Applied Mathematical Sciences, Vol. 2, 2008, no. 7, 305 - 332

A Multi-Agent System Formalization in Mobile Environments

Jianwen Chen

IBM Australia, Level 1, 55 Christie St
St Leonards Sydney, NSW 2065, Australia
jchen@au1.ibm.com

Abstract

This paper models and formalizes a mobile logic programming multi-agent system (MLPMAS) for mobile environments. Such a system consists of a number of agents connected via wire or wireless communication channels. Agents communicate with each other via passing answer sets obtained by updating the information received from connected agents with their own private information. The interactions between agents are also modeled in this formalization. Based on this model, knowledge based transaction can be studied in such a mobile multi-agent system. In addition, knowledge transaction is formally defined and modeled for these mobile multi-agent systems.

1. Introduction

Comparing to stationary environments, mobile environments have a few specific properties such as mobility and disconnection. The communication channels can be wire or wireless in mobile environments. The features of mobile environment have presented new challenges for researchers. We believe that research on multi-agent system and knowledge transaction in mobile environments is critical because this will help us to find a way to significantly improve current development of multi-agent system and mobile system.

However, there seems to be a separation between multi-agent systems and the intelligent agents community on one side, and the mobile system community on the other side [10, 14, 19, 21]. So far no formalization and model has been presented for multi-agent system in mobile environments and no study has been conducted for

knowledge transaction in mobile multi-agent system. On mobile system community side, currently the work in paper [3, 4, 13] has introduce calculus to describe the movement of processes and devices in mobile ambient, and the work in [2, 6, 11] has presented a number of Java packages, libraries, and frameworks to implement functionalities for programming distributed and mobile systems. The approaches above have following disadvantage: (1) They are not suitable for knowledge oriented processing in mobile environments; (2) They have no declarative semantics. They are low level algorithms for "how to do" and have no high level "what to do" intelligent functionality. (3) The details of transaction can't be specified. On multi-agent and intelligent agent community side, a lot of framework/model have been developed for problem solving, knowledge representation and reasoning such as stable model/answer set, SMODEL, DLV and XSB model in paper [7, 15, 18]. These models are knowledge oriented with declarative semantics, and their specification language can specified the details of knowledge transaction, but these models are only discussed and limited in stationary environments, and haven't be extended to mobile environments. The motivation of our work is to propose a new framework/model with the following features: (1) It is knowledge based with declarative semantics; (2) It can specify details of knowledge transaction; (3) It can be used in mobile environments.

In this paper we present a formalism and definition for a mobile logic programming multi-agent system (MLPMAS). Such a system is very useful for modeling decision-problems in mobile environments, not just the solutions of the problem but also the evolution of the beliefs of and the interactions between the agents in mobile environments. With respect to previous work, we can characterize major features in our approach as follows: (1) We use extending logic programs and answer set semantics in our formalization so our model is knowledge oriented and has declarative semantics inherited from logic programming; (2) It can specify details of knowledge transaction. By using knowledge rules, input, output and knowledge base itself can be specified; (3) Our model can be used to process knowledge transaction in mobile environment.

The rest of this paper is organized as follows. In section 2 presents and formalizes a mobile logic programming multi-agent system. Section 3 illustrates a study case to demonstrate how to specify a MLPMAS system in a particular problem domain. Section 4 formally defines and models the knowledge transaction in formalized MLPMAS systems. Then, three case studies are presented to address how to process a knowledge transaction in MLPMAS systems. Finally, section 5 summarizes the work in this paper.

2. A Mobile Logic Programming Multi-Agent System Formalization

As we discussed in our paper [5] that extended logic programming is a suitable tool for knowledge representation and study in mobile environments. In this paper, extended logic programming is employed as a mathematical tool to formalize a mobile logic programming multi-agent system in mobile environments. The definition and formalization of MLPMAS systems are based on a three-layer environment model presented in Figure 1.1. In this model it is assumed that every Mobile Host (MH) has its own Knowledge Base (KB) and Intelligent Agent (A11, A12, A13, A21, A22, A23), and every MSS has a knowledge base and agent residing on it. It is also assumed that MSS1 and MSS2 represent different MSSs in different geographic areas. In the Home Server (HS) level, there is a knowledge base that possesses a set of rules, and there is an agent residing on it. Every intelligent agent on a MH will work on behalf of that MH, and all the agents in the same geographic area (i.e. controlled by the same HS) will negotiate, communicate, and cooperate with each other to achieve the goal for themselves and their systems.
A MLPMAS system is shown in Figure 1.2. A mobile logic programming multi-agent

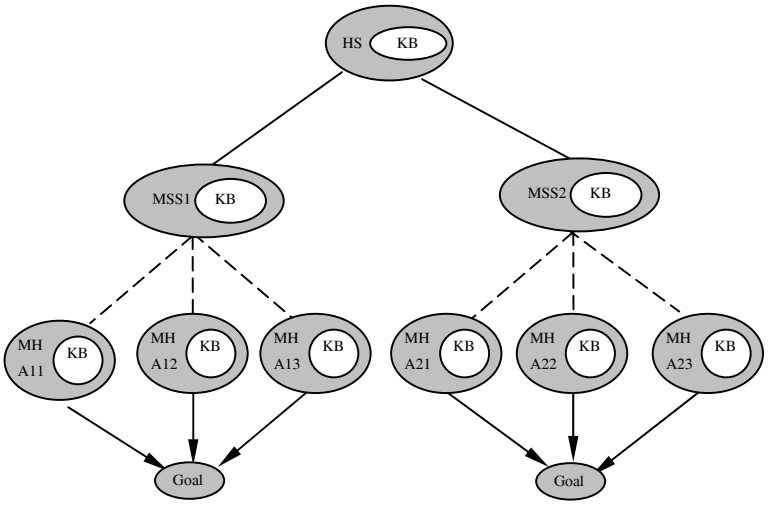

Figure 1.1: Environment model.

system includes the three levels, MH, MSS and HS, with the local knowledge base located on each level. The system consists of a set of agents in mobile environments. The agent resides on the MH, MSS and HS levels respectively, connected through communication channels. The agent on each level contains its own logic program representing its local information and reasoning method. Agents use information

received from their incoming channels as input for their reasoning, where the received information may be overridden by other concerns represented in their programs. Agents produce output to their outgoing communication channels.

Definition 1: *A mobile logic programming multi-agent system*, or *MLPMAS*, is a pair

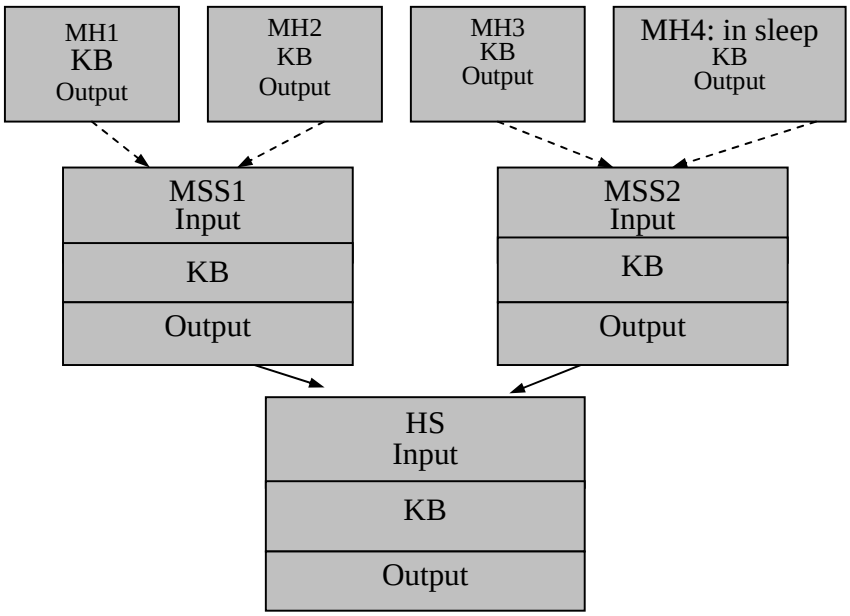

Figure 1.2: A mobile logic programming multi-agent system.

