Operational Semantics of Agents Sharing Action and Communication

Abstract. This paper introduces a new the protocol for agents to collaborate each other and the description of agent's actions including comunications in extended π-calculus[Mil91,KM01] in order to share actions. We focus on intentions of agents to define the protocol. In the protocol, an agent builds a group in which agents have the same intentions or the similar intentions. We formalize actions based on extended π-calculus and show the operational semantics of our proposed actions. Our formalization is based on π-calculus which provides a sound foundation to concurrent computations and a communication among parallel processes. Thus we assign a process to an action, it constructs the action including communications.

1 Introduction

The performance of a multi-agent system is bad when agents in it might individually perform actions. Though agents may communicate with each other to improve the performance, the performance might be worse because the communication augments ineffective actions [HCY99]. In order to avoid the situation, an agent has to communicate agents which have similar intentions[CLS].

In a BDI(belief, desire and intention) model, agents focus on their intention and collaborate with each other. The concepts of them have been described, but there were no clear description of how such a model could be used and effectively work [CLS,Nak99].

In this paper we propose a new protocol for agents to collaborate each other and describe the operational semantics of sharing action and communication according to it. In the protocol, an agent communicates with other agents to build groups and he builds a group in which agents have the same intentions or the similar intentions. The agents in one group can share their actions, because their intentions bring the same actions or the sharable actions. Our formalization is based on π-calculus[Mil91,KM01] which provides a sound foundation to concurrent computations and a communication among parallel processes[FG99,MUN98]. Thus we assign a process to an action, it constructs the action including communications.

2 Sharable Actions

"Sharable Action" means the action which can be performed by every agent in a group. We define the rule to decide "Sharable Action" to denote unsharable actions. The definition of unsharable actions is as follows:

Definition 1. ***Unsharable Actions***

- *An action influences an agent, which would perform it, is unsharable, like as a walk and a turn, etc.*
- *An action influences agents in a group is unsharable, like as a communication with neighbor agents, etc.*
- *An action which other agents cannot perform is unsharable, like as taking the neighbor object which adjoins a agent.*

The first rule means that if the action which is significant only to an agent and influences himself is performed by other agent, it may be waste of time. For instance, when an agent would like to move to a position and it performed by other agent indead of him, they may be back to previous position.

The second rule means that if the action which influences agents in the same group is performed, the results is undesirable. For instance, when an agent would like to communicate with neighbor agents and it performed by other agent, the agent, which built the action, cannot get necessary information to fail to communicate.

The third rule means that some agents can perform the action and the others cannot perform it. For instance, when an agent would like to clear rubble near him and it would be performed by other agent, the other agent cannot clear rubble, if he does not adjoin rubble.

Thus we call an action sharable, if it does not suit with any rules for unsharable actions.

3 Similar Intentions

Agents build groups according to their own intentions. The most simple way to build a group and to share actions is building a group with agents which have the same intentions. However, this condition is so hard that agents cannot build many groups. In order to ease the condition, we define similar intentions.

Before defining the similar intentions, we explain what an intention means in this paper. An intention means that the indicator to introduce actions from a goal for agents. In other words, an agent select actions according to a intention with the information of his environment and goal.

Definition 2. ***Intention***

$Intention : G \times E \rightarrow A$

where "G" means the current goal of the agent. "E" means the information including the environment around the agent. "A" means actions which have a layered structure.

Similar intentions are defined by the relation among intentions. We define the similar intentions which have some common sharable actions as a result of map.

Definition 3. ***Similar Intentions***
$SIntentions : I^n \times G^n \times E^n \rightarrow B$

where "B" means boolean value(true or false) and "I" means a function Intention in the definition 2.

If n equals 2, the function $SIntention$ is as follows:
$SIntentions(I_1, G_1, E_1, I_2, G_2, E_2) \equiv \left|Set(I_1(G_1, E_1)) \cup Set(I_2(G_2, E_2))\right| > \theta$
where I_n means the function Intention, Set is the function from a layered structure to a set as $Set : A \rightarrow \{a|a \in A\}$ and θ is a threshold.

