら避難を行っているときに、誘導対象者が自分から離れていれば

;; 誘導対象者の方向を向き、指示を送る。

;; 同時に、その誘導対象者の体の向きと距離を観測する。

;;

```

(defrule
  evac-guide-walk-1-1
  :priority 100
  (bind ?1 (is-guiding :id ?id :to ?to))
  (bind ?2 (?position :id ?id :from ?id :name_s ?name_s
                    :distance_range (!with-test! ?dis
                                          (equal? ?dis *DISTANCE_RANGE_OUT_OF_NEAR*))
                    :height (!with-test! ?height (equal? ?height *HEIGHT_EQ*))))
  (bind ?3 (route :mode ?mode))
  -->
  (!!turn :to ?to :angle_acceleration 360.0 :angle_velocity 360.0)
  (!!send :to_s ?to :sentence "私について来て下さい")
  (remove ?2)
  (test-func :cue (?observe :name_s ?to :action "approach"))
  (test-func :cue (?observe :name_s ?to :action "walk"))
  (!activate :cue (?position :from ?id :name_s ?name_s
                            :relative_angle '(270.0 450.0)
                            :distance_range *DISTANCE_RANGE_OUT_OF_NEAR* :height *HEIGHT_EQ*))
  (!activate :cue (?position :from ?id :name_s ?name_s

```

```

                :distance_range *DISTANCE_RANGE_NEAR* :height *HEIGHT_EQ*))
(modify ?3 (route :mode (!value-of! 'evac-stop))))

;; 誘導の対象者が近くまで来たら、避難経路を歩き始める。
(defrule
  evac-guide-walk-1-2
  :priority 100
  (bind ?1 (is-guiding :id ?id :to ?to))
  (bind ?2 (?position :id ?id :from ?id :name_s ?name_s
                    :distance_range (!with-test! ?dis
                                       (equal? ?dis *DISTANCE_RANGE_NEAR*))
                    :height (!with-test! ?height (equal? ?height *HEIGHT_EQ*)))
  (bind ?3 (route :id ?id :route ?route :mode ?mode))
  -->
  (bind ?velocity
    (cond ((member (cadr ?route) '(st-sd-1-u st-sd-2-u st-sd-3-u))
           (* *EVAC_WALK_VELOCITY* 0.5))
          ((member (cadr ?route) '(st-sd-1-c st-sd-2-c st-sd-3-c))
           (* *EVAC_WALK_VELOCITY* 0.6))
          (else *EVAC_WALK_VELOCITY*)))
  (bind ?width
    (cond ((member (cadr ?route) '(st-sd-1-u st-sd-2-u))
           (* *EVAC_WALK_WIDTH* 1.1))
          ((member (cadr ?route) '(st-sd-1-c st-sd-2-c))
           (* *EVAC_WALK_WIDTH* 1.2))
          (else *EVAC_WALK_WIDTH*)))
  (!!walk :route (car ?route) :width ?width :velocity ?velocity :acceleration *EVAC_WALK_ACCELERATION*
         :angle_velocity *EVAC_TURN_VELOCITY* :angle_acceleration *EVAC_TURN_ACCELERATION*)
  (remove ?2)
  (!activate :cue (?position :from ?id :name_s ?name_s
                           :distance_range *DISTANCE_RANGE_OUT_OF_NEAR* :height *HEIGHT_EQ*))
  (modify ?3 (route :mode (!value-of! 'evac-walk))))

(defrule
  evac-guide-walk-1-3
  :priority 110
  (bind ?1 (is-guiding :id ?id :to ?to))
  (bind ?2 (know-moving :id ?id :name_s (!with-test! ?name_s_move (not (equal? ?name_s_move '())))
                       :action (!with-test! ?action (member ?action '(("walk" "approach")("approach")))))
  (bind ?3 (route :id ?id :route ?route :mode ?mode))
  -->
  (bind ?velocity
    (cond ((member (cadr ?route) '(st-sd-1-u st-sd-2-u st-sd-3-u))
           (* *EVAC_WALK_VELOCITY* 0.5))
          ((member (cadr ?route) '(st-sd-1-c st-sd-2-c st-sd-3-c))
           (* *EVAC_WALK_VELOCITY* 0.5))))

```

```

        (* *EVAC_WALK_VELOCITY* 0.6))
        (else *EVAC_WALK_VELOCITY*))
(bind ?width
  (cond ((member (cadr ?route) '(st-sd-1-u st-sd-2-u))
        (* *EVAC_WALK_WIDTH* 1.1))
        ((member (cadr ?route) '(st-sd-1-c st-sd-2-c))
        (* *EVAC_WALK_WIDTH* 1.2))
        (else *EVAC_WALK_WIDTH*)))
(!!walk :route (car ?route) :width ?width :velocity ?velocity :acceleration *EVAC_WALK_ACCELERATION*
        :angle_velocity *EVAC_TURN_VELOCITY* :angle_acceleration *EVAC_TURN_ACCELERATION*)
(modify ?3 (route :id ?id :route ?route :mode (!value-of! 'evac-walk)))
(remove ?2)
(remove ?1))

(defrule
  evac-guide-walk-1-4
  :priority 110
  (bind ?1 (is-guiding :id ?id :to ?to))
  (bind ?2 (know-moving :id ?id :name_s (!with-test! ?name_s_move (equal? ?name_s_move '()))
                        :action (!with-test! ?action (member ?action ('("walk" "approach"))("approach")))
                        (route :id ?id :route ?route)
                        -->
                        (!!turn :to ?to :angle_acceleration 360.0 :angle_velocity 360.0)
                        (!!send :to_s ?to :sentence "私について来て下さい")
                        (remove ?2)
                        (test-func :cue (?observe :name_s ?to :action "approach"))
                        (test-func :cue (?observe :name_s ?to :action "walk")))))
;;
;; 避難完了確認済みのメッセージを受けたら、
;; 避難完了の述語を作成し、すべての移動アクションを終える。
;;
(defrule
  reach-goal-1
  :priority 30000
  (bind ?0 (request :id ?id :request-name (!with-test! ?r-name (or (equal? ?r-name "!send")(equal? ?r-name "避難完了を確認"))))
           :option-list (!with-test! ?options
                             (equal-opt-value? ?options "sentence"
                                                  "避難完了を
確認"))))
  (bind ?1 (agent-type :id ?id))
  (bind ?2 (in-evac :id ?id))
  -->
  (dbg "reach-goal-1")
  (!!finish :action "walk"))

```

```

(!!finish :action "approach")
(make (has-reached-goal :id ?id ))
(!follow :request ?0)
(remove ?0)
(remove ?1)
(remove ?2)
)

(defrule
reach-goal-2-1
:priority 1000
(has-reached-goal :id ?id)
(bind ?1 (route :id ?id))
-->
(remove ?1))

(defrule
reach-goal-2-2
:priority 1000
(has-reached-goal :id ?id)
(bind ?1 (?position :id ?id))
-->
(remove ?1))

(defrule
reach-goal-2-3
:priority 1000
(has-reached-goal :id ?id)
(bind ?1 (is-following :id ?id))
-->
(remove ?1))

(defrule
reach-goal-2-4
:priority 1000
(has-reached-goal :id ?id)
(bind ?1 (is-guiding :id ?id))
-->
(remove ?1))

(defrule
reach-goal-2-5
:priority 1000
(has-reached-goal :id ?id)
(bind ?1 (know-moving :id ?id))
-->
(remove ?1))

```

```

;;
;; 発話の要求があった場合、発話を実行するとともに、メッセージの送信も行う。
;;
(defrule
  send-with-speak-1
  :priority 120
  (agent-type :id ?id :type (!with-test! ?type (equal? ?type 'scenario-skyoto-special2)))
  (bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name
                                                                    '("!speak" "!!speak" "!sen
                                                                    :option-list (!with-test! ?options (member (get-opt-value ?options "sentence"
                                                                    '("あちらへ避難し
てください" "ただいまから避難訓練を開始します。みなさん階段を使って速やかに上の階まで
避難してください。")))))
  -->
  (dbg '(,?id send-with-speak))
  (!!send :to_s (get-opt-value ?options "to_s") :sentence (get-opt-value ?options "sentence"))
  (make-guide-action (list-include? (get-opt-value ?options "to_s") *do-guide-ps*)
                     (get-opt-value ?options "sentence") ?id)
  (!follow :request ?0)
  (remove ?0))

(defrule
  send-with-speak-2
  :priority 120
  (bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name
                                                                    '("!speak" "!!speak" "!sen
                                                                    :option-list (!with-test! ?options (equal-opt-value? ?options "sentence" "
ただいまから避難訓練を開始します。みなさん階段を使って速やかに上の階まで避難してくだ
さい。"))
  :cont ?cont))
  -->
  (dbg '(,?id send-with-speak-2))
  (set-opt-value! ?options "in" "to_s" (append (return-list (get-opt-value ?options "to_s")) *sp
  (!!send :to_s (get-opt-value ?options "to_s") :sentence (get-opt-value ?options "sentence"))
  (make-guide-action (list-include? (get-opt-value ?options "to_s") *do-guide-ps*)
                     (get-opt-value ?options "sentence") ?id)
  (!follow :request ?0)
  (remove ?0))

(defrule
  send-with-speak-3
  :priority 120
  (agent-type :id ?id :type (!with-test! ?type (equal? ?type 'scenario-skyoto-guide2)))
  (bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name
                                                                    '("!speak" "!!speak" "!sen

```

```

:option-list (!with-test! ?options (equal-opt-value? ?options "sentence" "
あちらへ避難してください))))))
-->
(dbg '(,?id send-with-speak-3))
(bind ?to (return-list (get-opt-value ?options "to_s")))
 (!!face :to ?to :angle_velocity 360.0 :angle_acceleration 360.0)
(set-opt-value! ?options "in" "to_s" ?to)
 (!!send :to_s ?to :sentence "あちらへ避難してください")
(make-guide-action (list-include? ?to *do-guide-ps*) "あちらへ避難してください" ?id)
(bind ?0 (modify ?0 (request :id ?id :request-name ?r-name :option-list ?options :cont ?cont)))
(!follow :request ?0)
(remove ?0))

;;
;; メッセージ送信の要求があった場合、そのままメッセージの送信を行う。
;;
(defrule
send-only
:priority 20
(bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name ('(!send" !!!send"
:option-list ?options)))
-->
(dbg '(,?id send-only))
(make-guide-action (list-include? (get-opt-value ?options "to_s") *do-guide-ps*)
(get-opt-value ?options "sentence") ?id)
(!follow :request ?0)
(remove ?0))

;;
;; 避難完了確認のための発話の要求は、相手を変更する（すでに避難完了のエージェントを対
象からはずす。)。
;;
(defrule
speak-finish-sentence
:priority 140
(bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name
('(!speak" !!!speak" !!!sen
:option-list (!with-test! ?options
(equal-opt-value? ?options "sentence" "
あなたの避難完了を確認しました。))))))
(bind ?1 (know_as_evacuee_finished :id ?id :evacuee ?evacuee))
-->
(dbg '(,?id speak-finish-sentence))
(bind ?to_s (check-finish-agents2 (get-opt-value ?options "to_s") ?evacuee ?id))
(bind ?evacuee

```

```

    (if (not (equal? (return-list ?id) ?to_s))
        (if (null? ?evacuee)
            ?to_s
            (append ?evacuee ?to_s))
        ?evacuee))
(dbg '(evacuee : ,?evacuee))
(modify ?1 (know_as_evacuee_finished :id ?id :evacuee ?evacuee))
(if (not (equal? (list ?id) ?to_s))
    (begin
        (set-opt-value! ?options "in" "to_s" ?to_s)
        (make-guide-action (list-include? ?to_s *do-guide-ps*)
            (get-opt-value ?options "sentence") ?id)
    )
    (set-opt-value! ?options "in" "to_s" ?id)
)
 (!!send :to_s ?to_s :sentence (get-opt-value ?options "sentence"))
(bind ?0 (modify ?0 (request :id ?id :request-name ?r-name :option-list ?options)))
(!follow :request ?0)
(remove ?0))

;;
;; 吸着誘導のための発話は、相手を記憶する。
;;
(defrule
speak-follow-me-sentence
:priority 140
(bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name
                                                                    '(!speak" !!!speak" !!!sen
                                                                    :option-list (!with-test! ?options
                                                                    (equal-opt-value? ?options "sentence" "
これから避難します。私について来て下さい。))))))
(not (is-guiding :id ?id))
-->
(dbg '(,?id speak-follow-me-sentence))
(bind ?name (return-one (get-opt-value ?options "to_s")))
 (!!turn :to ?name)
(make (is-guiding :id ?id :to (!value-of! ?name)))
(!activate :cue (?position :from ?id :name_s ?name
                        :distance_range *DISTANCE_RANGE_OUT_OF_NEAR* :height *HEIGHT_EQ*))
 (!!send :to_s (get-opt-value ?options "to_s") :sentence (get-opt-value ?options "sentence"))
(make-guide-action (list-include? (return-list ?name) *do-guide-ps*)
    (get-opt-value ?options "sentence") ?id)
(bind ?0 (modify ?0 (request :id ?id :options (!value-of! (set-opt-value! ?options "in" "to_s"
(!follow :request ?0)
(remove ?0))

