A Defeasible Reasoning System in Multi-Agent Environments

Chiaki Sakama
Dept. Computer and Communication Sci.
Wakayama University
Sakaedani, Wakayama 640 8510, Japan
sakama@sys.wakayama-u.ac.jp

Katsumi Inoue
Dept. Electrical and Electronics Eng.
Kobe University
Rokkodai, Nada, Kobe 657 8501, Japan
inoue@eedept.kobe-u.ac.jp

Koji Iwanuma
Dept. Computer Science and Media Eng.
Yamanashi University
Takeda, Kofu 400 8511, Japan
iwanuma@iw.media.yamanashi.ac.jp

Ken Satoh
Div. Electronics and Information Eng.
Hokkaido University
N13W8 Sapporo 060 8628, Japan
ksatoh@db-ei.eng.hokudai.ac.jp

Abstract

We introduce a multi-agent system based on logic programming. In this system an agent has a knowledge base written in an extended logic program. Then, an agent performs default reasoning in the situation that her belief is possibly rebutted by other agents. We present a logical framework of such systems and also characterize the system in situation calculus.

1 Introduction

Automated reasoning in a multi-agent environment is an important topic to construct cooperative intelligent systems. In a multi-agent environment, it is usually assumed that each agent has incomplete knowledge about the world. Then each agent performs reasoning by collecting necessary information through communication with other agents. However, communication between agents does not always end in success in real situations. It may happen that some agent takes much time in responding to a query because she is absent when the query is asked or there is an unexpected computational problem (e.g. falling into an infinite loop). Also, due to the unsecure communication environment, messages between agents might be lost or significantly delayed. In these circumstances, an intelligent agent may perform default reasoning using her own belief, rather than suspending a process and waiting for the arrival of missing information. On the other hand, such default assumption might be rebutted afterward by certain information from other agents.

Modelling the situation, Satoh *et al* [10] study a multi-agent system under incomplete communication environment. When communication is delayed or failed, an agent uses her default hypothesis as a tentative belief and continues computation. If contrary information to her belief is brought by another agent, she withdraws her belief and re-computes an alternative solution. They call such processes *speculative computation by default assumption* and realize it using *abduction*. This paper is a continuous study of the previous work and is intended to provide a logical framework of defeasible reasoning in multi-agent systems. We consider a multi-agent system in which each agent has a knowledge base written in an *extended logic program*. Then, the belief of each agent is characterized by *belief sets*, which dynamically change according to the information from other agents. Defeasible reasoning by an agent is characterized by an extended logic program using situation calculus, and belief sets of an agent are expressed by answer sets of the program.

The rest of this paper is organized as follows. Section 2 provides a logical framework of multi-agent systems. Section 3 characterizes the system in situation calculus. Section 4 presents related work and Section 5 summarizes the paper.

2 A Multi-Agent System

The language of our multi-agent system has an alphabet which is standard in logic programming. Each agent has a logic program which has the common syntax and semantics between agents, while an agent can have terms and predicates which are defined locally in her program. Let $Pred$ be the set of all predicates in the language of the system. Then, $Ext(\subseteq Pred)$ is the set of *external predicates* which are the common predicates shared by every agent.

The language has reserved terms called *identifiers*, which are symbols to distinguish each agent in a system.

A linearly ordered set T of constants in the language is defined such that t_0 is the smallest element in T and every element $t_i \in T$ has a successor $t_{i+1} \in T$. We write $t_i < t_j$ if $i < j$. Each element in T is called *timing*.

Definition 2.1 An *agent* A_t is defined as a triple $A_t = \langle\, \Pi, \Delta, \Gamma_t \,\rangle$ where

- Π is a *knowledge base* of the agent. It is given as an *extended logic program* (ELP) which is a set of rules of the form:

  $$L_0 \leftarrow L_1, \ldots, L_m, \mathit{not}\, L_{m+1}, \ldots, \mathit{not}\, L_n$$

  where each L_i $(0 \leq i \leq n)$ is a literal and *not* presents *negation as failure*. L_0 is the *head* and $L_1, \ldots, L_m, \mathit{not}\, L_{m+1}, \ldots, \mathit{not}\, L_n$ is the *body* of the rule. In the program, external predicates appear only in the bodies of rules. A rule with the empty body $L \leftarrow$ is also called a *fact* and is identified with the literal L. A rule with the empty head is also called an *integrity constraint*.