4 Agents Group

An agent communicates with other agents to build a group with them having same intentions or similar intentions. If an agent communicate with all agent in the system, the amount of communication among agents proliferates. However, an agent only has to communicate with surrounding agents, because the probability that surrounding agents have sharable actions is high. For instance, if an agent would like to move a block and collaborate with other agents which are in the distance, they might be unable to find the block or move to near him. The communication among them wastes the time.

By taking the prior conditions into consideration, we simplify Similar Intentions in the definition 3, because agents in a close zone have a similar environmental information . The definition of simplified $SimilarIntentions$ is as follows:

Definition 4. ***Similar Intentions'***
$SIntentions' : I^n \times G^n \times E_o \rightarrow B$

If n equals 2, the function $SIntention'$ is as follows:
$SIntentions(I_1, G_1, E_o, I_2, G_2) \equiv \left|Set(I_1(G_1, E_o)) \cup Set(I_2(G_2, E_o))\right| > \theta$
where I_n means the function Intention, Set is the function from a layered structure to a set as $Set : A \rightarrow \{a|a \in A\}$ and θ is threshold.

4.1 Build Groups

An agent builds a group with surrounding agents because of the previous conclusion. When an agent does not belong to a group and would like to collaborate with other agents, he builds a group as follows:

1. The agent, named agent L, asks the each surrounding agent about whether he belongs to any group.

2. Each agent sends the group identification to the agent L as the answer if he belongs to a group, otherwise sends his current goal .
3. The agent L chooses the member of a new group. The criterion to choose the member is the similarity between the intentions defined in the definition 4. A quantity in one communication is less than by using the definition 3.
4. The agent L sends the group identification to the chosen agents and asks them to send the actions.

4.2 Collaboration in Groups

The agent which asked other agents is the leader in the group. In subsection 4.1, the agent L is the leader.

The roles of the leader are the management of the member and allocating sharable actions. The details of the collaboration and the leader's roles are as follows:

1. The member in the group sends his actions to the leader, if he wants to continue belonging to the group, otherwise he informs the leader that he secedes from the group.
2. The leader analyzes the actions and creates new action set including the communication function which is written in π-calculus (the details are in the next section).
3. The leader sends the action set to the members including himself.
4. The members perform the action set. They send their next goal to the leader.
5. The leader tells to the member whose next goal differ from the leader's very much to secede from the group.
6. When the number of the member is less than some number γ, the leader informs all members that the group is deactivated and the works of the group is finished. Otherwise, the leader asks the other agents to send the actions and a situation returns to 1.

5 Description of Actions

The action set including communication is written in π-calculus and extended π-calculus[Mil91,KM01] which includes the effect of passing process and self-interpreter like 'eval' in Lisp. Because of the attributes, agents can transparently send the action set among them and easily perform actions and communication only by evaluating the action set in π-calculus.

Before proposing the description, we explain the layered structure of actions and how to share actions.

5.1 Layered Structure

The layer in actions means the relationship among the actions. The upper layer means the precondition to perform lower layers. Actions in the same layer can

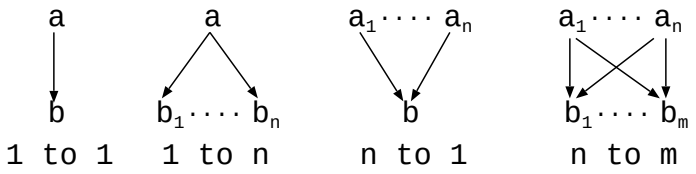

Fig. 1. Layered Structures of Actions

be performed without depending on those order. Figure 1 shows the actions with two layers.

In Figure 1, "a" and "a_i" are in the upper layer and "b" and "b_i" are in the lower layer. The actions "a_i" or "b_i" can be performed without depending on their order. Therefore, the actions in the same layer can be performed in parallel. The leader of a group can distribute the actions in the same layer to the members.

