Master Thesis

Modeling Action Rules through Participatory Simulation in Virtual Space

Supervisor Professor Toru Ishida

Department of Social Informatics
Graduate School of Informatics
Kyoto University

Yuki SUGIMOTO

Feburary 9, 2005

Modeling Action Rules through Participatory Simulation in Virtual Space

Yuki SUGIMOTO

Abstract

Multi-agent simulation is one of methods for experimenting on hard-to-reconstruct environments in the real world such as disaster simulation and traffic regulation. It need not consider restriction on how many times the experiment must be executed by mobilizing agents instead of humans, and it can treat such environments by constructing them in the virtual space. Since agents in the simulations must behave as humans do, human's action models must be obtained in advance. However, it is unrealistic to model human's actions through the real world experiments.

This study intends to develop an action modeling method utilizing participatory simulation in virtual space. Participatory simulation is such that avatars manipulated by humans are arranged in multi-agent simulation, in which interaction among humans and agents are permitted. This method is superior to that of real world experiment in the number of participants and in that accurate log data can be obtained, in addition to that hard-to-reconstruct environments can be constructed.

To utilize participatory simulation for action modeling, this research handles the following problems.

Quantitative restriction of log data Mobilizing humans in participatory simulation limits the experiment frequency. Therefore, it is difficult to construct a non-overfitting action model of a certain avatar inductively.

Keeping diversity In a case where all evacuees follow the same action model for supplementing the lack of the data, the variety of the obtained models is lost. But it is necessary to obtain also minority models in case of disaster evacuation.

The modeling process proposed by this research extracts the action rule of avatar from the already-known action rules in the domain and the log data about the avatar. The process consists of 3 steps; Domain knowledge acquisition step where general action rules in the domain are collected as domain knowledge, Log data processing step where the world states and the avatar's state changes which are to be the cause and the effects of rule-firing are extracted from the log data, and Action rule modeling step where the avatar's action rule set is obtained to seek for such rules as to explain the log data according to the framework of hypothetical reasoning. Domain knowledge can include such general but inconsistent rules as "Evade if there is congestion" and "Go to that if there is congestion". Different models mean different sets of rules, but every obtained model is a subset of the domain knowledge. A variety of action models can be acquired from the difference of the rule used though the action model obtained in this process is a subset of the domain knowledge.

In this paper, the above-mentioned action modeling process is applied to an evacuation experiment on virtual space. In the experiment, there are two roles, namely, evacuees and leaders. The leaders lead the evacuees in two leading methods to safely evacuate together. Domain knowledge is extracted from the interview to some of the participantss having participated in the experiment as evacuees. As a result, the modeling process was validated to apply the process to an avatar interviewed and compare the action rule sets that had been obtained from the domain knowledge and that from the interview to him.

The above-mentioned each problem was solved as follows by the modeling process advocated by this research.

- Already-known action rules in the domain are acquired as domain knowledge. An action model can be obtained to extract such rules as to explain the avatar's action from the domain knowledge even if the volume of the log data is small.
- Various models can be obtained to acquire one model from one agent log data.

In addition, the action model of humans who cannot be lead in safety with high possibility with the present leading methods was obtained applying the process to the avatar who had evacuated through a wrong exit. The obtained model can be utilized for establishing a new leading method through multi-agent simulations.

仮想空間を用いた参加型シミュレーションによる
行動ルールモデリング

杉本 悠樹

内容梗概

災害時の避難や，交通規制など，実世界では困難な実験を行うためのひとつの方法として，マルチエージェントシミュレーションが挙げられる．人間の代わりにエージェントを用いることで実験回数の制限を考慮する必要がなくなり，仮想空間上に実験環境を再現することで現実では困難な環境での実験を行うことが可能となる．このとき，エージェントは人間と同様の振る舞いをすることを期待されるため，予め人間の行動モデルを獲得しておく必要がある．しかしながら，実世界での実験が困難である場合，実験を通じた人間の行動モデリングは現実的ではない．

そこで本研究は，仮想空間での参加型シミュレーションを利用した行動モデリング手法を開発することを目的とする．参加型シミュレーションとは，人間の操作するアバターをマルチエージェントシミュレーションに配置し，環境内で，人間とエージェント間のインタラクションが行われるように設定したシミュレーションのことである．この手法は，実世界の実験を利用したモデリングに比べ，実世界では再現困難な環境を構築することができるという利点に加え，被験者の数や正確なログデータの獲得という点で，有利である．

参加型シミュレーションをモデリングに利用するために，本研究は以下の課題に取り組む．

訓練データ量の制限　参加型シミュレーションでは被験者を動員する必要があるため，実験回数に制約が課せられる．帰納学習によってモデリングを行うためには，大量の訓練データが必要となるが，実験回数の制約のため，帰納学習に適する量の訓練データを，ログデータから構築することは困難である

多様性の確保　データ量の不足を補うために，複数の被験者が同一のモデルに従うと仮定した場合，求まるモデルの多様性が失われてしまう．災害避難における誘導法の開発を考えると，参加型シミュレーションで，多くの避難者が指示に従って同じ方向へ向かう中，誤って間違った方向に向かう少

数の避難者がいたとき，この少数派のモデルも得られなければならない

本研究で提案するモデリングプロセスは，対象であるアバターに関するログデータと，対象領域での既知の人間の行動ルールから，実際にアバターが持つ行動ルールを抽出するものである．プロセスは3ステップからなり，それぞれ，対象領域における既知の人間の行動ルールを領域知識として収集する行動ルール収集ステップ，ルール発火の原因や結果となる，対象世界の状態やアバターの状態変化をログデータから抽出するログデータ加工ステップ，仮説推論の枠組みに従い，ログデータを説明するルールを求めることで，アバターが持っている行動ルール集合を獲得する行動モデル獲得ステップである．領域知識には，「混雑があれば回避する」「混雑があればそちらに行く」といった，一般的ではあるが人によって取捨選択が異なる行動ルールが含まれている．このプロセスで得られる行動モデルは，領域知識の部分集合であるが，用いたルールの違いから，様々な行動モデルが獲得できる．

本論文では，上記の行動モデリングプロセスを，仮想空間上の避難訓練に適用した．避難訓練には，避難者と誘導者の２つの役割がおり，誘導者が避難者を誘導し，共に安全に避難することを目的とする．領域知識は，避難者として参加した被験者数名に対するインタビューから抽出している．この結果，インタビューを行った被験者のアバターにモデリングプロセスを適用し，プロセスによって得られた行動ルール集合と，インタビューから得られた行動ルール集合の比較によって，モデリングプロセスの正当性を検証した．

本研究で提唱するモデリングプロセスにより，上述の各課題は以下のように解決された．

- 領域知識を利用した演繹的なモデリングプロセスにより訓練データ量の不足を解決した．領域知識は，文献，インタビューなどから得る，当該領域における人間の行動ルールのことである．
- 1体のアバターのログデータから１つの行動モデルを獲得するため，多様なモデルを得ることができる

更に，唯一避難の際に誤った出口から脱出したアバターにプロセスを適用することにより，従来の避難誘導法では安全に避難させられない可能性の高い人間の行動モデルを獲得することができた．獲得した行動モデルは，マルチエージェントシミュレーションを通じて，新たな避難誘導法の確立に利用することができる．

Modeling Action Rules through Participatory Simulation in Virtual Space

Contents

Chapter 1 Introduction

Multi-agent simulation is one of methods for experimenting on hard-to-reconstruct environments in the real world such as disaster simulation and traffic regulation. By mobilizing agents instead of humans, it need not consider restriction on how many times the experiment must be executed, and it can treat such difficult environments by constructing them in the virtual space. Since agents in the simulations must behave as humans do, human's action models must be obtained fully in advance.

The best way to get human action models is to gather people and to perform real world experiment to observe how those people act. Real world experiment, however, is sometimes difficult to perform as already mentioned.

This study intends to develop an action modeling method utilizing participatory simulation in virtual space. Participatory simulation is such that avatars manipulated by humans are arranged in multi-agent simulation, in which interaction among humans and agents are permitted. This method is superior to that of real world experiment in that accurate log data can be obtained, in addition to the fact that difficult environments can be constructed in virtual space.

Participatory simulation is easy to perform more times than that of real world in terms of the number of participants; it can utilize agents as participants. But it cannot be performed yet so many times as long as it mobilizes humans. To utilize participatory simulation for action modeling, this research handles the following problems.

Quantitative restriction of log data Mobilizing humans in participatory simulation limits the experiment frequency. Therefore, it is difficult to construct a non-overfitting action model of a certain avatar inductively with little amount of training data.

Keeping diversity In a case where all evacuees follow the same action model for supplementing the lack of the data, the variety of the obtained models is lost. But it is necessary to obtain also minority models in case of disaster evacuation.

This paper proposes a method to model human's action rules utilizing participatory simulation log data to overcome the problems above.

The construction of this paper showed below.

Chapter 2 explains participatory simulation in virtual space, which is focused in this study, and shows the characteristics of it and the problems in using it for modeling.Chapter 3 states the agent architecture of agents taking part in participatory simulation. Chapter 4 advocates the action rule modeling process for agents with the architecture presented in Chapter 3 to behave as same as the target avatar. Chapter 5 applies the modeling process to an exmaple of evacuation simulation. Chapter 6 is consideration on Chapter 5, and lastly this paper is concluded stating related works and future view.

Chapter 2 Participatory Simulation in Virtual Space

Participatory simulation in virtual space such simulation as to arrange avatars which human manipulates in multi-agent simulation in virtual space and to enable interaction among agents and avatars in it.

Figure 1: Evacuation simulation in virtual space

Participatory simulation treated in this study is evacuation training in virtual space (Figure 1).FreeWalk[1] is utilized as the platform of 3D virtual space.

an advantage in utilizing participatory simulation for action modeling are the fact that i) human action data in difficult domain to experiment in the real world, such as traffic regulation and emergency drill, can be obtained since the stage of simulation is on computers, ii) because agents take part, large-scale simulation can be done with less human participants than that of real world experiment, and iii) precise reproduction can be done with log data of simulation.

Though simulation can be done mobilizing less humans than real world experiment, there is limitation on how many times the simulation can be executed.

Consequently it is difficult to produce so much training data from obtained log data as to learn human's action model inductively.

To learn human action model from log data of participatory simulation, the problem that the size of training data set for learning is small.

Accordingly, this study models human's action rules explaining the training data with domain knowledge, which consists of already-known human action rules in the domain.

Chapter 3 Agent Architecture

The action rule modeling process intends to obtain the action rules of an avatar which human operates from the log data related to the avatar. An agent given the action rules behaves as the same way as the avatar.

This chapter states agent architecture for agents with action rules to behave.

Agents have several If-Then action rules and change their own states by conflict resolution and rule firing.

3.1 World State

This study supposes that an agent has a set of action rules, that it decides its action firing a rule reactively to the world state, and that agents have no memory on the world state.

Since the field of vision of an agent in virtual space is limited, an agent cannot observe the outside phenomena. Let S be the local world state that the agent recognizes, but S does not include the agent's own state. Generally, time t is continuous and S may change along with time t, however, since this study attaches importance to matching between S and action rules, time t is assumed to be discrete and incremented as S changes.

Let S_t be the local world state that the agent recognizes at the time t. A state consists of *predicates* and *constants*. Let $Pred$ be the set of predicates and constants to describe S.

S_t is a set of positive literals. S_t means that every literal in S_t is true at the time t. Because this is equivalent to the fact that the conjunction of all literals in S_t is true, set and conjunction are considered to be the same. S_t does not include negative literals. A literal which does not appear in S_t explicitly is false, that is, the negation of the literal is true.

3.2 Agent State

What an agent can only change by firing an action rule is its own state. Let $Pred_{agent}$ be the set of predicates and constants with which the agent's state can be described (therefore $Pred_{agent} \nsubseteq Pred$), and let S^{agent} be the agent's own state. The agent's state at the time t is describes as S_t^{agent}. S_t^{agent} includes the agent's action(walking forward, turning to the right, and so on). Since S_t does not include the agent's own state at the time t, $\forall t(S_t \cap S_t^{agent} = \emptyset)$ is true.

3.3 Action Rule

An agent changes its own state according to action rules. An action rule consists of the precondition and the effect. If the local world state satisfies the precondition of a rule, the agent's state is changed as described in the rule's effect.

Definition Action rule

An action rule a consists of the **precondition** $a^{precond}$ and the **effect** a^{effect}.

$$\begin{aligned} a &= (a^{precond}, a^{effect}) \\ \forall l(l \in a^{precond} &\Rightarrow (l \text{ is made from a predicate or a constant in } Pred)) \\ \forall l(l \in a^{effect} &\Rightarrow (l \text{ is made from a predicate or a constant in } Pred_{agent})) \end{aligned}$$

The symbol $\neg$ means "negation".