- Δ is a set of literals called *assumptions* such that $\Pi \cup \Delta$ is consistent, and every literal in Δ has an external predicate which has an identifier as an argument. When a literal contains variables, it is identified with its ground instances.

- Γ_t is a *response set at timing* $t \in T$, which satisfies the following conditions:

  1. $\Gamma_{t_0} = \emptyset$,
  2. $\Gamma_{t_i} \subseteq \Delta \cup \neg\Delta$ for any $t_i \in T$, where $\neg\Delta = \{\, \neg l \mid l \in \Delta \,\}$.
  3. If $l \in \Gamma_{t_i}$ and $\neg l \notin \Gamma_{t_{i+1}}$, then $l \in \Gamma_{t_{i+1}}$.

A *multi-agent system* is a finite set of agents.

An agent A_t has a parameter t, while in what follows we write A_t as A if the timing is not important in the context. An agent has a knowledge base Π which represents her knowledge. On the other hand, an agent has a set Δ of beliefs on other agents. When a literal $l \in \Delta$ contains an identifier B, an agent A has a belief l on the agent B. If the agent A wants to confirm the fact l in her reasoning, she asks a question to the agent B by communication.[1] The agent B replies either l or $\neg l$ if he can manage the question (Fig.1). The response set stores information returned by other agents, which is given as a subset of $\Delta \cup \neg\Delta$ at each timing t. The response set grows according to the increase of timing, and an old response is replaced by a new response when

[1] In this paper, we do not concern with a method of communication in detail.

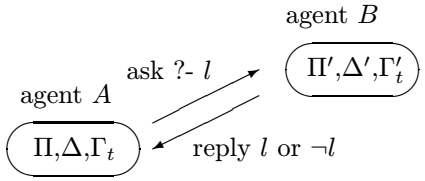

Figure 1: Communication between agents

they contradict each other. By the definition, the union of one's knowledge base Π and her beliefs Δ are consistent. This assumption is natural for a rational agent, otherwise she has inconsistent beliefs as a whole. However, her beliefs are possibly rebutted by information from other agents. In this sense, reasoning by an agent is *defeasible* and her beliefs *nonmonotonically* change at different timing. Dynamically changing state of beliefs is formally characterized in the next.

The semantics of our multi-agent system is based on the *answer set semantics* of extended logic programs [5]. We first review the definition of answer sets.

Given a *not*-free ELP (i.e., for each rule $m = n$) P and a set S of literals, S is a *consistent answer set* of P if S is a minimal set satisfying the conditions:

1. For each ground rule $L_0 \leftarrow L_1, \ldots, L_m$ from P, $\{L_1, \ldots, L_m\} \subseteq S$ implies $L_0 \in S$. In particular, $\{L_1, \ldots, L_m\} \not\subseteq S$ if L_0 is empty.

2. S does not contain a pair of complementary literals L and $\neg L$.

Next, for any ELP P and a set S of literals, a *not*-free ELP P^S is defined as follows: a rule $L_0 \leftarrow L_1, \ldots, L_m$ is in P^S iff there is a ground rule $L_0 \leftarrow L_1, \ldots, L_m, \mathit{not}\, L_{m+1}, \ldots, \mathit{not}\, L_n$ from P such that $\{\, L_{m+1}, \ldots, L_n \,\} \cap S = \emptyset$. Then, S is a *consistent answer set* of P if S is a consistent answer set of P^S. An ELP P is *consistent* if it has a consistent answer set; otherwise P is *inconsistent*. A consistent answer set is simply called an answer set hereafter.

Now we provide the *belief set semantics* of an agent.

Definition 2.2 Given an agent $A = \langle\, \Pi, \Delta, \Gamma_t \,\rangle$, suppose that $D \subseteq \Delta$ satisfies the following conditions:

1. $\Pi \cup D \cup \Gamma_t$ is consistent,

2. there is no $D \subset D' \subseteq \Delta$ such that $\Pi \cup D' \cup \Gamma_t$ is consistent.

Then, an answer set S_t of $\Pi \cup D \cup \Gamma_t$ is called a *belief set* of A at timing t.