The four cases in Figure 1 are divided into two cases according to the number of actions in the upper layer. The first case is the upper layer with one action, which is the two cases of the left-hand side in Figure 1. The other case is the upper layer with many actions. In the former case, the action b and b_i are performed after performing the action a. Therefore, the agent performing the action a should just tell other agents to perform the actions b and b_i. In the latter case, when all actions named a_i are finished, the action b and b_i can be performed. Therefore, it is necessary to confirm that all actions in a precondition are finished.

The leader of a group analyzes gathered actions and connects them by sharing same actions. The general style to connect actions is shown in Figure 2.

We abbreviate "the action set A" to "A", and use the same abbreviation to the others. "A-X" means the difference set between the sets "A" and "X".

In Figure 2, intersections among action sets not to be described do not have any element. For instance, the case $A \cap C = X \neq \emptyset$ implies the case which is $A \cap C = X \neq \emptyset$, $A \cap D = \emptyset$, $B \cap C = \emptyset$ $A \cap B = \emptyset$, and $C \cap D = \emptyset$.

1. The case $A \cap C = X \neq \emptyset$ means that each precondition has common actions X, then X must be the preconditions for B and D. If A equals C, then A is the preconditions for B and D.
2. The case $A \cap D = Y \neq \emptyset$ means that Y is the preconditions for B, but C must be finished to perform Y. If A equals D, then A is performed after finishing C. This situation is regarded as grafting the actions. There is the contrast case $B \cap C = Y' \neq \emptyset$.
3. The case $B \cap D = Z \neq \emptyset$ means that Z needs A and C as the preconditions. If B equals D, then B is performed after finishing A and C.
4. The case $A \cap C = X \neq \emptyset$, $A \cap D = Y \neq \emptyset$ and $X \cap Y = \emptyset$ is composed of the case 1 and 2.

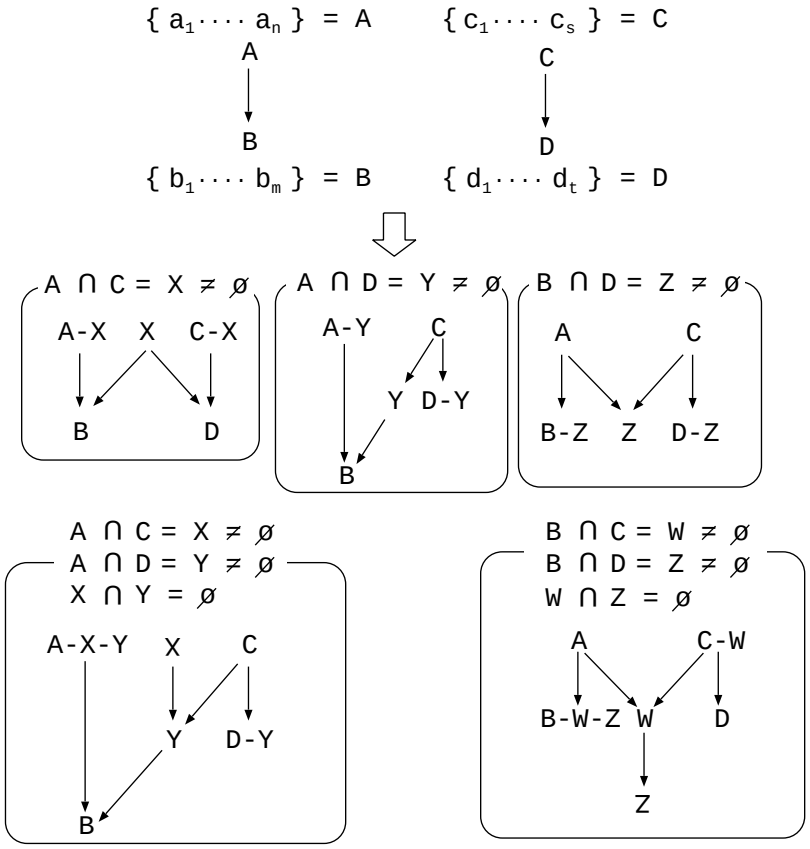

Fig. 2. Connecting Actions

5. The case $B \cap C = W \neq \emptyset$, $B \cap D = Z \neq \emptyset$ and $W \cap Z = \emptyset$ is composed of the case 3 and the contrast case of the case 2.