```

```

;;
;;
;;
(defrule
  speak-exam-completed
  :priority 100
  (know_as_evacuee_finished :id ?id :evacuee (!with-test! ?evacuee (>= (length ?evacuee) (length
  -->
  (!!send :to_s ?id :sentence "避難訓練を終了"))

(defrule
  send-message-2-1
  :priority 10000
  (bind ?1 (send-request :id ?id :to_s ?to_s :sentence ?sentence))
  -->
  (!!send :to_s ?to_s :sentence ?sentence)
  (remove ?1))

(defrule
  send-message-2-2
  :priority 10000
  (bind ?1 (guide-action :id ?id :to ?to :message ?message))
  -->
  (display "send-message-2-2")(newline)
  (!!speak :to_s ?to :sentence ?message)
  (!guide :type "email" :recipient (get-email-address ?to) :message ?message)
  (remove ?1))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; 誘導用ルール
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; 指差誘導
;;
(defrule
  is-pointing-1
  :priority 200
  (bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name ('(!point" "!!point"
  :option-list ?options))
  (route :id ?id :route ?route)
  (agent-type :id ?id :type (!with-test! ?type (equal? ?type 'scenario-skyoto-guide2)))
  -->
  (dbg '(,?id is-pointing-1))
  (make (is-pointing :id ?id :to_coordinate (!value-of! (caar ?route)) :time (!value-of! (current

```

```

(set-opt-value! ?options "in" "to_coordinate" (caar ?route))
 (!!turn :to_coordinate (caar ?route) :angle_velocity 360.0 :angle_acceleration 360.0)
(bind ?0 (modify ?0 (request :id ?id :option-list ?options)))
(!follow :request ?0)
(remove ?0)

```

```

(defrule
  is-pointing-2
  :priority 200
  (bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name '("!finish" "!!fin
    :option-list (!with-test! ?options (equal-opt-value? ?options "action" "point
  (bind ?1 (is-pointing :id ?id))
  -->
  (dbg '(,?id is-pointing-2))
  (remove ?1)
  (!follow :request ?0)
  (!behave :gesture "standing")
  (remove ?0))

```

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

```

```

(defrule
  turn-1
  :priority 200
  (bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name '("!turn" "!!turn
    :option-list ?options))
  (route :id ?id :route ?route)
  (agent-type :id ?id :type (!with-test! ?type (equal? ?type 'scenario-skyoto-guide2)))
  -->
  (set-opt-value! ?options "in" "to_coordinate" (caar ?route))
  (bind ?0 (modify ?0 (request :id ?id :option-list ?options)))
  (!follow :request ?0)
  (remove ?0))

```

```

(defrule
  turn-2-1
  :priority 200
  (bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name '("!turn" "!!turn
    :option-list (!with-test! ?options (not (equal? #f (get-opt-value ?options "
  (agent-type :id ?id :type (!with-test! ?type (member ?type '
    scenario-skyoto-evacuee1
    scenario-skyoto-evacuee2
    scenario-skyoto-evacuee3
    scenario-skyoto-evacuee4
    ))))

```

```

(not (is-turning :id ?id :to ?to))
-->
(display '(,?id turn-2-1))(newline)
(bind ?options (set-opt-value! ?options "in" "to" (return-one (get-opt-value ?options "to"))))
(bind ?0 (modify ?0 (request :id ?id :request-name ?r-name :options ?options)))
(!follow :request ?0)
(display '(,?id ,?options ,(get-opt-value ?options "to") ,(return-list (get-opt-value ?options
(display '(,?id ,(return-one (get-opt-value ?options "to"))))(newline)
(make (is-turning :id ?id :to (!value-of! (return-one (get-opt-value ?options "to")))
      :time (!value-of! (current-seconds))))
(test-func :cue (?observe :name_s (return-one (get-opt-value ?options "to"))
           :action "point"))

(remove ?0))

(defrule
turn-2-2
:priority 200
(bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name '(!"turn" "!!turn"
      :option-list (!with-test! ?options (not (equal? #f (get-opt-value ?options "
(bind ?1 (is-turning :id ?id :to ?to))
(agent-type :id ?id :type (!with-test! ?type (member ?type '(scenario-skyoto-evacuee1
      scenario-skyoto-evacuee2
      scenario-skyoto-evacuee3
      scenario-skyoto-evacuee4
      ))))
-->
(display '(,?id turn-2-2))(newline)
(bind ?options (set-opt-value! ?options "in" "to" (return-one (get-opt-value ?options "to"))))
(bind ?0 (modify ?0 (request :id ?id :request-name ?r-name :options ?options)))
(!follow :request ?0)
(modify ?1 (is-turning :id ?id :to (!value-of! (return-one (get-opt-value ?options "to")))
      :time (!value-of! (current-seconds))))
(test-func :cue (?observe :name_s (return-one (get-opt-value ?options "to"))
           :action "point"))

(remove ?0))

(defrule
check_point_angle-1
:priority 200
(bind ?1 (true :id ?id :cue (?observe :id ?id :name_s ?name_s :action (!with-test! ?action (eq
(bind ?2 (is-turning :id ?id :to (!with-test! ?to (number? ?to))))
(test (member ?to (return-list ?name_s)))
-->
(debug '(,?id check_point_angle-1))(newline)
(test-func :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle

```

```

(test-func :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle
(test-func :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle
(test-func :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle
(test-func :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle
(test-func :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle
(test-func :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle
(test-func :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle
(!activate :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle
(!activate :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle
(!activate :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle
(!activate :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle
(!activate :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle
(!activate :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle
(!activate :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle
(!activate :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle
(!activate :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle
(!activate :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle
(!activate :cue (?position :name_s ?to :from_coordinate '(0.0 0.0 -100000.0 0.0) :point_angle
(remove ?1)
)

```

;; 指差しの先を向き、その方向に通路があるかどうかを調べる。

```

;;
(defrule
  check-point-angle-2
  :priority 100
  (bind ?1 (know-pointing :id ?id :name_s ?to :point_angle ?point_angle))
  (bind ?2 (is-turning :id ?id :to ?to :angle ?angle))
  -->
  (dbg '(,?id check-point-angle-2 ?point_angle ,?point_angle))
  (!!turn :angle (/ (+ (car ?point_angle)(cadr ?point_angle)) 2.0) :angle_velocity 720.0 :angle_...
  (modify ?2 (is-turning :id ?id :to ?to :angle ?point_angle))
  (test-func :cue (?position :name_s ?id :from_coordinate '(375.5 -12.1 -42.0 0.0)
                  :relative_angle '(170.0 190.0)))
  (test-func :cue (?position :name_s ?id :from_coordinate '(378.5 -12.1 -42.0 0.0)
                  :relative_angle '(170.0 190.0)))
  (test-func :cue (?position :name_s ?id :from_coordinate '(371.5 -12.1 -42.0 0.0)
                  :relative_angle '(170.0 190.0)))
  (test-func :cue (?position :name_s ?id :from_coordinate '(379.5 -12.1 -2.0 0.0)
                  :relative_angle '(170.0 190.0)))
  (test-func :cue (?position :name_s ?id :from_coordinate '(371.5 -12.1 -2.0 0.0)
                  :relative_angle '(170.0 190.0)))
  (remove ?1))

```

;; 自分の向いている方向に通路があれば、その方向を向いているという述語を作る

;;

(defrule

check-point-angle-3-1

:priority 400

(bind ?1 (true :cue (?position :id ?id :name\_s ?id :from\_coordinate (!with-test! ?from\_co (list  
:relative\_angle (!with-test! ?r-angle (equal? ?r-angle '(170.0

(bind ?2 (is-turning :id ?id :to\_coordinate ?to\_co))

-->

(dbg '(,?id check-point-angle-3-1 ,?from\_co))

(modify ?2 (is-turning :id ?id :to\_coordinate ?from\_co))

(remove ?1))

;;

;;

(defrule

check-point-angle-3-2

:priority 600

(bind ?1 (false :cue (?position :id ?id :name\_s ?id :from\_coordinate (!with-test! ?from\_co (list  
:relative\_angle (!with-test! ?r-angle (equal? ?r-angle '(170.0

(is-turning :id ?id :to\_coordinate ?to\_co))

-->

(dbg '(,?id check-point-angle-3-2 ,?from\_co))

(remove ?1))

;; 他人に近づく依頼がきたとき、その近づく対象の相手の指差している方向に

;; 通路があることがわかっていれば近づかずにその通路のほうに歩く

(defrule

evac-approach-4-1-1

:priority 200

(bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name ('(!approach" "!!  
"!!walk" "!!walk"

:option-list ?options))

(bind ?1 (agent-type :id ?id :type (!with-test! ?type (member ?type (list 'scenario-skyoto-evac  
'scenario-skyoto-evac

(bind ?2 (is-turning :id ?id :to\_coordinate (!with-test! ?to\_co (list? ?to\_co))  
:angle (!with-test! ?angle (list? ?angle))))

(bind ?3 (route :route ?old\_route :mode ?mode))

-->

(dbg '(,?id evac-approach-4-1-1))(newline)(newline)(newline)

(bind ?0 (modify ?0 (request :id ?id :request-name (!value-of! "!!turn"

:option-list (!value-of! '(, (append (list "in" "angle") (/ (+ (car  
, (append (list "in" "angle\_velocity")  
, (append (list "in" "angle\_acceleration"  
))))))



```

(!follow :request ?0)
(bind ?route
  '(,(round-ex (+ 0.5 (* 2.0 (sin (deg->rad (/ (+ (car ?angle)
                                                    (cadr ?angle)) 2)))) 0.01)
    ,(round-ex (+ 0.5 (* 2.0 (cos (deg->rad (/ (+ (car ?angle)
                                                    (cadr ?angle)) 2)))) 0.01)))
  )
(!!walk :route (return-list ?route)
  :velocity (get-opt-value ?options "velocity")
  :acceleration (get-opt-value ?options "acceleration")
  :angle_velocity (return-one (get-opt-value ?options "angle_velocity"))
  :angle_acceleration (return-one (get-opt-value ?options "angle_acceleration"))
  :relative #t)
(remove ?2)
(remove ?0))

;; 他人に近づく依頼がきたとき、その近づく対象の相手が指差ししていなければ
;; その人に近づく
(defrule
  evac-approach-4-2-1
  :priority 200
  (agent-type :id ?id :type (!with-test! ?type (member ?type (list 'scenario-skyoto-evacuee3
                                                                    'scenario-skyoto-evacuee4))))
  (bind ?2 (is-turning :id ?id :to (!with-test! ?to (number? ?to))))
  (bind ?1 (false :id ?id :cue (?observe :id ?id :name_s ?name_s :action (!with-test! ?action (e
  (bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name ('!approach" "!!
    :option-list ?options))
  -->
  (dbg '(,?id evac-approach-4-2-1))(newline)(newline)(newline)
  (set-opt-value! ?options "in" "to_s" ?to)
  (bind ?0 (modify ?0 (request :id ?id :request-name ?r-name :option-list ?options)))
  (!follow :request ?0)
  (!activate :cue (?position :from ?id :name_s ?to
    :distance_range *DISTANCE_RANGE_OUT_OF_NEAR* :height *HEIGHT_EQ*))
  (remove ?0)
  (remove ?1)
  (remove ?2)
  (make (is-following :id ?id :to ?to :time (!value-of! (current-seconds))))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; 追従ルール
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; ついていく依頼が来たとき、まずその相手が視野の中にいるのか、近くにいるのかどうかを
調べ
;; いれば「追従中」の述語を作成しその相手についていく

```

```

;; いなければ、直ちに付随はやめ、ほかに対象者がいないかを観測し始める。
;;
;;
;; 追隨中に改札口のところまできたら追隨をやめ、自分で改札口まで歩いていく
;; (つじつまあわせ：修正要)
;;
(defrule
follow-0-0-1
:priority 1000
(in-evac :id ?id)
(bind ?2 (?position :id ?id :name_s ?id
:from_coordinate (!with-test! ?from_co (equal? ?from_co '(0.0 0.0 0.0 0.0))
:rectangle (!with-test! ?rectangle
(equal? ?rectangle '((370.0 -55.0)(378.0 -70.0)))
:height (!with-test! ?height (equal? ?height '(-8.0 -7.0))))))
(bind ?3 (route :id ?id :route (!with-test! ?route
(member (caaddr ?route) '(d-1 st-sd-1-u st-sd-2-u s
:mode ?mode)))
-->
(display "follow-0-0-1")(newline)
(modify ?3 (route :id ?id :route (!value-of! '((372.0 -65.0)) d-2-e (d-1 d-2-e)))
:mode (!value-of! 'evac-walk)))
 (!!walk :route '((372.0 -65.0)) :velocity 4.0)
(remove ?2))

;; 2つ以上の追隨述語ができていれば古いほうを削除
(defrule
follow-0-0-2
:priority 100000
(bind ?1 (is-following :id ?id :time ?time1))
(is-following :id ?id :time ?time2)
(test (< ?time1 ?time2))
-->
(display '(,?id delete-following))(newline)
(remove ?1))

;; 追隨する依頼で相手が指定されていれば、その相手についていく。
;; 同時にその相手に追隨中であるという述語を作成
;;
(defrule
follow-0-1-1
:priority 50
(bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name '("!!approach" "!!
:option-list (!with-test! ?options (not (equal-opt-value? ?options "to_s" '(

```

