5-2002

An Intelligence Representation in Agent Systems: An Extended π-Calculus

Shahram Rahimi
University of Southern Mississippi, rahimi@cs.siu.edu

Maria Cobb
University of Southern Mississippi

Dia Ali
University of Southern Mississippi

Huiquing Yang
University of Southern Mississippi

Frederick E. Petry
Tulane University of Louisiana

Published in Rahimi, S., Cobb, M., Ali, D., Yang, H., and Petry, F. E. (2002). An intelligence representation in agent systems: an extended π-calculus. Proceedings of the 2002 IEEE International Conference on Fuzzy Systems, 2002. FUZZ-IEEE'02, 185-190. doi: 10.1109/FUZZ.2002.1004984

An Intelligence Representation in Agent Systems: An Extended π-Calculus

Shahram Rahimi, Maria Cobb, Dia Ali, Huiqing Yang
Department of Computer Science and Statistics
University of Southern Mississippi
Hattiesburg, MS 39406, BOX 5106
name.lastname@usm.edu

Frederick E. Petry
Department of Electrical Engineering and Computer Science
Tulane University
New Orleans, LA 70118
petry@eecs.tulane.edu

Abstract - Intelligent mobile agent technology is one of the most promising of the newer software paradigms for providing solutions to complex, distributed computing problems. Agent properties of autonomy, intelligence and mobility provide a powerful platform for implementations that can utilize techniques involving collaborative problem solving and adaptive behavior. Although the technological tools and capabilities have advanced to this point, research into formal models and extensions to support representations of this new computing paradigm has not kept pace. Specifically, we find that current formal processing models are lacking in representation abilities for (1) intelligence capabilities, (2) team-based problem-solving approaches, and (3) mobility. In this paper, we present an extension of π-calculus that addresses the first of these deficiencies, the representation of intelligence.

I. INTRODUCTION

We believe that the mobile agents computing paradigm will enable a wide range of exciting new distributed applications. However, beyond the facilitation and basic engineering challenge, there are several concerns related to security (malicious agents and malicious hosts), validation, verification, and performance that need to be addressed. To obtain a solution to these concerns, it is necessary to introduce formal methods that can provide a mathematical framework useful for specifying and verifying these applications [1].

There are several formalisms addressing the problem of modeling mobile processes: the π-calculus and its extensions, the Ambient calculus, Petri nets, Actors, and the family of generative communication languages [1][2]. However, none of these models formalizes the intelligence aspect of mobile agents, which is of extreme importance.

The ability to perform reasoning and learning is one of the key aspects that distinguishes intelligent agents from other more "robotic" agents. As Belgrave [3] describes, reasoning implies that "an agent can possess the ability to infer and extrapolate based on current knowledge and experiences - in a rational, reproducible way." There are essentially six types of reasoning and learning scenarios:

- Rule-based reasoning, where agents use a set of user pre-conditions to evaluate conditions in the external environment,
- Knowledge-based reasoning, where agents are provided large sets of data about prior scenarios and resulting actions, from which they deduce their future moves,
- Simple statistical analysis for learning and reasoning,
- Fuzzy agents for reasoning when the information is imprecise or incomplete,
- Neural networks reasoning for unstructured data or noisy data, and
- Evolutionary computing to expand the learning by viewing the system from an inter-agent perspective and employing a generic algorithm.

There have been a number of areas for which fuzzy agents have been developed. These include neural fuzzy agents for data mining [5], for profile learning [4], and fuzzy agents for autonomous agents [7]. More relevant to our web-based agent spatial retrieval project are the recent approaches taken to fuzzy agents for e-commerce applications [6][8].

Intelligent agents are employed to solve certain kinds of complex distributed problems by cooperation. Combing the problem-solving capabilities is the key to many agent-based solutions. Cooperation sometimes occurs because the problem-solving activity covers a large geographic region (such as in telecommunication networks and military applications), where different agents have responsibility for different geographical areas. Sometimes it occurs because different agents have different "specialties" to bring to the problem-solving process, similar to the cooperation among human team members. The Multiple-Agent System (MAS) paradigm has proven a popular and effective method for

0-7803-7280-8/02/$10.00 ©2002 IEEE

building a cooperating team of agents. Each agent in the team is constructed as a software agent, conferring abilities of autonomy, self-knowledge, and acquaintance knowledge on the agent abilities useful for team-forming and cooperative problem solving.