$F = <A, C>$, where A is a set of agents: $A = A_{MH} \cup A_{MSS} \cup A_{HS}$, and $C \subseteq A \times A$ is a reflexive relation representing the communication channels between agents. For any $a_{1,} a_2 \in A$, if $<a_1, a_2> \in C$, then we say agents a_1 and a_2 have a *communication channel*. For each agent $a \in A$, there is an associated extended logic program *LocalKB(a),* which represents agent *a*'s *local knowledge base*.

Example 1: Definition of the MLPMAS system above can be explained by Example 1, in which an investor agent resides on a MH, a group agent resides on a MSS, and a fund manager agent resides on a HS. The investor agent manages the local knowledge base and provides output to the group agent on behalf of the MH. The group agent collects information from all involved investor agents, manages the local knowledge base on the MSS and sends output to the fund manager agent. The fund manager agent collects information from all involved group agents, does the investment

decision and manages the local knowledge base on the HS. The investor agent, group agent and fund manager agent are represented by $a_{MH,}$ a_{MS}S and a_{HS} respectively. Given a mobile logic programming multi-agent system $F = < A, C >$, with four mobile hosts MH1, MH2, MH3 and MH4,, the investor agent resides on each MH:

$$A_{MH} = \{a_{MH1}, a_{MH2}, a_{MH3}, a_{MH4}\}$$

There are two mobile support stations MSS1 and MSS2, with a group agent residing on each:

$$A_{MSS} = \{a_{MSS1}, a_{MSS2}\}$$

There is one home server HS, with a resident fund manager agent :

$$A_{HS} = \{a_{HS}\}$$

MH1 and MH2 are in the same geographic location as MSS1; MH3 and MH4 are in the same geographic location as MSS2. There is a wireless communication channel between the MHs and their MSS:

$$< a_{MH1}, a_{MSS1} > \in C,\ < a_{MH2}, a_{MSS1} > \in C,$$
$$< a_{MH3}, a_{MSS2} > \in C,\ < a_{MH4}, a_{MSS2} > \in C$$

It is a wire communication channel between the MSS and the HS:

$$< a_{MSS1}, a_{HS} > \in C,\ < a_{MSS2}, a_{HS} > \in C$$

It is assumed there is a communication channel between the MH and the HS as well:

$$< a_{MH1}, a_{HS} > \in C,\ < a_{MH2}, a_{HS} > \in C,$$
$$< a_{MH3}, a_{HS} > \in C,\ < a_{MH4}, a_{HS} > \in C$$

As mentioned earlier, each agent is associated with an extended logic program of its local knowledge base. If there is no answer set, it means the local knowledge base is not well designed. If there are multiple answer sets, each answer set represents a possible knowledge state.

The input and output of agents are defined in an MLPMAS as follows.

Definition 2: Let $F = < A, C >$ be a MLPMAS, where $A = A_{MH} \cup A_{MSS} \cup A_{HS}$. At the MH, MSS or HS level, for $\forall a \in A$, we have two parts to the input: message input and knowledge input, denoted by $MessageInput(a, X)$ and $KnowledgeInput(a, Y)$ respectively. That is,

$$Input(a) = < MessageInput(a, X), KnowledgeInput(a, Y) >$$

Here, $X \subseteq A, Y \subseteq A$, X, Y are subsets of A. Agent a collects message input from agents in X, and collects knowledge input from agents in Y, where

$$\forall b \in X, \text{ we have } < a, b > \in C, \text{ or } < b, a > \in C \text{ and}$$
$$\forall b' \in Y, \text{ we have } < a, b' > \in C, \text{ or } < b', a > \in C.$$

I.e. there is a communication channel between agent *a* and agent *b*, and agent *a* and agent b' respectively.

Message input is the information that an agent sends to another agent for the purpose of communication,. such as one agent informing another agent that it will move into another MSS geographic area. This information will not cause any influence to the other agent's local knowledge base. However, knowledge input is the information produced by the other agent's local knowledge base, and will be taken into the agent's local knowledge base, i.e. the answer set of a logic program.

For $\forall a \in A$, we have two parts to the output, message output and knowledge output, denoted by $MessageOutput(a,X)$ and $KnowledgeOutput(a,Y)$ respectively. That is

$$Output(a) = < MessageOutput(a,X), KnowledgeOutput(a,Y) >$$

here $X \subseteq A, Y \subseteq A$. Agent a sends message output to agents in X, and sends knowledge output to agents in Y.

Message output is information output for communication purposes, this information will not cause any influence to the other agent's local knowledge base. While knowledge output is the information that is produced by the agent's local knowledge base and will have an impact for the other agent's knowledge base.

Definition 3: The knowledge input and output in MLPMAS systems are defined on the MH level as follows:
There is no input for MHs at the MH level because this is the first level in MLPMAS systems, i.e.

$$KnowledgeInput(a_{MH}, Y) = \phi \qquad (1)$$

The knowledge output can be derived from the equation:

$$KnowledgeOutput(a_{MH}, a_{MSS})$$
$$= \text{an answer set of } [LocalKB(a_{MH}) \cup KnowledgeInput(a_{MH}, Y)] \qquad (2)$$

I.e. knowledge output is an answer set of the program formed by the local logic program of agent a_{MH} with extension of knowledge input from Y for agent a_{MH}.

$LocalKB(a)$ is an extended logic program as defined in Definition 1. $KnowledgeInput(a,Y)$ is a set of facts (beliefs). Note that $LocalKB(a_{MH}) \cup KnowledgeInput(a_{MH}, Y)$ is viewed as a new logic program while fact $e \in KnowledgeInput(a,Y)$ is treated as a rule $e \leftarrow$.

Definition 4: The knowledge input and output in MLPMAS systems on the MSS level are defined as follows:
The knowledge input can be derived from the equation:

$$KnowledgeInput(a_{MSS}, Y) = cons(\bigcup_{a_{MH} \in Y} KnowledgeOutput(a_{MH}, a_{MSS}), S_F) \quad (3)$$

where *cons(X)* returns a maximal consistent subset of knowledge output from Y to agent a_{MSS} with respect to the select function S_F.

S_F is the *selection function* of the system. For knowledge output, $\bigcup_{b \in Y} KnowledgeOutput(b, a)$ may be inconsistent. S_F is introduced to solve such inconsistency by taking proper preference in the domain. Note that S_F is domain dependent. It can be a special logic programming rule for a specific problem domain.

The knowledge output can be derived from the equation:

$$KnowledgeOutput(a_{MSS}, a_{HS}) = \text{an answer set of } [LocalKB(a_{MSS}) \cup KnowledgeInput(a_{MSS}, Y)] \quad (4)$$

I.e. knowledge output is an answer set of the program formed by the local logic program of agent a_{MSS} with extending of knowledge input of agent a_{MSS}.

Definition 5: *Knowledge input and output* in MLPMAS systems on the HS level is defined as follows:
The knowledge input can be derived from the equation:

$$KnowledgeInput(a_{HS}, Y) = cons(\bigcup_{a_{MSS} \in Y} KnowledgeOutput(a_{MSS}, a_{HS}), S_F) \quad (5)$$

I.e. knowledge input of a_{HS} is the maximal consistent subset of knowledge output from Y to agent a_{HS} with respect to the select function S_F.