By the definition, a belief set is an answer set which is obtained from the program by maximally assuming hypotheses as far as they are consistent with the response set. In particular, the belief sets of A coincide with the answer sets of $\Pi \cup \Delta$ at timing t_0.

A belief set nonmonotonically changes according to the transition of timing.

Example 2.1 Consider the following problem. (a) An agent A has a plan of meeting with his client-agents c_1, c_2, and c_3. (b) If every client is free to join the meeting, the meeting will be held as scheduled. (c) Else if some client is not free, the meeting must be cancelled. (d) Then, the agent A has to make a plan according to the conveniences of other clients.

Suppose that the agent A initially assumes that each client is free and tries to make her plan. The initial stage of the situation is represented by the agent $A = \langle\, \Pi, \Delta, \Gamma_{t_0} \,\rangle$ with

$$\begin{array}{ll} \Pi: & hold\_meeting \leftarrow join(c_1), join(c_2), join(c_3), \\ & cancel\_meeting \leftarrow not\, hold\_meeting, \\ & join(x) \leftarrow free(x), \\ \Delta: & free(c_1),\ free(c_2),\ free(c_3), \end{array}$$

where c_1, c_2, c_3 are identifiers. First, the response set at timing t_0 is $\Gamma_{t_0} = \emptyset$, so the belief set of A at timing t_0 is

$$\begin{array}{rcl} S_{t_0} & = & \{\, free(c_1),\ free(c_2),\ free(c_3),\ join(c_1), \\ & & join(c_2),\ join(c_3),\ hold\_meeting \,\}. \end{array}$$

Next, let $\Gamma_{t_2} = \{\, \neg free(c_2) \,\}$ be the response set at timing t_2. That is, a response from the agent c_2 saying he is not free is arrived at timing t_2. Then, the belief set of A at timing t_2 becomes

$$\begin{array}{rcl} S_{t_2} & = & \{\, free(c_1),\ \neg free(c_2),\ free(c_3),\ join(c_1), \\ & & join(c_3),\ cancel\_meeting \,\}. \end{array}$$

Thus, belief sets characterize the state of one's belief which dynamically changes in a multi-agent environment.

Let $\mathcal{A} = \{\, A_1, \ldots, A_n \,\}$ be the set of agents in a system. Then, the declarative semantics of $\mathcal{A}$ at timing t is defined as the set $\{\, \mathcal{BS}_{1,t}, \ldots, \mathcal{BS}_{n,t} \,\}$ where $\mathcal{BS}_{i,t}$ is the set of belief sets of an agent A_i at timing t.

3 Representing Agents in ELP

In this section, we characterize defeasible reasoning by an agent in an extended logic program whose answer sets express dynamically changing belief sets of the agent at each timing.

An ELP α is written using situation calculus, which contains atoms of the form:

$$hold(f, t)$$

where f is a fluent variable ranging over literals in the language of a system, and t is a situation variable ranging over the set of timing. Then, defeasible reasoning by an agent introduced in the preceding section is characterized by α as follows.

Definition 3.1 Given an agent $A = \langle\, \Pi, \Delta, \Gamma_t \,\rangle$, an ELP α consists of the following rules.

1. For each rule

   $$L_0 \leftarrow L_1, \ldots, L_m,\ not\, L_{m+1}, \ldots,\ not\, L_n$$

   in Π, α contains the rule:

   $$\begin{array}{c} hold(L_0, t) \leftarrow hold(L_1, t), \ldots, hold(L_m, t), \\ not\, hold(L_{m+1}, t), \ldots,\ not\, hold(L_n, t)\,. \end{array}$$

2. For each literal $L \in \Delta$, α contains the following *choice rules*:

   $$\begin{array}{c} hold(L, t) \leftarrow not\, \overline{hold(L, t)}, \\ \overline{hold(L, t)} \leftarrow not\, hold(L, t)\,, \end{array}$$

   where $\overline{hold(L, t)}$ is a literal uniquely associated with $hold(L, t)$.

3. For each literal $L \in \Gamma_t$, α contains the fact:

   $$hold(L, t)\,.$$

For convenience, we assume that $hold(\neg A, t)$ stands for $\neg hold(A, t)$ for any atom A.

The choice rule at the second step specifies the state of the assumption $hold(L, t)$ at timing t. It is valid if $hold(L, t)$ is true, while it is invalid if $\overline{hold(L, t)}$ is true. In the third step, it is assumed that the response set at every timing is already known.