```

                :cont ?cont))
(agent-type :id ?id :type (!with-test! ?type (member ?type '(scenario-skyoto-evacuee2
                                                                scenario-skyoto-evacuee3
                                                                scenario-skyoto-evacuee4))))

(in-evac :id ?id)
-->
(bind ?to (return-one (get-opt-value ?options "to_s")))
(set-opt-value! ?options "in" "to_s" ?to)
(bind ?0 (modify ?0 (request :id ?id
                             :option-list ?options)))
(!activate :cue (?position :from ?id :name_s ?name
                       :distance_range *DISTANCE_RANGE_OUT_OF_NEAR* :height *HEIGHT_EQ*))
(!follow :request ?0)
(make (is-following :id ?id :to ?to :time (!value-of! (current-seconds))))
(remove ?0)
)

```

;; 追隨する依頼で相手が指定されていなければ、周りのエージェントを観測。

;; 同時にだれにも追隨していないという述語を作成。

;;

```

(defrule
follow-0-1-2
:priority 50
(bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name '(!!"approach" "!"
                                                                :option-list (!with-test! ?options (equal-opt-value? ?options "to_s" '()))
                                                                :cont ?cont))
(agent-type :id ?id :type (!with-test! ?type (member ?type '(scenario-skyoto-evacuee2
                                                                scenario-skyoto-evacuee4))))

(in-evac :id ?id)
(not (is-facing :id ?id))
-->
(make (is-following :id ?id :to (!value-of! '()) :time (!value-of! (current-seconds))))
(bind ?angle_to_body (* 30.0 (- (random 7) 3)))
(make (is-facing :id ?id :angle_to_body ?angle_to_body))
;(!face :angle_to_body ?angle_to_body)
(test-func :cue (?observe :name_s '() :action "approach"))
(test-func :cue (?observe :name_s '() :action "walk"))
(test-func :cue (?position :name_s '() :from ?id
                          :angle_range *ANGLE_RANGE_VISIBLE*
                          :relative_angle '(-45.0 45.0)
                          :distance_range *DISTANCE_RANGE_NEAR* :height *HEIGHT_EQ*))

(remove ?0)
(call (apply (return-one ?cont) '(())))
)

```

;; 視野の中に歩行中のエージェントがいるなら  
;; そのエージェントに近づくとともに、追隨中の述語を作る。

```
(defrule
follow-0-1-3-1
:priority 1000
(agent-type :id ?id :type (!with-test! ?type (member ?type '(scenario-skyoto-evacuee2
                                                                    scenario-skyoto-evacuee4))))

(in-evac :id ?id)
(bind ?1 (see-in-view :id ?id :name_s (!with-test! ?name_s_seen (not (equal? ?name_s_seen '()))
                        :relative_angle (!with-test! ?relative_angle
                                            (equal? ?relative_angle '(-45.0 45.0))))))
(bind ?2 (know-moving :id ?id :name_s (!with-test! ?name_s_move (not (equal? ?name_s_move '()))
                        :action (!with-test! ?action (member ?action '(("walk" "approach")("approach")))))
(bind ?4 (is-facing :id ?id :angle_to_body ?angle_to))
(bind ?5 (is-following :id ?id :to ?to))
(test (not (null? (list-include? ?name_s_seen ?name_s_move))))
-->
(bind ?name (car (list-include? ?name_s_seen ?name_s_move)))
;(!face :angle_to_body 0.0)
(!turn :to ?name)
(!!approach :to_s ?name :distance *APPROACH_DISTANCE*
            :velocity (* *EVAC_WALK_VELOCITY* 1.2) :acceleration (* *EVAC_WALK_ACCELERATION* 2)
            :angle_velocity *EVAC_TURN_VELOCITY* :angle_acceleration *EVAC_TURN_ACCELERATION*)
(!activate :cue (?position :from ?id :name_s ?name
                    :distance_range *DISTANCE_RANGE_OUT_OF_NEAR* :height *HEIGHT_EQ*))
(modify ?5 (is-following :id ?id :to (!value-of! ?name) :time (!value-of! (current-seconds))))
(remove ?1)
(remove ?2)
(remove ?4)
)
```

;; 視野の中に歩行中のエージェントがいなければ  
;; 他の方向を向き、その方向にエージェントがいなければどうか観測する  
;;

```
(defrule
follow-0-1-3-2
:priority 100
(agent-type :id ?id :type (!with-test! ?type (member ?type '(scenario-skyoto-evacuee2
                                                                    scenario-skyoto-evacuee4))))

(in-evac :id ?id)
(bind ?1 (see-in-view :id ?id :name_s ?name_s_seen
                    :relative_angle (!with-test! ?relative_angle
                                            (equal? ?relative_angle '(-45.0 45.0))))))
```

```

(bind ?2 (know-moving :id ?id :name_s ?name_s_move
                      :action (!with-test! ?action (member ?action '(("walk" "approach"))("approach"))))
(bind ?4 (is-facing :id ?id :angle_to_body ?angle_to))
-->
(debug '(,?id follow-0-1-3-2))
(bind ?angle_to (if (>= ?angle_to 0.0)(- ?angle_to 40.0)(+ ?angle_to 50.0)))
(!!turn :right ?angle_to :angle_velocity 360.0 :angle_acceleration 360.0)
(modify ?4 (is-facing :id ?id :angle_to_body ?angle_to))
(test-func :cue (?observe :name_s '() :action "walk"))
(test-func :cue (?observe :name_s '() :action "approach"))
(test-func :cue (?position :name_s '() :from ?id
                       :angle_range *ANGLE_RANGE_VISIBLE*
                       :relative_angle ?relative_angle
                       :distance_range *DISTANCE_RANGE_NEAR* :height *HEIGHT_EQ*))

(remove ?1)
(remove ?2)
)

```

```
;;
```

```
;; 追跡中に距離が離れれば、追跡の相手の姿が見えるかどうか観測する。
```

```
;;
```

```

(defrule
follow-1-4-1-1-1
:priority 100
(bind ?1 (?position :id ?id :name_s ?name_s :from ?id
                   :distance_range (!with-test! ?dis (equal? ?dis *DISTANCE_RANGE_OUT_OF_NEAR*))
                   :height (!with-test! ?height (equal? ?height *HEIGHT_EQ*))))
(agent-type :id ?id :type (!with-test! ?type (member ?type '(scenario-skyoto-evacuee2
                                                            scenario-skyoto-evacuee4))))

(in-evac :id ?id)
(is-following :id ?id :to (!with-test! ?to (not (equal? ?to '()))))
-->
(debug '(,?id follow-1-4-1-1-1))(newline)
(!!turn :to ?to :angle_velocity 720.0 :angle_acceleration 720.0)
(test-func :cue (?see :name (return-one ?to)))
(remove ?1)

```

```
;;
```

```
;; 無駄なルール
```

```
;; 余計な観測を行ってしまったので無視する
```

```
;;
```

```

(defrule
follow-1-4-1-1-2

```

```

:priority 100
(bind ?1 (?position :id ?id :name_s ?name_s :from ?from
          :distance_range (!with-test! ?dis (equal? ?dis *DISTANCE_RANGE_OUT_OF_NEAR*
          :height (!with-test! ?height (equal? ?height *HEIGHT_EQ*))))
(agent-type :id ?id :type (!with-test! ?type (member ?type '(scenario-skyoto-evacuee2
          scenario-skyoto-evacuee4))))

(in-evac :id ?id)
(is-following :id ?id :to (!with-test! ?to (equal? ?to '())))
-->
(dbg '(,?id follow-1-4-1-1-2))(newline)
(remove ?1)

;;
;;(追隨中に距離が離れ、)追隨の相手が見えれば、その相手へいそいで近づく
;;
(defrule
follow-1-4-1-2
:priority 100
(agent-type :id ?id :type (!with-test! ?type (member ?type '(scenario-skyoto-evacuee2
          scenario-skyoto-evacuee4))))

(in-evac :id ?id)
(bind ?1 (true :id ?id :cue (?see :id ?id :name ?name_s)))
(is-following :id ?id)
-->
(dbg '(,?id follow-1-4-1-2))(newline)
(!!approach :to_s ?name_s :distance *APPROACH_DISTANCE*
            :velocity (* *EVAC_WALK_VELOCITY* 1.2) :acceleration (* *EVAC_WALK_ACCELERATION* 2)
            :angle_velocity *EVAC_TURN_VELOCITY* :angle_acceleration *EVAC_TURN_ACCELERATION*)
(!activate :cue (?position :name_s ?name_s :from ?id
                    :distance_range *DISTANCE_RANGE_OUT_OF_NEAR*
                    :height *HEIGHT_EQ*))

(remove ?1)

;;
;; 追隨中に距離が離れ、追隨の相手が見えなければ、その相手への追隨をやめ
;; 周辺をもう一度観測しなおす。
;;
(defrule
follow-1-4-1-3
:priority 100
(agent-type :id ?id :type (!with-test! ?type (member ?type '(scenario-skyoto-evacuee2
          scenario-skyoto-evacuee4))))

(in-evac :id ?id)
(bind ?1 (false :id ?id :cue (?see :id ?id :name ?name)))
(bind ?2 (is-following :id ?id :to ?to))

```

```

-->
(dbg '(,?id follow-1-4-1-3))(newline)
(!finish :action "turn")
(bind ?angle_to (round (* (- (random 7) 3.5) 30.0)))
(make (is-facing :id ?id :angle_to_body ?angle_to))
(test-func :cue (?observe :name_s '() :action "approach"))
(test-func :cue (?observe :name_s '() :action "walk"))
(test-func :cue (?position :name_s '() :from ?id
                    :angle_range *ANGLE_RANGE_VISIBLE*
                    :relative_angle '(-45.0 45.0)
                    :distance_range *DISTANCE_RANGE_NEAR* :height *HEIGHT_EQ*))
(modify ?2 (is-following :id ?id :to (!value-of! '())))
(remove ?1)

(defrule
follow-1-5-1
:priority 200
(bind ?1 (is-following :id ?id :to ?to))
(bind ?2 (?position :id ?id :name_s ?to :from_coordinate (!with-test ?from (equal? ?from '(0.0
:rectangle (!with-test! ?rec (equal? ?rec '((376.0 -70.0)(390.0 -55.0))))
:height (!with-test! ?height (equal? ?height '(-8.0 -7.0))))))
-->
(dbg '(,?id follow-1-5))(newline)
(!finish :action "approach")
(remove ?2)
(remove ?1)
(!!walk :route '((373.0 -65.0)) :velocity 3.0))

(defrule
follow-1-5-2
:priority 200
(bind ?1 (is-following :id ?id :to ?to))
(bind ?2 (?position :id ?id :name_s ?to :from_coordinate (!with-test ?from (equal? ?from '(0.0
:rectangle (!with-test! ?rec (equal? ?rec '((376.0 -100.0)(390.0 -57.0))))
:height (!with-test! ?height (equal? ?height '(-12.5 -11.5))))))
-->
(!finish :action "approach")
(remove ?2)
(remove ?1)
(!!walk :route '((377.5 -60.0)(377.3 -70.0)(377.0 -90.0)) :velocity 3.0))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; 視野内のエージェントの認識（主語のエージェントに対する向きも付加）
;;

```

```

(defrule
see-in-view-1-1
:priority 10000
(bind ?1 (true :id ?id :cue (?position :id ?id :name_s ?name_s :from ?id
                                :angle_range (!with-test! ?angle_range (equal? ?angle_r
                                :relative_angle ?relative_angle
                                :distance_range (!with-test! ?dis (equal? ?dis *DISTANC
                                :height (!with-test! ?height (equal? ?height *HEIGHT_EQ*))

(not (see-in-view :id ?id :name_s ?n))
-->
(make (see-in-view :id ?id :name_s ?name_s
                    :relative_angle (!value-of! ?relative_angle)))

(remove ?1))

(defrule
see-in-view-1-2
:priority 10000
(bind ?1 (true :id ?id :cue (?position :id ?id :name_s ?name_s :from ?id
                                :angle_range (!with-test! ?angle_range (equal? ?angle_r
                                :relative_angle ?relative_angle
                                :distance_range (!with-test! ?dis (equal? ?dis *DISTANC
                                :height (!with-test! ?height (equal? ?height *HEIGHT_EQ

(bind ?2 (see-in-view :id ?id :name_s ?n))
-->
(modify ?2 (see-in-view :id ?id :name_s ?name_s
                        :relative_angle (!value-of! ?relative_angle)))

(remove ?1))

(defrule
see-in-view-2
:priority 10000
(bind ?1 (false :id ?id :cue (?position :id ?id :name_s ?name_s :from ?id
                                :angle_range (!with-test! ?angle_range (equal? ?angle_r
                                :relative_angle ?relative_angle
                                :distance_range (!with-test! ?dis (equal? ?dis *DISTANC
                                :height (!with-test! ?height (equal? ?height *HEIGHT_EQ

(not (see-in-view :id ?id))
-->
(make (see-in-view :id ?id :name_s (!value-of! '()))
                    :relative_angle (!value-of! ?relative_angle)))

(remove ?1))