5.2 Description of Actions in π-calculus

We define the descriptions of actions according to action sets in previous subsection. We use π-calculus and extended π-calculus [Mil91,KM01] to describe transparent communications.

Definition 5. ***Leader's Actions***

$$Leader \stackrel{def}{=} \ !\big(Leader(x).(eval \ \ x)\big)$$

where Leader is the name to get a process and $(eval \ \ x)$ *means the self-interpreter like 'eval' in Lisp.*

Definition 6. ***Member's Actions***

$Agent_n \stackrel{def}{=} \ !\big(Agent_n(x).(eval\ \ x)\big)$

where $Agent_n$ *is the agent identification and* $(eval\ \ x)$ *means the self-interpreter like 'eval' in Lisp.*

The definitions 5 and 6 mean an agent get a process through networks and perform it by evaluating it.

Definition 7. ***Actions to Members***

$Actions(T) \stackrel{def}{=} Code(P)$
$P \equiv \overline{system}[t_1].\overline{t_1} \mid \cdots \mid \overline{system}[t_n].\overline{t_n} \mid t_1.\ \cdots\ .t_n.\overline{Leader}[Code(Q)]$
$Q \equiv \overline{t_1} \mid \cdots \mid \overline{t_n}$

where $T = \{t_1, t_2, \ldots t_{n-1}, t_n\}$.

In the definition 7, $Code(P)$ means the coded process from P. Two processes P and $(eval\ Code(P))$ are equivalent. $\overline{system}[t_i]$ means accessing the action named t_i in the system. The detail of the process is in the next section.

The definitions 8, 9, 10, 11 and 12 correspond to the connectiong actions 1, 2, 3, 4 and 5 in previous subsection, respectively. These defined processes are used by the leader agent according to the action structures. In them, A_i means the action set $\{a_k, a_{k+1}, \cdots, a_{l-1}, a_l\}(k \leq l)$ which is assigned to the $Agent_i$. The other descriptions B_i, C_i, D_i, W_i, X_i, Y_i and Z_i are the same.

For instance, the definition 8 means that the leader assigns the action sets A_i, C_i, X_i to $Agent_i$ at first, after finishing the action sets X and A he assigns the action sets B_i to $Agent_i$ or finishing the action sets X and C he assigns the action sets D_i to $Agent_i$. We will explain the detail of an operational semantics in next Section.

Definition 8. ***Common Precondition***

$Common \stackrel{def}{=} \ x_1.x_2 \cdots x_{k-1}.x_k.\big((a_i.\ \cdots\ .a_j.P) \mid (c_x.\ \cdots\ .c_y.Q)\big) \mid R$
$P \equiv \overline{Agent_1}[Actions(B_1)] \mid \cdots \mid \overline{Agent_n}[Actions(B_n)]$
$Q \equiv \overline{Agent_1}[Actions(D_1)] \mid \cdots \mid \overline{Agent_n}[Actions(D_n)]$
$R \equiv \overline{Agent_1}[Actions(A_1C_1X_1)] \mid \cdots \mid \overline{Agent_n}[Actions(A_nC_nX_n)]$

where $X = A \cap C$, $A = \bigcup_{i=1}^{n} A_i$ *and* $A_i \cap A_j = \emptyset (i \neq j)$. *And the other sets are the same.*

Definition 9. ***Merged Precondition***

$Mearged \stackrel{def}{=} \ (a_i.\ \cdots\ .a_j.y_t.\ \cdots\ .y_s.P) \mid (c_x.\ \cdots\ .c_y.Q) \mid R$

$P \equiv \overline{Agent_1}[Actions(B_1)] \mid \cdots \mid \overline{Agent_n}[Actions(B_n)]$
$Q \equiv \big(\overline{Agent_1}[Actions(D_1)] \mid \cdots \mid \overline{Agent_n}[Actions(D_n)]$
$\quad \mid \overline{Agent_1}[Actions(Y_1)] \mid \cdots \mid \overline{Agent_n}[Actions(Y_n)]\big)$