Mobile agent technology is an excellent tool for adequate implementation of distributed artificial intelligent systems. However, we lack a conceptual and modeling tool to adequately describe, evaluate and compare different approaches to distributed AI using intelligent agents.

In this paper we propose an extension to π-calculus that addresses intelligent capabilities of mobile agents. To do so, the concept of knowledge unit is introduced into π-calculus (we chose π-calculus because of its high popularity among the researchers). A knowledge unit includes a knowledge base (rules) and a set of facts. The knowledge base contains the domain knowledge needed to solve problems, and the set of facts are used by the knowledge base for production of decisions.

This extended π-calculus provides the modeling facilities for adding/dropping facts to/from the fact list and modification of the knowledge base as well as facilities for calling the knowledge units and more.

The remaining parts of the paper are organized as follows. First, a brief description of our motivation and an overview of π-calculus are given. Then, the intelligence extension is presented. Next, some illustrative examples are given. Finally, a brief summary concludes the paper.

II. MOTIVATION

The problem of automatically conflating digital map data has presented a rich set of computational challenges for two decades now. The term "conflation" in Geographical Information Systems is used to refer to the integration of data from different sources. Without the ability to conflate/integrate data from different sources, users are faced with extensive duplication of effort and unnecessary cost. More recently, researchers have turned to fuzzy logic, rough sets and other methods for handling reasoning abilities under conditions of uncertainty to help solve general conflation problems [9]. This approach certainly shows greater promise for producing a wider range of acceptable results.

The aim of our current research project is to develop an autonomous agent-based updating system that retrieves, filters, conflates and validates geospatial data from multiple heterogeneous sources by using fuzzy logic and rough sets methods. A formal representation of such a system is needed to allow us to address different implementation issues of the system. As we mentioned earlier, current formal processing models are lacking in representation capabilities for intelligence, mobility, abstraction, resource control, security and team-based problem solving approaches. This compelled us to implement an extension to π-calculus, which provides the above capabilities.

III. BASIC π-CALCULUS

The π-calculus is process algebra that focuses on process mobility. Processes communicate using channels. "The π-calculus is a way of describing and analyzing systems consisting of agents which interact among each other, and whose configuration or neighborhood is continually changing [2]." The channels define the configuration of the system. They are basic entities with no structure, with which more complex entities called processes are built.

A π-calculus process is given by the following syntax:

$$P \equiv \sum_{i \in I} \alpha_i.P_i \mid P_1 \mid P_2 \mid P_1 + P_2 \mid (x)P \mid !P$$

$$\alpha \equiv x(y) \mid \bar{x}y$$

Where x, y stand for names, '.' is the prefixing operator, '+' is the sum operator, and '|' is the parallel operator. $(x)P$ makes the name x local to P which means that only P can use x, i.e., x is used nowhere except in P. '!' is the replication operator (means $P \mid P \mid \ldots$). Prefixes α are atomic actions; they are of two forms: (1) input prefix $x(y)$ which means that the name y is received over channel x, (2) output prefix $\bar{x}\langle y\rangle$ which means that the name y is sent over channel x. Names refer to channels. Given a π-calculus P, the bound names of P, noted $bn(P)$ are names restricted by $(\,)$ or names y appearing in input prefixes $x(y)$. The free names of P, noted $fn(P)$ are the names appearing in P but not bound. For example, process $\bar{x}\langle y\rangle \mid x(u).\bar{u}\langle v\rangle$ will behave like $\bar{y}\langle v\rangle$ after the channel name y has been sent. Indeed, the first process $\bar{x}\langle y\rangle$ sends the channel name y along channel x, while the second one $x(u).\bar{u}\langle v\rangle$ is waiting for the channel name u along the channel x, in order to use it for sending name v.