(defrule
see-in-view-3-1
:priority 10000

```

```

(bind ?1 (?position :id ?id :name_s ?name_s :from ?id
               :angle_range (!with-test! ?angle_range (equal? ?angle_range *ANGLE_RANGE_V*))
               :relative_angle ?relative_angle
               :distance_range (!with-test! ?dis (equal? ?dis *DISTANCE_RANGE_NEAR*))
               :height (!with-test! ?height (equal? ?height *HEIGHT_EQ*))))
(not (see-in-view :id ?id))
-->
(make (see-in-view :id ?id :name_s ?name_s
                  :relative_angle (!value-of! ?relative_angle)))
(remove ?1))

(defrule
see-in-view-3-2
:priority 10000
(bind ?1 (?position :id ?id :name_s ?name_s :from ?id
               :angle_range (!with-test! ?angle_range (equal? ?angle_range *ANGLE_RANGE_V*))
               :relative_angle ?relative_angle
               :distance_range (!with-test! ?dis (equal? ?dis *DISTANCE_RANGE_NEAR*))
               :height (!with-test! ?height (equal? ?height *HEIGHT_EQ*))))
(bind ?2 (see-in-view :id ?id :name_s ?n))
-->
(modify ?2 (see-in-view :id ?id :name_s ?name_s
                       :relative_angle (!value-of! ?relative_angle)))
(remove ?1))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; 歩行エージェントの認識 (walk と approach を実行中のエージェントを認識)
;;; 空間上すべてを認識してしまうため現実世界のルールとは異なるが
;;; 計算の簡略化、大量の依頼による負荷の軽減のため採用
;;; (現実には近づけるためには上記の視野内のエージェント一体一体に対して観測を行えばよいと考える)
;;

(defrule
know-moving-1-1
:priority 100
(bind ?1 (true :id ?id :cue (?observe :id ?id :name_s ?name_s
                                   :action (!with-test! ?action (member ?action ("walk" "ap

(not (know-moving :id ?id))
-->
(make (know-moving :id ?id :name_s (!value-of! ?name_s) :action (!value-of! (list ?action))))
(remove ?1))

(defrule

```

```

know-moving-1-2
:priority 100
(bind ?1 (true :id ?id :cue (?observe :id ?id :name_s ?name_s
                             :action (!with-test! ?action (member ?action ("walk" "ap
(bind ?2 (know-moving :id ?id :name_s ?old_name_s :action ?old_action))
-->
(modify ?2 (know-moving :id ?id :name_s (!value-of! (append (return-list ?name_s) (return-list
                                                         :action (!value-of! (list-unique (append (return-list ?action) (return-
(remove ?1))

(defrule
know-moving-2-1
:priority 100
(bind ?1 (false :id ?id :cue (?observe :id ?id :name_s ?name_s
                             :action (!with-test! ?action (member ?action ("walk" "a
(not (know-moving :id ?id))
-->
(make (know-moving :id ?id :name_s (!value-of! '()) :action (!value-of! (list ?action))))
(remove ?1))

(defrule
know-moving-2-2
:priority 100
(bind ?1 (false :id ?id :cue (?observe :id ?id :name_s ?name_s
                             :action (!with-test! ?action (member ?action ("walk" "a
(bind ?2 (know-moving :id ?id :name_s ?old_name_s :action ?old_action))
-->
(modify ?2 (know-moving :id ?id :name_s ?old_name_s
                    :action (!value-of! (list-unique (append (return-list ?action) (return-
(remove ?1))

(defrule
know-moving-3-1
:priority 100
(bind ?1 (?observe :id ?id :name_s ?name_s
         :action (!with-test! ?action (member ?action ("walk" "approach")))))
(not (know-moving :id ?id))
-->
(make (know-moving :id ?id :name_s (!value-of! ?name_s) :action ?action))
(remove ?1))

(defrule
know-moving-3-2
:priority 100
(bind ?1 (?observe :id ?id :name_s ?name_s

```

```

                :action (!with-test! ?action (member ?action ('("walk" "approach")))))
(bind ?2 (know-moving :id ?id :name_s ?old_name_s :action ?old_action))
-->
(modify ?2 (know-moving :id ?id :name_s (!value-of! (append (return-list ?name_s) (return-list
                :action (!value-of! (list-unique (append (return-list ?action) (return-
(remove ?1))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; エージェントの絶対座標に対する体の向きを取得
;;FreeWalk には現在の体の向きに対する移動などができないため、空間座標系での体の向き
を知る必要がある
;;

(defrule
is-turning-1-1
:priority 10000
(bind ?1 (true :id ?id :cue (?position :id ?id :name_s ?id
                :from_coordinate (!with-test! ?from (equal? ?from '(0.0
                :relative_angle (!with-test! ?relative_angle (list? ?re
(not (is-turning :id ?id :to (!with-test! ?to (not (number? ?to))))))
-->
(debug '(,?id is-turning-1-1))
(make (is-turning :id ?id
                :relative_angle (!value-of! ?relative_angle)))
(remove ?1))

(defrule
is-turning-1-2
:priority 10000
(bind ?1 (true :id ?id :cue (?position :id ?id :name_s ?id
                :from_coordinate (!with-test! ?from (equal? ?from '(0.0
                :relative_angle (!with-test! ?relative_angle (list? ?re
(bind ?2 (is-turning :id ?id :to (!with-test! ?to (not (number? ?to))) :relative_angle ?old_re
-->
(debug '(,?id is-turning-1-2))
(modify ?2 (is-turning :id ?id
                :relative_angle (!value-of! ?relative_angle)))
(!activate :cue (?position :name_s ?id :from_coordinate '(0.0 0.0 -100000.0 0.0)
                :relative_angle ?old_relative_angle))
(remove ?1))

(defrule
is-turning-2
:priority 10000

```

```

(bind ?1 (false :id ?id :cue (?position :id ?id :name_s ?id
                                :from_coordinate (!with-test! ?from (equal? ?from '(0.0 0.0 -100000.0 0.0)
                                :relative_angle (!with-test! ?relative_angle (list? ?relative_angle)))
(not (is-turning :id ?id :relative_angle (!with-test! ?relative_angle (list? ?relative_angle)))
-->
(debug '(,?id is-turning-2))
(remove ?1))

(defrule
is-turning-3-1
:priority 10000
(bind ?1 (?position :id ?id :name_s ?id
            :from_coordinate (!with-test! ?from (equal? ?from '(0.0 0.0 -100000.0 0.0)
            :relative_angle (!with-test! ?relative_angle (list? ?relative_angle))))
(not (is-turning :id ?id :to (!with-test! ?to (not (number? ?to))))))
-->
(debug '(,?id is-turning-3-1))
(make (is-turning :id ?id
                  :relative_angle (!value-of! ?relative_angle)))
(remove ?1))

(defrule
is-turning-3-2
:priority 10000
(bind ?1 (?position :id ?id :name_s ?id
            :from_coordinate (!with-test! ?from (equal? ?from '(0.0 0.0 -100000.0 0.0)
            :relative_angle (!with-test! ?relative_angle (list? ?relative_angle))))
(bind ?2 (is-turning :id ?id :to (!with-test! ?to (not (number? ?to))) :relative_angle ?old_re
-->
(debug '(,?id is-turning-3-2))
(modify ?2 (is-turning :id ?id
                      :relative_angle (!value-of! ?relative_angle)))
(!activate :cue (?position :name_s ?id :from_coordinate '(0.0 0.0 -100000.0 0.0)
               :relative_angle ?old_relative_angle))
(remove ?1))

```

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; エージェント座標に対する指差しの向きを取得
;;; FreeWalk には現在の体の向きに対する移動などができないため、空間座標系での体の向き
を知る必要がある
;;

```

```

(defrule

```

```

know-pointing-1-1
:priority 10000
(bind ?1 (true :id ?id :cue (?position :id ?id :name_s ?name_s
                             :from_coordinate (!with-test! ?from (equal? ?from '(0.0
                             :point_angle (!with-test! ?point_angle (list? ?point_angle)))
(not (know-pointing :id ?id))
-->
(debug '(,?id know-pointing-1-1))
(make (know-pointing :id ?id :name_s ?name_s
                    :point_angle (!value-of! ?point_angle) :time (!value-of! (current-seconds)))
(remove ?1))

(defrule
know-pointing-1-2
:priority 10000
(bind ?1 (true :id ?id :cue (?position :id ?id :name_s ?name_s
                             :from_coordinate (!with-test! ?from (equal? ?from '(0.0
                             :point_angle (!with-test! ?point_angle (list? ?point_angle)))
(bind ?2 (know-pointing :id ?id :name_s ?old_name_s :point_angle ?old_point_angle))
-->
(debug '(,?id know-pointing-1-2))
(modify ?2 (know-pointing :id ?id :name_s ?name_s
                          :point_angle (!value-of! ?point_angle) :time (!value-of! (current-seconds)))
(remove ?1))

(defrule
know-pointing-2
:priority 10000
(bind ?1 (false :id ?id :cue (?position :id ?id :name_s ?name_s
                              :from_coordinate (!with-test! ?from (equal? ?from '(0.0
                              :point_angle (!with-test! ?point_angle (list? ?point_angle)))
;(not (know-pointing :id ?id))
-->
(debug '(,?id know-pointing-2))
;(make (know-pointing :id ?id
                    :point_angle (!value-of! ?point_angle) :time (!value-of! (current-seconds)))
(remove ?1))

(defrule
know-pointing-3-1
:priority 10000
(bind ?1 (?position :id ?id :name_s ?name_s
          :from_coordinate (!with-test! ?from (equal? ?from '(0.0 0.0 -100000.0 0.0))
          :point_angle (!with-test! ?point_angle (list? ?point_angle))))
(not (know-pointing :id ?id))

```

```

-->
(dbg '(,?id know-pointing-3-1))
(make (know-pointing :id ?id :name_s ?name_s
                    :point_angle (!value-of! ?point_angle) :time (!value-of! (current-seconds))
                    (remove ?1))

(defrule
know-pointing-3-2
:priority 10000
(bind ?1 (?position :id ?id :name_s ?name_s
                 :from_coordinate (!with-test! ?from (equal? ?from '(0.0 0.0 -100000.0 0.0))
                 :point_angle (!with-test! ?point_angle (list? ?point_angle))))
(bind ?2 (know-pointing :id ?id :name_s ?old_name_s :point_angle ?old_point_angle))
-->
(dbg '(,?id know-pointing-3-2))
(modify ?2 (know-pointing :id ?id :name_s ?name_s
                        :point_angle (!value-of! ?point_angle) :time (!value-of! (current-seconds))
                        (remove ?1))

(defrule
face-1
:priority 200
(bind ?0 (request :id ?id :request-name (!with-test! ?r-name (member ?r-name (list "!face" "!!face")))
                 :option-list (!with-test! ?options (list? (get-opt-value ?options "to_coordinate")))
                 (route :route ?route))
-->
(dbg '(,?id face-1))
(set-opt-value! ?options "in" "to_coordinate" (caar ?route))
(bind ?0 (modify ?0 (request :id ?id :request-name ?r-name :option-list ?options)))
(!follow :request ?0)
(remove ?0)
(dbg '(,?id face-1 ok))
(display '(load-rule finished))(newline)

```

### A.3 エージェントモデルを想定していない場合のプロトコル

```

;;
;; -- scheme --
;;
;; 地下鉄京都駅避難訓練シナリオファイル
;; file   : skyoto-scenario-20050201-normal.q
;; creator: Konishi Shinji
;; date   : 2005/02/01

```

```

(load "../Data/Scenario/skyoto_walk/ps-define-1.q")

(define
  (wait-for x)
  (do ((cs (+ (current-seconds) x) cs))(> (current-seconds) cs))
  x)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; エージェントの基本モデル。
;; インタラクションは開始と終了のみでその他のインタラクションはない。
;; 基本の骨組みを理解しておくためだけに設計。
;;
(display "(defscenario scenario-skyoto-base)")(newline)
(defscenario
  scenario-skyoto-base
  ($area $scenario-type $emailaddress)
  (let
    (($route '((0.0 0.0)) , $area (,$area)))
    ($agents '())
    ($loop-end #f)
    ($agent-rp agent-rp)
    ($agent-rr agent-rr)
    ($agent-slr agent-slr)
    ($agent-str agent-str)
    ($walk-w walk-w)
    ($walk-v walk-v)
    ($walk-a walk-a)
    ($turn-v turn-v)
    ($turn-a turn-a)
    ($tmp 0))
  (scene_init
    (#t
      (!!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a
              :angle_velocity $turn-v :angle_acceleration $turn-a)
      (go scene_normal_walk)))
  (scene_normal_walk
    ((?hear :from_s $agents :word *start-keyword*)
     (!finish :action "walk")
     (set! $agents '())
     (set! $walk-w 0.8)
     (set! $walk-v 6.0)
     (set! $turn-v *evac-turn-v*)
     (set! $turn-a *evac-turn-a*)
     (make-evac-route-sp1 self $route 0 #t)
     (!!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a :angle_v