$R \equiv \overline{Agent_1}[Actions(A_1C_1)] \mid \cdots \mid \overline{Agent_n}[Actions(A_nC_n)]$

where $X = A \cap D$. A_i, B_i, C_i, D_i *and* Y_i *are the same as above.*

Definition 10. ***Common Posterior Action***

$CommonPost \stackrel{def}{=} \big(a_i. \cdots .a_j.(\overline{a'} \mid P)\big) \mid \big(c_x. \cdots c_y.(\overline{c'} \mid Q)\big) \mid \big(a'.c'.R\big) \mid S$

$P \equiv \overline{Agent_1}[Actions(B_1)] \mid \cdots \mid \overline{Agent_n}[Actions(B_n)]$
$Q \equiv \overline{Agent_1}[Actions(D_1)] \mid \cdots \mid \overline{Agent_n}[Actions(D_n)]$
$R \equiv \overline{Agent_1}[Actions(Z_1)] \mid \cdots \mid \overline{Agent_n}[Actions(Z_n)]$
$S \equiv \overline{Agent_1}[Actions(A_1C_1)] \mid \cdots \mid \overline{Agent_n}[Actions(A_nC_n)]$

where $Z = B \cap D$. A_i, B_i, C_i, D_i *and* Z_i *are the same as above.*

Definition 11. ***Common Precondition & Merged Preconditions***

$Common\&Merged \stackrel{def}{=} x_1. \cdots .x_k.\Big((a_i. \cdots .a_j.y_t. \cdots .y_s.P) \mid (c_x. \cdots .c_y.Q)\Big) \mid R$

$P \equiv \overline{Agent_1}[Actions(B_1)] \mid \cdots \mid \overline{Agent_n}[Actions(B_n)]$
$Q \equiv \big(\overline{Agent_1}[Actions(D_1)] \mid \cdots \mid \overline{Agent_n}[Actions(D_n)]$
$\quad \mid \overline{Agent_1}[Actions(Y_1)] \mid \cdots \mid \overline{Agent_n}[Actions(Y_n)]\big)$

$R \equiv \overline{Agent_1}[Actions(A_1C_1X_1)] \mid \cdots \mid \overline{Agent_n}[Actions(A_nC_nX_n)]$

where $X = A \cap C$ *and* $Y = A \cap D$. A_i, B_i, C_i, D_i, X_i *and* Y_i *are the same as above.*

Definition 12. ***Common Posterior Action& More Action***

$CommonPost\&More \stackrel{def}{=} \big(a_i. \cdots .a_j.(\overline{a'} \mid P)\big) \mid \big(c_x. \cdots c_y.(\overline{c'} \mid Q)\big) \mid \big(a'.c'.(R \mid S)\big) \mid T$

$P \equiv \overline{Agent_1}[Actions(B_1)] \mid \cdots \mid \overline{Agent_n}[Actions(B_n)]$
$Q \equiv \overline{Agent_1}[Actions(D_1)] \mid \cdots \mid \overline{Agent_n}[Actions(D_n)]$
$R \equiv \overline{Agent_1}[Actions(W_1)] \mid \cdots \mid \overline{Agent_n}[Actions(W_n)]$
$S \equiv w_1. \cdots .w_n.\big(\overline{Agent_1}[Actions(Z_1)] \mid \cdots \mid \overline{Agent_n}[Actions(Z_n)]\big)$
$T \equiv \overline{Agent_1}[Actions(A_1C_1)] \mid \cdots \mid \overline{Agent_n}[Actions(A_nC_n)]$

where $W = B \cap C$ *and* $Z = B \cap D$. A_i, B_i, C_i, D_i, W_i *and* Y_i *are the same as above.*

6 Operational Semantics

The operation semantics of the definitions uses the basic definitions of π-calculus and extended π-calculus with an additional operation in terms of the process *system*: $\overline{system}[a]$ which means accessing the action named a in the system. This process can be performed without a symmetric process as the process $system(x)$.