With this setting, the program α specifies the change of beliefs in an agent, and answer sets of α express the history of change at each timing. Belief sets of a particular timing is computed by answer sets in the following manner. Let S be an answer set of an ELP α. Then, we define

$$S^t_\alpha = \{\, L \mid hold(L, t) \in S \,\}.$$

An answer set S is called Δ-*maximal* if there is no answer set T such that $S \cap D \subset T \cap D$, where $D = \{\, hold(L, t) \mid L \in \Delta \,\}$.

Theorem 3.1 *Let $A = \langle\, \Pi, \Delta, \Gamma_t \,\rangle$ be an agent and α its ELP expression. Then, A has a belief set S_t at timing t iff α has a Δ-maximal answer set T such that $S_t = T^t_\alpha$.*

Proof: (Sketch) Suppose that S_t is an answer set of $\Pi \cup D \cup \Gamma_t$ where $D \subseteq \Delta$. Then, D is a maximal subset of Δ such that $\Pi \cup D \cup \Gamma_t$ is consistent. In α, on the other hand, for each literal $L \in \Delta$, the choice rule can select $hold(L,t)$ or $\overline{hold(L,t)}$ at timing t. If L is consistent with the response set Γ_t, it is included in S_t. In this case, the corresponding literal $hold(L,t)$ is included in the Δ-maximal answer set T. Else if L is inconsistent with the response set Γ_t, it is not included in S_t. In this case, the corresponding literal $\overline{hold(L,t)}$ is included in the Δ-maximal answer set T. Thus, an assumption L is in S_t iff $hold(L,t)$ is in a Δ-maximal answer set T. For any rule in Π and any fact in Γ_t, the corresponding rules and facts are in α. Then, for any literal L, $L \in S_t$ iff $hold(L,t) \in T$. Hence, the result follows. □

Example 3.1 In the agent $A = \langle\, \Pi, \Delta, \Gamma_{t_i} \,\rangle$ of Example 2.1, α becomes

$$
\begin{aligned}
&hold(hold\_meeting, t) \leftarrow hold(join(c_1), t),\\
&\qquad\qquad hold(join(c_2), t),\\
&\qquad\qquad hold(join(c_3), t).\\
&hold(cancel\_meeting, t) \leftarrow\\
&\qquad\qquad not\, hold(hold\_meeting, t).\\
&hold(join(x), t) \leftarrow hold(free(x), t).\\
&\neg hold(free(c_2), t_2).\\
&hold(free(c_j), t) \leftarrow not\, \overline{hold(free(c_j), t)}\\
&\qquad\qquad (j = 1, 2, 3).\\
&\overline{hold(free(c_j), t)} \leftarrow not\, hold(free(c_j), t)\\
&\qquad\qquad (j = 1, 2, 3).
\end{aligned}
$$

Then, α has the Δ-maximal answer set T where

$$
\begin{aligned}
T^t_\alpha \;=\; \{\, &free(c_1),\, free(c_2),\, free(c_3),\, join(c_1),\\
&join(c_2),\, join(c_3),\, hold\_meeting \,\}
\end{aligned}
$$

at timing $t < t_2$, and

$$
\begin{aligned}
T^t_\alpha \;=\; \{\, &free(c_1),\, \neg\, free(c_2),\, free(c_3),\, join(c_1),\\
&join(c_3),\, cancel\_meeting \,\}
\end{aligned}
$$

at timing $t \geq t_2$.

Next we consider the problem in some particular situation.

A *normal logic program* (NLP) is a set of rules of the form:

$$A_0 \leftarrow A_1, \ldots, A_m,\, not\, A_{m+1}, \ldots,\, not\, A_n$$

where each $A_i\, (0 \leq i \leq n)$ is an atom.

Let $A^{strat} = \langle\, \Pi, \Delta, \Gamma_t \,\rangle$ be an agent structure which is the same as in Definition 2.1 except that Π is a *locally stratified* NLP[2] and Δ is a set of atoms.

In this case, an NLP expression of an agent is defined as follows.

[2] We refer the reader to [9] for the precise definition of a local stratified NLP. Note that a locally stratified NLP does not contain integrity constraints by its definition.