The knowledge output can be derived from the equation:

$$KnowledgeOutput(a_{HS}) = \text{an answer set of } [LocalKB(a_{HS}) \cup KnowledgeInput(a_{HS}, Y)] \quad (6)$$

I.e. knowledge output is an answer set of the program formed by the local logic program of agent a_{HS} with extending of knowledge input of agent a_{HS}.

Example 2: (Example 1 continued). Given the same scenario described in Example 1. To invest share1, MH1, MH2, MH3 and MH4 are involved investors. It is shown how input and output are derived on the MSS level based on their definition above.

On the MSS level, the group agent has input from involved investor agents on behalf of MH. Group agent a_{MSS1} on MSS1 has the input as below according to Definition 2:

$$Input(a_{MSS1}) = < MessageInput(a_{MSS1}, X), KnowledgeInput(a_{MSS1}, Y) >$$

Here $X = \{a_{MH1}, a_{MH2}\}$ is a subset of A, including all investor agents who have message input for the group agent on MSS1. $Y = \{a_{MH1}, a_{MH2}\}$ is a subset of A, including all investor agents who have knowledge input for the agent on MSS1.

According to equation (3), knowledge input on MSS1 can be derived as below:

$$KnowledgeInput(a_{MSS1}, Y)$$
$$= cons(\bigcup_{a_{MH} \in Y} KnowledgeOutput(a_{MH}, a_{MSS1}), S_F)$$
$$= cons(KnowledgeOutput(a_{MH1,} a_{MSS1}) \cup KnowledgeOutput(a_{MH2,} a_{MSS1}), S_F)$$

The group agent on MSS1 has message output and knowledge output according to Definition 2:

$$Output(a_{MSS1}) = < MessageOutput(a_{MSS1}, X), KnowledgeOutput(a_{MSS1}, Y) >$$

Here $X = \{a_{HS}\}$, $Y = \{a_{HS}\}$ are subsets of A. The agent a_{MSS1} sends message output to the agent a_{HS} in X and sends knowledge output to the agent a_{HS} in Y.

The knowledge output can be derived from equation (4):

$$KnowledgeOutput(a_{MSS1}, a_{HS})$$
$$= \text{an answer set of } [LocalKB(a_{MSS1}) \cup KnowledgeInput(a_{MSS1}, Y)]$$

Thus, an agent sends its full set of belief outputs over all outgoing communication channels. On the other hand, an agent receives as input, the beliefs of all agents connected to its incoming channels.

3. A Case Study for MLPMAS Systems

Study case 1 is presented to illustrate how to specify a MLPMAS system in a specific problem domain based on the formalization above.

Study Case 1: This case study is presented in a specific investment problem domain. The MLPMAS system has shown in Figure 1.2. At the MH level, there are MH1,

MH2, MH3 and MH4. MH1 and MH2 are in the cell of MSS1, MH3 and MH4 are in the cell of MSS2. MSS1 and MSS2 are connected to the same HS. At the MH level, each MH has a local knowledge base that includes a set of investment rules, and has an investor agent residing on it. At the MSS level, a MSS has it's own knowledge base, and accepts the input from MHs and produces the output based on this input and it's own belief. The HS has it's own local knowledge base. It accepts input from the MSS level and makes the investment decision. For the initial status, it is assumed that MH1 and MH2 are all alive when a transaction is processed in a MSS1 cell. In a MSS2 cell, the MH3 is alive, while MH4 is in sleep at the moment the HS is requesting the transaction information from all related MH agents. The HS will need information from MH4 by the time it does the decision making. The logic programming rules for local knowledge bases on MH, MSS, and HS is specified when input and output of agents are discussed on every level. Note that, all the equations and definitions quoted in this case study are those formalized in MLPMAS systems in section 2 of this paper.

MH Level:

Local knowledge base at MH level:

The logic programming rules of local knowledge base on MH level are as follow.

The local knowledge base of MH1 has rules *r1-r3* related to the share1 investment:

$$\begin{Bmatrix} r1:\ holds(profit(share1)) \leftarrow \\ r2:\ holds(risk(share1)) \leftarrow \\ r3:\ \neg holds(cost(share1)) \leftarrow \end{Bmatrix}$$

The local knowledge base of MH2 has rules *r4-r6* related to the share1 investment:

$$\begin{Bmatrix} r4:\ holds(profit(share1)) \leftarrow \\ r5:\ \neg holds(risk(share1)) \leftarrow \\ r6:\ \neg holds(cost(share1)) \leftarrow \end{Bmatrix}$$

The local knowledge base of MH3 has rules *r7-r9* related to the share1 investment:

$$\begin{Bmatrix} r7:\ holds(profit(share1)) \leftarrow \\ r8:\ \neg holds(risk(share1)) \leftarrow \\ r9:\ \neg holds(cost(share1)) \leftarrow \end{Bmatrix}$$

The local knowledge base of MH4 has rules *r10-r12* related to the share1 investment:

$$\begin{Bmatrix} r10:\ holds(profit(share1)) \leftarrow \\ r11:\ \neg holds(risk(share1)) \leftarrow \\ r12:\ \neg holds(cost(share1)) \leftarrow \end{Bmatrix}$$

Input at MH level:

According to equation (2), there is no input for MHs at the MH level because this is the first level in MLPMAS systems. We have

$$KnowledgeInput(a_{MH}, Y) = \phi$$

i.e.

$$KnowledgeInput(a_{MH1}) = \phi,\ KnowledgeInput(a_{MH2}) = \phi,$$
$$KnowledgeInput(a_{MH3}) = \phi,\ KnowledgeInput(a_{MH4}) = \phi$$

Output at MH level:

According to equation (2), we have

$$KnowledgeOutput(a_{MH}, a_{MSS})$$
$$= \text{an answer set of } [LocalKB(a_{MH}) \cup KnowledgeInput(a_{MH}, Y)]$$
$$= \text{an answer set of } [LocalKB(a_{MH}) \cup \phi]$$
$$= \text{an answer set of } [LocalKB(a_{MH})]$$

I.e. at the MH level, the knowledge output is an answer set of local knowledge base. Based on rules *r1-r9* of the local knowledge base of MHs, the knowledge outputs are derived as below on MH1, MH2, MH3 and MH4.

$$KnowledgeOutput(a_{MH1}, a_{MSS1}) = \{profit(share1), risk(share1), \neg cost(share1)\}$$

i.e. it is high profit, high risk and low cost to invest share1 on MH1.

$$KnowledgeOutput(a_{MH2}, a_{MSS1}) = \{profit(share1), \neg risk(share1), \neg cost(share1)\}$$

i.e. it is high profit, low risk and low cost to invest share1 on MH2.

$$KnowledgeOutput(a_{MH3}, a_{MSS2}) = \{profit(share1), \neg risk(share1), \neg cost(share1)\}$$

i.e. it is high profit, low risk and low cost to invest share1 on MH3. MH4 is at sleep at the moment the information is retrieved from it.

MSS level:

Local knowledge base at MSS level:

On MSS1, we have rule *r13* related to this investment in its knowledge base:

$$\{r13:\ holds(info-requested(HS, MHi)) \leftarrow holds(slept(MHi))\}$$

On MSS2, we have rule *r14* similar to *r13* related to this investment in its knowledge base:

$$\{r14:\ holds(info-requested(HS, MHi)) \leftarrow holds(slept(MHi))\}$$

Rules *r13* and *r14* denote that if MHi is in sleep at the time the HS agent requests transaction information from MHs, HS will request information from MHi when HS does the decision making for the transaction.

