MOLECULAR SENSORS IN INFORMATION FUSION: AGENTS-BASED INTERFACE

E.V.Krishnamurthy
Australian National University, Canberra, Australia
Evk.Krishnamurthy@anu.edu.au

ABSTRACT

We describe a distributed agent-based interface, for hybrid (soft and hard) computation involved in molecular logic based sensor information (data) fusion. The computations are interpreted as the outcome arising out of deterministic, nondeterministic or stochastic interaction among the agents, in an environment containing multiple molecular sensors. These interactions are like chemical reactions and the agents can achieve the data fusion arising in a complex sensory system to achieve a required outcome

KEYWORDS
Agents, Information fusion, Molecular logic gates. Multi-agent tool kits.

1. INTRODUCTION

Molecular logic gates and molecular sensors are currently gaining importance [1], [2] in many situations involving multi-sensory evaluation. However, there are two important differences between electronic hardware and molecular hardware:
(i) ***Coupling***: Firstly, in many of the molecular logic gates, the inputs are chemicals and outputs are optical -they work on ionic input and optical (fluorescence) output. Since the nature of the inputs and outputs are different, it is difficult to pass the output from one molecular gate, as the input to the second. Therefore, iterated molecular logic gates cannot be constructed at present, unless we can find another molecular means to generate ionic output from fluorescence input. This makes physical integration of molecular logic gates difficult. Although very recently invented photonic multifunctional molecular logic devices by Andreasson et al [3], take optical (photons) inputs and produce optical (photons) outputs- still the inputs and outputs belong to different wavelength ranges requiring additional manipulations for coupling the devices.

(ii) ***Resetting***: Secondly, when chemical inputs are used and optical outputs are obtained, unlike an electronic device, which recovers to its initial state once the signal (which is usually a delta function) passes on in molecular devices the bound ions remain in its binding state unless it is removed or reset through resetting reagents. This problem of resetting is easier and faster (although not eliminated) in the case of photonic molecular gates, since light pulses can be used for resetting.

The above two features make it difficult at the present to speed up the molecular gates and work with them as the main hardware in the computational set-up. Therefore it becomes necessary to introduce a software agent (based on electronic hardware) as an interface. In this paper we describe an agent-based paradigm that can obviate these two difficulties and provide us a hybrid set-up that use electronic hardware, software, and molecular hardware. Such a set-up can be useful in many applications. Although the structure of the agents we describe are very

powerful, in the context of molecular information fusion we need very simple agents with restricted abilities for many practical applications. They can be designed in a simple manner either as a software patch or as a piece of hardware depending upon the applications.

This feature arises due to Agent's event loop:
Agents can deal with "events " as in Java. An event is anything that can occur asynchronously within the agent and external to it, to which the agent might want to respond.

Accordingly an agent creates an event loop.

while the agent is still active:
wait for the next event
process the event

This is called an **event loop**. The agent waits for events to happen. When an event occurs, the agent calls handle events appropriately, through *"Event-handlers"*.

Thus a key aspect of an agent is "**perception and action"** cycle. The perception can be external or internal. Thus an agent is assumed to be capable of observing and reacting to events external and internal to it. This feature is useful to couple molecular logic devices.

Fusion of sensor information plays an important role in several applications, Ovaska and Sick [4], Ovaska [5], Hall and Llinas [6]. In this paper we describe a distributed agent -paradigm for realizing the soft and hard computations involved in multi sensor information fusion.

The multi-agent paradigm for Information (data) fusion (MAIF) proposed in this paper consists of the following features:
(i) A set that contains agents (called the agent-space) whose information is structured in an appropriate way to suit the problem at hand
(ii) A set of interaction rules that prescribes the context for the applicability of the rules to the agents. Each rule consists of a left-hand side (a pattern or property or attribute) describing the conditions under which the agents can communicate and interact, and a right hand side describes the actions to be performed by the agents, if the rule becomes applicable, based on some deterministic or probabilistic criteria.
(iii) A control strategy that specifies the manner in which the agents will be chosen and interaction rules will be applied, the kinetics of the rule- interference (inhibition, activation) and a way of resolving conflicts that may arise when several rules match at once.
(iv) A coordinating agent evaluates the performance of the agents to determine the effectiveness of rule application.
This agent ensures that the contract among the different agents hold; if the contract fails, the coordinator can rescue, abort or restart as in i-Contract or in Eiffel.
The MAIF is applicable to physical, chemical, biological and agent-based computational problems, since it has the following computational features:
(i) Interaction -Based: The computations are interpreted as the outcome of interacting agents to produce new agents (or same agents with modified attributes) according to specific rules. Hence the intrinsic (genotype) and acquired properties due to interaction (phenotype) can both be incorporated in the agent space. Since the interaction rules are inherently parallel, any number of actions can be performed *cooperatively or competitively* among the subsets of agents.