The particularity of π-calculus is to allow names of channels to be passed as parameters. If a process moves, its neighborhood changes and with it, the channels it uses for communication move.

Transition $x(y).P \xrightarrow{x<z>} P\{z/y\}$ means that message z is sent along channel x. If we consider that z is not a simple value but a channel name, then the resulting process $P\{z/y\}$ is able to use this name as a channel for further communications. The actual value of the channel is instantiated during the execution of the process [2].

IV. INTELLIGENCE EXTENSION

In this section we define the syntax and semantics of the intelligence representation of the extension to π-calculus. In

this extension we introduce the concept of knowledge unit. A knowledge unit consists of a knowledge base and a set of facts. Mobile agents have the capability to add/drop facts to/from the facts list and modify the knowledge base by adding new rules or eliminating existing ones. Each mobile agent is capable of carrying one or more knowledge units and sending and receiving them to/from other agents.

We begin with a quick overview of the new primitives of the extension for manipulation of knowledge units (K_i represents a knowledge unit):

- Knowledge unit call, $K_i\langle\vec{y}\rangle(\vec{z}).P$,
- Receiving a knowledge unit, $x(K_i).P$,
- Sending a knowledge unit, $\bar{x}\langle K_i\rangle.P$,
- Adding a fact to the fact list or a rule to the knowledge base, $K_i(x)$,
- Dropping a fact from a fact list or a rule from a knowledge base, $\bar{K}_i\langle x\rangle$.

We adopt the condition "$[C]\ P_1{:}P_2$" instead of the match "$[x{=}y]\ P$". We assume an infinite set $\mathcal{N}$ of names and use letters x, y, and z, possibly with the prime or subscripts, to range over names. Names refer to variables, channels, facts, or rules. We adopt notation $\vec{x}$ which stands for a finite sequence $x_1, x_2, \ldots, x_n$ of names. K_i represents knowledge unit i which may be or may not be a local name to a specific process.

Definition I (Process):

The following expression define a process:

p	$\equiv$	τ	(no action)
	$\mid$	$\alpha.P$	(action prefix)
	$\mid$	$P_1 + P_2$	(summation process)
	$\mid$	$[C]P_1 : P_2$	(conditional process)
	$\mid$	$(x)P$	(name restriction)
	$\mid$	$(K_i)P$	(knowledge name restriction)
	$\mid$	$!P$	(replication)

Letters P, P_1, P_2, ... are used to denote processes. P is a set of processes. τ is the no action process. This is the process that can do nothing. α is called an action prefix (definition III). $\alpha.P$ performs the action α and then behaves like P. $P_1 + P_2$ is a summation process. This process acts like either P_1 or P_2. $[C]P_1 : P_2$ is a conditional process. If C is logically true, this process acts like P_1. x, y stand for names (channels, variables, facts, or rules). $(x)P$ makes the name x local to P which means that only P can use x. $(K_i)P$ indicates that K_i is a knowledge unit name local to process P, which means that we may have more than one K_i in a multi-agent system (more in examples). $P!$ is the replicated processes (means $P_1 \mid P_2 \mid \ldots$). For instance $P_1 \mid P_2$ consists of P_1 and P_2 acting in parallel. The components may act independently; also, an output action of P_1 at any output port $\bar{x}$ may synchronize with an input action of P_2 at x, to create a silent, τ, action of the composite agent $P_1 \mid P_2$.

Definition II (knowledge units):

A knowledge unit consists of a knowledge base (rules) and a set of facts. A knowledge unit reacts to any new fact(s) added to its facts list. K_1, K_2,... represents knowledge units. K^i denotes the set of knowledge units that belong to process i (P_i).

The grammar of knowledge units:

K	$\equiv$	0	*(empty knowledge unit)*
	$\mid$	r	*(a single rule)*
	$\mid$	K_1+K_2	*(knowledge units summation)*

0 is called an empty knowledge unit. An empty knowledge unit is produced if all the rules and facts are deleted from it. A knowledge unit may consist of a single rule. K_1+K_2 is called a knowledge unit summation.