The precondition $a^{precond}$ is the set of literals that must be true for firing the action rule $a^{precond}$. What $a^{precond}$ is satisfied means that every literal in $a^{precond}$ is true, which is equivalent to the fact that the conjunction of all literals in $a^{precond}$ is true. Since both S_t and S_t^{agent} have literals which are true as positive literals in them and do not have negative literals, $a^{precond}$ has literals which should be included in $S_t \cup S_t^{agent}$ as the positive literals and has literals which should not be included in $S_t \cup S_t^{agent}$ as the negative literals in it. Let $a_i^{precond,-}$ be the set of all negative literals in $a^{precond}i$, $a_i^{precond,+}$ be the set of all positive literals in $a_i^{precond}$. Let $Positive(l)$ be a predicate which is true if and only if the literal l is positive, $Negative(l)$ be a predicate which is true if and only if l is negative. $a_i^{precond,-}$ and $a_i^{precond,+}$ are described as follows.

$$\begin{aligned} a^{precond,-} &= \{l \in a^{precond} | Negative(l)\} \\ a^{precond,+} &= \{l \in a^{precond} | Positive(l)\} \end{aligned}$$

$a^{precond}$, $a^{precond,-}$ and $a^{precond,+}$ satisfy the following equations.

$$a^{precond} = a^{precond,-} \cup a^{precond,+}$$
$$a^{precond,-} \cap a^{precond,+} = \emptyset$$

The effect a^{effect} is the set of literals, which shows how the agent's state changes when the action rule a are fired. Literals which are deleted from S^{agent} are descrined as the negative literals and literals which are added to S^{agent} are described as the positive literals. Let $a^{effect,-}$ be the set of all negative literals in a^{effect} and $a^{effect,+}$ be the set of all positive literals in a^{effect}. $a^{effect,-}$ and $a^{effect,+}$ are defined by the following equation.

$$\begin{aligned} a^{effect,-} &= \{l \in a^{effect} | Negative(l)\} \\ a^{effect,+} &= \{l \in a^{effect} | Positive(l)\} \end{aligned}$$

a^{effect}, $a^{effect,-}$ and $a^{effect,+}$ satisfies the following equation.

$$a^{effect} = a^{effect,-} \cup a^{effect,+}$$
$$a^{effect,-} \cap a^{effect,+} = \emptyset$$

Let A^{agent} be the set of all the action rules that the agent has.

3.4 Firing Rules

An agent fires an action rule a when the set $S_t \cup S_t^{agent}$ satisfies the precondition $a^{precond}$. The reason why the union of S_t and S_t^{agent} is used is that trigger of rule firing is related not only what the agent has observed but also what the agent is now. Therefore, $a^{precond}$ is allowed to include literals on the agent's own state.

If a rule is fired, the agent's state changes to be the state of the next time $t+1$, S_{t+1}^{agent}. If no rule is fired, $S_t^{agent} = S_{t+1}^{agent}$. Total order $\leq_{agent}$ is defined for A_{agent} that is the agent's action rule set. In a case where two or more rules are firable, the most prior rule according to $\leq_{agent}$ is fired. Consequently, An agent's behavior is determined by a totally ordered set $(A_{agent}, \leq)$. As for $a_i, a_j \in A_{agent}$, if both $a_i \leq_{agent} a_j$ and $a_i \neq a_j$ are true, rule a_j is fired prior to a_i. "$a_i \leq_{agent} a_j$ and $a_i \neq a_j$" is described merely as $a_i <_{agent} a_j$.

Rule firing is defined as follows.

Definition Firable rule

Given S_t and S_t^{agent}, action rule a is **firable** at the time t if and only if the precondition $a^{precond}$ is true.

$$\text{"}a \text{ is firable at the time } t\text{"} \Leftrightarrow ((S_t \wedge S_t^{agent}) \Rightarrow a^{precond})$$

Seeing S_t, S_t^{agent} and $a^{precond}$ are also sets of literals, The definition is rewrited as follows.

$$\begin{aligned} \text{"}a \text{ is firable at the time } t\text{"} \quad &\Leftrightarrow \quad (a^{precond,+} \subseteq (S_t \cup S_t^{agent})) \\ &\wedge \quad \forall l(l \in a^{precond,-} \Rightarrow \neg l \notin (S_t \cup S_)^{agent}) \end{aligned}$$

It requires that all literals in $a^{precond,+}$ should appear in $(S_t \cup S_t^{agent})$ and that the negations of all literals in $a^{precond,-}$ (note that all the literals in $a^{precond,-}$ are negative) should not appear in $(S_t \cup S_t^{agent})$

If there are two or more firable rules in A_{agent}, conflict resolution must be done. The set of all firable rules is called the **instantiation set**. Let I_t be the instantiation set at the time t.

Definition Instantiation

$$I_t = \{a | (a \in A_{agent}) \wedge (a \text{ is firable at the time } t)\}$$

An agent decides the rule to be fired by choosing one rule from I_t. According to the order $\leq_{agent}$, the most prior rule is fired.

Definition Selection of the firing rule (Conflict resolution)

The rule that is fired at the time t is

$$a = \mathrm{max(I_t, \leq_{agent})}$$

in which $\mathrm{max(I, \leq) = a\,such that\,\forall b(b \in I \Rightarrow b \leq a)}$.

If a rule is fired, the agent's state is changed as described in the rule's effect. Let a be the firing rule. Since S_{t+1}^{agent} is produced be deleting all literals that are negative literals in a^{effect} from S_t^{agent} and adding all positive literals in a^{effect} to it, the following equation is true.

$$S_{t+1}^{agent} = (S_t^{agent} - \{l | \neg l \in a^{effect,-}\}) \cup a^{effect,+}$$

If $I_t = \emptyset$, rule firing does not occur, and therefore, $S_{t+1}^{agent} = S_t^{agent}$.

3.5 Action Mechanism

According to the definitions above, action mechanism of agents is described below. The action mechanism is also called a process to produce the sequence $\{S_t^{agent}\}$. The agent's state at the time 0, S_0^{agent} is assumed to be given.

State decision process of agent

```
1. $t \leftarrow 0$
2. $S_t$ is obtained by the agent's observation
3. $I_t \leftarrow \{a | (a \in A_{agent}) \wedge (a \text{ is firable at the time } t)\}$
4. If $I_t = \emptyset$, $S_{t+1}^{agent} \leftarrow S_t^{agent}$. Otherwise, $a \leftarrow \mathrm{max(I_t, \leq_{agent})}$ and $S_{t+1}^{agent} \leftarrow (S_t^{agent} - \{l | \neg l \in a^{effect,-}\}) \cup a^{effect,+}$.
5. $t \leftarrow t + 1$ and go to 2
```

First, the agent observes the world in the current state. Since the field of view of an agent with human body model in virtual reality is limited, the world state observed is local, not global (step 2).

Next, the agent searches for the firable rules on $S_t \cup S_t^{agent}$ in the agent's action rule set A_{agent}. I_t is the set of the firable rules (step 3).

If there are two or more firable rules in I_t, the agent executes conflict resolution to choose one firing rule. Between the rules the agent has, there is an rule firing priority order $\leq$ *agent*. The most prior rule is fired. After rule firing, the agent changes its own state S_t^{agent} according to the fired rule. If there is no rule firable, rule firing is not occur and the agent's state does not change (step 4).

Chapter 4 Action Rule Modeling Process

This chapter states the action rule modeling process. An agent with the architecture shown in Chapter 3 behaves as the same way as the target avatar if given the action model obtained from the process.

The action rule modeling process consists of 3 steps; Domain knowledge acquisition step where general action rules in the domain are collected as domain knowledge, Log data processing step where the world states and the avatar's state changes which are to be the cause and the effects of rule-firing are extracted from the log data, and Action rule modeling step where the avatar's action rule set is obtained to seek for such rules as to explain the log data according to the framework of abduction[2].

4.1 Domain Knowledge Acquisition

In Domain knowledge acquisition step, action rules which are general for human in the domain are collected as domain knowledge A.

Domain knowledge in this study is a set of If-then form action rules which claims "humans generally act like this in the domain". The rules are not always true in the domain, and people may have different opinion on whether the rule is right or wrong. Domain knowledge can be obtained by documents, by interviews with people related to the domain, by specialists' opinions.

As is the case with agent architecture (Chapter 3), avatars which humans manipulate are assumed to behave reactively to their observation by firing rules. A rule has the condition that the local world state should satisfy as its precondition, and the avatar's state change after rule firing as its effect. Because this study focuses on avatar's action, avatar's state consists of action description such as "walking forward", "turning".

4.2 Log Data Processing

Log data processing step makes symbolic description of the world states $\{S_t\}$ and the avatar's state $\{S_t^{avatar}\}$ which are to be the cause and the effects of rule-firing from the log data.

Since an avatar's action rule is fired by $\{S_t\}$ and $\{S_t^{avatar}\}$ and changes its own state $\{S_t^{avatar}\}$, $\{S_t\}$ and $\{S_t^{avatar}\}$ should be described with the vocabulary utilized to describe the domain knowledge.

Due to the characteristics of virtual space, agents' position in the space and users' key inputs can be precisely obtained as log data. In this study, sequences of position of avatars and agents are utilized.

4.3 Action Rule Modeling

The action rule modeling process calculates the avatar's action rule set and its order by which sequence of the avatar's states $\{S_t^{avatar}\}$ is lead from sequence of the local world state $\{S_t\}$ and rule firing. The avatar's action rule set A_{avatar} is a subset of the domain knowledge.

The action rule set A_{avatar} and order $\leq_{avatar}$, which are the output of the process, are required to satisfy the following requirements.

- $A_{avatar} \subseteq A$
- Guarantee of reproducing the avatar's behavior

  When the action model $(A_{avatar}, \leq_{avatar})$ and $\{S_t\}$ are given to an agent which follows "state decision process of agent" in 3.5, the agent always generates $\{S_t^{agent}\}$ such that $\forall t(0 \leq t \leq t_{end} \Rightarrow S_t^{agent} = S_t^{avatar})$ is true (t_{end} is the end time).

Though an agent's action rule set is assumed to be an totally ordered set, some pairs of rules may be exist between which rule firing priority cannot be determined from only one example of log data. Therefore, $\leq_{avatar}$ is allowed to be a partially ordered set.

The partially ordered set $(A_{avatar}, \leq_{avatar})$ that satisfy the above requirements is calculated using $\{S_t\}$, $\{S_t^{avatar}\}$ and the domain knowledge of action rules A by the framework of *abduction*[2]. Assume that A is a set of already-known action rules extracted from documents or interviews and is a finit set.

4.3.1 Abduction

Abduction (or abductive reasoning, hypothetical reasoning) is a inference mechanism where if there is a proposition with whether it is true or false unknown, a hypothesis that it is true is generated and if observation is completely explained

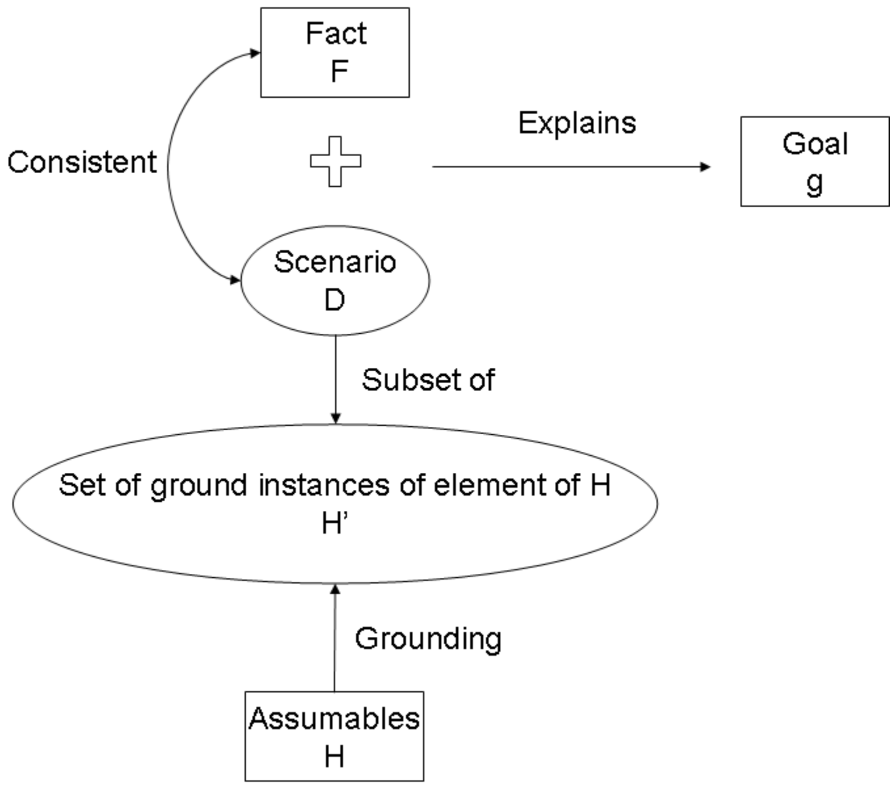

Figure 2: Abduction

without contradiction, the hypothesis is certainly true (Figure 2).