(ii) Content-based activation of rules: The next set of rules to be invoked is determined solely by the contents of the agent-space, as in the context of chemical reactions.
(iii) Pattern matching: Search takes place to bind the variables in such a way to satisfy the left hand side of the rule. It is this characteristic of pattern (or attribute) matching that gives the agent-based paradigm its distinctive capabilities for innovative computing.
(iv) Suitable for deterministic, non-deterministic, fuzzy and probabilistic evolutionary modes:
This paper is organized as follows: In Sections 2 and 3, general properties multi-agent systems are described. Section 4 deals with the computational aspects MAIF. Section 5 we describe the connectivity structure that can arise among many sensory agents during information fusion process. Section 6 briefly describes the currently available software-agent tools. Section 7 is the conclusion.

2. MULTI-AGENT SYSTEM

The AOIS (agent oriented information system community) defines an agent as a system that is capable of perceiving events in its environment, or representing information about the current state of affairs and of acting in its environment guided by perceptions and stored information. The multi-agent system consists of the following single agent-system, Fig.1. Thus whenever several agents N are involved i = 1,2,3, … N then each of the agents will be denoted with a label (i), Woolridge [7].

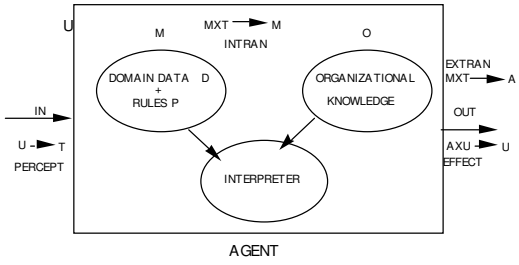

Figure 1

(1) Worldly states or environment U:
Those states which completely describe the universe containing all the agents.

(2) Percept: Depending upon the molecular sensory capabilities (input interface to the universe or environment) an agent can receive from U an input T (a standard set of messages), using a sensory function Perception (PERCEPT): PERCEPT : $U \rightarrow T$.

PERCEPT can involve various types of sensory perception arising from molecular sensors: see, read, hear, smell. In the context of molecular logic, it can be the recognition or identification of a particular wavelength. The messages are assumed to be of standard types based on an interaction language that is interpreted identically by all agents. Since U includes both the environment and other agents the input can be either from the agents directly or from the environment that has been modified by other agents. Thus we can deal with agents that can communicate directly, as well as, indirectly through the environment as in active walker model and affect the behavior of another agent (see also EFFECT) or the environment.

(3) Epistemic states or Mind M:
We assume that the agent has a mind M (that is essentially a problem domain knowledge consisting of an internal database for the problem domain data and a set of problem domain rules) that can be clearly understood by the agent without involving any sensory function. The database D sentences are in first order predicate calculus (also known as extensional database) and agents mental actions are viewed as inferences arising from the associated rules that result in an intentional database, that changes (revises or updates) D.
An ordered pair of elements (D, P) represents the agent's state of belief at a certain time. Here, D is a set of beliefs about objects, their attributes and relationships stored as an internal database and P is a set of rules expressed as preconditions and consequences (conditions and actions). When T is input, if the conditions given in the left-hand side of P match T the elements from D that correspond to the right-hand side are taken from D and suitable actions are carried out locally (in M) as well as on the environment. The nature of internal production rules P, mode of application and the action set determines whether an agent is deterministic, nondeterministic, probabilistic or fuzzy. Rule application policy in a production system P can be modified by:
(1) Assigning probabilities/fuzziness to apply the rule
(2) Assigning strength to a rule using a measure of its past success
(3) Introducing a support for a rule by using a measure of its likely relevance to the current situation.
The above three factors provide for competition and cooperation among the different rules. Such a model is useful for many applications. If the choice of the rules and corresponding actions are deterministic we, have a deterministic system suitable for hard computation. If, however, there are several competing choices, a nondeterministic choice or a probabilistic choice of the rules is made and the corresponding actions are carried out; then the agent is nondeterministic or stochastic and is suitable for soft computation in Complex Systems (e.g., Active-walker, Self-organization models). Also, we assume that each agent can carry out basic computations- having memory, simple addition, comparison, control rules and the generation of random numbers. Such mechanisms can help us simulate tumbling, as well as running of organisms for foraging and can be useful in designing Molecular machines, [12,13].