Definition III (Actions):

In addition to send and receive actions in π-calculus the extension includes Knowledge Actions (KA). KAs consist of knowledge unit calls, receiving knowledge units, sending knowledge units, adding/dropping facts and rules to/from a knowledge unit. Letters $\alpha_1, \alpha_2, \ldots$ are used to represent action prefixes. Let A denote the set of all actions in the calculus:

- $x(\vec{y})$ is an input prefix. x stands for a name of an input port (channel) of a process which contains it. $x(\vec{y}).P$ inputs arbitrary names $\vec{z}$ at the port x and then behave like $P\{\vec{z}/\vec{y}\}$. All free occurrences of the names $\vec{y}$ in P are bound by the input action prefix $x(\vec{y})$ in P.

- $\bar{x}\langle\vec{y}\rangle$ is an output prefix. A name x is thought of as an output port of a process which contains it; $\bar{x}\langle\vec{y}\rangle.P$ outputs the names $\vec{y}$ at the port x and then behaves like P.

- $(K_i)P$ makes the knowledge name K_i private(local) to P.

- $K_i\langle\vec{y}\rangle(\vec{z})$ is a knowledge unit call. $K_i\langle\vec{y}\rangle(\vec{z}).P$ calls the knowledge unit, K_i, passing a list of facts,

$\vec{y}$. The result of this call is placed in $\vec{z}$. All free occurrences of $\vec{z}$ in P are bound by the prefix $K_i\langle\vec{y}\rangle(\vec{z})$ in P.

- $x(K_i).P$ is an input prefix which receives a knowledge unit, K_i, from a process that is sending it through port x. $x(K'_i).P$ inputs a knowledge unit, K_i, at the port x and then behaves like $P\{K_i / K'_i\}$.

- $\bar{x}\langle K_i\rangle$ is an output prefix which sends a knowledge unit to a process that is receiving through port x. $\bar{x}\langle K_i\rangle.P$ outputs the knowledge unit K_i through port x and then acts like P.

- $K_i(x)$ is a prefix that adds x to the facts list of K_i, if x is a fact, or to the rule list of K_i if x is a rule. $K_i(x).P$ adds x to the facts list or rule base of K_i and then acts like P.

- $\overline{K}_i\langle x\rangle$ is a prefix which drops x from the facts list (if x is a fact) or from the rule base (if x is a rule). $\overline{K}_i\langle x\rangle.P$ drops x from the facts list or the rule base of K_i and then acts like P.

We need conditional expressions to describe conditional processes to deal with case analysis. As we mentioned earlier, in the π-calculus the match operator $[x=y]$ is used for this purpose. $[x=y]P$ behaves like P if the names x and y are identical. Otherwise it behaves like the deadlock process. We adopt the condition instead of the match. The conjunction $(C_1 \wedge C_2)$ and the negation $(\neg C)$ are also used for the easier task of programming or specification.

Definition VI: (structural laws)

Over what π-calculus offers, the structural laws of the extended calculus are as follows:

- $K + 0 \equiv K$
- $K_1 + (K_2 + K_3) \equiv (K_1 + K_2) + K_3$
- $K_1 + K_2 \equiv K_2 + K_1$
- $(K_1)(K_2)P \equiv (K_2)(K_1)P$
- $(K_1)(K_2)(K_3)P \equiv (K_1, K_2, K_3)P$

V. ILLUSTRATIVE EXAMPLES

In this section, we shall present some examples to demonstrate the basic anatomy of the extended calculus. Moreover, we will demonstrate the use of the new formalism in a system for solving the mapping conflation issue, which includes five intelligent agents.

A. Example 1: (knowledge unit passing)

The agent P has a link x to agent Q and wishes to pass knowledge unit K_1 along its link to Q. Q is willing to receive it. Thus P may be $\bar{x}\langle K_1\rangle.P'$ and Q may be $x(K'_1).Q'$ (figure 1).