An assumption-based scheme is a pair $\langle F, H \rangle$ where

F is a set of closed formulae called the 'Facts',

H is a set of formulae called the 'assumables' or the 'possible hypotheses'. Let H' be the set of ground instances of element of H.

Definition Scenario

A **scenario** of $\langle F, H \rangle$ is a subset D of H' such that $F \cup D$ is consistent.

Definition Explanation

If g is a ground formula, an **explanation** of g from $\langle F, H \rangle$ is a scenario of $\langle F, H \rangle$ that together with F implies g.

Thus, if g is a closed formula, an explanation of g from $\langle F, H \rangle$ is a set D of elements of H' such that

- $F \cup D \models g$ and
- $F \cup D \not\models false$

Definition Extension

An **extension** of $\langle F, H \rangle$ is the consequence of F together with a maximal (with respect to set inclusion) scenario of $\langle F, H \rangle$.

Definition Minimal Explanation

A **minimal explanation** of g is an explanation of g such that no strict subset is also an explanation of g.

A straight-forward implementation to find a minimal explanation is as follows if H' is a finite set; $H' = h_0, h_1, \ldots, h_n$.

Function $\mathrm{Abduction(i, D)}$

```
1. If $F \cup D \not\models false$ then return $fail$
2. If $F \cup D \models g$ then return $D$
3. If $i > n$ then return $fail$
4. If $\mathrm{Abduction(i+1, D) \neq fail}$ then return $\mathrm{Abduction(i+1, D)}$
5. return $\mathrm{Abduction(i+1, D \cup h_i)}$
```

Each element defined in this paper corresponds to each element in abduction as follows.

- Goal g: $(A_{avatar}, \leq_{avatar})$ reproduces the avatar's action
- Fact F: $A_{avatar} \subseteq A$ is true and $\leq_{avatar}$ has no contradiction
- Assumables H: $H = \{a \in A_{avatar}, b \leq_{avatar} c\}$ where a, b, c are free variables
- H'; the set of ground instances of element of H: $H' = \{a_0 \in A_{avatar}, a_1 \in a_{avatar}, \ldots, a_0 \leq_{avatar} a_1, a_0 \leq_{avatar} a_1, a_1 \leq_{avatar} a_2, \ldots\}$

In the implementation, the number of hypotheses (the cardinality of H') is limited by calculating all firable rules in A at the time t.

4.3.2 Implementation of Action Rule Modeling

First, a rule set of rules to be fired and a rule set of rules firable is calculated at each time t. C_t, which is a rule set of rules firable, is the set of firable rules at the time t in A. P_t, which is a rule set of rules to be fired, is the set of rules which are firable at the time t and lead S^{avatar}_{t+1} correctly. The avatar is considered to fire a rule in P_t at the time t. After calculating $\{C_t\}$ and $\{P_t\}$, the process calls function GetModel, which returns the avatar's action rule set and its order based on $\{C_t\}$ and $\{P_t\}$. GetModel calculates the avatar's action rule set and its order recursively.

Function Action rule modeling $\mathrm{ActionRuleSet}(\mathrm{t_{end}}, \{\mathrm{S_t}\}, \{\mathrm{S_t^{avatar}}\})$

Inputs

t_{end}: the end time

$\{S_t\}$: a sequence of the local world state

$\{S_t^{avatar}\}$: a sequence of the avatar's state

A: domain knowledge which is a set of action rules

Outputs

$(A_{avatar}, \leq_{avatar})$: an action rule set which the avatar may utilize A_{avatar} and a partial order $\leq_{avatar}$

Process

```
1. $N \leftarrow \{\},\ t \leftarrow 0$
2. If $t = t_{end}$, ActionRuleSet finishes retuning GetModel$(0, \{\}, \{\})$
3. Let $C_t$ be the set of firable action rules in $A$. $C_t \leftarrow \{a | (a \in A) \wedge (a \text{ is firable at the time } t)\}$
4. If $S_t^{avatar} = S_{t+1}^{avatar}$, $N \leftarrow N \cup C_t$, $t \leftarrow t+1$ and go to 2
5. Let $P_t$ be the set of rules which make the avatar's state equal to $S_{t+1}^{avatar}$.
   $P_t \leftarrow \{a \in C_t | S_{t+1}^{avatar} = (S_t^{avatar} - \{l | \neg l \in a^{effect,-}\}) \cup a^{effect,+}$
6. $t \leftarrow t+1$ and go to 2
```

The variable N is the set of rules which are firable when $S_t^{avatar} = S_{t+1}^{avatar}$, that is, no rule firing should occur. Since it is not probable for the avatar to have such rules, the avatar's action rule set is necessarily a subset of $A - N$.

Function GetModel has recursive structure. It determines an action rule to be included the avatar's action rule set and also determines the order between rules so as not to be inconsistent with that of the previous time. The definition is as follows. Suppose that $\{C_t\}$, $\{P_t\}$, N and t_end are given by ActionRuleSet.

Function $\mathrm{GetModel}(\mathrm{t}, \mathrm{A_{avatar}}, \mathrm{R})$

Inputs

t: time

A_{avatar}: an action rule set
R: a partial order

Outputs
$(A_{avatar}, \leq_{avatar})$: an action rule set which the avatar may utilize A_{avatar} and a partial order $\leq_{avatar}$

Process

```
1. If $t = t_{end}$, $\leq \leftarrow R$ and GetModel finishes returning $(A_{avatar}, \leq)$
2. If $S_t^{avatar} = S_{t+1}^{avatar}$, return $\mathrm{GetModel(t+1, A_{avatar}, R)}$
3. Suppose $P_t = \{a_1, a_2, \ldots, a_m\}$
4. **for** $i \leftarrow 0$ **to** $m$
   (a) If $a_i \in N$, **next** $i$
   (b) $R' \leftarrow R \cup \{(a_j, a_i) | a_j \in C_t - N\}$
   (c) $R'' \leftarrow R' \cup \{(a_x, a_y) | \exists a_z((a_x, a_z) \in R' \wedge (a_z, a_y) \in R')\}$
   (d) If $R' \neq R''$, $R' \leftarrow R''$ and go to 4c
   (e) If $R'$ as a partial order has no contradiction and $\mathrm{GetModel(t{+}1, A_{avatar} \cup \{a_i\}, R') \neq fail}$, return $\mathrm{GetModel(t+1, A_{avatar} \cup \{a_i\}, R')}$
   (f) **next** $i$
5. GetModel finishes retuning $fail$
```

In step 4e, GetModel has to judge whether R' is a partial order or not. Because the pair of A_{avatar} and R, which are the arguments of GetModel, (A_{avatar}, R) must be guaranteed to be a partially ordered set, based on the arguments in the next call of GetModel, whether $(A_{avatar} \cup \{a_i\}, R')$ is a partially ordered set or not is judged. If R' satisfies the following characteristics, $(A_{avatar} \cup \{a_i\}, R')$ is a partially ordered set. Otherwise it is a partially ordered set.

Reflexivity $\forall a(a \in A_{avatar} \cup \{a_i\} \Rightarrow (a, a) \in R')$

Antisymmetry $\forall a \forall b(b \in A_{avatar} \cup \{a_i\} \wedge b \in A_{avatar} \cup \{a_i\} \wedge (a, b) \in R' \wedge (b, a) \in R' \Rightarrow a = b)$

Transitivity $\forall a \forall b \forall c (a \in A_{avatar} \cup \{a_i\} \wedge b \in A_{avatar} \cup \{a_i\} \wedge c \in A_{avatar} \cup \{a_i\} \wedge (a,b) \in R' \wedge (b,c) \in R' \Rightarrow (a,c) \in R')$

The fact that the loop consisting of step 4c and step 4d finishes in finite time is proved as follows.

Proof Escape of loop

The fact that when GetModel$(0, \{\}, \{\})$ is called, in all recursive calls of GetModel

$$R \subseteq A \times A \tag{1}$$

is true and the loop finishes in finite time is proved.

1. If $t = 0$, $R = \{\}$ because of GetModel(0,{},{}). Therefore (1) is true.
   In step 4b, $R' = R \cup \{(a, a_i) | a \in C_0 - N\}$. Due to $C_0 \subseteq A$ and $a_i \in P_0 \subseteq A$, as to R', similarly $\forall a \forall b ((a,b) \in R' \Rightarrow a \in A \wedge b \in A)$, that is,

   $$R' \subseteq A \times A \tag{2}$$

   is true. In step 4c, $R'' = R' \cup \{(a,b) | \exists c((a,c) \in R' \wedge (c,b) \in R')\}$. a, b and c in this formula satisfy $a, b, c \in A$ because of (1). Thus, as to R'',

   $$\begin{aligned} R'' &\subseteq A \times A \qquad (3) \\ R' &\subseteq R'' \qquad (4) \end{aligned}$$

   is true. If $R' \neq R''$, $|R'| + 1 \leq |R''|$. Because R' in the next loop is R'' ($R' \leftarrow R''$), R' satisfies (2) again, and the next R'' satisfies (3) and (4), and $|R'| + 1 \leq |R''|$. As long as $R' \neq R''$ the number of elements in R' increases by 1 or more. Because of the restriction of (2) and that A is a finite set, $|R'| \leq |A|^2$ regardless of the number times of the loop. Thus, the loop always finishes in finite time.
2. if $t = k \, (0 \leq k < t_{end})$, assume that (1) is true and the loop finishes in finite time.
   By step 4e, R' at the time $t = k$ becomes R at the time $t = k + 1$. Since R' at the time $t = k$ satisfies (2) by the same argument as mentioned above, R at the time $t = k+1$ also satisfies (1) and the loop finishes in finite time.

By mathematical induction, at any time t such that $0 \leq t \leq t_{end}$, R satisfies (1), and the loop consisting of step item-transitivity and step 4d finishes in finite time.□

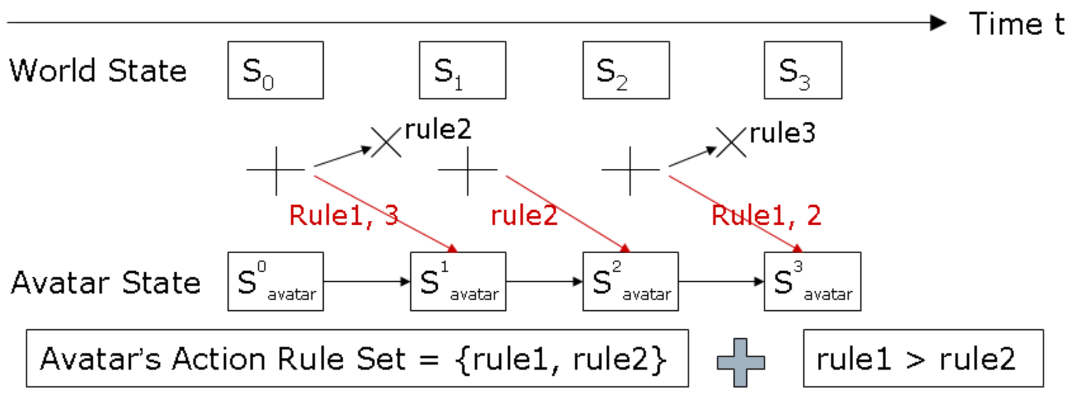

Figure 3: Action rule modeling

Figure 3 shows behavior of ActionRuleSet and GetModel with an easy example.

$t = 1$. Rule1, rule2 and rule3 are firable in the domain knowledge A, and firing rule1 or rule3 leads the avatar's next state S_2^{avatar} correctly, but in a case where rule2 is fired, the next state does not equal to S_2^{avatar}.

$t = 2$. Only rule2 is firale and firing rule2 leads S_3^{avatar} correctly.

$t = 3$. Rule1, rule2 and rule3 are firable, but rule3 must not be fired.

Given such $\{S_t\}$, $\{S_t^{avatar}\}$ and A, ActionRuleSet returns set$\{rule1, rule2\}$(firing priority: $rule2 < rule1$).