(4) Organizational Knowledge (O): Since each agent needs to communicate with the external world or other agents, we assume that O contains all the information about the relationships among the different agents. For example, the connectivity relationship for communication, the data dependencies between agents, interference among agents with respect to rules. Information about the location of different domain rules is in O.

(5) INTRAN:
On the receipt of T, the action in the agent M is suitably revised or updated by the function called Internal transaction (INTRAN).
Revision: means acquisition of new information about the environment; it requires a change in the rule system P. This may result in changes in the database D.
Example: Inclusion of a new tax-rule in Tax system.
Update: means adding new entries to the database D; the rules P **are not changed**.
Example: Inclusion of a new tax -payer in Tax system.
Both revision and update can be denoted in set-theoretic notation by: INTRAN: M X T $\rightarrow$ M(D,P)
T can be interpreted as a transaction for updating or revising a set of database instances. Hence, if one or several interaction conditions hold for several non- disjoint subsets of objects in the agent at the same time, the choice made among them can be nondeterministic or probabilistic. This leads to *competitive parallelism.* The actions on the chosen subset are executed atomically and

committed. In other words, the chosen subset undergoes an **'asynchronous atomic update'**. This ensures that the process of matching and the follow-up actions satisfy the four important properties used in *Transaction Processing*, namely, ACID properties: Atomicity (indivisibility and either all or no actions or carried out), Consistency (before and after the execution of a transaction), Isolation (no interference among the actions), Durability (no failure). Once all the actions are carried out and committed the next set of conditions are considered.
As a result of the actions followed by commitment, we may revise or update and obtain a new database for each agent; this may satisfy new conditions of the text and the actions are repeated by initiating a new set of transactions. These set of transformations halt when there are no more transactions executable or the databases does not undergo a change for two consecutive steps indicating a new consistent state of the databases.
However, if the interaction condition holds for several disjoint subsets of elements in the database at the same time, the actions can take place independently and simultaneously. This leads to *cooperative parallelism.*

(6) EXTRAN: External action is defined through a function called global or external transaction (EXTRAN) that maps an epistemic state and a partition from an external state into an action performed by the agent. That is: EXTRAN:
M X T $\rightarrow$ A. This means that the current state of mind and a new input activates an external action from the action set A.

(7) EFFECT: The agent also can affect U by performing an action from a set of actions A (ask, tell, hear, read, write, speak, send, smell, taste, receive, silent), or more complex actions. Such actions are carried out according to a particular agent's role and governed by an etiquette called protocols. The effect of these actions is defined by a function EFFECT that modifies the world states through the actions of an agent: EFFECT: A X U $\rightarrow$U; EFFECT can involve additions, deletions and modifications to U.

Example: The effect can be a restoration of the initial state of the Molecular gate by triggering a suitable chemical reagent [2] or a suitable resetting light pulse [3].
Also it can be a triggering of a new ion source for onward transmission to other molecular gates.
Thus the Percept and Effect functions are used in obviating the two difficulties mentioned earlier.
Thus an agent is defined by a set of nine entities, called a 9-tuple:
(U,T,M(P,D),O,A,PERCEPT,INTRAN,EXTRAN, EFFECT).
The interpreter repeatedly executes selected rules in P, until no rule can be fired.

3. MULTIAGENT COMPUTATION

A multi-agent system can be defined as a loosely coupled network of agents that interact among them and through the environment to solve a problem. Operationally, the multiagent system carries out distributed computation by sending, receiving, handshaking and acknowledging messages and performing some local computations and has the following features:
1. An agent has the structure as described in Figure 1.
2. There is a seeding agent who initiates the computation process.
3. Each agent can be active or inactive.
4. Initially all agents are inactive except for a specified seeding agent that initiates the computation.
5. An active agent can do local computation, send and receive messages and can spontaneously become inactive.
6. An inactive agent becomes active if and only if it receives a message.

7. Each agent may retain its current belief or revise its belief as a result of receiving a new message by performing a local computation. If it revises its belief, it communicates its revised state of belief to other concerned agents; else it does not revise its solution and remains silent.

Hence the basic agent model can realise:

(i) Reactive agent that make decisions at run time with a limited amount of information,

(ii) Deliberating agent that has an internal representation of the environment and has a logical inference mechanism for decision making and planning and

(iii) Interacting agent that is capable of coordinating the activities with other agents through communication and negotiation.

4. INFORMATION FUSION

Three crucial properties of Agents make them suitable for the multi sensor information fusion:

(i) *Autonomy*: Make decisions on actions they want to do without explicit control from the user,

(ii) *Reactive*: Respond appropriately depending upon the context, and

(iii) *Proactive*: Act in anticipation of future goals to meet the specified objectives.