Input at MSS level:
At MSS level, according to Definition 2 of MLPMAS systems, input of MSS1 agent equals:

$$Input(a_{MSS1}) =< MessageInput(a_{MSS1}, X), KnowledgeInput(a_{MSS1}, Y) >$$

where $X=\{a_{MH1}, a_{MH2}\}$ is a subset of A, including all investor agents who have message input for the group agent on MSS1. $Y=\{a_{MH1}, a_{MH2}\}$ is a subset of A, including all investor agents who have knowledge input for the group agent on MSS1.

According to equation (3), knowledge input on MSS1 is as below:

$$\begin{aligned}
&KnowledgeInput(a_{MSS1}, Y)\\
&= cons(\cup_{a_{MH} \in Y} KnowledgeOutput(a_{MH}, a_{MSS1}), S_F)\\
&= cons(KnowledgeOutput(a_{MH1,} a_{MSS1}) \cup KnowledgeOutput(a_{MH2}, a_{MSS1}), S_F)\\
&= cons(\{profit(share1), risk(share1), \neg cost(share1)\} \cup\\
&\{profit(share1), \neg risk(share1), \neg cost(share1)\}, S_F)
\end{aligned}$$

For agent a_{MSS1}, $risk(share1)$ is a belief in output of a_{MH1}, while $\neg risk(share1)$ is a belief in output of a_{MH2}, they are inconsistent. Here it is assumed that the selection function S_F takes the positive atom as the higher preference for investment risk, in which case, $risk(share1)$ will become the input of a_{MSS1}.

Therefore, the knowledge input of a_{MSS1} is:

$$KnowledgeInput(a_{MSS1,} Y) = \{profit(share1), risk(share1), \neg cost(share1)\}$$

Without considering the selection function the knowledge input of a_{MSS1} will be different:

$$KnowledgeInput(a_{MSS1,} Y) = \{profit(share1), \neg cost(share1)\}$$

The different knowledge input is derived by considering the selection function in a specific problem domain, therefore a different answer set is derived for decision making due to the selection function.

Input to MSS2 agent equals:

$$Input(a_{MSS2}) =< MessageInput(a_{MSS2}, X), KnowledgeInput(a_{MSS2}, Y) >$$

where $X=\{a_{MH3}, a_{MH4}\}$ is a subset of A, including all investor agents who have message input for the group agent on MSS2. $Y=\{a_{MH3}, a_{MH4}\}$ is a subset of A,

including all investor agents who have knowledge input for the group agent on MSS2.

The MH4 is at sleep at the moment, so knowledge input of MSS2 agent equals:

$$KnowledgeInput(a_{MSS2,}Y)$$
$$= cons(\bigcup_{a_{MH}\in Y} KnowledgeOutput(a_{MH}, a_{MSS2}), S_F)$$
$$= cons(KnowledgeOutput(a_{MH3,}a_{MSS2}) \cup KnowledgeOutput(a_{MH4,}a_{MSS2}), S_F)$$
$$= cons(\{profit(share1), \neg risk(share1), \neg cost(share1)\} \cup \phi, S_F)$$
$$= \{profit(share1), \neg risk(share1), \neg cost(share1)\}$$

Output at MSS level:

Based on Definition 2 of a MLPMAS system, there are two parts to the output of the MSS, message output and knowledge output.
According to equation (4), the knowledge output can be derived:

$$KnowledgeOutput(a_{MSS}, a_{HS})$$
$$= \text{an answer set of } [LocalKB(a_{MSS}) \cup KnowledgeInput(a_{MSS}, Y)]$$

The knowledge output of MSS1is derived as below:

$$KnowledgeOutput(a_{MSS1}, a_{HS})$$
$$= \{profit(share1), risk(share1), \neg cost(share1)\}$$

The knowledge output of MSS2 is derived as below:

$$KnowledgeOutput(a_{MSS2}, a_{HS})$$
$$= \{profit(share1), \neg risk(share1), \neg cost(share1), info-requested(HS, MH4)\}$$

A new belief, *info-requested(HS, MH4),* is added to the answer set on MSS2 because of rule *r14* in the local knowledge base of MSS2.

HS level:

Local knowledge base at HS level:
There are rules *r15-r21* in the local knowledge base of HS.

$$\begin{cases} r15:\ holds(invest(share1)) \leftarrow holds(profit(share1)), \neg holds(risk(share1)), \\ \neg holds(cost(share1)), holds(info-get(MHi), res(request-info(MHi))) \\ r16:\ \neg holds(invest(share1)) \leftarrow holds(risk(share1)) \\ r17:\ \neg holds(invest(share1)) \leftarrow holds(cost(share1)) \\ r18:\ \neg holds(risk(share1)) \leftarrow notholds(risk(share1)) \\ r19:\ \neg holds(cost(share1)) \leftarrow notholds(cost(share1)) \\ r20:\ \neg holds(invest(share1)) \leftarrow holds(info-requested(HS, MHi)), \\ \neg holds(info-get(MHi), res(request-info(MHi))), \\ r21:\ \neg holds(info-get(MHi)) \leftarrow notholds(info-get(MHi)), \\ holds(info-requested(MHi)), holds(timeout(MHi)) \end{cases}$$

Rule *r15* denotes that if it is high profit, low risk, low cost to invest share1, and the HS gets requested information from every sleeping MHi, the decision to invest share1 will be made. Rules *r16, r17* and *r20* denote that if share1 is high risk or high cost on any MHi, or information is not available from every sleeping MHi, then the HS will make the decision that share1 will not be invested. Rules *r18* and *r19* denote that if share1 hasn't been specified to be high risk or high cost for any MHi, then it is considered to be low risk or low cost. Rule *r21* denotes that if the HS hasn't got requested information from sleeping MHi before time is out, then the HS will assume no information is available from MHi.

Input at HS level:

On the HS level, based on Definition 2 of a MLPMAS system, input of an HS agent equals:

$$Input(a_{HS}) = < MessageInput(a_{HS}, X), KnowledgeInput(a_{HS}, Y) >$$

Where $X = \{a_{MSS1}, a_{MSS2}\}$ is a subset of A, including all involved agents who have message information input for the agent on the HS. $Y = \{a_{MSS1}, a_{MSS2}\}$ is a subset of A, including all involved agents who have knowledge input for the agent on the HS.

Based on equation (5), knowledge input of a HS agent equals:

$$KnowledgeInput(a_{HS}, Y)$$

$$= cons(\bigcup_{a_{MSS} \in Y} KnowledgeOutput(a_{MSS}, a_{HS}), S_F)$$

$$= cons(KnowledgeOutput(a_{MSS1}, a_{HS}) \cup KnowledgeOutput(a_{MSS2}, a_{HS}), S_F)$$

$$= cons(\{profit(share1), risk(share1), \neg cost(share1)\} \cup$$

$$\{profit(share1), \neg risk(share1), \neg cost(share1), info-requested(HS, MH4)\}, S_F)$$

$$= \{profit(share1), risk(share1), \neg cost(share1), info-requested(HS, MH4)\}$$

$risk(share1)$ is a belief of input on the HS with consideration of the selection function.

Output at HS level:

Based on Definition 2 of a MLPMAS system, the output of a HS agent has two parts, message output and knowledge output. The knowledge output is derived as below according to equation (6):

$$KnowledgeOutput(a_{HS})$$

$$= \text{an answer set of } [LocalKB(a_{HS}) \cup KnowledgeInput(a_{HS}, Y)]$$

$$= \text{an answer set of } [LocalKB(a_{HS}) \cup \{profit(share1), risk(share1), \neg cost(share1),$$

$$info-requested(HS, MH4)\}]$$

$$= \{\neg invest(share1), profit(share1), risk(shareq), \neg cost(share1)\}$$

Where $[LocalKB(a_{HS}) \cup KnowledgeInput(a_{HS}, Y)] \models \{\neg invest(share1)\}$.