```

```

    (make-evac-route-sp1 self $route 0 #f)
    (go scene_evac_walk))
  ((?finish :action "walk")
   (!!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a :angle_v
    (make-route-one $route)
    (go scene_walk)))
  (scene_evac_walk
   ((?hear :from_s $agents :word *finish-keyword*)
    (set! $agents '())
    (go scene_end))
   ((?finish :action "walk")
    (!!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a :angle_v
     (make-evac-route-sp1 self $route 0 #f)
     (go scene_end))))
  (scene_end
   (#t
    )))

;;
;; 特殊エージェント1
;; シナリオの進行管理を行う
;; また、避難完了エージェントの管理も行っている。
;; 他者とのインタラクションは
;; ・ 避難訓練開始、終了メッセージの送信
;; ・ 避難完了エリアへのエージェントの移動を観測 避難完了確認メッセージ送信
;; である
;;
(display "(defscenario scenario-skyoto-special1)"(newline))
(defscenario
 scenario-skyoto-special1
 ($area $scenario-type $emailaddress)
 (let
  (($route '(((0.0 0.0)) , $area (, $area)))
   ($agents '())
   ($loop-end #f)
   ($known-evacuee '())
   ($tmp 0))
  (scene_init
   (#t
    (go scene_normal)))
  (scene_normal
   ((?input :key "return" :mode "temporarily")
    (if (not (member self *special-agents*))
        (set! *special-agents* (list-unique (append (return-list *special-agents*)
                                                    (list self))))))

```

```

(guard ((?position :name_s $agents :from_coordinate '(0.0 0.0 0.0 0.0) :rectangle '((-1000
      (otherwise (set! $agents *all-agents*)))
(display '(!speak :to_s ,$agents :sentence *evacuation-exam-start-sentence-1* :loudness 10
(!send :to_s $agents :sentence "ただいまから避難訓練を開始します。上の階ま
で避難してください。" )
  (!!send :to_s self :sentence "避難訓練を開始")
(display "evacuee-agents: ")(display *evacuee-agents*)(newline)
(display "avatars      : ")(display *avatars*)(newline)
(display "guide-agents  : ")(display *guide-agents*)(newline)
(display "special-agents : ")(display *special-agents*)(newline)
(set! *all-agents* $agents)
(set! $agents '())
(display '(,self start))(newline)
(go scene_evac)))
(scene_evac
  ((?receive :from_s self :word "訓練を終了")
    (go scene_end))
  ((?position :name_s $agents
    :from_coordinate '(0.0 0.0 0.0 0.0)
    :rectangle '((360.0 -90.0)(385.0 -57.0))
    :height '(-8.0 -7.0))
    (set! $agents (return-list $agents))
    (set! $tmp '())
    (guard ((?position :name_s $tmp
      :from_coordinate '(0.0 0.0 0.0 0.0)
      :rectangle '((365.0 -70.0)(385.0 -60.0))
      :height '(-12.5 -10.5))
      (set! $tmp (return-list $tmp))
      (append! $agents $tmp))
      (otherwise (set! $tmp '()))))
    (set! $tmp '())
    (guard ((?position :name_s $tmp
      :from_coordinate '(0.0 0.0 0.0 0.0)
      :rectangle '((365.0 -20.0)(385.0 10.0)))
      (set! $tmp (return-list $tmp))
      (append! $agents $tmp))
      (otherwise (set! $tmp '()))))
    (set! $tmp '())
    (go scene_send_confirmed_sentence))
  ((?position :name_s $agents
    :from_coordinate '(0.0 0.0 0.0 0.0)
    :rectangle '((365.0 -70.0)(385.0 -57.0))
    :height '(-12.5 -10.5))
    (set! $agents (return-list $agents))
    (set! $tmp '())

```

```

(guard ((?position :name_s $tmp
            :from_coordinate '(0.0 0.0 0.0 0.0)
            :rectangle '((360.0 -90.0)(385.0 -60.0))
            :height '(-8.0 -7.0))
        (set! $tmp (return-list $tmp))
        (append! $agents $tmp))
  (otherwise (set! $tmp '())))
(set! $tmp '())
(guard ((?position :name_s $tmp
            :from_coordinate '(0.0 0.0 0.0 0.0)
            :rectangle '((365.0 -20.0)(385.0 10.0))
            :height '(-8.0 -7.0))
        (set! $tmp (return-list $tmp))
        (append! $agents $tmp))
  (otherwise (set! $tmp '())))
(set! $tmp '())
(go scene_send_confirmed_sentence)
((?position :name_s $agents
            :from_coordinate '(0.0 0.0 0.0 0.0)
            :rectangle '((365.0 -20.0)(385.0 10.0))
            :height '(-8.0 -7.0))
  (set! $agents (return-list $agents))
  (set! $tmp '())
  (guard ((?position :name_s $tmp
              :from_coordinate '(0.0 0.0 0.0 0.0)
              :rectangle '((365.0 -70.0)(385.0 -60.0))
              :height '(-12.5 -10.5))
          (set! $tmp (return-list $tmp))
          (append! $agents $tmp))
    (otherwise (set! $tmp '())))
  (set! $tmp '())
  (guard ((?position :name_s $tmp
              :from_coordinate '(0.0 0.0 0.0 0.0)
              :rectangle '((360.0 -90.0)(385.0 -57.0))
              :height '(-8.0 -7.0))
          (set! $tmp (return-list $tmp))
          (append! $agents $tmp))
    (otherwise (set! $tmp '())))
  (set! $tmp '())
  (go scene_send_confirmed_sentence))
)
(scene_send_confirmed_sentence
 (#t
  (set! $agents (check-finish-agents2 $agents $known-evacuee self))
  (display '(,self (check-finish-agents , $agents)))(newline)
  (if (not (equal? (list self) $agents))
      (begin

```

```

        (set! $known- evacuee
          (if (null? $known- evacuee)
              $agents
              (append $known- evacuee $agents)))
        (display '(,self -> , $agents))(newline)
        (!!send :to_s $agents :sentence "こちらであなたの避難完了を確認しまし
た。")
        (display '(evacuee finished : , $known- evacuee))(newline)
        (set! *finish- agents* $known- evacuee))
    (set! $agents '())
    (if (>= (length $known- evacuee)(- (length *agent- list*) (length *special- agents*)))
        (begin
            (set! $loop- end #t)
            (!!send :to_s self :sentence "訓練を終了"))
        (wait- for 1)
        (go scene_ evac)
        ))
    (scene_ end
     (#t
      (display '(,self all- evacuee- finished , *all- agents*))(newline)
      (!speak :to_s *all- agents* :sentence "全員の避難が完了しました。これで避難
訓練を終了いたします。" :loudness 10.0))))
;;
;; アバターシナリオ
;; シナリオの進行に応じて、状態を遷移させるが、その行動は操作者に従う。
;; 他者とのインタラクションは
;; ・避難訓練開始、終了メッセージの受信
;; である
;;
(display "(defscenario scenario- skyoto- avatar1)")(newline)
(defscenario
 scenario- skyoto- avatar1
 ($area $scenario- type $emailaddress)
 (let
  (($route '(((0.0 0.0)) , $area (, $area)))
   ($agents '())
   ($loop- end #f)
   ($tmp 0))
  (scene_ init
   (#t
    (go scene_ walk)))
  (scene_ walk
   ((?receive :from_s $agents :word "避難訓練を開始")
    (set! $agents '())

```

```

    (go scene_evac)))
(scene_evac
  ((?receive :from_s $agents :word "避難完了を確認")
    (set! $agents '())
    (go scene_end)))
(scene_end
  (#t
    (display '(,self end))(newline)
  ))
))

```

;; 誘導者エージェント1

;; ・基本的に現在地から出口まで避難。

;; ・開始と同時に近くにいる避難者エージェントについてくるように指示をし、避難を開始する。

;; ・誘導対象者が自分に近づく様子がなく、立ち止まっていれば、再度指示する。

;; ・誘導対象者が自分以外のエージェントについているようならば、

;; そのエージェントの誘導をやめる。

;; ・避難経路誘導中、誘導対象者との距離により、適切な動作（減速、停止+振り向き+指示、停止+接近+指示）をし、避難する。

;;

```
(display "(defscenario scenario-skyoto-guide1)")(newline)
```

```
(defscenario
```

```
  scenario-skyoto-guide1
```

```
  ($area $scenario-type $emailaddress)
```

```
  (let
```

```
    (($route '(((0.0 0.0)) , $area (, $area)))
```

```
    ($prev-route '(((0.0 0.0)) , $area (, $area)))
```

```
    ($agents '())
```

```
    ($loop-end #f)
```

```
    ($agent-rp agent-rp)
```

```
    ($agent-rr agent-rr)
```

```
    ($agent-slr agent-slr)
```

```
    ($agent-str agent-str)
```

```
    ($walk-w *evac-walk-w*)
```

```
    ($walk-v *evac-walk-v*)
```

```
    ($walk-a *evac-walk-a*)
```

```
    ($turn-v *evac-turn-v*)
```

```
    ($turn-a *evac-turn-a*)
```

```
    ($from-co '(0.0 0.0 0.0 0.0))
```

```
    ($to-guide '())
```

```
    ($tmp 0))
```

```
(scene_init
```

```

(#t
  (display '(,self ready))(newline)
  (go scene_normal)))
(scene_normal
  ((?receive :from_s $agents :word "避難訓練を開始")
    (display '(,self start))(newline)
    ;(!send :to_s self :sentence "避難訓練を開始")
    (!finish :action "walk")
    (set! $agents '())
    (set! $walk-w *evac-walk-w*)
    (set! $walk-v *evac-walk-v*)
    (set! $turn-v *evac-turn-v*)
    (set! $turn-a *evac-turn-a*)
    (set! $route (make-evac-route-one self $route 1 #f))
    ;(!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a :angle_
    ;(make-evac-route-one self $route 1 #t)
    (set! $prev-route (return-list $route))
    (set! $from_co (list (car (list-ref (car $route) (- (length (car $route)) 1)))
                        (get-height-with-areaid (cadr $route))
                        (cadr (list-ref (car $route) (- (length (car $route)) 1)))
                        0.0))

    (go scene_evac_not_walk_without_guide)))
(scene_evac_not_walk_without_guide
  ((?receive :from_s $agents :word "避難完了を確認")
    (!send :to_s self :sentence "避難完了を確認")
    (go scene_end))
  ((?position :name_s self :from_coordinate '(0.0 -7.4 0.0 0.0)
             :rectangle '((367.5 -75.0)(380.5 -57.0)) :height '(-0.3 0.3))
    (set! $route '((372.5 -63.0)) d-2-w (d-1 d-2-w))
    (!walk :route (car $route)
           :width $walk-w :velocity $walk-v :acceleration $walk-a
           :angle_velocity $turn-v :angle_acceleration $turn-a)
    (go scene_evac_walk_without_guide))
  ((?position :from self :name_s $agents :distance_range '(0.0 3.5)
             :angle_range '(40.0 32.0) :height '(-1.0 1.0))
    (set! $agents (list-delete $agents *guide-agents*))
    (if (null? $agents)
      (begin
        (guard ((?position :name_s self :from_coordinate $from_co
                          :distance_range '(0.0 1.5))
              (set! $prev-route (return-list $route))
              (set! $from_co (list (car (list-ref (car $route) (- (length (car $route)) 1))
                                      (get-height-with-areaid (cadr $route))
                                      (cadr (list-ref (car $route) (- (length (car $route)) 1))
                                      0.0))

```

```

        (set! $route (make-evac-route-one self $route 1 #f))
      (otherwise
        (set! $agents '()))
      (guard ((?position :name_s $agents :distance_range '(0.0 4.0) :angle_range '(-60.0 60.0))
        (set! $agents (return-list $agents))
        (cond ((> (length $agents) 10)
          (set! $walk-v (* 0.5 *evac-walk-v*)))
          ((> (length $agents) 7)
          (set! $walk-v (* 0.7 *evac-walk-v*)))
          ((> (length $agents) 4)
          (set! $walk-v (* 0.9 *evac-walk-v*)))
          (else
          (set! $walk-v (* 1.0 *evac-walk-v*))))
        (set! $agents '()))
      (otherwise
        (set! $walk-v *evac-walk-v*)))
      (!!walk :route (car $prev-route) :width $walk-w :velocity $walk-v :acceleration walk-a
        :angle_velocity $turn-v :angle_acceleration $turn-a)
      (set! $agents '())
      (go scene_evac_walk_without_guide))
(begin
  (set! $to-guide (car (return-list $agents)))
  (!turn :to $to-guide :angle_velocity 180.0 :angle_acceleration 360.0)
  (!!speak :to_s $agents :sentence "避難します。私について来て下さい。")
  (!behave :gesture "come")
  (!send :to_s $agents :sentence "避難します。私について来て下さい。")
  (!!face :angle_to_body 0.0)
  (!behave :gesture "standing")
  (!!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a
    :angle_velocity $turn-v :angle_acceleration $turn-a)
  (set! $agents '())
  (wait-for 3)
  (go scene_evac_with_guide))))
((?position :name_s self :from_coordinate $from_co
  :distance_range '(0.0 1.5))
  (guard ((?position :name_s $agents :distance_range '(0.0 4.0) :angle_range '(-60.0 60.0))
    (set! $agents (return-list $agents))
    (cond ((> (length $agents) 10)
      (set! $walk-v (* 0.5 *evac-walk-v*)))
      ((> (length $agents) 7)
      (set! $walk-v (* 0.7 *evac-walk-v*)))
      ((> (length $agents) 4)
      (set! $walk-v (* 0.9 *evac-walk-v*)))
      (else
      (set! $walk-v (* 1.0 *evac-walk-v*))))