The operation rule of the process is extending the reduction rule of π-calculus as follows:

$$\text{SYSTEM} : \overline{system}[a].P \overset{\text{perform } a}{\longrightarrow} P$$

The reduction mark $\overset{\text{perform } a}{\longrightarrow}$ means that after observing the process named "a" the process is reduced. However, the rule has no influence upon calculating processes in π-calculus.

Firstly, we give the opereational semantics of the definiton 8. Suppose the action set A_i equal $\{a_i\}$ to ease the explanation. The other action sets are on the same supposition.

The leader has the following processes:

$Leader \equiv\ !\big(Leader(x).(eval\ x)\big) \mid x_1.\cdots.x_n\big((a_1.\cdots.a_n.P) \mid (c_1.\cdots.c_n.Q)\big) \mid R$
$P \equiv \overline{Agent_1}[Actions(b_1)] \mid \cdots \mid \overline{Agent_n}[Actions(b_n)]$
$Q \equiv \overline{Agent_1}[Actions(d_1)] \mid \cdots \mid \overline{Agent_n}[Actions(d_n)]$
$R \equiv \overline{Agent_1}[Actions(a_1c_1x_1)] \mid \cdots \mid \overline{Agent_n}[Actions(a_nc_nx_n)]$

The member numbered i has the following processes:

$Agent_i \equiv !\big(Agent_i(x).(eval\ x)\big)$

The names $Leader$ and $Agent_i$ mean the communication ports to the leader or agents.

In the first step, the leader perform the processes $\overline{Agent_i}[Actions(a_ic_ix_i)]$. Each agent named i gets $Actions(a_ic_ix_i)$ and evaluates it by $(eval\ Actions(a_ic_ix_i))$. These operations are as follows:

$$\begin{aligned}&!\big(Leader(x).(eval\ x)\big) \mid x_1.\cdots.x_n\big((a_1.\cdots.a_n.P) \mid (c_1.\cdots.c_n.Q)\big) \mid R\\ \rightarrow\ &!\big(Leader(x).(eval\ x)\big) \mid x_1.\cdots.x_n\big((a_1.\cdots.a_n.P) \mid (c_1.\cdots.c_n.Q)\big)\end{aligned}$$

$$!\big(Agent_i(x).(eval\ x)\big) \rightarrow\ !\big(Agent_i(x).(eval\ x)\big) \mid (eval\ Actions(a_ic_ix_i))$$

The next step is that each agent performs action by $(eval\ Actions(a_ic_ix_i))$ as follows:

$$\begin{aligned}&(eval\ Actions(a_ic_ix_i))\\ \rightarrow\ &\overline{system}[a_i].\overline{a_i} \mid \overline{system}[c_i].\overline{c_i} \mid \overline{system}[x_i].\overline{x_i} \mid a_i.c_i.x_i.\overline{Leader}[Code(S_i)]\\ \overset{a_i,c_i,x_i}{\longrightarrow}\ &\overline{a_i} \mid \overline{c_i} \mid \overline{x_i} \mid a_i.c_i.x_i.\overline{Leader}[Code(S_i)] \rightarrow\ \overline{c_i} \mid \overline{x_i} \mid c_i.x_i.\overline{Leader}[Code(S_i)]\\ \rightarrow\ &\overline{x_i} \mid x_i.\overline{Leader}[Code(S_i)] \rightarrow\ \overline{Leader}[Code(S_i)]\end{aligned}$$

The leader gets $Code(S_i) \equiv Code(\overline{a_i} \mid \overline{c_i} \mid \overline{x_i})$ through the prot $Leader$. The next operation is as follows:

$$\begin{aligned}&!\big(Leader(x).(eval\ x)\big) \mid (eval\ Code(S_i))\\ &\mid x_1.\cdots.x_n\big((a_1.\cdots.a_n.P) \mid (c_1.\cdots.c_n.Q)\big)\\ \rightarrow\ &!\big(Leader(x).(eval\ x)\big) \mid \overline{a_i} \mid \overline{c_i} \mid \overline{x_i} \mid x_1.\cdots.x_n\big((a_1.\cdots.a_n.P) \mid (c_1.\cdots.c_n.Q)\big)\end{aligned}$$