Definition 3.2 Given an agent $A^{strat} = \langle\, \Pi, \Delta, \Gamma_t \,\rangle$, an NLP β consists of the following rules.

1. For each rule

   $$A_0 \leftarrow A_1, \ldots, A_m,\, not\, A_{m+1}, \ldots,\, not\, A_n$$

   in Π, β contains the rule:

   $$
   \begin{aligned}
   hold(A_0, t) \leftarrow\; &hold(A_1, t), \ldots, hold(A_m, t),\\
   &not\, hold(A_{m+1}, t), \ldots,\, not\, hold(A_n, t)\,.
   \end{aligned}
   $$

2. For each atom $A \in \Delta$, β contains the fact:

   $$hold(A, t_0)\,,$$

   and the following *inertia rules*:

   $$
   \begin{aligned}
   &hold(A, t_{i+1}) \leftarrow hold(A, t_i),\, not\, \overline{hold(A, t_{i+1})},\\
   &\overline{hold(A, t_{i+1})} \leftarrow \overline{hold(A, t_i)},\, not\, hold(A, t_{i+1})\,,
   \end{aligned}
   $$

   where $\overline{hold(A,t)}$ is an atom uniquely associated with $hold(A,t)$.

3. For each literal $L \in \Gamma_t$, β contains the fact:

   $$
   \begin{aligned}
   &hold(A,t)\,, && \text{if } L = A \text{ with an atom } A,\\
   &\overline{hold(A,t)}\,, && \text{if } L = \neg A \text{ with an atom } A.
   \end{aligned}
   $$

The main difference between α and β is at the second step. Instead of the choice rules in α, the inertia rules are considered for each assumption in Δ together with the facts $hold(A, t_0)$ as the initial assumptions. An answer set of β is equivalent to a *stable model* in an NLP [4]. Let S be a stable model of an NLP β. Then, we define

$$S^t_\beta = \{\, A \mid\; hold(A,t) \in S \,\} \cup \{\, \neg A \mid\; \overline{hold(A,t)} \in S \,\}\,.$$

Then we have the following result.

Theorem 3.2 *Let $A^{strat} = \langle\, \Pi, \Delta, \Gamma_t \,\rangle$ be an agent and β its NLP expression. Then, A^{strat} has a belief set S_t at timing t iff β has a stable model T such that $S_t = T^t_\beta$.*

Proof: (Sketch) In a stratified program Π, the change of beliefs from A to $\neg A$ on some agent does not make a program inconsistent. (Note that any assumption appears only in the body of a rule.) Then, an initial assumption in Δ holds at each timing unless its contrary is known by a response. The inertia rules represent the situation. In this case, there is a 1-1 correspondence between a belief set S_t and the collection of atoms T^t_β for a stable model T. □

Note that in contrast to Theorem 3.1 the assumption of Δ-maximality is unnecessary in Theorem 3.2.

Example 3.2 The agent $A = \langle\, \Pi, \Delta, \Gamma_{t_i} \,\rangle$ of Example 2.1 is in fact A^{strat}, hence we can consider an alternative translation β as:

$$
\begin{aligned}
&hold(hold\_meeting, t) \leftarrow hold(join(c_1), t),\\
&\qquad\qquad hold(join(c_2), t),\\
&\qquad\qquad hold(join(c_3), t).\\
&hold(cancel\_meeting, t) \leftarrow\\
&\qquad\qquad not\, hold(hold\_meeting, t).\\
&hold(join(x), t) \leftarrow hold(free(x), t).\\
&hold(free(c_1), t_0).\\
&hold(free(c_2), t_0).\\
&hold(free(c_3), t_0).\\
&\overline{hold(free(c_2), t_2)}.\\
&hold(free(c_j), t_{i+1}) \leftarrow hold(free(c_j), t_i),\\
&\qquad\qquad not\, \overline{hold(free(c_j), t_{i+1})}\\
&\qquad\qquad\qquad (j = 1, 2, 3).\\
&\overline{hold(free(c_j), t_{i+1})} \leftarrow \overline{hold(free(c_j), t_i)},\\
&\qquad\qquad not\, hold(free(c_j), t_{i+1})\\
&\qquad\qquad\qquad (j = 1, 2, 3).
\end{aligned}
$$

Then, β has the stable model T where

$$
\begin{aligned}
T^t_\beta \;=\; \{\, &free(c_1),\ free(c_2),\ free(c_3),\ join(c_1),\\
&join(c_2),\ join(c_3),\ hold\_meeting \,\}
\end{aligned}
$$

at timing $t < t_2$, and

$$
\begin{aligned}
T^t_\beta \;=\; \{\, &free(c_1),\ \neg free(c_2),\ free(c_3),\ join(c_1),\\
&join(c_3),\ cancel\_meeting \,\}
\end{aligned}
$$

at timing $t \geq t_2$.