The $risk(share1)$ is a part of the knowledge input of the HS, and according to rule *r10* of the knowledge base on the HS, it is known that share1 can not be invested if there is a high risk to invest it. From the output of HS, the fund manager agent on HS makes the decision that share1 won't be invested. In this case, no matter what information is received from MH4, HS will make the decision that share1 can't be invested, otherwise the information from MH4 will be considered as input to HS, and the decision making will be made based on all inputs and the local knowledge base itself. It is also shown that the selection function will impact the decision making. If no selection function is used in this example, $risk(share1)$ will not be part of the input to the HS agent, therefore share1 may be invested.

After the HS has made the decision that share1 will not be invested, the transaction decision will be sent to the MSS, and all involved MHs will be notified by broadcasts from the MSS.

4. Transaction Processing in MLPMAS Systems

4.1 Defining Knowledge Transaction in MLPMAS Systems

Based on Definition 1 of a MLPMAS system presented above, A Mobile Logic Programming Multi-Agent System (MLPMAS) is a pair $F = <A, C>$, where A is a set of agents: $A = A_{MH} \cup A_{MSS} \cup A_{HS}$, and $C \subseteq A \times A$ is a reflexive relation representing the communication channels between agents. Recall the formalization in our paper [5]: the knowledge transaction model on domain D is defined as a pair $\sum = (R, I(D))$ in mobile environments, where R is a set of generic knowledge transaction rules, and $I(D)$ is a finite set of initial facts and rules with respect to the problem domain D. Thus knowledge transaction can be defined in mobile logic programming multi-agent systems as follows:

Definition 6: In mobile logic programming multi-agent systems, the knowledge transaction can be defined as $T = < A, C, R, I(D) >$, where A is a set of agents: $A = A_{MH} \cup A_{MSS} \cup A_{HS}$, $C \subseteq A \times A$ is a reflexive relation representing the communication channels between agents, R is a set of generic knowledge transaction processing rules, and $I(D)$ is a finite set of initial facts and rules with respect to problem domain D.

4.2 Transaction Processing Language in Mobile Environments

In our paper [5], a language $\mathcal{L}$ is formalized to process knowledge based transaction in mobile environments. Firstly recall the syntax of language $\mathcal{L}$ as follows. A language $\mathcal{L}$ contains variables of three sorts: situation variable *s*, fluent variable *f*, and action variable *a*. It has one predicate *holds(f, s)*, where *f* is a fluent function, and *s* is a situation. The predicate *holds(f, s)* means that fluent *f* is true at situation *s*. $\mathcal{L}$ also has a resulting function *res(a, s)*, which denotes a situation resulting from situation *s* by performing action *a*. To begin with, $\mathcal{L}$ has one initial situation constant S_0.

Based on language $\mathcal{L}$, the following action functions are defined in MLPMAS systems:
Write (MHi): denotes MHi has an update transaction request.
Submit (MSSi): denotes MSSi submits the transaction request to HS.
acquire-lockc(MHi): denotes MHi acquires lock for transaction updating.
do-trans(HS): denotes HS starts the transaction.

request-info(MHi): denotes MHi is requested to provide its information.
*cal-output(MHi):*denotes MHi calculates its output.
sent-info(MHi): denotes MHi sends its information as output.
cal-input(MSSi): denotes MSSi calculates its input.
cal-output(MSSi): denotes MSSi calculates its output.
cal-input(HS): denotes HS calculates its input.
cal-output(HS): denotes HS calculates its output.
make-decision (HS): denotes HS makes the decision for the transaction.
abort-trans (HS): denotes HS aborts the transaction.
notice-abort (HS, MSSi): denotes HS notices MSSi transaction abort.
broadcast (MSSi, MHi): denotes MSSi broadcasts transaction status to MHi.
update-knowledge (MHi): denotes MHi updates its knowledge base.

The following fluent functions are defined in MLPMAS systems:
update-requested(MHi): denotes MHi has an update transaction request.
registered(MSSi, MHi): denotes MHi has registered in MSSi cell.
trans-submitted (MSSi): denotes MSSi has submitted the transaction request to HS.
locked (MHi): denotes MHi has got the lock for the transaction.
trans-start (HS): denotes transaction start on HS.
info-requested (MHi): denotes MHi has been requested to provide its information.
output-derived (MHi): denotes MHi has derived its output.
output-sent(MHi): denotes MHi has sent its output to MSSi.
input-derived(MSSi): denotes MSSi has derived its input.
output-derived(MSSi): denotes MSSi has derived its output.
input-derived(HS): denotes HS has derived its input.
output-derived(HS): denotes HS has derived its output.
decision-made(HS): denotes HS has made the transaction decision.
trans-aborted(HS): denotes HS has aborted the transaction.
knowledge-update(HS): denotes HS has updated its knowledge base.
abort-noticed(HS, MSSi): denotes HS has sent the transaction abort notice to MSSi.
abort-broadcasting(MSSi, MHi): denotes MSSi has broadcast the transaction abort to MHi.
knowledge-update(MHi): denotes MHi has updated its knowledge base.

4.3 Transaction Processing Rules in MLPMAS Systems

In our paper [5], a set of rules are specified and imposed to formalize a knowledge transaction processing model in mobile environments, which models all transaction processing activities, requests, results and constraints on MH, MSS, and HS levels.

Firstly we recall some of those transaction rules, which will be used in the study cases in later sections of this paper.

The following rules are imposed in paper [5] to specify knowledge transaction processing in mobile environments:

At the register stage of a transaction, when MH moves into MSS cell, it is registered. The rule for this is:

t1: holds(registered(MSS, MH), res(move(MSS, MH), s)) ←

After an action *move(MSS, MH)* happens, *res(move(MSS, MH), s)* becomes the current situation, and *registered(MSS, MH)* is true.

Then MH requests to start an update transaction. The rule is:

t2: holds(update-requested(MH), res(write(MH), s)) ←

After an action *write(MH),* i.e., the MH submits a write request, *res(write(MH), s)* becomes the current situation, and *update-requested(MH)* becomes true.

After the MH requests a query or update, the MSS submits this transaction request to the HS on behalf of the MH. If it is a write request, the lock needs to be acquired firstly to submit this transaction.

t3: holds(trans-submitted(MSS), res(submit(MSS), s)) ← holds(locked(MH), s),holds(update-requested(MH), s))

If both *update-required(MH)* and *locked(MH)* are true, action *submit(MSS)* will happen and *trans-submitted(MSS)* then becomes true.

After the MH requests a query or update transaction and the MSS submits this transaction request to the HS, the HS starts the transaction. The rule is:

t4: holds(trans-start(HS), res(do-trans(HS), s)) ← holds(trans-submitted(MSS), s)

After transaction request is submitted by the MSS, i.e., *trans-submitted(MSS)* is true, action *do-trans(HS)* happens, and *trans-start(HS)* then becomes true, the transaction starts.

If the decision is made that the transaction will be aborted at the HS level, HS updates its knowledge base to reflect this accordingly. The rules to denote this are *t5* and *t6:*

t5: holds(trans-aborted(HS),res(abort-trans (HS),s))←holds(decesion-made(HS),s)

t6: ¬holds(knowledge-update(HS),s) ← holds(trans-aborted(HS),s)

The HS sends transaction abort notice to the MSS. The rule is:

$$t7:\ holds(abort-noticed(HS, MSS), res(notice-abort(HS, MSS), s)) \leftarrow holds(trans-aborted(HS), s)$$

The MSS broadcasts the transaction abort information to involved MHs. The rule is:

$$t8:\ holds(abort-broadcasting(MSS,MH),res(broadcast(MSS,MH),s)) \leftarrow holds(abort-noticed(HS,MSS),s)$$

The MHs update their local knowledge base accordingly, the rule reflecting this is:

$$t9:\ \neg holds(knowledge-update(MH),res(update-knowledg(MH),s)) \leftarrow holds(abort-broadcasting(MSS,MH),s)$$

A set of new rules are imposed in this paper to specify the activities and capture the features of MLPMAS systems in mobile environments. The rules are specified as follows. Note that the equations mentioned below are those presented in section 2 for the formalization of a MLPMAS system.