The result of calculation of ActionRuleSet is as follows.

$$\begin{aligned} C_0 &= \{rule1, rule2, rule3\}, & P_0 &= \{rule1, rule3\} \\ C_1 &= \{rule2\}, & P_1 &= \{rule2\} \\ C_2 &= \{rule1, rule2, rule3\}, & P_2 &= \{rule1, rule2\} \end{aligned}$$

GetModel calculates the answer as follows.

- $t = 0$. $Rule1$ is selected from P_0 to be added to A_{avatar}. $rule2 < rule1$, $rule3 < rule1$
- $t = 1$. $Rule2$ is selected from P_1 to be added to A_{avatar}

- $t = 2$. $P_2 = \{rule1, rule2\}$. If $rule2$ is selected, "$rule1 < rule2$" contradicts "$rule2 < rule1$" at $t = 0$. On the other hand, selecting $rule1$ does not have any contradiction. Thus $rule1$ is selected and $A_{avatar} = \{rule1, rule2\}(rule2 < rule1)$ is returned

The fact that this action rule set certainly reproduces the avatar's state changes can be confirmed as follows.

- $t = 0$. The firable rules in A_{avatar} by $S_0 \cup S_0^{avatar}$ are $rule1$ and $rule2$. The firing rule is $rule1$ because of the order. Thus, the next state equals to S_1^{avatar}.
- $t = 1$. The firable rule by $S_1 \cup S_1^{avatar}$ is $rule2$. $Rile2$ is fired and the next state becomes S_2^{avatar}.
- $t = 2$. The firable rules in A_{avatar} by $S_2 \cup S_2^{avatar}$ are $rule1$ and $rule2$. $Rule1$ is selected as the firing rule, and firing $rule1$ makes the next state equal to S_3^{avatar}.

The avatar's state changes can be reproduces certainly by the action rule set obtained using ActionRuleSet.

At $t = 0$ in GetModel, because of $P_0 = \{rule1, rule3\}$, there is possibility that $rule3$ is selected. But since the order becomes $rule1 < rule3$ and $rule2 < rule3$, contradiction is occurred at $t = 2$.

4.4 Restriction of Avatar's model

The fact that the action model $(A_{avatar}, \leq_{avatar})$ calculated by ActionRuleSet satisfies the following requirements is proved.

- $A_{avatar} \subseteq A$
- Guarantee of reproducing the avatar's behavior
  When the action model $(A_{avatar}, \leq_{avatar})$ and $\{S_t\}$ are given to an agent which follows "state decision process of agent" in 3.5, the agent always generates $\{S_t^{agent}\}$ such that $\forall t(0 \leq t \leq t_{end} \Rightarrow S_t^{agent} = S_t^{avatar})$ is true (t_{end} is the end time).

Proof $A_{avatar} \subseteq A$

$A_{avatar} \subseteq A - N$ is proved. N is calculated by ActionRuleSet.

The process of GetModel must be considered. Let A^n_{avatar} be the argument A_{avatar} at the time $t = n\,(0 \leq n \leq t_{end})$.

1. $t = 0$

   $t = 0$ occurs when call of GetModel at step 2 in ActionRuleSet. Thus $A_{avatar} = \{\}$ and $A^0_{avatar} \subseteq A - N$.

2. $t = k\,(0 \leq k < t_{end})$.

   Assume that $A^k_{avatar} \subseteq A - N$.

   (a) If $S^{avatar}_k = S^{avatar}_{k+1}$, $A^{k+1}_{avatar} = A^k_{avatar}$ by step 2 in GetModel. Therefore, $A^{k+1}_{avatar} \subseteq A - N$ by the assumption.

   (b) If $S^{avatar}_k \neq S^{avatar}_{k+1}$,

   $C_k \subseteq A$ because $C_n \subseteq A$ for arbitrary $n\,(0 \leq n < t_{end})$ by step 3 in ActionRuleSet. $P_k \subseteq C_k$ by step 5. $a_i \in P_k$ by step 3 in GetModel, thus $\{a_i\} \subseteq A$. On the other hand, $A^{k+1}_{avatar} = A^k_{avatar} \cup \{A^i\}$ by step 4e, and this a_i satisfies $a_i \notin N$ by 4a. Therefore, $\{a_i\} \subseteq A - N$, and $A^{k+1}_{avatar} \subseteq A - N$ by the assumption.

As showed above, assuming $A^k_{avatar} \subseteq A - N$ makes $A^{k+1}_{avatar} \subseteq A - N$ true. $A^t_{avatar} \subseteq A - N$ at any time t such that $0 \leq t \leq t_{end}$ by mathematical induction.

A_{avatar} finally obtained is proved to be $A^{t_{end}}_{avatar}$ by step 1 in GetModel. Therefore $A_{avatar} \subseteq A - N \subseteq A$. □

Proof Guarantee of reproducing the avatar's behavior

An agent produces $\{a^{agent}_t\}$ according to the following process.

State decision process of agent

1. $t \leftarrow 0$
2. S_t is obtained by the agent's observation
3. $I_t \leftarrow \{a | (a \in A_{agent}) \wedge (a \text{ is firable at the time } t)\}$
4. If $I_t = \emptyset$, $S^{agent}_{t+1} \leftarrow S^{agent}_t$. Otherwise, $a \leftarrow \max(\mathrm{I_t}, \leq_{\mathrm{agent}})$ and $S^{agent}_{t+1} \leftarrow (S^{agent}_t - \{l | \neg l \in a^{effect,-}\}) \cup a^{effect,+}$.
5. $t \leftarrow t + 1$ and go to 2

By comparing an agent's output to which the action model $(A_{avatar}, \leq_{avatar}$) obtained by ActionRuleSet is given and the avatar's actual state sequence, the fact that $S^{agent}_t = S^{avatar}_t$ at any time t $(0 \leq t \leq t_{end})$ when the model

$(A_{agent}, \leq_{agent})$ in the above process equals to $(A_{avatar}, \leq_{avatar})$. As the initial agent state cannot be obtained and must be given in advance, let $S_0^{agent} = S_0^{avatar}$.

1. $t = 0$. $S_0^{agent} = S_0^{avatar}$.
2. $t = k\,(0 \leq k < t_{end})$. Assume that $S_k^{agent} = S_k^{avatar}$.
   (a) If $S_{k+1}^{avatar} = S_k^{avatar}$
   Every firable rule at the time k in A is included in N by step 3 and step 4 in ActionRuleSet

$$\begin{aligned} C_k &= \{a|(a \in A) \wedge (a \text{ is firable at the time } k)\} \\ C_k &\subseteq N \end{aligned}$$

   Meanwhile, by the proof of $A_{avatar} \subseteq A$, $A_{avatar} \subseteq A-N$. Thus A_{avatar} does not include any element in N.

$$\begin{aligned} A_{avatar} \subseteq A - N &= \{a|a \in A \wedge a \notin N\} \\ A_{avatar} \cap N &= \emptyset \end{aligned}$$

   Therefore

$$A_{avatar} \cap C_k = \emptyset$$

   and there is no rule firable at the time k in A by $A_{avatar} \subseteq A$. By $A_{agent} = A_{avatar}$,

$$I_k = \{a|(a \in A_{agent}) \wedge a \text{ is firable at the time } k\} = \emptyset$$

   Thus, $S_{k+1}^{agent} = S_k^{agent}$ and $S_{k+1}^{agent} = S_k^{agent} = S_k^{avatar} = S_{k+1}^{avatar}$ by the assumption.
   (b) If $S_{k+1}^{avatar} \neq S_k^{avatar}$
   The firing rule is $\mathrm{max(I_k, \leq_{agent})} = \mathrm{max(I_k, \leq_{avatar})}$. $I_k = \{a|(a \in A_{agent}) \wedge (a \text{ is firable at the time } k)\}$ by definition. By $A_{agent} = A_{avatar}$ and $A_{avatar} \subseteq A-N$ proved above, $I_k \subseteq \{a|(a \in A-N) \wedge (a \text{ is firable at the time } k)\}$
   By this and
   $C_k = \{a|(a \in A) \wedge (a \text{ is firable at the time } k)\}$, $I_k \subseteq C_k - N$. $\mathrm{max(C_k - N, \leq_{avatar}) \in A_{avatar}}$ by step 4b and step 4e and $\mathrm{max(C_k - N, \leq_{avatar}}$

$) \in \mathrm{I_k}$. Thus, $\mathrm{max(C_k - N, \leq_{avatar}) = max(I_k, \leq_{avatar})}$. Because of $\mathrm{max(C_k - N, \leq_{avatar}) \in P_k}$, $\mathrm{max(I_k, \leq_{avatar}) \in P_k}$. By step 5 in ActionRuleSet, in a case where the agent's state at the time k is S_k^{avatar}, after firing an arbitrary rule in P_k, the agent's next state S_{k+1}^{agent} equals to S_k^{avatar} because S_k^{agent} equals to S_k^{avatar}. $S_{k+1}^{agent} = S_{k+1}^{avatar}$.

At any time t such that $0 \leq t \leq t_{end}$, $S_t^{agent} = S_t^{avatar}$ is true by mathematical induction.□

Chapter 5 Action Rule Modeling in Evacuation Domain

This chapter states application of the action rule modeling process to evacuation simulation in virtual space.

5.1 Sugiman Experiment

This study applied the modeling process advocated in this paper to participatory simulation in 3-dimension virtual space based on evacuation drill experiment by Professor Sugiman (called Sugiman experiment in the following) [3].

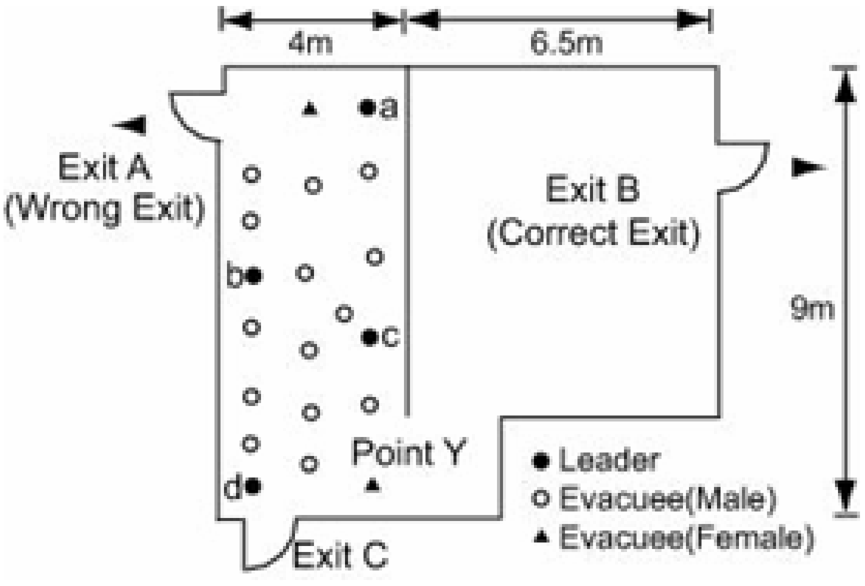

Figure 4: Sugiman Experiment[3]

Sugiman experiment is performed to compare effectivity of two evacuation method; Follow-direction method and Follow-me method. Follow-direction method is such frequently used method as to point to the exit and say "The exit is over there" in a loud voice. Follow-me method is such as to speak "Please follow me" to a few near evacuees and evacuate together.

Figure 4 shows the plan of basement that is the environment of the experiment and initial position of evacuees and evacuation leaders.

After leaders and evacuees enter this basement from Exit C, it is closed. Exit A and Exit B are opened when the siren of evacuation beginning stops, but Exit A is a wrong exit. Only leaders know that the right exit is Exit B. Evacuees evacuates according to leaders' indication. Effectiveness of the evacuation methods is evaluated based on the time all members in the basement take to escape from it.

In Sugiman experiment, every participant are given indication on the way of leading if she or he is a leader, or on the way of escaping otherwise in advance. Since humans behave according to restriction produced by indication and them own decision, humans' action mechanism is divided into protocol, which is social restriction, and action rules of their own.

Simulation on FreeWalk[1] where environment as same as Sugiman experiment was reproduces to obtain the same result was performed. Scenario Description Language Q[4] was utilized to control agents. Initial protocol and action rules were obtained describing agents' actions by scenario description process[5] and dividing agents' actions into protocol and action rules(Figure 5, Table 1). Initial protocol were made from materials related to Sugiman experimant and action rules were coordinated to obtain the same result as Sugiman experiment through the simulation[6].