If the agent $Agent_1$ finishes actions then

$$\begin{aligned}\rightarrow\ &!\big(Leader(x).(eval\ x)\big)\\ &\mid \overline{a_1} \mid \overline{c_1} \mid \overline{x_1} \mid \overline{a_i} \mid \overline{c_i} \mid \overline{x_i} \mid x_1.\cdots.x_n\big((a_1.\cdots.a_n.P) \mid (c_1.\cdots.c_n.Q)\big)\\ \rightarrow\ &!\big(Leader(x).(eval\ x)\big)\\ &\mid \overline{a_1} \mid \overline{c_1} \mid \overline{a_i} \mid \overline{c_i} \mid \overline{x_i} \mid x_2.\cdots.x_n\big((a_1.\cdots.a_n.P) \mid (c_1.\cdots.c_n.Q)\big)\end{aligned}$$

If all agent $Agent_i$ finishes actions then

$\rightarrow !\big(Leader(x).(eval\ x)\big) \mid \overline{a_1} \mid \overline{c_1} \mid \cdots \mid \overline{a_n} \mid \overline{c_n} \mid (a_1.\cdots.a_n.P) \mid (c_1.\cdots.c_n.Q)$

$\rightarrow !\big(Leader(x).(eval\ x)\big) \mid P \mid Q$

$\rightarrow !\big(Leader(x).(eval\ x)\big) \mid \overline{Agent_1}[Actions(b_1)] \mid \cdots \mid \overline{Agent_n}[Actions(b_n)]$
$\quad \mid \overline{Agent_1}[Actions(d_1)] \mid \cdots \mid \overline{Agent_n}[Actions(d_n)]$

In above operations, agents have performed the actions a_i, c_i and x_i. After above operations, the leader sends $Actions(b_i)$ and $Actions(d_i)$ to the agent $Agent_i$. The agent $Agent_i$ performs the action b_i and d_i as before operations. The results show as follows: the actions A, C and X are performed at first, then B and D are performed. It means that the operations satisfies the conditions for the actions.

The other actions can be performed as before. We abbreviate the detail because of limited space.

7 Conclusion

We have shown in this paper that the protocol for agents to collaborate each other and the description of agent's actions including comunications in extended π-calculus in order to share actions.

We have focused on intentions of agents to define the protocol. In the protocol, an agent builds a group in which agents have the same intentions or the similar intentions. We have formalized actions based on extended π-calculus and have shown the operational semantics of our proposed actions.

Some specific issues (to find common process, communication performance, to find the most suitable number of members $\cdots$) have been left out and will be described in a forthcoming paper.

References

[CLS] P. Cohen, H. Levesque, and I. Smith. On team formation.

[FG99] Jacques Ferber and Olivier Gutknecht. Operational semantics of multi-agent organizations. In *Agent Theories, Architectures, and Languages*, pages 205–217, 1999.

[HCY99] S. Hayden, C. Carrick, and Q. Yang. Architectural design patterns for multiagent coordination, 1999.

[KM01] Yoshinobu Kawabe and Ken Mano. Executing coded π-calculus processes. In *Proceedings of the ACIS 2nd International Conference on Softwre Engineering, Artificial Intelligence, Networking & Parallel/Distributed computing(SNPD'01)*, pages 25–32, August 2001.

[Mil91] R. Milner. Polyadic π-calculus:a Tutorial. LFCS Report Series ECS-LFCS-91-180, Laboratory for Foundation of Computer Science, University of Edinburgh, 1991.

[MUN98] MUNINDAR P. SINGH. Applying the Mu-Calculus in Planning and Reasoning about Action. *Journal of Logic and Computation*, 8:425–445, 1998.

[Nak99] Yasuo Nakayama. Communication and attitude change. In *The Second International Conference on Cognitive Science and The 16th Annual Meeting of the Japanese Cognitive Science Society Joint Conference (ICCS/JCSS99)*, 1999.