After the MH agent is requested by the HS to provide problem domain related information to the HS, the MH agent calculates the MH's output based on equation (2), then sends its output to the MSS if the MH is not at sleep. The rules specifying this are:

$$t10:\ holds(output-derived(MHi),res(cal-output(MHi),s)) \leftarrow holds(info-requested(MHi),s)$$

$$t11:\ holds(output-sent(MHi),res(sent-output(MHi),s_0)) \leftarrow holds(output-derived(MHi),s_0)$$

Rules *t10* and *t11* denote: If the MH agent is requested to provide its information, action *cal-output(MH)* is taken to calculate the output, *output-derived(MH)* becomes true. Then output is sent to the MSS by action *sent-output(MH),* ,i.e., *output-sent(MH)* becomes true.

The MSS agent collects information from all related MHs and calculates the input based on the equation (3). The rule is:

$$t12:\ holds(input-derived(MSSi),res(cal-input(MSSi),s)) \leftarrow holds(output-sent(MHi),s)$$

Rule *t12* denotes: After output of the MH is sent to the MSS, the MSS agent takes action *cal-input(MSS.,* This derives the input of the MSS, i.e., *input-derived(MSS)* is true.

Then the output of the MSS is calculated and derived based on equation (4) after its input is derived. The rule is:

$$t13:\ holds(output-derived(MSSi),res(cal-output(MSSi),s)) \leftarrow holds(input-derived(MSSi),s))$$

Rule *t13* denotes: After input of the MSS is derived, the MSS agent takes action *cal-output(MSS),* then its output is derived, i.e., *output-derived(MSS)* becomes true.

The HS agent gets the output of the MSS, and calculates the input of the HS based on equation (5). The rule is:

$$t14:\ holds(input-derived(HS), res(cal-input(HS), s)) \leftarrow holds(output-derived(MSSi), s)$$

Rule *t14* denotes: After the output of the MSS is known to the HS, the HS agent takes action *cal-input(HS),* then the input of the HS has been derived , i.e., *input-derived(HS)* is true.

The output of the HS will be derived after the calculation base on equation (6) by considering the input of the HS and its local knowledge base. The rules are:

$$t15:\ holds(output-derived(HS), res(cal-output(HS), s)) \leftarrow holds(input-derived(HS), s))$$

Rule *t15* denotes: After input of the HS is derived, the HS agent takes action *cal-output(HS),* then its output is derived, i.e., *output-derived(HS)* becomes true.

The HS will do the decision-making based on its output. The rule is:

$$t16:\ holds(decision-made(HS), res(make-decision(HS), s)) \leftarrow holds(output-derived(HS), s)$$

Rule *t16* denotes: After *output-derived(HS)* becomes true, the HS will take action *make-decision(HS),* then the decision is made, i.e. *decision-made(HS)* becomes true.

4.4 Transaction Study Case in MLPMAS systems

Study case 2 demonstrates how a knowledge transaction is processed in a specific problem domain in MLPMAS systems.

Study Case 2: Given the same scenario and investment domain as study case 1 in this paper. The initial status is the same: MH1, MH2 and MH3 is alive, while MH4 is at sleep. At the time the HS agent requests a transaction, MH1 initializes an update transaction by requesting to invest a share (*share1*). Here, we will present, step by step, the activities involved in this transaction together with the generic transaction rules and the specific rules that are applied in this investment domain.

As the initial facts, MH1 and MH2 are registered with MSS1, while MH3 and MH4 are registered with MSS2. Based on rule *t1* specified in section 4.3, the rules *i1-i4* below denote the initial status:

i1: holds(registered(MSS1, MH1), s_0) ←
i2: holds(registered(MSS1, MH2), s_0) ←
i3: holds(registered(MSS2, MH3), s_0) ←
i4: holds(registered(MSS2, MH4), s_0) ←

The rules for the local knowledge base in the MH, MSS and HS are domain related rules. Recall these rules as below.

Local knowledge base of MH**:**
The local Knowledge base of MH1 has rules *r1-r3* related to share1 investment:

$$\begin{Bmatrix} r1:\ holds(profit(share1), s_0) \leftarrow \\ r2:\ holds(risk(share1), s_0) \leftarrow \\ r3:\ \neg holds(cost(share1), s_0) \leftarrow \end{Bmatrix}$$

The local Knowledge base of MH2 has rules *r4-r6* related to share1 investment:

$$\begin{Bmatrix} r4:\ holds(profit(share1), s_0) \leftarrow \\ r5:\ \neg holds(risk(share1), s_0) \leftarrow \\ r6:\ \neg holds(cost(share1), s_0) \leftarrow \end{Bmatrix}$$

The local Knowledge base of MH3 has rules *r7-r9* related to share1 investment:

$$\begin{Bmatrix} r7:\ holds(profit(share1), s_0) \leftarrow \\ r8:\ \neg holds(risk(share1), s_0) \leftarrow \\ r9:\ \neg holds(cost(share1), s_0) \leftarrow \end{Bmatrix}$$

The local Knowledge base of MH4 has rules *r10-r12* related to share1 investment:

$$\begin{Bmatrix} r10:\ holds(profit(share1), s_0) \leftarrow \\ r11:\ \neg holds(risk(share1), s_0) \leftarrow \\ r12:\ \neg holds(cost(share1), s_0) \leftarrow \end{Bmatrix}$$

The local knowledge base of the MSS:
On MSS1, rule *r13* is related to this investment in its knowledge base:

$$\{r13:\ holds(info-requested(HS, MHi), s_0) \leftarrow holds(slept(MHi), s_0)\}$$

On MSS2, rule *r14* is related to this investment in its knowledge base:

$$\{r14:\ holds(info-requested(HS, MHi), s_0) \leftarrow holds(slept(MHi), s_0)\}$$

Rules *r13* and *r14* denote that if MHi is at sleep at the time the HS agent requests transaction information from the MHs, the HS will request information from MHi when it does the decision making for the transaction.

The local knowledge base of the HS:
There are rules *r15-r21* in local knowledge base of HS:

$$\left\{\begin{array}{l} r15:\ holds(invest(share1),s_0) \leftarrow holds(profit(share1),s_0),\neg holds(risk(share1),s_0), \\ \neg holds(cost(share1),s_0),holds(info-get(MHi),res(request-info(MHi),s_0)) \\ r16:\ \neg holds(invest(share1),s_0) \leftarrow holds(risk(share1),s_0) \\ r17:\ \neg holds(invest(share1),s_0) \leftarrow holds(cost(share1),s_0) \\ r18:\ \neg holds(risk(share1),s_0) \leftarrow notholds(risk(share1),s_0) \\ r19:\ \neg holds(cost(share1),s_0) \leftarrow notholds(cost(share1),s_0) \\ r20:\ \neg holds(invest(share1),s_0) \leftarrow holds(info-requested(HS,MHi),s_0), \\ \neg holds(info-get(MHi),res(request-info(MHi),s_0)), \\ r21:\ \neg holds(info-get(MHi),s_0) \leftarrow notholds(info-get(MHi),s_0), \\ holds(info-requested(MHi),s_0),holds(timeout(MHi),s_0) \end{array}\right\}$$

In the following discussion of transaction processing, all the rules have been formally specified in section 4.3.