```

```

        (set! $agents '())
        (otherwise
          (set! $walk-v *evac-walk-v*))
        (!!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a
          :angle_velocity $turn-v :angle_acceleration $turn-a)
        (set! $prev-route (return-list $route))
        (set! $from_co (list (car (list-ref (car $route) (- (length (car $route)) 1)))
          (get-height-with-areaid (cadr $route))
          (cadr (list-ref (car $route) (- (length (car $route)) 1)))
          0.0))
        (set! $route (make-evac-route-one self $route 1 #f))
        (set! $agents '())
        (go scene_evac_walk_without_guide))
      (otherwise
        (guard ((?position :name_s self :from_coordinate $from_co
          :distance_range '(0.0 1.5))
          (set! $prev-route (return-list $route))
          (set! $from_co (list (car (list-ref (car $route) (- (length (car $route)) 1)))
            (get-height-with-areaid (cadr $route))
            (cadr (list-ref (car $route) (- (length (car $route)) 1)))
            0.0))
          (set! $route (make-evac-route-one self $route 1 #f)))
          (otherwise
            (set! $agents '()))
          (guard ((?position :name_s $agents :distance_range '(0.0 4.0) :angle_range '(-60.0 60.0))
            (set! $agents (return-list $agents))
            (cond ((> (length $agents) 10)
              (set! $walk-v (* 0.5 *evac-walk-v*)))
              ((> (length $agents) 7)
              (set! $walk-v (* 0.7 *evac-walk-v*)))
              ((> (length $agents) 4)
              (set! $walk-v (* 0.9 *evac-walk-v*)))
              (else
              (set! $walk-v (* 1.0 *evac-walk-v*))))
            (set! $agents '()))
            (otherwise
              (set! $walk-v *evac-walk-v*)))
            (!!walk :route (car $prev-route) :width $walk-w :velocity $walk-v :acceleration walk-a
              :angle_velocity $turn-v :angle_acceleration $turn-a)
            (set! $agents '())
            (go scene_evac_walk_without_guide)))
        (scene_evac_walk_without_guide
          (!!receive :from_s $agents :word "避難完了を確認")
          (!!send :to_s self :sentence "避難完了を確認")
          (go scene_end))

```

```

((?position :name_s self :from_coordinate '(0.0 -7.4 0.0 0.0)
           :rectangle '((367.5 -75.0)(380.5 -57.0)) :height '(-0.3 0.3))
(set! $route '((372.5 -63.0) d-2-w (d-1 d-2-w)))
(!!walk :route (car $route)
       :width $walk-w :velocity $walk-v :acceleration $walk-a
       :angle_velocity $turn-v :angle_acceleration $turn-a)
(go scene_evac_walk_without_guide))
((?position :from self :name_s $agents :distance_range '(0.0 3.5)
           :angle_range '(40.0 32.0) :height '(-1.0 1.0))
(set! $agents (list-delete $agents *guide-agents*))
;(display '(,self guide , $agents ,*guide-agents*))
(if (null? $agents)
    (begin
      (guard ((?position :name_s self :from_coordinate $from_co
                      :distance_range '(0.0 1.5))
              (set! $prev-route (return-list $route))
              (set! $from_co (list (car (list-ref (car $route) (- (length (car $route)) 1))
                                   (get-height-with-areaaid (cadr $route))
                                   (cadr (list-ref (car $route) (- (length (car $route)) 1))
                                   0.0))
              (set! $route (make-evac-route-one self $route 1 #f)))
      (otherwise
       (set! $agents '()))
      (guard ((?position :name_s $agents :distance_range '(0.0 4.0) :angle_range '(-60.0 60.0))
              (set! $agents (return-list $agents))
              (cond ((> (length $agents) 10)
                     (set! $walk-v (* 0.5 *evac-walk-v*)))
                    ((> (length $agents) 7)
                     (set! $walk-v (* 0.7 *evac-walk-v*)))
                    ((> (length $agents) 4)
                     (set! $walk-v (* 0.9 *evac-walk-v*)))
                    (else
                     (set! $walk-v (* 1.0 *evac-walk-v*))))
              (set! $agents '()))
      (otherwise
       (set! $walk-v *evac-walk-v*)))
      (!!walk :route (car $prev-route) :width $walk-w :velocity $walk-v :acceleration walk-a
              :angle_velocity $turn-v :angle_acceleration $turn-a)
      (set! $agents '())
      (go scene_evac_walk_without_guide))
    (begin
      (set! $to-guide (car (return-list $agents)))
      (!turn :to $to-guide)
      (!!speak :to_s $agents :sentence "避難します。私について来て下さい。")
      (!behave :gesture "come"))

```

```

(!send :to_s $agents :sentence "避難します。私について来て下さい。")
 (!!face :angle_to_body 0.0)
(!behave :gesture "standing")
 (!!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a
         :angle_velocity $turn-v :angle_acceleration $turn-a)
(set! $agents '())
(wait-for 3)
(go scene_evac_with_guide))))
((?position :name_s self :from_coordinate $from_co
            :distance_range '(0.0 1.5))
 (guard ((?position :name_s $agents :distance_range '(0.0 4.0) :angle_range '(-60.0 60.0))
        (set! $agents (return-list $agents))
        (cond ((> (length $agents) 10)
              (set! $walk-v (* 0.5 *evac-walk-v*)))
              ((> (length $agents) 7)
              (set! $walk-v (* 0.7 *evac-walk-v*)))
              ((> (length $agents) 4)
              (set! $walk-v (* 0.9 *evac-walk-v*)))
              (else
              (set! $walk-v (* 1.0 *evac-walk-v*))))
        (set! $agents '())))
 (otherwise
  (set! $walk-v *evac-walk-v*)))
 (!!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a
         :angle_velocity $turn-v :angle_acceleration $turn-a)
(set! $prev-route (return-list $route))
(set! $from_co (list (car (list-ref (car $route) (- (length (car $route)) 1)))
                    (get-height-with-areaid (cadr $route))
                    (cadr (list-ref (car $route) (- (length (car $route)) 1)))
                    0.0))
(set! $route (make-evac-route-one self $route 1 #f))
(set! $agents '())
(go scene_evac_walk_without_guide))
(otherwise
 (guard ((?position :name_s self :from_coordinate $from_co
                 :distance_range '(0.0 1.5))
        (set! $prev-route (return-list $route))
        (set! $from_co (list (car (list-ref (car $route) (- (length (car $route)) 1)))
                            (get-height-with-areaid (cadr $route))
                            (cadr (list-ref (car $route) (- (length (car $route)) 1)))
                            0.0))
        (set! $route (make-evac-route-one self $route 1 #f)))
 (otherwise
  (set! $agents '()))))
(guard ((?position :name_s $agents :distance_range '(0.0 4.0) :angle_range '(-60.0 60.0))

```

```

(set! $agents (return-list $agents))
(cond (> (length $agents) 10)
      (set! $walk-v (* 0.5 *evac-walk-v*))
      (> (length $agents) 7)
      (set! $walk-v (* 0.7 *evac-walk-v*))
      (> (length $agents) 4)
      (set! $walk-v (* 0.9 *evac-walk-v*))
      (else
       (set! $walk-v (* 1.0 *evac-walk-v*))))
(set! $agents '())
(otherwise
 (set! $walk-v *evac-walk-v*))
(!!walk :route (car $prev-route) :width $walk-w :velocity $walk-v :acceleration walk-a
 :angle_velocity $turn-v :angle_acceleration $turn-a)
(set! $agents '())
(go scene_evac_walk_without_guide))
(scene_evac_with_guide
 ((?receive :from_s $agents :word "避難完了を確認")
  (!!send :to_s self :sentence "避難完了を確認")
  (go scene_end))
 ((?position :name_s $agents :distance_range '(0.0 0.6) :angle_range '(-90.0 0.0))
  (!turn :left 45.0 :angle_velocity 360.0 :angle_acceleration 360.0)
  (!!walk :route (car $prev-route) :width $walk-w :velocity $walk-v :acceleration walk-a
   :angle_velocity $turn-v :angle_acceleration $turn-a)
  (go scene_evac_with_guide))
 ((?position :name_s $agents :distance_range '(0.0 0.6) :angle_range '(0.0 90.0))
  (!turn :right 45.0 :angle_velocity 360.0 :angle_acceleration 360.0)
  (!!walk :route (car $prev-route) :width $walk-w :velocity $walk-v :acceleration walk-a
   :angle_velocity $turn-v :angle_acceleration $turn-a)
  (go scene_evac_with_guide))
 ((?position :name_s self :from_coordinate '(0.0 -7.4 0.0 0.0)
   :rectangle '((367.5 -75.0)(380.5 -57.0)) :height '(-0.3 0.3))
  (set! $route '(((372.5 -63.0)) d-2-w (d-1 d-2-w)))
  (!!walk :route (car $route)
   :width $walk-w :velocity $walk-v :acceleration $walk-a
   :angle_velocity $turn-v :angle_acceleration $turn-a)
  (go scene_evac_walk_without_guide))
 ((?position :name_s $to-guide :distance_range '(5.0 100.0))
  (!turn :to $to-guide :angle_velocity 360.0 :angle_acceleration 360.0)
  (!!speak :to_s $to-guide :sentence "しっかりと私について来て下さい。")
  (!behave :gesture "come")
  (!send :to_s $to-guide :sentence "しっかりと私について来て下さい。")
  (!behave :gesture "standing")
  (guard ((?position :name_s self :from_coordinate $from_co
   :distance_range '(0.0 1.5))

```

```

(guard ((?position :name_s $agents :distance_range '(0.0 4.0) :angle_range '(-60.0
  (set! $agents (return-list $agents))
  (cond ((> (length $agents) 10)
    (set! $walk-v (* 0.5 *evac-walk-v*))
    ((> (length $agents) 7)
    (set! $walk-v (* 0.7 *evac-walk-v*))
    ((> (length $agents) 4)
    (set! $walk-v (* 0.9 *evac-walk-v*))
    (else
    (set! $walk-v (* 1.0 *evac-walk-v*))))
  (set! $agents '()))
  (otherwise
  (set! $walk-v *evac-walk-v*)))
(set! $prev-route (return-list $route))
(set! $from_co (list (car (list-ref (car $route) (- (length (car $route)) 1)))
  (get-height-with-areaid (cadr $route))
  (cadr (list-ref (car $route) (- (length (car $route)) 1)))
  0.0))
(set! $route (make-evac-route-one self $route 1 #f))
(otherwise
  (set! $agents '()))
(guard ((?position :name_s $to-guide :relative_angle '(160.0 200.0))
  (guard ((?observe :name_s $to-guide :action "approach")
    (!!walk :route (car $prev-route)
      :width $walk-w :velocity $walk-v :acceleration walk-a
      :angle_velocity $turn-v :angle_acceleration $turn-a)
    (go scene_evac_with_guide))
    ((?observe :name_s $to-guide :action "walk")
    (!!walk :route (car $prev-route)
      :width $walk-w :velocity $walk-v :acceleration walk-a
      :angle_velocity $turn-v :angle_acceleration $turn-a)
    (go scene_evac_with_guide))
    (otherwise
    (if (member $to-guide *avatars*)
      (go scene_evac_with_guide)
      (begin
        (set! $to-guide '())
        (go scene_evac_not_walk_without_guide))))))
  (otherwise
  (set! $to-guide '())
  (go scene_evac_not_walk_without_guide))))
((?position :name_s $to-guide :distance_range '(0.0 5.0))
  (guard ((?position :name_s self :from_coordinate $from_co
    :distance_range '(0.0 1.5))
    (guard ((?position :name_s $agents :distance_range '(0.0 4.0) :angle_range '(-60.0

```

```

(set! $agents (return-list $agents))
(cond ((> (length $agents) 10)
      (set! $walk-v (* 0.5 *evac-walk-v*))
      ((> (length $agents) 7)
       (set! $walk-v (* 0.7 *evac-walk-v*)))
      ((> (length $agents) 4)
       (set! $walk-v (* 0.9 *evac-walk-v*)))
      (else
       (set! $walk-v (* 1.0 *evac-walk-v*))))
(set! $agents '())
(otherwise
 (set! $walk-v *evac-walk-v*))
(set! $prev-route (return-list $route))
(set! $from_co (list (car (list-ref (car $route) (- (length (car $route)) 1)))
                    (get-height-with-areaid (cadr $route))
                    (cadr (list-ref (car $route) (- (length (car $route)) 1)))
                    0.0))
(set! $route (make-evac-route-one self $route 1 #f))
(otherwise
 (set! $agents '()))
(!!walk :route (car $prev-route) :width $walk-w :velocity $walk-v :acceleration walk-a
        :angle_velocity $turn-v :angle_acceleration $turn-a)
(set! $agents '())
(go scene_evac_with_guide))
)
(scene_end
 (#t
  (display '(,self end))(newline)
  (!appear :position '(,+ 360.0 (random 20)) -7.0 ,(- -75.0 (random 15))))
  (!finish :action "walk")
  (!finish :action "approach")
  )))