In reactive fusion, the system has to react to various kinds of events, signals and conditions that are often distributed and concurrent. Also they can be time critical exhibiting both digital and analog (or hybrid) behavior. In addition the reactive system, as in cell biological system can contain components that signal each other and also repeatedly created and destroyed.

The fusion process is sensitive to the order of events. In order to speed up the use of the multi-agent fusion paradigm we need to consider how to permit multiple agent execution concurrently. This offers the possibility of carrying out parts or all of computations in parallel on distinct processors or performing multiple-sensory functions simultaneously. Such possibilities would require the analysis of the rules as to how the rules interfere with each other.

There are four ways in which such interference can take place:

1. **Enabling dependence (ED)**: Agent A(i) and agent A(j) are called enable dependent (or dataflow dependent) if the messages from A (i) creates the required precondition in A(j) to carry out a specific action .

2.**Inhibit dependence (ID)**: Agents A (i) and A (j) are called inhibit dependent, if the actions of A (i) creates the required precondition in A(j) to prevent it from executing a specific action.

3. **INTRAN Conflict (IC)** : Agents A (i) and A (j) are opposition dependent (also called data-output dependent) through A(k)), if the order in which A (i) and A (j) enable A(k) and update A(k) produce different results in A(k); that is the objects A(i) and A (j) perform operations on A(k) that are not order reversible. That is, local serializability is not ensured in the INTRAN within A(k), if the actions are carried out within an agent in different order.

4. EXTRAN Conflict (EC): Agents A (i) and A(j) are data antidependent through A(k) if the order in which A(i) enables (inhibits) A(k), and A(j) enables (inhibits) A(k) result in different external actions (EXTRAN) by A(k) on the environment. That is the order in which information arrives from the environment and other agents affects the global serializability of the actions of an agent.

4.1. Concurrency and Conflicts

In distributed computing and transaction processing: we require that the following two conditions are satisfied for global serialization when concurrent operations take place.

1. At each agent the actions in local actions are performed in the non-conflicting order **(Local serializability**).

2. At each agent the serialization order of the tasks dictated by every other agent is not violated. That is, for each pair of conflicting actions among transactions p and q, an action of p precedes

an action of q in any local schedule, if and only if, the preconditions required for p do not conflict with those preconditions required for execution of the action q in the required ordering of **all tasks in all agents (Global serializability).**

The above two conditions require that the preconditions for actions in different agents A(i) and A(j) do not interfere or cause conflicts. These conditions are necessary for the stabilization of the multi-agent systems so that the computations are locally and globally consistent.

Termination: For the termination of agent –based program, the interaction among the agents must come to a halt. of agents. Then we have an equilibrium state (or a fixed point).

Non–termination: These cases arise when the agents continue to interact indefinitely as in chemical oscillations, biological reactions, and sensory signal processing.

Conflicts: Resolution or compromise?

The conflicts arising in INTRAN and EXTRAN require resolution or compromise. e.g, the actions, may need a compromise, or a blending of the behaviour of actions if the quantitative parameters can be suitably averaged over. These rules should be based on the context.

Vector, Pipeline and Data Parallelism

1. Vector parallelism: If all the agents are independent then we can apply all the rules simultaneously, e.g., a vector addition.
2. Pipeline parallelism: Here multiple agents enable each other and passing data in a pipeline fashion – e.g. multi- enzyme reactions, where at each membrane an " imprisoned" enzyme performs a given operation and then sends it on to the next stage.
3. Data parallelism: Multiple identical agents are activated in parallel based on distinct data- e.g., fusing multiple sensory information.

4.2. Advantages

(i) **Multi Agent Information Fusion** (MAIF), starts with simple rules of interaction among the individual components that drive the system to the complex behavior observed. It works bottom up by examining what low-level rules and what kind of heterogeneous, autonomous agents are required to synthesize the required higher-level behavior. Thus, MAIF is useful for simulating active walker models where each walker (e.g., ants with scent) can influence (repel or attract) other walkers using a shared landscape.

(ii) Agent-based fusion enables us to make predictions about the processes occurring at the intermediate mesoscopic scale due to the interplay between the microscopic dynamics and the macroscopic environment.

(iii) The fusion of deterministic, chaotic/ stochastic systems are possible.

(iv) Different models where iterated application of simple rules can generate complex forms can be studied.

(v) Agent-based hybrid fusion (for soft and hard computation) can be used to study the global behavior of a complex adaptive system from the local interactive behavior of its components.