Table 1: Rules for evacuation scenaios[6]

Agent	Rule
Leader	Follow the protocol
	After getting out of room, confirm whether the target evacuee is out of the room
	If the target is out of the room, walk toward the next exit
	If the target is not out of the room, slow down
	If the target gets out of the room, speed up to walk toward the next exit
Evacuee	Follow the protocol
	If the exits open, observe who is around myself
	If the nearest person moves, follow him or her
	After observing the target get out of the room, walk toward the exit
	After passing through the exit, follow the target again

After that, participatory simulation where 12 evacuee agents out of 16 were replaced with avatars which human manipulated and the same initial protocol

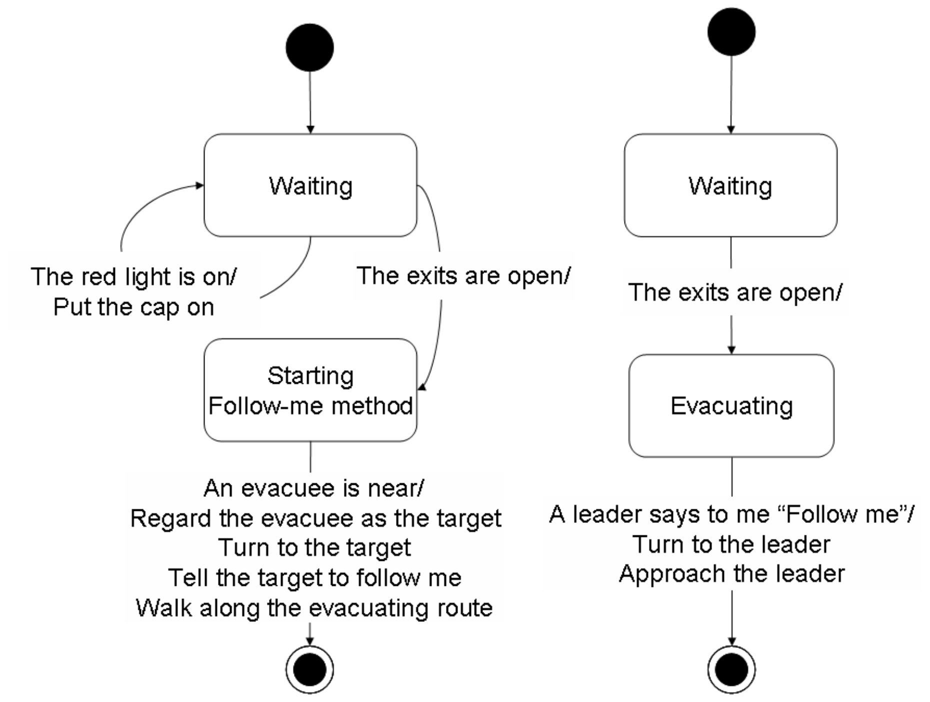

Figure 5: Protocols of Follow-me method cases in Sugiman Experiment[6]; leader's protocol (left) and evacuee's protocol (right)

was given to all participants was performed. The rest of evacuees and leaders were agents.

Of the results of the simulation, a trial with follow-me method differed from that of Sugiman experiment. Figure 6 represents the log data played on 2D simulator FlatWalk. An avatar labeled "9" (the avatar with a circle in the lower part of Figure 6) is looking around in the beginnig, but after many avatars and agents start going toward the right exit, it goes toward the wrong exit against them.

From the difference between this result and the result of the multi-agent simulation, it is found that humans have such action rules that action rules assumed in the multi-agent simulation do not include.

5.2 Applying Action Rule Modeling Process

This chapter applies the action rule modeling process to evacuation simulation in virtual space. Domain knowledge is made from interviews performed with

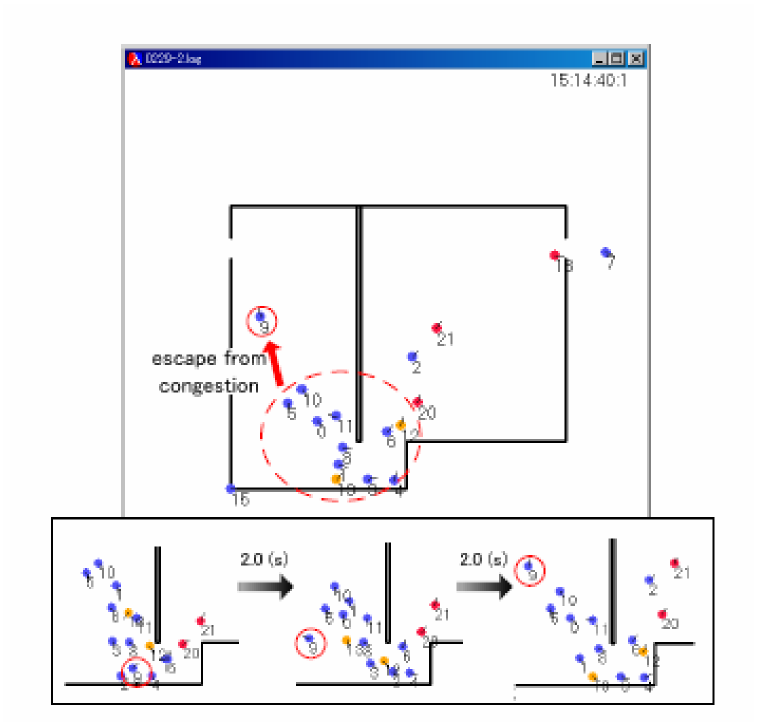

Figure 6: Result of participatory simulation [6] (FlatWalk by Tomoyuki Kawasoe)

participants after the experiment. Log data is sequences of agents' position recorded every 0.1 seconds.

5.2.1 Domain Knowledge Acquisition

Domain knowledge in this domain, which is a set of human action rules on Sugiman experiment on FreeWalk is extracted from interviews with participants. 6 partcipants who manipulated evacuee avatars in the participatory simulation were interviewed. An interviewer shows movie which records the picture the interviewee saw during the simulation and asks the interviewee "what s/he looked at", "what s/he was going to do" and "what s/he worried about" at each 0.1 seconds. The participant who had escaped from the wrong exit was not included in the 6 interviewees.

Action rules extracted from interviews are described using predicated as follows.

"Turn toward a person who is moving"

$$(\{\neg Noop(x), OnLeftOf(x, Self), Noop(Self)\}, \{\neg Noop(Self), TurnToLeft(Self)\})$$
$$(\{\neg Noop(x), OnRightOf(x, Self), Noop(Self)\}, \{\neg Noop(Self), TurnToRight(Self)\})$$

Action rules are described using the form of $a = (a^{precond}, a^{effect})$ according to 3.3. Though variables appear in rules, those are binded to possible agent names.

The rules above are equivalent to the following.

$$\begin{aligned}&(\{\neg Noop(Agent01), OnLeftOf(Agent01, Self), Noop(Self)\},\\ &\qquad\qquad \{\neg Noop(Self), TurnToLeft(Self)\})\\ &(\{\neg Noop(Agent02), OnLeftOf(Agent02, Self), Noop(Self)\},\\ &\qquad\qquad \{\neg Noop(Self), TurnToLeft(Self)\})\end{aligned}$$
$$\cdots$$
$$\begin{aligned}&(\{\neg Noop(Agent01), OnRightOf(Agent01, Self), Noop(Self)\},\\ &\qquad\qquad \{\neg Noop(Self), TurnToRight(Self)\})\\ &(\{\neg Noop(Agent02), OnRightOf(Agent02, Self), Noop(Self)\},\\ &\qquad\qquad \{\neg Noop(Self), TurnToRight(Self)\})\end{aligned}$$
$$\cdots$$

Action rules in the form of "I took such an action because the observation had satisfied the condition" extracted from the interviews are described utilizing predicates(Table 2). The first row shows the obtained rules in natural language, the second shows the preconditions, the third shows the literals which are to be deleted from the avatar's state as the effect of rule firing, and the fourth shows the literals which are to be added to the avatar's state. For a certain rule a, the first row corresponds to the precondition $a^{precond}$, the second to the delete list of the effect $a^{effect,-}$, the third to the add list of the effect $a^{effect,+}$. When the rule is fired, all literals in the delete list are deleted from the avatar's state first, then all literals in the add list are added. When no rule is fired, the avatar's state does not change since all the literals at the previous time remain.

"$Self$" in the table is a special constant: it represents the agent itself which

Table 2: Predicates and constants to write the domain knowledge

Predicate or Constant	Explanation
$Leader(x)$	Leader x is in the avatar's sight
$Evacuee(x)$	Evacuee x is in the avatar's sight
$Exit(x)$	Exit x is in the avatar's sight
$Walk(x)$	x is walking
$Turn(x)$	x is changing its direction
$Noop(x)$	x is not moving
$ToRight(x)$	x's direction is right in view of the avatar
$ToLeft(x)$	x's direction is left in view of the avatar
$Forward(x)$	x's direction is forward in view of the avatar
$Backward(x)$	x's direction is backward in view of the avatar
$OnRightOf(x, y)$	x is on the right of y in view of the avatar
$OnleftOf(x, y)$	x is on the left of y in view of the avatar
$InFrontOf(x, y)$	x is in front of y in view of the avatar
$Behind(x, y)$	x is in the rear of y in view of the avatar
$Near(x, y)$	The position of x is near to y's
$Far(x, y)$	The position of x is far from y's
$ManyPeopleFacingToRight$	Many people are facing to right in view of the avatar
$ManyPeopleFacingToLeft$	Many people are facing to left in view of the avatar
$ManyPeopleFacingBackward$	Many people are facing backward in view of the avatar
$ManyPeopleFacingForward$	Many people are facing forward in view of the avatar
$ManyPeopleWalkingToRight$	Many people are walking rightward in view of the avatar
$ManyPeopleWalkingToleft$	Many people are walking leftward in view of the avatar
$ManyPeopleWalkingForward$	Many people are walking forward in view of the avatar
$ManyPeopleWalkingRearward$	Many people are walking toward the avatar
$ManyPeopleGatherInFront$	Many people gather in front of the avatar
$ManyPeopleGatherInRight$	Many people gather on the right of the avatar
$ManyPeopleGatherInLeft$	Many people gather on the left of the avatar
$NoLeaderInSight$	No leader is in the avatar's sight
$NoExitInSight$	No exit is in the avatar's sight
$WalkForward(x)$	x(Normally, the avatar) is walking forward
$WalkBackward(x)$	x(Normally, the avatar) is walking backward
$TurnToRight(x)$	x(Normally, the avatar) is turning to right
$TurnToLeft(x)$	x(Normally, the avatar) is turning to left
$TurnBackward(x)$	x(Normally, the avatar) is turning backward

has the rule. If the name of the agent which has the rule is "$Agent01$", then $Self = Agent01$

The obtained rules counted up to 18. Since these rules were obtained through interviews, they all are described in natural language. Because one rule changes into many rules when described utilizing predicates, rules described with predicates amounted to 107.

5.2.2 Log Data Processing

Action model is an action rules set which combines local world state the avatar observed an the avatar's own state. Each action rule is fired if observation satisfies the precondition of it to raise the avatar's state change.

At log data processing step, the avatar's observations and actions are reproduced along with time series using log data, and described using predicates and constants, which were necessary to describe action rules of domain knowledge (Table 2).

The form of log data is *S-Expression* (symbolic expression), the detail of which is described below.

Log data

= ((time-ID time position-of-agent0 position-of-agent1 $\cdots$ position-of-agent$n$) $\cdots$)

Position of agent

= (X-coordinate Y-coordinate angle)

$0 \leq angle < 360$. Angle describes the angle consisting of the agent's direction vector and vector (0, 1) counterclockwise.

Field of view World state at the time t can include only the avatar's screen information which is displayed at that time. Thus, it is necessary to the avatar's field of view. It can be calculated using the avatar's position at that time, the angle range of vision and the shape of the basement (Figure 7). In this paper's simulation, the angle range is 60°.