```

;; 誘導者エージェント2

;; ・基本的に現在地から出口まで避難。

;; ・避難中に誘導対象者がいれば、出口の方向を指示する。

;; ・誘導対象者が誘導方向に移動し、付近に誰もいなければ自らも避難を開始する。

;;

```
(display "(defscenario scenario-skyoto-guide2)")(newline)
```

```
(defscenario
```

```
  scenario-skyoto-guide2
```

```
  ($area $scenario-type $emailaddress)
```

```
  (let
```

```
    (($route '(((0.0 0.0)) , $area (, $area)))
```

```

($prev-route '(((0.0 0.0)) , $area (, $area)))
($agents '())
($loop-end #f)
($agent-rp agent-rp)
($agent-rr agent-rr)
($agent-slr agent-slr)
($agent-str agent-str)
($walk-w walk-w)
($walk-v walk-v)
($walk-a walk-a)
($turn-v turn-v)
($turn-a turn-a)
($from_co '(0.0 0.0 0.0 0.0))
($tmp 0))
(scene_init
  (#t
    (display '(,self ready))(newline)
    (go scene_normal)))
(scene_normal
  ((?receive :from_s $agents :word "避難訓練を開始")
    (display '(,self start))(newline)
    (!!send :to_s self :sentence "避難訓練を開始")
    (set! $agents '())
    (set! $walk-w *evac-walk-w*)
    (set! $walk-v *evac-walk-v*)
    (set! $turn-v *evac-turn-v*)
    (set! $turn-a *evac-turn-a*)
    (set! $route (make-evac-route-one self $route 1 #f))
    (set! $from_co (list (car (list-ref (car $route) (- (length (car $route)) 1)))
      (get-height-with-areaid (cadr $route))
      (cadr (list-ref (car $route) (- (length (car $route)) 1)))
      0.0))
    (set! $prev-route (return-list $route))
    (go scene_evac_walk_without_guide)))
(scene_evac_walk_without_guide
  ((?receive :from_s $agents :word "避難完了を確認")
    (!!send :to_s self :sentence "避難完了を確認")
    (go scene_end))
  ((?position :name_s self :from_coordinate '(0.0 -7.4 0.0 0.0)
    :rectangle '((367.5 -75.0)(380.5 -57.0)) :height '(-0.3 0.3))
    (set! $route '(((372.5 -63.0)) d-2-w (d-1 d-2-w)))
    (!!walk :route (car $route)
      :width $walk-w :velocity $walk-v :acceleration $walk-a
      :angle_velocity $turn-v :angle_acceleration $turn-a)
    (go scene_evac_walk_without_guide)))

```

```

((?position :name_s $agents :distance_range '(0.0 3.5)
           :angle_range '(30.0 330.0))
 (set! $agents (list-delete $agents *guide-agents*))
 (if (null? $agents)
     (begin
       (guard ((?position :name_s $agents :distance_range '(0.0 4.0) :angle_range '(-60.0 60.0)
                        (set! $agents (return-list $agents))
                        (cond ((> (length $agents) 10)
                              (set! $walk-v (* 0.5 *evac-walk-v*))
                              ((> (length $agents) 7)
                               (set! $walk-v (* 0.7 *evac-walk-v*))
                              ((> (length $agents) 4)
                               (set! $walk-v (* 0.9 *evac-walk-v*))
                              (else
                               (set! $walk-v (* 1.0 *evac-walk-v*))))
                        (set! $agents '()))
         (otherwise
          (set! $walk-v *evac-walk-v*)))
       (guard
        ((?position :name_s self :from_coordinate $from_co :distance_range '(0.0 2.5))
         (set! $prev-route (return-list $route))
         (set! $from_co (list (car (list-ref (car $route) (- (length (car $route)) 1)))
                              (get-height-with-areaid (cadr $route))
                              (cadr (list-ref (car $route) (- (length (car $route)) 1)))
                              0.0))
         (set! $route (make-evac-route-one self $route 1 #f)))
         (otherwise
          (set! $agents '()))
         (!!walk :route (car $prev-route) :width $walk-w :velocity $walk-v :acceleration walk-a
                :angle_velocity $turn-v :angle_acceleration $turn-a)
         (go scene_evac_walk_without_guide))
     (begin
       (!finish :action "walk")
       (guard
        ((?position :name_s self :from_coordinate $from_co :distance_range '(0.0 2.5))
         (set! $prev-route (return-list $route))
         (set! $from_co (list (car (list-ref (car $route) (- (length (car $route)) 1)))
                              (get-height-with-areaid (cadr $route))
                              (cadr (list-ref (car $route) (- (length (car $route)) 1)))
                              0.0))
         (set! $route (make-evac-route-one self $route 1 #f)))
         (otherwise)
         (!!turn :to_coordinate '(,(car $from_co) ,(caddr $from_co))
                :angle_velocity 180.0 :angle_acceleration 180.0)
         (!!point :to_coordinate '(,(car $from_co) ,(caddr $from_co)) :limit_angle_to_body 90.0)

```

```

      (!!face :to (car (return-list $agents)) :limit_angle_to_body 120.0
        :angle_velocity 180.0 :angle_acceleration 180.0)
      (!!speak :to_s $agents :sentence "あちらへ避難してください")
      (!send :to_s $agents :sentence "あちらへ避難してください")
      (wait-for 3)
      (!!face :angle_to_body 0.0 :angle_velocity 180.0 :angle_acceleration 180.0)
      (set! $agents '())
      (go scene_evac_guide))))
((?position :name_s self :from_coordinate $from_co :distance_range '(0.0 2.5))
 (guard ((?position :name_s $agents :distance_range '(0.0 4.0) :angle_range '(-60.0 60.0))
  (set! $agents (return-list $agents))
  (cond ((> (length $agents) 10)
    (set! $walk-v (* 0.5 *evac-walk-v*)))
    ((> (length $agents) 7)
    (set! $walk-v (* 0.7 *evac-walk-v*)))
    ((> (length $agents) 4)
    (set! $walk-v (* 0.9 *evac-walk-v*)))
    (else
    (set! $walk-v (* 1.0 *evac-walk-v*))))
  (set! $agents '()))
 (otherwise
  (set! $walk-v *evac-walk-v*)))
 (!!walk :route (car $route) :width $walk-w :velocity $walk-v :acceleration walk-a
  :angle_velocity $turn-v :angle_acceleration $turn-a)
(set! $prev-route (return-list $route))
(set! $from_co (list (car (list-ref (car $route) (- (length (car $route)) 1)))
  (get-height-with-areaid (cadr $route))
  (cadr (list-ref (car $route) (- (length (car $route)) 1)))
  0.0))
(set! $route (make-evac-route-one self $route 1 #f))
(go scene_evac_walk_without_guide))
(otherwise
 (!!walk :route (car $prev-route) :width $walk-w :velocity $walk-v :acceleration walk-a
  :angle_velocity $turn-v :angle_acceleration $turn-a)
 (set! $agents '())
 (go scene_evac_walk_without_guide)))
(scene_evac_guide
 (!!receive :from_s $agents :word "避難完了を確認")
 (!!send :to_s self :sentence "避難完了を確認")
 (go scene_end))
 (!!position :name_s self :from_coordinate '(0.0 -7.4 0.0 0.0)
  :rectangle '((367.5 -75.0)(380.5 -57.0)) :height '(-0.3 0.3))
 (set! $route '((372.5 -63.0) d-2-w (d-1 d-2-w)))
 (!!walk :route (car $route)
  :width $walk-w :velocity $walk-v :acceleration $walk-a

```

```

        :angle_velocity $turn-v :angle_acceleration $turn-a)
    (go scene_evac_guide))
((?position :name_s $agents :distance_range '(0.0 0.6) :angle_range '(-90.0 0.0))
 (!turn :left 45.0 :angle_velocity 360.0 :angle_acceleration 360.0)
 (!!walk :route (car $prev-route) :width $walk-w :velocity $walk-v :acceleration walk-a
        :angle_velocity $turn-v :angle_acceleration $turn-a)
 (wait-for 1)
 (!finish :action "walk")
 (go scene_evac_guide))
((?position :name_s $agents :distance_range '(0.0 0.6) :angle_range '(0.0 90.0))
 (!turn :right 45.0 :angle_velocity 360.0 :angle_acceleration 360.0)
 (!!walk :route (car $prev-route) :width $walk-w :velocity $walk-v :acceleration walk-a
        :angle_velocity $turn-v :angle_acceleration $turn-a)
 (wait-for 1)
 (!finish :action "walk")
 (go scene_evac_guide))
((?position :from self :name_s $agents :distance_range '(0.0 2.0)
        :angle_range '(30.0 330.0))
 (set! $agents (list-delete $agents *guide-agents*))
 (if (null? $agents)
     (begin
      (guard ((?position :name_s $agents :distance_range '(0.0 4.0) :angle_range '(-60.0 60.0)
              (set! $agents (return-list $agents))
              (cond ((> (length $agents) 10)
                     (set! $walk-v (* 0.5 *evac-walk-v*)))
                    ((> (length $agents) 7)
                     (set! $walk-v (* 0.7 *evac-walk-v*)))
                    ((> (length $agents) 4)
                     (set! $walk-v (* 0.9 *evac-walk-v*)))
                    (else
                     (set! $walk-v (* 1.0 *evac-walk-v*))))
              (set! $agents '()))
            (otherwise
             (set! $walk-v *evac-walk-v*)))
      (!!walk :route (car $prev-route) :width $walk-w :velocity $walk-v :acceleration walk-a
              :angle_velocity $turn-v :angle_acceleration $turn-a)
      (set! $agents '())
      (!finish :action "point")
      (!behave :gesture "standing")
      (go scene_evac_walk_without_guide))
     (begin
      (!!face :to (car (return-list $agents)) :limit_angle_to_body 120.0
              :angle_velocity 180.0 :angle_acceleration 180.0)
      (!!speak :to_s $agents :sentence "あちらへ避難してください")
      (!send :to_s $agents :sentence "あちらへ避難してください")

```

```

        (wait-for 3)
        (!face :angle_to_body 0.0 :angle_velocity 180.0 :angle_acceleration 180.0)
        (set! $agents '())
        (go scene_evac_guide))))
(otherwise
  (!finish :action "point")
  (!behave :gesture "standing")
  (!!walk :route (car $prev-route) :width $walk-w :velocity $walk-v :acceleration walk-a
    :angle_velocity $turn-v :angle_acceleration $turn-a)
  (set! $agents '())
  (go scene_evac_walk_without_guide)))
(scene_end
  (#t
    (display '(,self end))(newline)
    (!appear :position '(,+ 360.0 (random 20)) -7.0 ,(- -75.0 (random 15))))
    (!finish :action "walk")
    (!finish :action "approach")
  ))))

;;
;; 避難者シナリオ 4
;; 避難訓練開始とともに、避難を開始する。
;; 他者とのインタラクションは
;; ・ 避難訓練開始のメッセージを受信する 避難行動開始
;; ・ 誘導者の誘導に従う
;; ・ 避難完了確認メッセージを受信する 姿を消す
;; である。
;; 単純
(display "(defscenario scenario-skyoto-evacuee4)")(newline)
(defscenario
  scenario-skyoto-evacuee4
  ($area $scenario-type $emailaddress)
  (let
    (($route '(((0.0 0.0)),$area (,$area)))
     ($agents '())
     ($agent-rp agent-rp)
     ($agent-rr agent-rr)
     ($agent-slr agent-slr)
     ($agent-str agent-str)
     ($walk-w walk-w)
     ($walk-v walk-v)
     ($walk-a walk-a)
     ($turn-v turn-v)
     ($turn-a turn-a)
     ($follow-to '()))

```

```

($tmp 0)
($tmp2 0)
($tmp3 0))
(scene_init
  (#t
    (display '(,self ready))(newline)
    (go scene_normal)))
(scene_normal
  ((?receive :from_s $agents :word "避難訓練を開始")
    (!!send :to_s self :sentence "避難訓練を開始")
    (display '(,self start))(newline)
    (set! $agents '())
    (set! $walk-w *evac-walk-w*)
    (set! $walk-v *evac-walk-v*)
    (set! $turn-v *evac-turn-v*)
    (set! $turn-a *evac-turn-a*)
    (wait-for 2)
    (go scene_evac)))
(scene_evac
  ((?receive :from_s $agents :word "避難完了を確認")
    (!!send :to_s self :sentence "避難完了を確認")
    (go scene_end))
  ((?position :name_s self :from_coordinate '(0.0 -7.4 0.0 0.0)
    :rectangle '((367.5 -75.0)(380.5 -57.0)) :height '(-0.3 0.3))
    (set! $route '(((372.5 -63.0)) d-2-w (d-1 d-2-w)))
    (!!walk :route (car $route)
      :width $walk-w :velocity $walk-v :acceleration $walk-a
      :angle_velocity $turn-v :angle_acceleration $turn-a)
    (go scene_evac))
  ((?position :name_s self :from_coordinate '(0.0 -12.0 0.0 0.0)
    :rectangle '((377.5 -55.0)(380.5 -41.9)) :height '(-0.3 0.3))
    (set! $route '(((377.5 -70.0)) s-7-e (s-1-e s-7-e)))
    (!!walk :route (car $route)
      :width $walk-w :velocity $walk-v :acceleration $walk-a
      :angle_velocity $turn-v :angle_acceleration $turn-a)
    (go scene_evac))
  ((?position :name_s self :from_coordinate '(0.0 -12.0 0.0 0.0)
    :rectangle '((368.0 -55.0)(372.1 -41.9)) :height '(-0.3 0.3))
    (set! $route '(((370.5 -70.0)) s-7-w (s-1-w s-7-w)))
    (!!walk :route (car $route)
      :width $walk-w :velocity $walk-v :acceleration $walk-a
      :angle_velocity $turn-v :angle_acceleration $turn-a)
    (go scene_evac))
  ((?receive :from_s $agents :word "ついて来て下さい")
    (set! $agents (car (if (list? $agents) $agents (list $agents)))))