A knowledge unit with a private (local) name: Ki

A knowledge unit with a non-private name: Ki

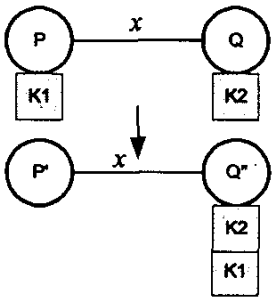

Figure 1: Example 1

In this case, the transition is:

$$\bar{x}\langle K_1\rangle.P' \mid x(K'_1).Q' \xrightarrow{\tau} P' \mid Q'\{K_1 / K'_1\}$$

So Q'' in the diagram is $Q'\{K_1 / K'_1\}$. If K_1 was private to agent P', then just a copy of the knowledge unit would be sent.

B. *Example 2: (passing a copy of knowledge unit)*

As in example one, P has a knowledge unit K_i, but now suppose that this knowledge unit name is private (local). P wishes to pass a copy of K_i to Q along its link x. In this case:

$$(K_1)(\bar{x}\langle K_1\rangle.P') \mid x(K'_1).Q' \xrightarrow{\tau} (K_1)(P' \mid Q'\{K_1/K'_1\})$$

C. Example 3: (knowledge unit call)

In this example agent P receives a fact, y, from agent Q and then calls knowledge unit K_1 by adding the fact to K_1 facts list and then behaving like P' which in fact is the same as $P\{y/z\}$.

$$x(z).P \mid \bar{x}\langle y\rangle.Q \xrightarrow{\tau} P\{y/z\} \mid Q;$$
$$K_i\langle y\rangle(y_o).P' \xrightarrow{\tau} P'$$

D. Example 4: (Scope extrusion)

Now let's assume that agent P wishes to send a knowledge unit K_i, which is not a private name, to the agent Q which already has a knowledge unit with the same name (a private name).

In this case, agent Q changes its private knowledge unit name, K_1, to K'_1 (figure 2).

$$\bar{x}\langle K_1\rangle.P' \mid (K_1)(x(K_2).Q') \xrightarrow{\tau}$$
$$.P' \mid (K'_1)(Q'\{K'_1/K_1\}\{K_1/K_2\})$$

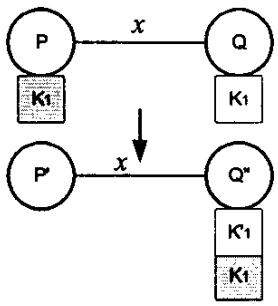

Figure 2: Example 4, first part.

Now let's consider a group of three agents, P, Q, and R. Agent P has a private link, x, to R. P also has a non-private link, y, to Q, which has a non-private link x as well. Agent Q would like to pass its knowledge unit K_1 to both P and R at the same time, and both P and R are willing to accept this knowledge unit. K_1 is a private name to agent Q. In this scenario, agent P may pass channel x to Q, so Q can contact both P and R. Because Q already has a non-private channel with name x, they (P, R and Q) have to rename the private channel x to x' (figure 3).

In this case:

$$(x)(\bar{y}\langle x\rangle.P' \mid R) \mid y(z)Q' \xrightarrow{\tau}$$
$$(x')(P' \mid R \mid Q'\{x'/z\})$$

and

$$(x')((K_1)(\bar{x}'\langle K_1\rangle.Q') \mid x'(K'_1).P' \mid x'(K'_1).R \xrightarrow{\tau}$$
$$(x')(K_1)(Q' \mid P'\{K_1/K'_1\} \mid R\{K_1/K'_1\})$$

It should be emphasized that in case of name duplication, the private name is always the one that changes. This is to be expected, because other agents may use a non-private name in the system and changes to a non-private name may cause system irregularities.