The outside limitation of vision can be calculated from the avatar's position and direction and angle range. In a case where the room cut off the view, it can be obtained from the position of poles (corresponding to terminations of segments in log data) and the avatar's position. Removing the cut off from the

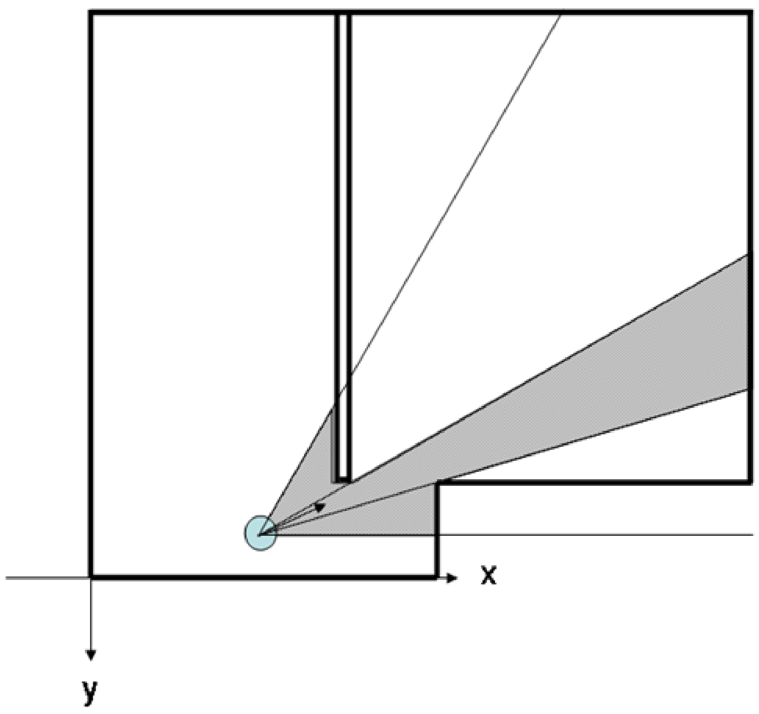

Figure 7: Avatar's view area

limitation of vision generates a polygon which represents the field of view. The details are shown below.

In log data, the room is described as position of poles in clockwise way. First, the intersections of two lines of vision limitation and the lines of room are calculated. The polygon consisting of the two intersection and the internal vertices (position of pole) is defined as "temporary vision". Drawing segments from the avatar's position to the vertices in the temporary vision, if a segment is extensible toward the vertex, that is, the termination point of segment slightly extended toward the vertex is also in the room, the segment is extended to calculate the new intersection. Since area surrounded by intersection point before extending, extended segment, and the room is out of vision, new temporary vision is calculated by removing this area from the temporary vision (Figure 8).

Whether an agent is in or out of vision is decided by the field of vision and the agent's position. Drawing a long line from the agent's position, if the number of intersections of the line and the segment of vision polygon is odd, the agent is inside. Otherwise, outside. In a case where intersection is generated

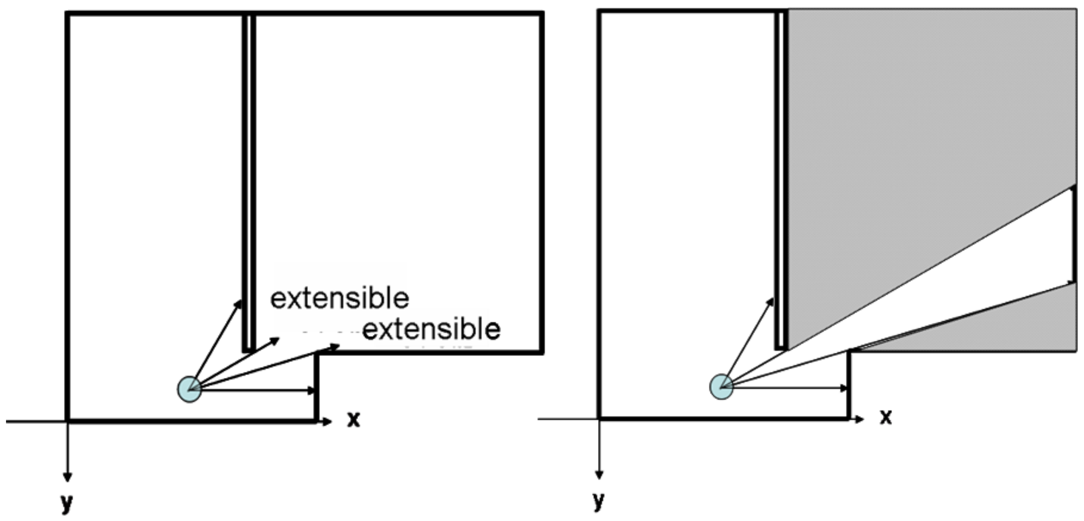

Figure 8: Making avatar's view area

on a vertex of vision polygon, if the neighboring vertices exist in the same side in terms of the long line, the intersection is considered to be two intersections (Figure 9).

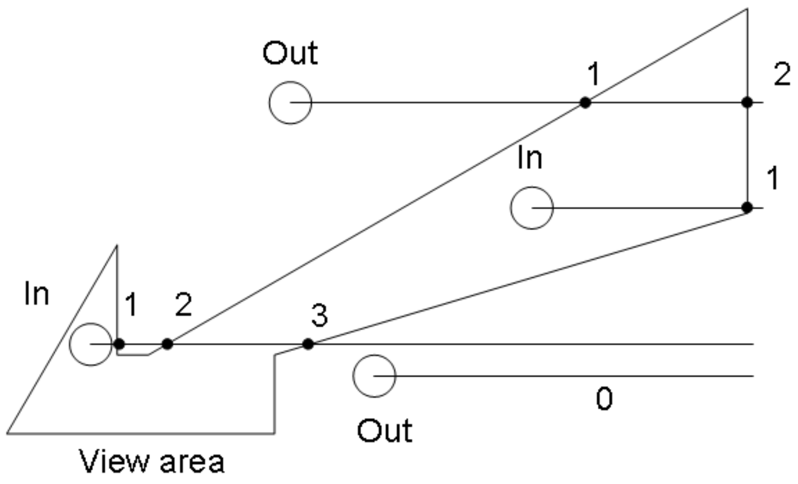

Figure 9: Deciding inside or outside

World state In the simulation, every participant is given role in advance; a leader with a white cap $Leader(x)$, a evacuee $Evacuee(x)$. If a leader A is in sight, $Leader(A)$ becomes true. Symbol $Exit(x)$ represents that a exit x is in sight, which can be obtained from the shape of the room.

To calculate an agent's sequence of behavior, position log data is utilized. If the position changes, the agent is walking ($Walk(x)$) at that time, if the position does not change and the direction changes, the agent is turning ($Turn(x)$). Neither the position nor the direction changes, the agent is regarded to do nothing ($Noop(x)$)

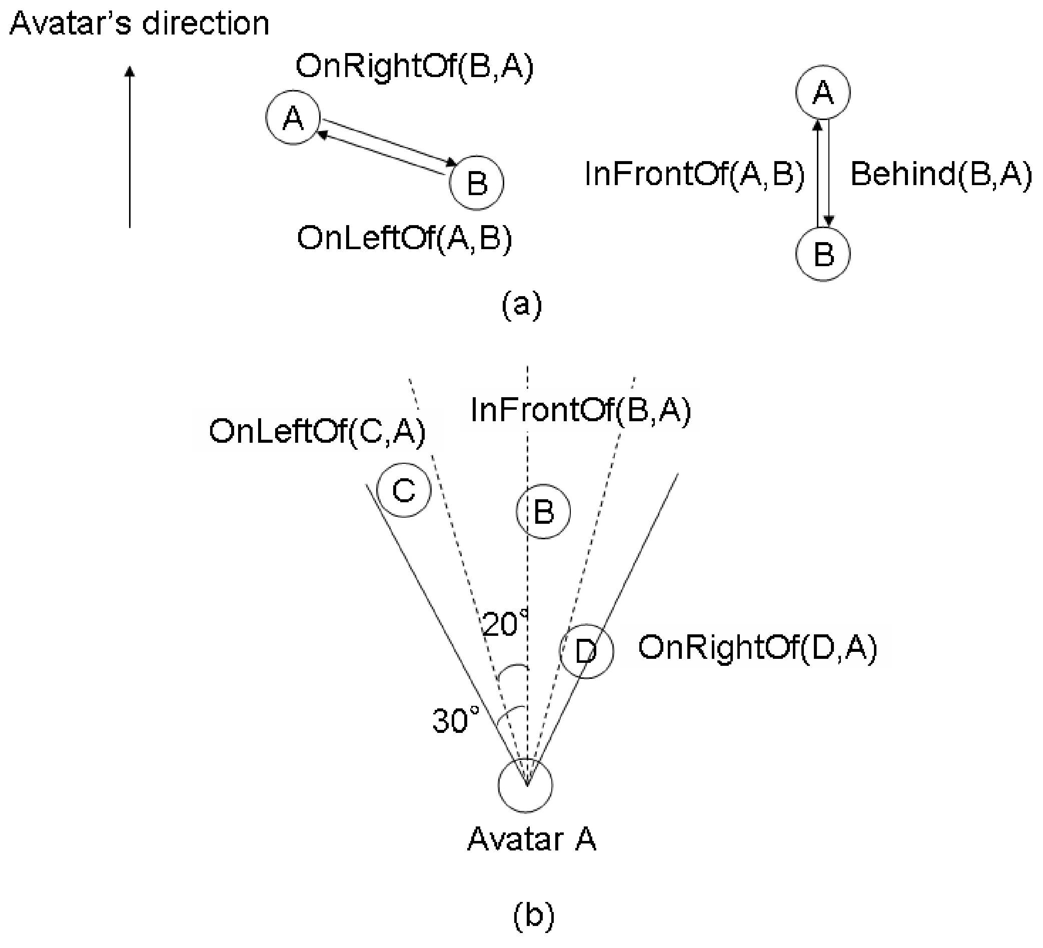

Figure 10: Position relationship; among agents(a), between the avatar and an agent(b)

Position relationship represents how positional relation between an agent x and an agent y is in the viewpoint of the avatar; x is in front of y ($InFrontOf(x, y)$), behind y ($Behind(x, y)$), on the left of y ($OnLeftOf(x, y)$), or on the right of y ($OnRightOf(x, y)$). These relations can be acquired from the avatar's direction and agents' position. Positional relation between agents can be obtained by the angle consisting of the avatar's direction vector and the vector whose starting point is one agent's position and terminating point is the other agent's position (Figure 10(a)). On the other hand, calculating positional relation between an agent and the avatar must consider not only position but also the angle range of vision because it is determined according to the position on screen. Since the angle range is 60°, the avatar's field of vision is divided into three parts; center area consists of 20° right area to the center line and 20° left area to the center line, right area is the right side of center area, and left area is the left side. Positional relation is determined according to which area the agent exists in (Figure 10(b)).

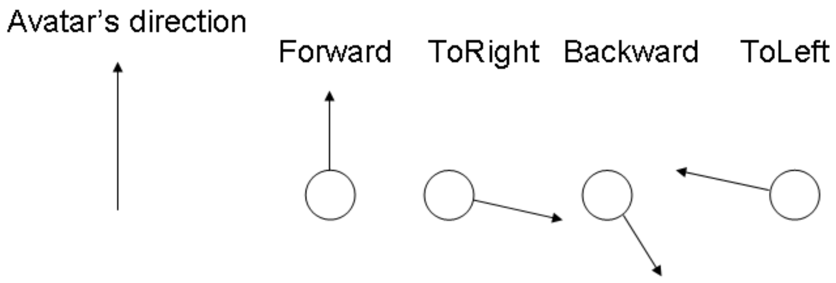

Figure 11: Direction of agents

Direction represents the direction of the target in terms of the avatar; the target's direction is forward ($Forward(x)$), rearward ($Backward(x)$), leftward ($ToLeft(x)$) or rightward ($ToRight(x)$). This can be calculated from the avatar's direction and agents' direction (Figure 11).

Distance between agents can be obtained the agents' position; if distance between two agents is under threshold, the two agents are near ($Near(x, y)$), the two are far ($Far(x, y)$) otherwise.

An example world state (Figure 12) is described using predicates mentioned above. The owner of this vision is supposed to be Agent D and relationship to be described is restricted to that of Agent A, Agent B, agent C and Agent D.