```

```

(!turn :to $agents :angle_velocity $turn-v :angle_acceleration $turn-a)
 (!!approach :to_s $agents
            :distance 1.0 :velocity $walk-v :acceleration $walk-a
            :angle_velocity $turn-v :angle_acceleration $turn-a)
(set! $follow-to $agents)
(set! $agents '())
(go scene_evac)
((?receive :from_s $agents :word "あちらへ避難してください")
 (set! $agents (car (if (list? $agents) $agents (list $agents))))
 (!turn :to $agents :angle_velocity $turn-v :angle_acceleration $turn-a)
 (guard ((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
                  :point_angle '(0.0 20.0))
        (set! $tmp '(0.0 25.0)))
        ((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
                  :point_angle '(20.0 40.0))
        (set! $tmp '(20.0 45.0)))
        ((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
                  :point_angle '(40.0 60.0))
        (set! $tmp '(40.0 65.0)))
        ((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
                  :point_angle '(60.0 80.0))
        (set! $tmp '(60.0 85.0)))
        ((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
                  :point_angle '(80.0 100.0))
        (set! $tmp '(80.0 105.0)))
        ((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
                  :point_angle '(100.0 120.0))
        (set! $tmp '(100.0 125.0)))
        ((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
                  :point_angle '(120.0 140.0))
        (set! $tmp '(120.0 145.0)))
        ((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
                  :point_angle '(140.0 160.0))
        (set! $tmp '(140.0 165.0)))
        ((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
                  :point_angle '(160.0 180.0))
        (set! $tmp '(160.0 185.0)))
        ((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
                  :point_angle '(180.0 200.0))
        (set! $tmp '(180.0 205.0)))
        ((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
                  :point_angle '(200.0 220.0))
        (set! $tmp '(200.0 225.0)))
        ((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
                  :point_angle '(220.0 240.0))

```

```

(set! $tmp '(220.0 245.0))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
:point_angle '(240.0 260.0))
(set! $tmp '(240.0 265.0))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
:point_angle '(260.0 280.0))
(set! $tmp '(260.0 285.0))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
:point_angle '(280.0 300.0))
(set! $tmp '(280.0 305.0))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
:point_angle '(300.0 320.0))
(set! $tmp '(300.0 325.0))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
:point_angle '(320.0 340.0))
(set! $tmp '(320.0 345.0))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
:point_angle '(340.0 360.0))
(set! $tmp '(340.0 365.0))
(otherwise
(display '(,self point otherwise))(newline)
(guard ((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(0.0 20.0))
(set! $tmp '(0.0 25.0))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(20.0 40.0))
(set! $tmp '(20.0 45.0))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(40.0 60.0))
(set! $tmp '(40.0 65.0))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(60.0 80.0))
(set! $tmp '(60.0 85.0))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(80.0 100.0))
(set! $tmp '(80.0 105.0))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(100.0 120.0))
(set! $tmp '(100.0 125.0))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(120.0 140.0))
(set! $tmp '(120.0 145.0))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(140.0 160.0))
(set! $tmp '(140.0 165.0))

```

```

((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
           :relative_angle '(160.0 180.0))
 (set! $tmp '(160.0 185.0)))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
           :relative_angle '(180.0 200.0))
 (set! $tmp '(180.0 205.0)))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
           :relative_angle '(200.0 220.0))
 (set! $tmp '(200.0 225.0)))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
           :relative_angle '(220.0 240.0))
 (set! $tmp '(220.0 245.0)))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
           :relative_angle '(240.0 260.0))
 (set! $tmp '(240.0 265.0)))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
           :relative_angle '(260.0 280.0))
 (set! $tmp '(260.0 285.0)))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
           :relative_angle '(280.0 300.0))
 (set! $tmp '(280.0 305.0)))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
           :relative_angle '(300.0 320.0))
 (set! $tmp '(300.0 325.0)))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
           :relative_angle '(320.0 340.0))
 (set! $tmp '(320.0 345.0)))
((?position :name_s $agents :from_coordinate '(0.0 0.0 -10000.0 0.0)
           :relative_angle '(340.0 360.0))
 (set! $tmp '(340.0 365.0)))
(otherwise
 (display '(,self body otherwise))(newline)
 (set! $tmp '(0.0 0.0))))))
(set! $tmp3
 (round-ex (+ 0.0
            (+
             (/ (+ (car $tmp) (cadr $tmp)) 2.0)
             ;(/ (+ (car $tmp2) (cadr $tmp2)) 2)
            ))
 0.01))
(!turn :angle $tmp3 :angle_velocity 360.0 :angle_acceleration 360.0)
(guard ((?position :name_s self
                 :rectangle '((374.0 -39.0)(377.5 -43.0)))
 (set! $route '(((375.0 -43.0)(375.0 -46.0)) s-1-c-e (s-2-c-e s-1-c-e)))
 (!!walk :route (car $route)

```

```

        :width $walk-w :velocity $walk-v :acceleration $walk-a
        :angle_velocity $turn-v :angle_acceleration $turn-a))
((?position :name_s self :from_coordinate '(375.0 -12.0 -41.0)
        :relative_angle '(160.0 200.0))
(set! $route '(((375.0 -41.0)) s-1-c-e (s-2-c-e s-1-c-e)))
(!!walk :route (car $route)
        :width $walk-w :velocity $walk-v :acceleration $walk-a
        :angle_velocity $turn-v :angle_acceleration $turn-a))
((?position :name_s self :from_coordinate '(378.0 -12.0 -41.0)
        :relative_angle '(160.0 200.0))
(set! $route '(((378.0 -41.0)(377.5 -50.0)) s-6-e (s-2-c-e s-1-e s-6-e)))
(!!walk :route (car $route)
        :width $walk-w :velocity $walk-v :acceleration $walk-a
        :angle_velocity $turn-v :angle_acceleration $turn-a))
((?position :name_s self :from_coordinate '(371.0 -12.0 -41.0)
        :relative_angle '(160.0 200.0))
(set! $route '(((371.0 -41.0)(370.5 -50.0)) s-6-w (s-2-c-w s-1-w s-6-w)))
(!!walk :route (car $route)
        :width $walk-w :velocity $walk-v :acceleration $walk-a
        :angle_velocity $turn-v :angle_acceleration $turn-a))
((?position :name_s self :from_coordinate '(379.0 -12.0 -5.0)
        :relative_angle '(160.0 200.0))
(set! $route '(((379.0 -5.0)) s-5-e (s-2-c-e s-5-e)))
(!!walk :route (car $route)
        :width $walk-w :velocity $walk-v :acceleration $walk-a
        :angle_velocity $turn-v :angle_acceleration $turn-a))
((?position :name_s self :from_coordinate '(371.0 -12.0 -5.0)
        :relative_angle '(160.0 200.0))
(set! $route '(((371.0 -5.0)) s-5-w (s-2-c-e s-5-w)))
(!!walk :route (car $route)
        :width $walk-w :velocity $walk-v :acceleration $walk-a
        :angle_velocity $turn-v :angle_acceleration $turn-a))
(otherwise
(guard ((?position :name_s self :from_coordinate '(0.0 0.0 -10000.0 0.0)
        :relative_angle '(0.0 20.0))
(set! $tmp2 '(0.0 20.0)))
((?position :name_s self :from_coordinate '(0.0 0.0 -10000.0 0.0)
        :relative_angle '(20.0 40.0))
(set! $tmp2 '(20.0 40.0)))
((?position :name_s self :from_coordinate '(0.0 0.0 -10000.0 0.0)
        :relative_angle '(40.0 60.0))
(set! $tmp2 '(40.0 60.0)))
((?position :name_s self :from_coordinate '(0.0 0.0 -10000.0 0.0)
        :relative_angle '(60.0 80.0))
(set! $tmp2 '(60.0 80.0)))

```

```

((?position :name_s self :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(80.0 100.0))
(set! $tmp2 '(80.0 100.0)))
((?position :name_s self :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(100.0 120.0))
(set! $tmp2 '(100.0 120.0)))
((?position :name_s self :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(120.0 140.0))
(set! $tmp2 '(120.0 140.0)))
((?position :name_s self :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(140.0 160.0))
(set! $tmp2 '(140.0 160.0)))
((?position :name_s self :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(160.0 180.0))
(set! $tmp2 '(160.0 180.0)))
((?position :name_s self :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(180.0 200.0))
(set! $tmp2 '(180.0 200.0)))
((?position :name_s self :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(200.0 220.0))
(set! $tmp2 '(200.0 220.0)))
((?position :name_s self :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(220.0 240.0))
(set! $tmp2 '(220.0 240.0)))
((?position :name_s self :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(240.0 260.0))
(set! $tmp2 '(240.0 260.0)))
((?position :name_s self :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(260.0 280.0))
(set! $tmp2 '(260.0 280.0)))
((?position :name_s self :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(280.0 300.0))
(set! $tmp2 '(280.0 300.0)))
((?position :name_s self :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(300.0 320.0))
(set! $tmp2 '(300.0 320.0)))
((?position :name_s self :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(320.0 340.0))
(set! $tmp2 '(320.0 340.0)))
((?position :name_s self :from_coordinate '(0.0 0.0 -10000.0 0.0)
:relative_angle '(340.0 360.0))
(set! $tmp2 '(340.0 360.0)))
(otherwise
(set! $tmp2 '(0.0 0.0)))
(set! $tmp3

```

```

      '(((round-ex (+ 0.0
                    (+
                      (* 3.0 (sin (deg->rad (/ (+ (car $tmp) (cadr $tmp)) 2))))
                      (* 3.0 (sin (deg->rad (/ (+ (car $tmp2) (cadr $tmp2)) 2))))
                    0.01)
      ,(round-ex (+ 0.0
                    (+
                      (* 3.0 (cos (deg->rad (/ (+ (car $tmp) (cadr $tmp)) 2))))
                      (* 3.0 (cos (deg->rad (/ (+ (car $tmp2) (cadr $tmp2)) 2))))
                    0.01))))
      (if (< (+ (abs (caar $tmp3)) (abs (cadar $tmp3))) 1.0)
          (set! $tmp3
              (begin
                  ;(display '($tmp ,$tmp)($tmp2 ,$tmp2))(newline)
                  '(((round-ex (+ 0.4 (* 3.0 (sin (deg->rad (/ (+ (car $tmp)
                                                                    (cadr $tmp)) 2)))) 0.0
                              ,(round-ex (+ 0.4 (* 3.0 (cos (deg->rad (/ (+ (car $tmp)
                                                                    (cadr $tmp)) 2)))) 0.0
                              )))
              ))
      (!!walk :route $tmp3
              :width $walk-w :velocity $walk-v :acceleration $walk-a
              :angle_velocity $turn-v :angle_acceleration $turn-a
              :relative #t)))
      (set! $agents '())
      (wait-for 2)
      (go scene_evac))
      ((?position :name_s self :from_coordinate '(0.0 0.0 0.0 0.0)
                :height '(-11.95 -7.6))
      (guard ((?position :name_s $agents :distance_range '(0.0 4.0) :angle_range '(-60.0 60.0))
              (set! $agents (return-list $agents))
              (cond ((> (length $agents) 10)
                     (set! $walk-v (* 0.5 *evac-walk-v*)))
                    ((> (length $agents) 7)
                     (set! $walk-v (* 0.7 *evac-walk-v*)))
                    ((> (length $agents) 4)
                     (set! $walk-v (* 0.9 *evac-walk-v*)))
                    (else
                     (set! $walk-v (* 1.0 *evac-walk-v*))))
              (set! $agents '()))
      (otherwise
      (set! $walk-v *evac-walk-v*)))
      (set! $route '(((373.6 -60.0)) d-1 (st-sd-2-u d-1)))
      (!!walk :route '((-0.6 -6.0))
              :width $walk-w :velocity $walk-v :acceleration $walk-a
              :angle_velocity $turn-v :angle_acceleration $turn-a

```

```

        :relative #t)
      (go scene_evac))
    ((?position :name_s $agents :distance_range '(0.0 2.5)
              :angle_range '(-40.0 40.0) :relative_angle '(-40.0 40.0))
      (set! $agents (car (if (list? $agents) $agents (list $agents))))
      (guard ((?observe :name_s $agents :action "walk")
              (!turn :to $agents :angle_velocity $turn-v :angle_acceleration $turn-a)
              (!!approach :to_s $agents
                        :distance 1.0 :velocity $walk-v :acceleration $walk-a
                        :angle_velocity $turn-v :angle_acceleration $turn-a))
              ((?observe :name_s $agents :action "approach")
              (!turn :to $agents :angle_velocity $turn-v :angle_acceleration $turn-a)
              (!!approach :to_s $agents
                        :distance 1.0 :velocity $walk-v :acceleration $walk-a
                        :angle_velocity $turn-v :angle_acceleration $turn-a))
              (otherwise
                (set! $agents '()))))
      (set! $follow-to $agents)
      (set! $agents '())
      (go scene_evac))
    )
  (scene_end
    (#t
      (display '(,self end))(newline)
      (!finish :action "walk")
      (!finish :action "approach")
      )))

```