E. Example 5: (an example of a small geographical conflation system)

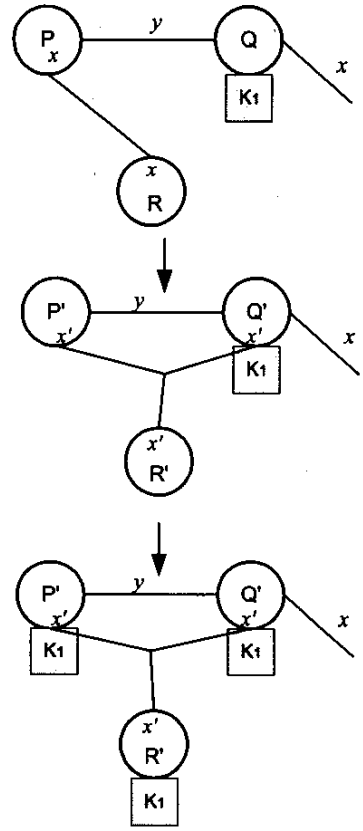

Figure 3: Example 4, second part.

In this scenario, there are five agents involved (figure 4):

- Manager Agent: M
- Conflation Agents: C_1 and C_2
- Query/Conversion Agents: Q_1 and Q_2

The manager agent divides this specific conflation task to point and line conflations by sending the line conflation knowledge unit, K_L, to C_2 and the point conflation knowledge unit, K_P, to C_1.

The query/conversion agents have responsibility to get the data from two different data sources, one in vector format and the other one in raster format. These agents then convert the data to a proper format and pass it to the conflation agents. Again, it is the manager agent that sends the query agents the proper knowledge units for vector and raster format conversion (K_V and K_R).

Next, the point and line conflation agents (C_1 and C_2) perform the conflation process on the data they receive from query agents and pass the results to the manager agent (figure 4). It is particularly in the case of the conflation agents that we will implement the needed extensions required for fuzzy agent mechanisms. Based on our current conflation approach that uses fuzzy logic for the spatial data integration, we can use the formal agent representations developed here to extend

the fuzzy spatial techniques to the conflation agents [9]. The overall model is as follows:

(1) Manager Agent sends the knowledge units to conflation and query agents:

$$-\bar{x}_1\langle K_P\rangle.\bar{x}_2\langle K_L\rangle.\bar{z}_1\langle K_V\rangle.\bar{z}_2\langle K_R\rangle.M' \mid x_1(K'_P).C'_1 \mid x_2(K'_L).C'_2 \mid z_1(K'_V).Q'_1 \mid z_2(K'_R).Q'_2\ ;$$

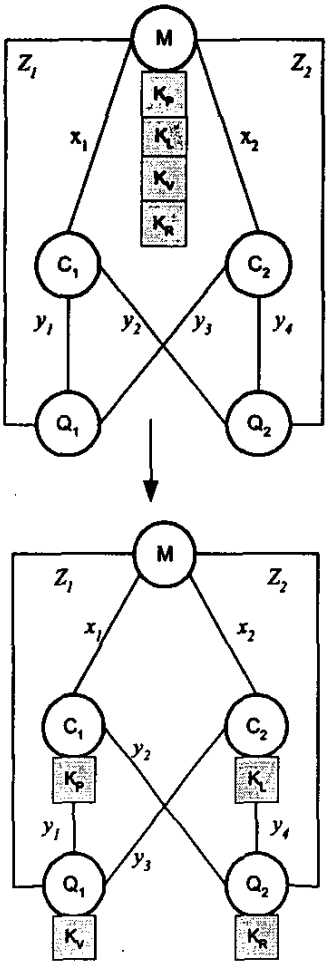

Figure 4: Example5.