The translation β cannot be used for an agent A which has an unstratified knowledge base.

Example 3.3 Let $A = \langle\, \Pi, \Delta, \Gamma_t \,\rangle$ be the agent such that

$$
\begin{aligned}
\Pi:&\quad r \leftarrow p(a), q, not\, r,\\
&\quad q \leftarrow not\, p(b),\\
\Delta:&\quad p(a),\ p(b),
\end{aligned}
$$

where a and b are identifiers. Suppose that the response set changes from $\Gamma_{t_0} = \emptyset$ to $\Gamma_{t_1} = \{\, \neg p(b) \,\}$. Then, the belief set of A at each timing becomes

$$
\begin{aligned}
S_{t_0} &= \{\, p(a),\ p(b) \,\},\\
S_{t_1} &= \{\, \neg p(b),\ q \,\}.
\end{aligned}
$$

In this case, the corresponding ELP α:

$$
\begin{aligned}
&hold(r, t) \leftarrow hold(p(a), t), hold(q, t),\\
&\qquad\qquad not\, hold(r, t).\\
&hold(q, t) \leftarrow not\, hold(p(b), t).\\
&\neg hold(p(b), t_1).\\
&hold(p(x), t) \leftarrow not\, \overline{hold(p(x), t)}\ (x = a, b).\\
&\overline{hold(p(x), t)} \leftarrow not\, hold(p(x), t)\ (x = a, b).
\end{aligned}
$$

has the Δ-maximal answer set:

$$
\begin{aligned}
\{\, &hold(p(a), t_0),\ hold(p(b), t_0),\ \neg hold(p(b), t_1),\\
&\overline{hold(p(a), t_1)},\ \overline{hold(p(b), t_1)},\ hold(q, t_1), \ldots\},
\end{aligned}
$$

which expresses the belief sets of each timing, while the corresponding NLP β:

$$
\begin{aligned}
&hold(r, t) \leftarrow hold(p(a), t), hold(q, t),\\
&\qquad\qquad not\, hold(r, t).\\
&hold(q, t) \leftarrow not\, hold(p(b), t).\\
&hold(p(a), t_0).\\
&hold(p(b), t_0).\\
&\overline{hold(p(b), t_1)}.\\
&hold(p(x), t_{i+1}) \leftarrow hold(p(x), t_i),\\
&\qquad\qquad not\, \overline{hold(p(x), t_{i+1})}\ (x = a, b).\\
&\overline{hold(p(x), t_{i+1})} \leftarrow \overline{hold(p(x), t_i)},\\
&\qquad\qquad not\, hold(p(x), t_{i+1})\ (x = a, b).
\end{aligned}
$$

has no stable model. In fact, $hold(p(a), t_1)$ and $hold(q, t_1)$ become true, and the first rule makes the program inconsistent.

This example illustrates the situation in which the change of beliefs on one agent affects beliefs on other agents. In such cases, the translation α is more expressive that β.

Using situation calculus, we can extend the present language α to express various situations. For example, suppose that an agent expects a response f from some agent a at timing between t_1 and t_n but the exact timing is unknown. The situation is expressed by the following k-rules:

$$
\begin{aligned}
hold(f(a), t_1) \leftarrow\ & not\, hold(f(a), t_2),\\
&\cdots\\
&not\, hold(f(a), t_n);\\
\cdots&\\
hold(f(a), t_n) \leftarrow\ & not\, hold(f(a), t_1),\\
&\cdots\\
&not\, hold(f(a), t_{n-1})\,,
\end{aligned}
$$

which is equivalent to the disjunctive rule:

$$
hold(f(a), t_1)\ \vee\ \cdots\ \vee\ hold(f(a), t_n)
$$

in effect.