The following values are calculated from position log data (Figure 13)¿

$$\begin{aligned} \text{The distance between D and A} &= 1.65 \\ \text{The distance between D and B} &= 3.90 \end{aligned}$$

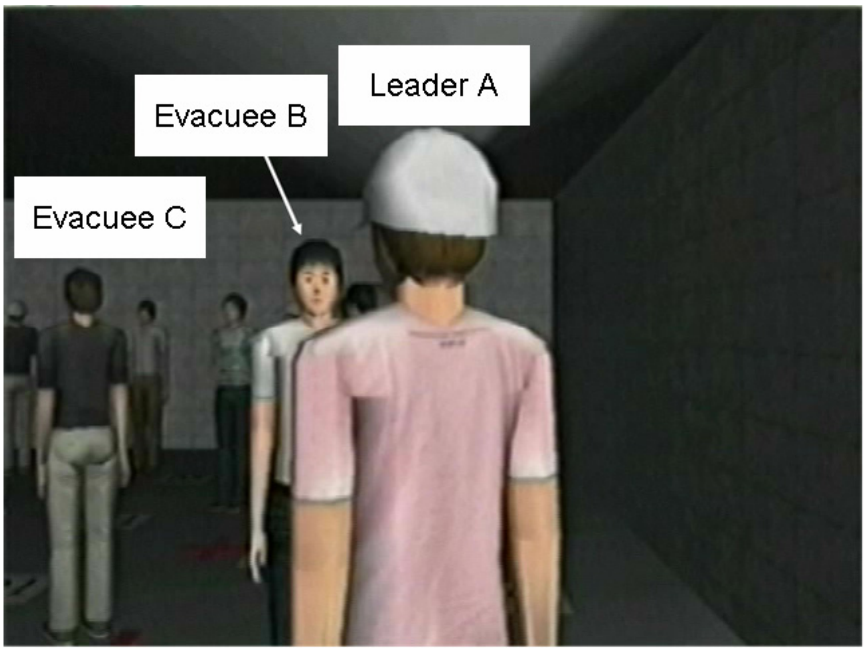

Figure 12: Local world state for Agent D

The world state 0.1 seconds before

Position of Avatar D:$(1.5, -9.9, 0.0)$

Position of Agent A:$(1.5, -8.25, 10.0)$

Position of Agent B:$(1.55, -6.0, 180.0)$

Position of Agent C:$(2.98, -5.88, 40.0)$

$\Downarrow$

The current world state

Position of Avatar D:$(1.5, -9.9, 0.0)$

Position of Agent A:$(1.5, -8.25, 10.0)$

Position of Agent B:$(1.55, -6.0, 180.0)$

Position of Agent C:$(3.0, -5.85, 40.0)$

Figure 13: Log data of Figure 12

$$\begin{aligned}
\text{The distance between D and C} &= 4.32\\
\text{The distance between A and B} &= 2.25\\
\text{The distance between A and C} &= 2.83\\
\text{The distance between B and C} &= 1.46
\end{aligned}$$

Let threshold for distance be 2.5, the following literals are true.

$$\begin{aligned}
&\{Near(A,D), Near(D,A), Far(B,D), Far(D,B),\\
&Far(C,D), Far(D,C), Near(A,B), Near(B,A),\\
&Near(C,B), Near(B,C), Far(A,C), Far(C,A)\}
\end{aligned}$$

The following values on action are calculated from the current position and the position 0.1 seconds before.

$$\begin{aligned}
\text{The difference of A's position} &= (0.0, 0.0, 0.0)\\
\text{The difference of B's position} &= (0.0, 0.0, 0.0)\\
\text{The difference of C's position} &= (0.02, 0.03, 0.0)
\end{aligned}$$

Thus, the fact that A and B are stopping and C is walking is acquired.

$$\{Noop(A), Noop(B), Walk(C)\}$$

Let AB be the vector whose starting point is the position of A and whose terminating point is the position of B, The following values on positional relationship are obtained.

$$\begin{aligned}
DA &= (0, 1.65)\\
DB &= (0.05, 3.9)\\
DC &= (1.5, 4.05)\\
AB &= (0.05, 2.25)\\
AC &= (1.5, 2.4)\\
BC &= (1.45, 0.15))
\end{aligned}$$

The following literals is true since Agent D's angle is 0.0.

$$\{InFrontOf(A,D), Behind(D,A), InFrontOf(B,D), Behind(D,B),$$
$$OnLeftOf(C,D), OnRightOf(D,C), InFrontOf(B,A), Behind(A,B),$$
$$OnLeftOf(C,B), OnRightOf(B,C), Behind(A,C), InFrontOf(C,A)\}$$

D's angle and angle of each agent provides direction.

$$Forward(A), Backward(B), Forward(C)$$

$NoExitInSight$ is true since no exit is in D's sight.

From the above, the local world state for Agent D is described as follows.

$$\begin{aligned} S = \; & \{Leader(A), Near(A,D), Near(D,A), Forward(A), Noop(A), \\ & Evacuee(B), Far(B,D), Far(D,B), Backward(B), Noop(B), \\ & Evacuee(C), Far(C,D), Far(D,C), Forward(C), Walk(C), \\ & InFrontOf(A,D), Behind(D,A), InFrontOf(B,D), Behind(D,B), \\ & OnLeftOf(C,D), OnRightOf(D,C), InFrontOf(B,A), Behind(A,B), \\ & Near(A,B), Near(B,A), OnLeftOf(C,B), OnRightOf(B,C), \\ & Near(C,B), Near(B,C), Behind(A,C), InFrontOf(C,A), \\ & Far(A,C), Far(C,A), NoExitInSight\} \end{aligned}$$

Shown in Table 2, constants such as "no exit is in sight ($NoExitInSight$)" are also available.

Avatar state The modeling process explains the avatar's state change by rule matching with world state. Therefore, the avatar's state must be described separately. Avatar's state is restricted within the avatar's behavioral state for action rules in domain knowledge, such as "walking", "turning" and so on.

For output of the process in 3.5, direction of action is also important; literals for representing walking forward and backward ($WalkForward(x)$, $WalkBackward(x)$), turning to the left and the right ($TurnToLeft(x)$, $TurnToRight(x)$) and no action ($Noop(x)$) are prepared. Which state the avatar belongs to is decided the difference between the avatar's position at the time t and at the previous time

$t-1$. Turning rearward ($TurnBackward(x)$) is defined to be true if and only if $TurnToLeft(x)$ or $TurnToRight(x)$ is true and consequently the avatar's direction is changed in more degrees than certain threshold.

5.2.3 Action Rule Modeling

Obtained log data includes position log data of the avatar that an interviewed participant manipulated. This study applied action rule modeling process to the avatar of which action rules are already known through interview.

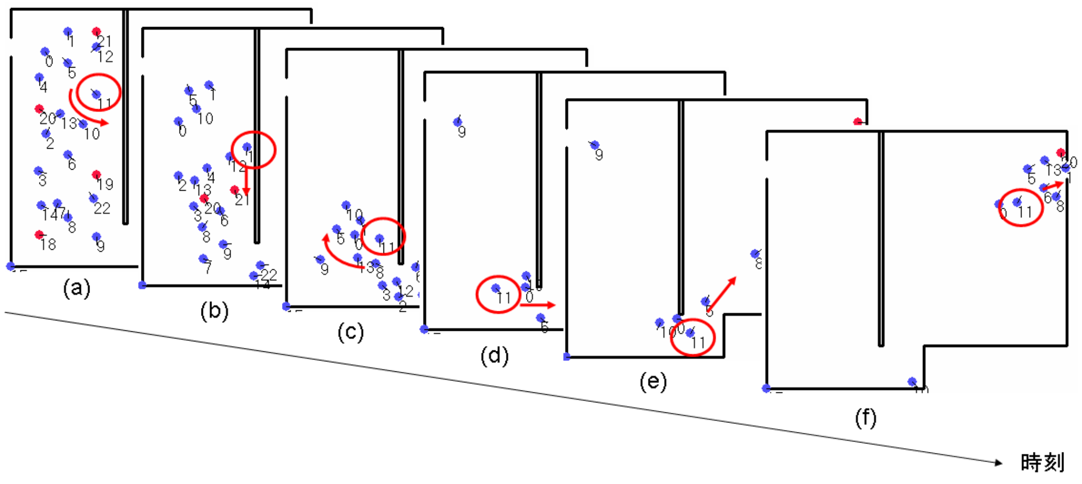

Figure 14: Tracing positions and actions of Avatar 11

Avatar11 which the interviewed participant manipulated evacuated through the correct exit B according to route the leaders took (Figure 14).

Sequence of avatar11's states is shown in Table 5.2.3. The fifth row means the avatar's state S_t^{avatar}. The first row and the second row indicate starting and ending time t, respectively, while S_t^{avatar} does not change. Thus the first column means

$$S_0^{avatar} = S_1^{avatar} = \ldots = S_6^{avatar} = S_7^{avatar} = \{(NoopSelf)\}$$

Time t is discretized according to change of local world state which the avatar observes as shown in 3.1. Therefore the variable t does not correspond to real world time. The third row and the fourth row display starting time and ending time in the real world, respectively, while S_t^{avatar} does not change.

Table 3: State changes of Avatar 11

Start t	End t	Start time	End time	S_t^{avatar}
0	7	0	72	$\{Noop(Avatar11)\}$
8	8	73	73	$\{WalkBackward(Avatar11)\}$
9	12	74	85	$\{TurnBackward(Avatar11)\}$
13	13	86	93	$\{Noop(Avatar11)\}$
14	20	94	100	$\{TurnToRight(Avatar11)\}$
21	23	101	105	$\{Noop(Avatar11)\}$
24	29	106	111	$\{TurnToRight(Avatar11)\}$
30	34	112	117	$\{Noop(Avatar11)\}$
35	38	118	121	$\{TurnToLeft(Avatar11)\}$
39	42	122	126	$\{Noop(Avatar11)\}$
43	45	127	133	$\{WalkForward(Avatar11)\}$
46	47	134	136	$\{Noop(Avatar11)\}$
48	50	137	139	$\{TurnToRight(Avatar11)\}$
51	53	140	142	$\{Noop(Avatar11)\}$
54	58	143	151	$\{WalkForward(Avatar11)\}$
59	64	152	157	$\{Noop(Avatar11)\}$
65	67	158	160	$\{TurnToLeft(Avatar11)\}$
68	81	161	177	$\{Noop(Avatar11)\}$
82	86	178	184	$\{WalkForward(Avatar11)\}$
87	103	185	205	$\{Noop(Avatar11)\}$
104	105	206	208	$\{TurnToRight(Avatar11)\}$
106	106	209	210	$\{Noop(Avatar11)\}$
107	112	211	216	$\{WalkForward(Avatar11)\}$
113	113	217	217	$\{TurnToLeft(Avatar11)\}$
114	120	218	224	$\{Noop(Avatar11)\}$
121	123	225	227	$\{TurnToLeft(Avatar11)\}$
124	152	228	259	$\{Noop(Avatar11)\}$
153	158	260	266	$\{TurnToRight(Avatar11)\}$
159	160	267	269	$\{Noop(Avatar11)\}$
161	165	270	275	$\{TurnToRight(Avatar11)\}$
166	167	276	278	$\{Noop(Avatar11)\}$
168	172	279	283	$\{TurnToRight(Avatar11)\}$
173	180	284	296	$\{Noop(Avatar11)\}$
181	194	297	315	$\{TurnToLeft(Avatar11)\}$
195	196	316	320	$\{Noop(Avatar11)\}$
197	200	321	325	$\{TurnToRight(Avatar11)\}$
201	204	326	333	$\{WalkForward(Avatar11)\}$
205	205	334	334	$\{Noop(Avatar11)\}$

Start t	End t	Start time	End time	S_t^{avatar}
206	208	335	347	$\{TurnToLeft(Avatar11)\}$
209	210	348	350	$\{Noop(Avatar11)\}$
211	213	351	361	$\{WalkForward(Avatar11)\}$
214	218	362	369	$\{TurnToLeft(Avatar11)\}$
219	220	370	373	$\{Noop(Avatar11)\}$
221	224	374	379	$\{TurnToRight(Avatar11)\}$
225	225	380	380	$\{Noop(Avatar11)\}$
226	228	381	386	$\{WalkForward(Avatar11)\}$
229	229	387	387	$\{Noop(Avatar11)\}$
230	234	388	394	$\{TurnToLeft(Avatar11)\}$
235	236	395	396	$\{Noop(Avatar11)\}$
237	241	397	402	$\{WalkForward(Avatar11)\}$
242	257	403	418	$\{Noop(Avatar11)\}$
258	260	419	421	$\{TurnToRight(Avatar11)\}$
261	269	422	431	$\{WalkForward(Avatar11)\}$
270	275	432	437	$\{TurnToLeft(Avatar11)\}$
276	276	438	440	$\{Noop(Avatar11)\}$
277	299	441	467	$\{WalkForward(Avatar11)\}$
300	301	468	469	$\{Noop(Avatar11)\}$
302	304	470	472	$\{WalkForward(Avatar11)\}$
305	306	473	475	$\{Noop(Avatar11)\}$
307	307	476	477	$\{TurnToRight(Avatar11)\}$
308	314	478	484	$\{WalkForward(Avatar11)\}$
315	318	485	488	$\{TurnToRight(Avatar11)\}$
319	327	489	498	$\{WalkForward(Avatar11)\}$
328	332	499	503	$\{TurnToLeft(Avatar11)\}$
333	333	504	504	$\{Noop(Avatar11)\}$
334	341	505	513	$\{WalkForward(Avatar11)\}$
342	342	514	514	$\{Noop(Avatar11)\}$
343	345	515	517	$\{TurnToRight(Avatar11)\}$
346	348	518	520	$\{Noop(Avatar11)\}$
349	350	521	522	$\{TurnToRight(Avatar11)\}$
351	362	523	538	$\{Noop(Avatar11)\}$
363	364	539	543	$\{WalkForward(Avatar11)\}$
365	365	544	544	$\{Noop(Avatar11)\}$
366	366	545	547	$\{TurnToRight(Avatar11)\}$
367	367	548	548	$\{Noop(Avatar11)\}$
368	372	549	553	$\{WalkForward(Avatar11)\}$
373	373	554	556	$\{TurnToRight(Avatar11)\}$
374	374	557	558	$\{Noop(Avatar11)\}$

Start t	End t	Start time	End time	S_t^{avatar}
375	375	559	566	$\{TurnToRight(Avatar11)\}$
376	376	567	569	$\{Noop(Avatar11)\}$
377	378	570	572	$\{WalkForward(Avatar11)\}$

From interview and movie that captures the participant's screen, broadly speaking, avatar11 is considered to have acted as follows. Immediately after the experiment had started, since she could not find leader with a white hat, she turned to the back to search leaders. The initial position of avatar11 is Figure 14(a) and she was faced to exit A, only narrow field of view can be obtained. But after turning to the back, she could capture the flow. Having seen many people including leaders go toward the right space (the right room of Figure 4), she followed them. Because many people had passed exit B, she confirmed that exit B is the right exit and evacuated through exit B.