Step 1: MH1 requests to start an update transaction to invest share1. The rule is:

g1: holds(update-requested(MH1), res(write(MH1), s_0)) ←

Step 2: After MH1 requests an update transaction and the lock is acquired, MSS1 submits this transaction request to the HS. The transaction rule is:

g2: holds(trans-submitted(MSS1), res(submit(MSS1), s_0)) ←
holds(update-requested(MH1), s_0),
holds(locked(MH1), res(acquire-lock(MH1), res(write(MH1), s_0)))

Step 3: After the MSS submits the transaction to the HS, the HS starts the transaction. The transaction rule is:

g3: holds(trans-start(HS), res(do-trans(HS), s_0)) ←
holds(trans-submitted(MSS1), s_0)

Step 4: Based on its knowledge base, the HS will know which MH agents are related to this investment request, then the HS sends this investment request to all involved MH agents and asks these agents to provide their related investment information. The rule is:

$$g4:\ holds(info-requested(MHi),res(request-info(MHi),s_0)) \leftarrow$$
$$holds(invest-involved(MHi),s_0)$$

Step 5: After the MH agent receives the request from the HS, via MSS, the agent calculates the MH's output and then sends its output to the MSS if the MH is not at sleep. The rules are:

$$g5:\ holds(output-derived(MHi),res(cal-output(MHi),s_0)) \leftarrow holds(info-requested(MHi),s_0)$$

$$g6:\ holds(output-sent(MHi),res(sent-output(MHi),s_0)) \leftarrow holds(output-derived(MHi),s_0)$$

The outputs of MH1 and MH2 are sent to MSS1, the output of MH3 is sent to MSS2, while MH4 is at sleep at the moment.

Step 6: The MSS agent collects information from all related MHs and calculates the input. MSS1 collects input from MH1 and MH2, while MSS2 collects input from MH3 and MH4. The rule is:

$$g7:\ holds(input-derived(MSSi),res(cal-input(MSSi),s_0)) \leftarrow holds(output-sent(MHi),s_0)$$

The inputs of MSS1 and MSS2 are derived accordingly.

Step 7: The output of the MSS is derived by considering its input and its own knowledge base. The rule is:

$$g8:\ holds(output-derived(MSSi),res(cal-output(MSSi),s_0)) \leftarrow holds(input-derived(MSSi),s_0))$$

Step 8: The HS agent gets the output of MSS1 and MSS2, and calculates the input of the HS, the rule is as below:

$$g9:\ holds(input-derived(HS),res(cal-input(HS),s_0)) \leftarrow holds(output-derived(MSSi),s_0)$$

In this study case, MH4 is at sleep at the moment MSS2 collects information from it, MSS2 sends this information to the HS as a belief of its output and the HS reflects this in its input.

Step 9: The output of the HS will be derived by considering its input and its own knowledge base. The rules are:

$$g10:\ holds(output-derived(HS),res(cal-output(HS),s_0)) \leftarrow holds(input-derived(HS),s_0))$$

$$g11:\ holds(decision-made(HS),res(make-decision(HS),s_0)) \leftarrow holds(output-derived(HS),s_0)$$

From the discussion of study case 1 of this paper, we know that ¬*invest(share1)* is in every answer set, which means the HS will make the decision that share1 can't be invested.

Step 10: After the decision has been made (that share1 will not be invested), the transaction will be aborted at the HS level and the HS will update its knowledge base to reflect this accordingly. The rules t12 for this are:

g12: holds(trans-aborted(HS),res(abort-trans (HS),s_0)) ←
holds(decesion-made(HS),s_0)
g13: ¬holds(knowledge-update(HS),s_0) ← holds(trans-aborted(HS),s_0)

Step 11: The HS sends a transaction abort notice to the MSS. The rule reflecting this is:

$$g14:\ holds(abort-noticed(HS,MSSi),res(notice-abort(HS,MSSi),s_0)) \leftarrow holds(trans-aborted(HS),s_0)$$

Step 12: The MSS broadcasts the transaction abort information to involved MHs The rule for this is:

$$g15:\ holds(abort-broadcasting(MSSi,MHi),res(broadcast(MSSi,MHi),s_0)) \leftarrow holds(abort-noticed(HS,MSSi),s_0)$$

Step 13: The MHs update their local knowledge base accordingly. The rule reflecting this is:

$$g16:\ \neg holds(knowledge-update(MHi),res(update-knowledg(MHi),s_0)) \leftarrow holds(abort-broadcasting(MSSi,MHi),s_0)$$

As defined previously, a knowledge transaction in a MLPMAS system can be denoted as $T=<A,C,R,I(D))$. Study case 2 presented the steps in how a transaction is processed in a MLPMAS systems showing how agents, rules, initial facts, input and output work together allowing the HS to make a decision. In this study case, agents are on each MH, MSS and HS level, and reflexive communication channels are between the MH, MSS, and HS. The initial fact and domain related rules are reflected as *I(D)={i1-i4,r1-r21},* which includes initial facts of the transaction and every local knowledge base on the MH, MSS and HS. The generic transaction rules are reflected as *R={g1-g16}*, therefore the following fact φ is in answer set:

φ={ ¬*holds(invest (share1),* s_0*),*
holds(trans-aborted (HS), s_0*),*
¬*holds(knowledge-updated(MH),* s_0*)* }

i.e. the facts that share1 isn't invested, the transaction is aborted on the HS, and the knowledge base isn't updated on the MHs are concluded from the Answer set of the logic programs.

4.5 Disconnection and Mobility in MLPMAS Systems

In Comparison to stationary environments, disconnection and mobility are two distinguished features of mobile environments. The Mobile Host can be disconnected from the network due to voluntary or involuntary sleep, or network connection issues. Also, the Mobile Host can migrate to another MSS cell after a physical movement. Study cases 3 and 4 are presented to demonstrate how these two features can be addressed for knowledge transaction processing in the specific problem domain in the formalized MLPMAS systems.

Study Case 3: This study presents how a transaction is processed when MH4 is at sleep. Consider the same scenario as study case 1,, but now the select function S_F will take a negative atom as high preference for investment risk. Thus, $\neg$*invest (share1)* will become the input of a_{MSS1}, and the input of a_{HS} becomes:

$$KnowledgeInput(a_{HS}, Y)$$

$$= cons(\bigcup_{a_{MSS} \in Y} KnowledgeOutput(a_{MSS}, a_{HS}), S_F)$$

$$= cons(KnowledgeOutput(a_{MSS1,} a_{HS}) \cup KnowledgeOutput(a_{MSS2,} a_{HS}), S_F)$$

$$= \{profit(share1), \neg risk(share1), \neg cost(share1), info-requested(HS, MH4)\}$$

In this case, the HS requests information from MH4. The HS needs to know if *holds(info-get(MH4))* is true, and if *holds(profit(share1)), ¬holds(risk(share1)) and ¬holds(cost(share1))* are still true when considering the input from MH4. It is assumed that MH4 is still at sleep after the time bound. Thus ¬*holds(info-get(MH4))* becomes true, according to the rule *r20* in local knowledge base of the HS. In this case, ¬*invest (share1)* is always true in every answer set, that is, the HS will make the decision that share1 cannot be invested and the transaction will be aborted accordingly. In this study case, the rules for transaction processing are almost the same as in study case 2, the only difference being an extra rule, added after rule *g9,* for the HS to request the information from MH4:

$$f1: holds(info-requested(MH4), res(request-info(MH4), s_0)) \leftarrow$$
$$holds(input-derived(HS), s_0)$$

Rule *f1* denotes: after the input of the HS is derived, the HS takes the action *request-info(MH4)* to request information from MH4, then *info-requested (MH4)* becomes true.