Suppose another situation that an agent asks the same question f to several agents, and uses the quickest response from them. The situation is expressed by the following rule:

$$
\begin{aligned}
hold(f, t_i) \leftarrow\ & hold(f(x), t_i),\\
&not\, \neg hold(f(y), t_j),\ t_j < t_i\,.
\end{aligned}
$$

The above rule presents that a response $f(x)$, meaning "f is true", from some agent x is assumed unless its contrary $\neg f$ has already been assumed by another agent y at earlier timing.

4 Related Work

Kowalski and Sadri [8] propose a multi-agent system using extended logic programs. It has a mechanism of forward reasoning for performing behaviors reactive to the environment as well as backward reasoning for planning or problem solving within an agent. Given an input to be executed, it is propagated to agents and each agent computes a possible action using *abduction*. In this system, a knowledge base of each agent is assumed to be unchanged in the process of reasoning.

Dell'Acqua and Pereira [3] introduce a multi-agent system which has the ability of updating agents' knowledge. It combines Kowalski and Sadri's multi-agent system and the technique of *dynamic logic programming*. A knowledge base of one agent can be updated by a series of update programs which specify update requests by other agents. In their system each agent computes an action through abduction, which is in contrast with our default reasoning.

Situation calculus is usually used for reasoning about actions and changes [6, 1]. By contrast, we used the calculus for characterizing defeasible reasoning in multi-agent environments. A program transformation from an agent to an ELP α is similar to the one in [7], in which an abductive logic program, called a *knowledge system*, is transformed to an ELP. We used the transformation for expressing defeasible reasoning in multi-agent systems.

A multi-agent system presented in this paper is a slight modification of the one introduced in [10]. In that paper, a top-down abductive procedure which reflects changes in belief is introduced, but the semantic issue is left open.

5 Summary

This paper presented a logical framework of defeasible reasoning in multi-agent systems. Each agent has an ability to perform default reasoning based on her own belief, and also can revise her belief according to the information brought by other agents. We characterized the system in situation calculus, in which answer sets express dynamically changing belief sets of an agent.

In [2] the authors say:

> The ability to express the current situation, record facts and perform hypothetical reasoning makes a language a viable candidate for use in designing intelligent agents capable of planning and acting in a changing environment.

In our language, the current situation is expressed by a knowledge base, new information is recorded by a response set, and hypothetical reasoning is performed by each agent. In this sense, the language introduced in this paper is considered expressive for designing intelligent agents. On the other hand, in the present system an agent does not have an ability to update her knowledge base, and the communication between agents is restricted on the confirmation of assumptions. Future research includes enhancing the ability of agents to cope with more complicated situations.

Acknowledgment

This research is partly supported by Grand-in-Aid for Scientific Research, "Construction of Logical Multi-agent Systems under Incomplete Communication Environments", The Ministry of Education, Science, Sports and Culture, Japan.

References

[1] C. Baral.. Rule based updates on simple knowledge bases. In: *Proc. AAAI-94*, pp. 136–141, 1994.

[2] C. Baral, M. Gelfond, and A. Provetti. Representing actions: laws, observations and hypothesis. *J. Logic Programming* 31:201–243, 1997.

[3] P. Dell'Acqua and L. M. Pereira. Updating agents. In: *Proc. ICLP'99 Workshop on Multi-Agent Systems in Logic Programming*, 1999.

[4] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In: *Proc. 5th Int'l Conf. and Symp. on Logic Programming*, pp. 1070–1080, MIT Press, 1988.

[5] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9:365–385, 1991.

[6] M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *J. Logic Programming* 17:301–321, 1993.

[7] K. Inoue. Hypothetical reasoning in logic programs. *J. Logic Programming* 18:191-227, 1994.

[8] R. A. Kowalski and F. Sadri. From logic programming to multi-agent systems. *Annals of Mathematics and Artificial Intelligence* 25:391–419, 1999.

[9] T. C. Przymusinski. On the declarative semantics of deductive databases and logic programs. In: J. Minker (ed.), *Foundations of Deductive Databases and Logic Programming* pp. 193–216, Morgan Kaufmann, 1988.

[10] K. Satoh, K. Inoue, K. Iwayama, and C. Sakama. Speculative computation by abduction under incomplete communication environments. In: *Proc. 4th Int'l Conf. on Multi-Agent Systems*, to appear, 2000.