(2) Query/conversion agents call their knowledge units to convert the data format. The knowledge units return the converted data, which is split to points and lines to be sent to point and line conflation agents:

$$-K'_V\langle D_{I1}\rangle(\bar{D}_{O1}).Q'_1 \mid K'_R\langle D_{I2}\rangle(\bar{D}_{O2}).Q'_2\ ;$$

$$\bar{D}_{O1} \equiv [D_{P1}, D_{L1}] \ \ and \ \ \bar{D}_{O2} \equiv [D_{P2}, D_{L2}]$$

(3) Next the query agents send their data to conflation agents:

$$-\bar{y}_1\langle D_{P1}\rangle.\bar{y}_3\langle D_{L1}\rangle.Q'_1 \mid \bar{y}_2\langle D_{P2}\rangle.\bar{y}_4\langle D_{L2}\rangle.Q'_2 \mid y_1(D'_{P1}).y_2(D'_{P2}).C'_1 \mid y_3(D'_{L1}).y_4(D'_{L2}).C'_2;$$

(4) Then the point and line conflation agents (C_P and C_L) call their knowledge units to perform the conflation process:

$$-K'_P\langle\bar{D}_P\rangle(D_{FP}).C'_1 \mid K'_L\langle\bar{D}_L\rangle(D_{FL}).C'_2\ ;$$

$$\bar{D}_P \equiv [D'_{P1}, D'_{P2}] \ \ and \ \ \bar{D}_L \equiv [D'_{L1}, D'_{L2}]$$

(5) Finally, conflation agents send the results of the conflation process to the manager agent:

$$-\bar{x}_1\langle D_{FP}\rangle.C'_1 \mid \bar{x}_2\langle D_{FL}\rangle.C'_2 \mid x_1(D'_{FP}).x_2(D'_{FL}).M';$$

VI. SUMMARY AND CONCLUSION

A computing paradigm based on intelligent mobile agents is a powerful problem-solving tool. A lack of formalisms for representing advanced agent capabilities such as intelligence/knowledge representation and collaboration can hinder proper verification and validation of agent system design. In this paper, we have presented an extension to the π-calculus that provides support for the representation of intelligence based on knowledge units and knowledge actions for mobile agent systems. We then demonstrated the use of the new formalism in a system for solving the mapping conflation issue. We believe this calculus-based modeling tool, including further planned extensions for collaboration and mobility representation, will significantly enhance the ability to reliably and accurately represent these more abstract, yet integral, properties of mobile agent systems.

References:

[1] G. D. M. Serugendo, M. Muhugusa, C. Tschudin, J. Harms,"A Survey of Theories for Mobile Agents," World Wide Web Journal, Special Issue: Applications and Techniques of Web Agents, pp. 139-153, 1998.

[2] R. Milner, "The Polyadic π-Calculus: a Tutorial," Logic and Algebra of Specication, Hamer, Brauer, and Schwichtenberg, Eds., Springer-Verlag, Germany, pp. 1-49, 1993.

[3] M. Belgrave, "The Unified Agent Architecture: A White Paper," URL: http://www.ee.mcgill.ca/~belmarc/uaa_paper.html, 1999.

[4] S. Mitaim and B. Kosko, " Neural Fuzzy Agents for Profile Learning and Adaptive Object Matching," Presence, Vol 7, #6, pp 617-637, 1998.

[5] M. Nikravesh and F. Aminzadeh, "Data Mining and Fusion with Integrated Neuro-Fuzzy Agents: Rock Properties and Seismic Attenuation," Joint Conf. Information Sciences, Research Triangle Park, NC, October 1998.

[6] R. Lee, J. Liu, B. Do, and H. Chan, "A New Era of Internet Shopping using Fuzzy Agents," Proceedings of International Conference on Artificial Intelligence (ICAI'2001), June 25-28, pp. 47-50, Las Vegas, 2001.

[7] A. Skarmeta, H. Barbera and M. Alonso, "A Fuzzy Agents Architecture for Autonomous Mobile Robots," Proceedings of IFSA, pp 675-682, Taipei, Taiwan, 1999.

[8] R. Yager, "Target E-commerce Marketing using Fuzzy Intelligent Agents," IEEE Intelligent Systems, Vol 15, pp 42-45, Nov. 2000.

[9] M. Paprzycki, M. Cobb, D. Ali, S. Rahimi, C. Bjursell, R. Angryk, F. Petry, "Methodology for Spatial Data Integration Agents," to appear in Proceedings of the 27th Summer School on Mathematics in Applications, Sozopol, ulgraia, June, 2001.