Study Case 4: This study case addresses how a transaction is processed after a MH's migration from one cell to another. Given the same scenario as study case 1, the difference is MH4 will move to another cell managed by MSS3 during sleep, then wake up and register with a new mobile support station, MSS3. In this case, when the HS requests information from MH4 via MSS2 for decision making, MSS2 tells the HS that MH4 has left its cell. Thus the HS needs to find in which cell MH4 is currently located using one of the following algorithms: broadcast [8, 9], central service [12], home bases [17], and forwarding pointer [9], which are four basic mechanisms for determining the current address of a mobile host. After knowing that MH4 is located in cell MSS3, the HS will request MH4's information from MSS3, and eventually it will get MH4's information from MSS3. After the transaction is processed according to rules g1-g9 as in study case 2, the following extra rules are added to address the migration scenario:

f2: holds(info-requested(MH4), res(request-info(MH4), s_0))←
holds(input-derived(HS),s_0)

Rule *f2* denotes: based on its derived input, the HS takes action *request-info(MH4)* to request information from MH4, then *info-requested(MH4)* becomes true.

f3: holds(cell-located(HS, MH4), res(locate-cell(HS, MH4),s_0)) ←
holds(cell-migrated(HS, MH4),s_0)

Rule *f3* denotes: After the HS realises that MH4 has migrated from MSS2, i.e., *cell-migrated(HS, MH4)* is true, the HS takes action *locate-cell(HS, MH4)* to locate MH4 according to some algorithm and, then, the location of MH4 is known, i.e., *cell-located(HS, MH4)* becomes true.

f4: holds(info-get(MH4), res(request-info(MH4),s_0)) ←
holds(cell-located(HS, MH4),s_0)

Rule *f4* denotes: after the location of MH4 is established, i.e., *cell-located(HS, MH4)* becomes true, the HS takes action *request-info(MH4)* to request information from MH4 via the new MSS whereupon information from MH4 becomes available, i.e., *info-get(MH4)* is true.

From the local knowledge base of MH4, given in study case 1, it is known that MH4's output is $\{profit(share1), \neg risk(share1), \neg cost(share1)\}$, and by considering the output from MH4, the input of the HS becomes:

$$KnowledgeInput(a_{HS}, Y) = \{profit(share1), \neg risk(share1), \neg cost(share1)\}.$$

info-requested(HS, MH4) is no longer a belief in the HS's input.

Thus, *holds(invest(share1), s_0)* is entailed by the HS's logic programs. That is, the HS will make the decision that share1 will be invested, and the transaction will be committed, whereupon the knowledge will be updated accordingly in both the HS and the MH level.

5. Summary

This paper presented and formalized a mobile logic programming multi-agent system (MLPMAS) for mobile environments. This formalization can be used to process knowledge transactions in such kinds of mobile multi-agent systems. Extended logic program was employed as a specification method so this model is knowledge oriented and has declarative semantics. The formalized MLPMAS system is very useful for modeling decision-making problems for mobile applications, not just the solutions of the problem but also the evolution of the beliefs of and the interactions between the agents in mobile environments. Based on the formalized MLPMAS system, this paper formally defined the knowledge transaction in this kind of multi-agent systems. A few study cases were illustrated to demonstrate how knowledge transactions can be processed in a particular problem domain in MLPMAS systems.

With respect to previous works, major advantages of the formalized MLPMAS system can be characterized as follows: (1) This model can be used to process knowledge transactions in multi-agent systems for mobile environments. This differs from many investigations on multi-agents and intelligent agents in that currently developed languages and models for knowledge representation, reasoning and problem solving are only limited in conventional environments and haven't been extended to mobile environments, such as stable model/answer set, SMODEL, DLV and XSB model in [7, 15, 16]. (2) Extending logic programs is adopted in this formalization so this model is knowledge oriented and has declarative semantics. (3) It can specify details of knowledge transactions, input/ output and the knowledge base using knowledge rules. (2) and (3) make the formalization different from most other works on mobile agent systems in that they are not knowledge oriented processing, such as Telescript [20], Aglets [11], Mole [1] and KLAVA [2, 6, 11].

Reference

[1] Baumann, J., and Rothermel, K., "The Shadow Approach: An Orphan Detection Protocol for Mobile Agents," *Proceedings of the Second International Workshop on Mobile Agents (MA'98)*, pp. 2-13, September 1998.
[2] Bettini, L., et al, "KLAVA: a Java package for distributed and mobile applications," *Software Practice and Experience,* 32 (2002) 1365-1394.
[3] Cardelli, L., A. Gordon, D., "Mobile Ambients," *Theoretical Computer Science*, 240 (2000), 177-213.
[4] Cardelli, L., A. Gordon, D., "Types for the Ambient Calculus," *Information and Computation* 177 (2002), 160-194.
[5] Jianwen Chen and Yan Zhang, "A Rule Based Knowledge Transaction Model in Mobile Environments", *Proceeding of 2nd IASTED International Conference on Information and Knowledge Sharing,* Scottsdale, U.S.A., November, 2003.
[6] Deugo D.,"Choosing a mobile agent messaging model," *Proceedings of ISADS 2001*. IEEE Press, 2001;278-286.
[7] Eiter, T., and et al., "A deductive system for nonmonotonic reasoning," *in Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR97*), pp 363-374. LNAI, Vol. 1265, 1997.
[8] Imielinski, T. and Badrinath, B.R., "Data Management for Mobile Computing", *SIGMOD Record*, 22(1), pp. 34-39, March 1993.
[9] Ioannidis, J.Duchamp, D. and Maguire, G.Q. Jr., "IP-Based Protocols for Mobile Internetworking," *Proceedings of the ACM SIGCOMM'91 Conference on Communications Architectures and Protocols,* pp.235-245, September 1991.
[10] Komiya, T., et al, "Mobile Agent Model for Transaction Processing on Distributed Objects," *Information Sciences*, 2003, pp.1-16.
[11] Lange D, Oshima M. *Programming and Deploying* Java *Mobile Agents with Aglets,* Addision-Wesley: Reading, MA, 1998.
[12] Ma, C., "On Building Very Large Nameing *Systems", Proceeding of 5th SIGOPS Workshop on Models and Paradigms for Distributed Systems Structuring*, Sep. 1992.
[13] Milner, R., et al, "A calculus of mobile processes, Parts 1-2," *Information and Computation*, 100 (1) (1992) 1-77.
[14] Milojicic, D., "Mobile Agent Applications," *IEEE Concurrency*, 1999, pp. 80-90.
[15] Nemela, I., and Simons, P., "Efficient implementation of the well-founded and stable model semantics," *in Proceeding of the International Joint Conference and Symposium on Logic Programming,* pp 289-303. MIT Press, 1996.

[16] Rao, P., et al, "XSB: A System for Efficiently Computing Well-founded Semantics," *Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning,* pp 2-17. LNAI, vol. 1265, 1997.
[17] Teraoka. F. and Tokoro. M., "Host Migration Transparency in IP networks: the VIP Approach", *Computer Communication Review*, 23(1), pp.45-65, Jan. 1993.
[18] Vos, M. D., and Vermeir, D., "Extending Answer Sets for Logic Programming Agents," *in Proceedings of the Logic in Artificial Intelligence (Jelia2000) workshop*, 2000.
[19]Weiss, G., *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, The MIT Press, Cambridge, Massachusetts, 1999.
[20] White, J.E., "Telescript Technology: Mobile Agents," General Magic White Paper, Appeared in Bradshaw, J., *Software Agents,* AAAI/MIT Press, 1996.
[21] Wooldridge, M., *An Introduction to Multiagent Systems*, John Wiley & Sons, LTD, 2002.

Received: June 11, 2007