From interview to the participant who used avatar11 and movie that captures the participant's screen, the participant proved to have the following action rules in domain knowledge.

- If no leader can be seen, turn to the left or the right or the back to search for leaders
- Go toward the direction many people are faced to
- Go toward the direction many people are walking to
- Go toward the place where many people gather
- Go toward the direction a leader is walking to
- Follow the agent or avatar in front
- If an exit is in sight, go toward it

ActionRuleSet could not obtain the answer from avatar11's log data.

One of the reasons is that N, which is the set of rules which is not included in the answer rule set, included necessary rules to explain the avatar's behavior. If P_t consists of only one rule, the rule must be included in A_{avatar}. In the log data, $|P_t| = 1$ appeared 6 times, while 5 rules of the 6 are included in N. $|N| = 63$ is big size considering $|A| = 107$ where A is the domain knowledge.

Another reason is that rule priority may change according to circumstances. Executing GetModel with $N = \emptyset$ cannot get the answer either. Since $P_t \neq$

at all times when rule firing should occur, there are some rules to explain the avatar's state change. But, to fire the rule, the order required contradicts that previous constructed.

Then, focusing only on action rule set, $\{P_t\}$ in the action rule modeling process is compared with the rules that were obtained through interview.

A_{avatar}, the answer of the modeling process, is calculated by ActionRuleSet where rule set $\{P_t\}$ that explains the avatar's state change correctly is obtained and GetModel where rules are selected from P_t at each time t to construct an action rule set. Therefore, A_{avatar} varies according to selected rules from $\{P_t\}$. Table 4 shows rule sets which are minimal explanations and recall and precision against the rule set obtained through interview. Actually, all rules are described with predicated. A rule with "$\sharp$" is a rule in the rule set obtained through interview.

Let $A_{interview}$ be the set of rules obtained through interview, Since A_{avatar} is necessary action rule set to explain the log data, $Recall = |A_{interview} \cap A_{avatar}|/|A_{interview}|$ means how many rules which actually the participant have were obtained, and $Precision = |A_{interview} \cap A_{avatar}|/|A_{avatar}|$ means how much the rules obtained through interview can explain the log data.

Table 4: Minimal explanation on Avatar 11's behavior

Action rule set	Recall	Precision
♯ If no leader can be seen, turn to the left or the right or the back to search for leaders If no exit can be seen, turn to the left or the right or the back to search for exit ♯ Follow the agent or avatar in front ♯ If an exit is in sight, go toward it ♯ Go toward the direction a leader is walking to If there is congestion in front, try to pass through	57%	67%
♯ If no leader can be seen, turn to the left or the right or the back to search for leaders If no exit can be seen, turn to the left or the right or the back to search for exit ♯ Follow the agent or avatar in front ♯ If an exit is in sight, go toward it ♯ Go toward the direction a leader is walking to If there is an agent or an avatar who is going the same way in front and moves slowly, try to go ahead of him or her	57%	67%
♯ If no leader can be seen, turn to the left or the right or the back to search for leaders If no exit can be seen, turn to the left or the right or the back to search for exit ♯ Follow the agent or avatar in front ♯ If an exit is in sight, go toward it Before passing through exit, turn to the left or the right If there is congestion in front, try to pass through	43%	50%
♯ If no leader can be seen, turn to the left or the right or the back to search for leaders If no exit can be seen, turn to the left or the right or the back to search for exit ♯ Follow the agent or avatar in front ♯ If an exit is in sight, go toward it Before passing through exit, turn to the left or the right If there is an agent or an avatar who is going the same way in front and moves slowly, try to go ahead of him or her	43%	50%

Chapter 6 Discussion

Action rule modeling process can obtain action rules which the target has from log data of participatory simulation due to utilizing domain knowledge. However, various improvements and verifications on assumptions are needed.

In a case where domain knowledge is obtained enough to explain avatar's behavior, the reason why the action modeling process can not get answers is that rules necessary to explain avatar's behavior are included in N, which is a set of rules forbidden to be fired or that rule firing priority is not unchangeable but it may change under some circumstances. To overcome the former problem, it is necessary to introduce a mechanism where it can happen that rule firing does not occur even if there exist instantiations available. To overcome the latter, it is necessary to change agent architecture so that rule firing priority can change according to the avatar's state.

The action modeling process assumes that humans manipulating avatars can observe all events in the field of view and react to observation instantly as well as agents. However, humans have limitation of reaction time, and cannot understand completely the whole area of view. This problem is difficult to overcome, but it is possible to calculate reaction time in the domain and recognizable events in the view from a psychological and sociological standpoint.

This study restricts agents' and avatars' states to them actions. However, the result of interviews includs such rules as "Since I had seen an exit on the right a little while before the time (and the exit could not been seen at the time), I turned to the right". To include such rules into domain knowledge, description of memory and recognition must be included in states. $TurnBackward$, which is an action to turn backward, ends and an agent's action becomes $Noop$ when the agent finishes turning to the back. It is necessary to extend state description to represent finish of actions as action rules. However, it is difficult to construct state description related to memory and recognition from postion log data.

ActionRuleSet cannot detect operation mistakes and meaningless actions. Therefore, it happens that the process makes an over-fitting action rule sets or cannot find such action rules in domain knowledge as to explain the agent's actions since the process tries to explain all of the actions. Inductive learning can obtain only important action rules if a large data set of training data is obtained.

Chapter 7 Related Work

Application of machine learning to human action modeling is in strong demand in both academic and business domain. Applying machine learning to user modeling is called ML4UM (Machine Learning for User Modeling) or MLIRM (Machine Learning, Information Retrieval, User Modeling), and several researches for application are now going on.

In academic domain, results of the researches are utilized for digital human, architecture, psychology, disaster prevention, to confirm the reliability of human models which experts constructed with real data, or to modify and refine the models.

In business domain, there exists recommendation system of Amazon.com, which utilizes collaborative filtering[7]. This system assumes that every user of the shopping site has the same model to recommend goods to a user which the user may buy with high possibility based on purchasing history of the user. The system adopts inductive learning method to construct user model which takes purchasing history data for the training data.

User modeling is classified as follows according to its purpose[8].

1. Describing the cognitive process that underlie the user's action
2. Describing the difference between the user's skills and expert skills
3. Describing the user's behavioral patterns or preferences
4. Describing the user's characteristics

In recent years, there are a lot of researches to aim for 3, and so is this research, because this learns action models to simulate human's behavior in several environment in multi-agent simulation.

Another important viewpoint to classify user modeling is whether they model individual user or communities of users. Recommendation system of Amazon.com is a famous example of the latter. It regards the users of the site as one community for modeling. In this standpoint, large set of training data is not difficult to obtain and thus inductive learning method can be utilized.

On the other side, this study belongs to the former because it models individual avatar. Since the target is an individual, it is unrealistic to obtain large training data set.

One of researches to overcome lack of training data is knowledge-based learning approach. [9] models students by capture student error with domain knowledge. it is also one of the characteristics that theory refinement is used to refine knowledge base not to contradict with training data.

This research differs from previous researches in that domain knowledge means appropriate human action rules in the domain, which are not always right for everyone, does not mean knowledge which is always true in the domain.

Chapter 8 Conclusion

This paper proposed an action modeling method which utilizes participatory simulation in virtual space. This method is superior to that of real world experiment in the number of participants and in that accurate log data can be obtained, in addition to that hard-to-reconstruct environments can be constructed.

Contributions of this action rule modeling process are as follows.

- Already-known action rules in the domain are acquired as domain knowledge. An action model can be obtained to extract such rules as to explain the avatar's action from the domain knowledge even if the volume of the log data is small.
- Various models can be obtained to acquire one model from one agent log data.

In this paper, the above-mentioned action modeling process is applied to an evacuation experiment on virtual space. Domain knowledge is extracted from the interview to some of the participants having participated in the experiment as evacuees. As a result, the modeling process was validated to apply the process to an avatar interviewed to compare the action rule sets that had been obtained from the domain knowledge and that from the interview to her.

The action model of humans who cannot be lead in safety with high possibility with the present leading methods was obtained applying the process to an avatar who had evacuated through a wrong exit. The obtained model can be utilized for establishing a new leading method through multi-agent simulations.

Future works are, for example, investigation of the assumptions on the action model, classification of human's action pattern in addition. In this paper, agents are assumed to behave reactively to environment and not to have memory on world state or their own state. Thus, in domains where memory is made much of, the action rule modeling process may not be able to acquire an action rule set.

An action rule set obtained through the modeling process can explain log data related to the target avatar, but if much noise is in the log data, the obtained rule set may be over-fitting. To overcome this problem, a lot of action rule sets which are classified as the same action pattern is needed to obtain a more general action rule set by inductive learning.

Acknowledgments

I would like to express my sincere gratitude to Professor Toru Ishida at Graduate School of Informatics, Kyoto University, for invaluable advice and discussion.

I am also grateful to my research adviser, Professor Hajime Kita at Academic Center for Computing and Media Studies, Kyoto University, and Mr. Tomohiro Kuroda at Department of Medical Informatics, Kyoto University Hospital.

I would like to express special thanks to Mr. Hideyuki Nakanishi at Ishida Laboratory for providing me with the interview movies of the Sugiman Experiment on FreeWalk. I would also like to thank Assistant Professor Sachiyo Arai for worthy advice.

I would like to extend many thanks to all the members of Ishida Laboratory at Kyoto University for their help.

This research was done as a part of the project ”Universal Design of Digital City” of CREST, JST.

References

[1] Nakanishi, H.: FreeWalk: A Social Interaction Platform for Group Behavior in a Virtual Space, *International Journal of Human Computer Studies (IJHCS)*, Vol. 60, No. 4, pp. 421–454 (2004).

[2] Poole, D.: Who Choose the Assumptions?, *Abduction* (O'Rorke, P.(ed.)), AAAI/MIT Press (1997).

[3] Sugiman, T. and Misumi, J.: Development of a New Evacuation Method for Emergencies: Control of Collective Behavior by Emergent Small Groups, *Journal of Applied Psychology*, Vol. 73, No. 1, pp. 3–10 (1988).

[4] Ishida, T.: *Q*: A Scenario Description Language for Interactive Agents, *IEEE Computer*, Vol. 35, pp. 54–59 (2002).

[5] Murakami, Y., Ishida, T., Kawasoe, T. and Hishiyama, R.: Scenario Description for Multi-Agent Simulation, *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*, pp. 369–376 (2003).

[6] Murakami, Y. and Ishida, T.: 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の設計, *Third Joint Agent Workshops & Symposium (JAWS 2004)*, p. 04006 (2004).

[7] Linden, G., Smith, B. and York, J.: Amazon.com recommendations: Item-to-item collaborative Filtering, *IEEE Internet Computing*, Vol. 7, No. 1, pp. 76–80 (2002).

[8] Webb, G. I., Pazzani, M. J. and Billsus, D.: Machine Learning for User Modeling, *User Modeling and User-Adapted Interaction*, Vol. 11, pp. 19–29 (2001).

[9] Baffes, P. and Mooney, R.: Refinement-based Student Modeling and Automated Bug Library Construction, *Journal of Artificial Intelligence in Education*, Vol. 7, pp. 75–116 (1996).

Appendix

A.1　研究経過報告書