4.3. Special Purpose Agents in Simulation

Agents can be designed to realize special purpose functionalities. In building molecular logic models based on agents, it will be more convenient to use special purpose agents –such as sensor agents, goal agents, skill agents and compose these modules to build reusable, composable behaviors. In particular, we can have:

1.Relay agents that simply pass the message to the next signaling component
2.Messenger agents that carry the signal from one part of the system to another.
3.Adaptor agent: Link one signaling agent to another
4.Amplifier agents: Increase the signal levels causing a signal cascade

5.Transducer agents: Convert the signal from one for to another
6.Bifurcation agents: Spread the signal from one pathway to another
7.Integrator agents: Receive two or more signals and integrate them and transmit them.
8.Modulator agents: These modify the intra-agent signaling agents and thereby regulate the strength of the signal along the pathway.

4.4. Production, Demand and Consumption rule: Controlling Molecular Logic

In implementing Molecular logic it becomes necessary to generate the required ionic or light source in the environment. This can be realized by the producer- consumer model using interacting agents thus:
1. A *monitoring agent* checks the availability of the required reagent/ions or resetting light pulse in the environment. It enables the producer agent to produce the required reagents.
2.There is a *producer agent* that generates the ions and puts them in a buffer agent.
3. There is a *consumer agent* that consumes the ions /light pulses from the buffer agent.
4. The *buffer agent* checks the consumption of ions from the buffer. If the buffer is full it inhibits the producer agent; if the buffer is empty, it enables the producer agent to produce. If the buffer is partly full it optimizes the rate of consumption and production.

5. CONNECTIVITY PATTERNS IN FUSION

In multi sensor fusion the information arrival is nondeterministic, fuzzy or probabilistic. The communication or interconnection patterns among the agents play a key role for applications to various fusion aspects. The fusion agents therefore modify the pattern of their communication pathways, namely, the topology and geometry at will. Here we need to study the Graph model to analyze the connectivity structure among the agents in a network arising from cooperative and competitive interactions.

6. MULTI-AGENT ARCHITECTURE AND TOOLKITS

Shakshuki et al. [8] evaluate multiagent tool kits, such as: Java Agent development framework (JADE), Zeus Agent building toolkit and JACK Intelligent Systems. They consider Java support, and performance evaluation. The number of agents they consider is of the order of 32. For the implementation of the paradigm described here, further developments are needed in Agent technology, since we need a very large number of agents to simulate many real-life scientific applications.

Gorton et al [9] have evaluated agent architectures: Adaptive Agent architecture (AAA), Aglets developed by IBM, and the Java based architecture Cougaar. The paradigm described here is well-suited for implementing in Cougaar, a Java based agent architecture, since Cougaar is based on human reasoning. A Cougaar agent consists of a blackboard that facilitates communication and operational modules called plug-in that communicate with one another through the blackboard and contain the logic for the agent's operations. The use of blackboard and direct communication are useful for simulating the problems in Synthetic biology.

Many other recent developments include Repast, North et al [10], Einstein, Ilachinski, in [11], and other agent based software tools, Adamsky and Komosinski [11].
Repast is object oriented and has a discrete event scheduler, 2D visualization, and can be used with a variety of languages: java, C#, managed C++, Prolog etc and is available for several Platforms. Yet another tool is Star Logo, in [11].

7. CONCLUSION

This paper described a multiagent Information fusion (MAIF) methodology suitable for molecular sensory evaluation. We described the important features of agents, the cooperative and competitive parallelism, nondeterministic and probabilistic connectivity structure that can arise among the agents during the multi-sensory information fusion. We also briefly described the software tools available for agent -based information fusion. These will be useful in realizing molecular machines [12],[13].

REFERENCES

[1] A.P.de Silva, *Molecular Logic gates* (2011), Focus Reviews, ,Chem Asian Vol.6,pp.750-766.
[2] V.Bojinov and N/.Georgiev(2011), *Molecular Sensors and molecular Logic gates* ,J. Univ Chem. Technology and Metallurgy, Vol.45, No.1., pp.3-26.
[3] J.Andreasson et al., *All Photonic Multifunctional Logic devices*, J.Am Chem Soc.,Vol.133, pp.11641-11648, 2011
[4] S.J.Ovaska and B.Sick (2006), *Fusion of soft and hard computing, in Computational Intelligence*,pp.47-72, G.Yen and D.B.Fogel (Eds),IEEE Computational Intelligence Society, IEEE Press, New York.
[5] S.J Ovaska,(Ed (2004), ,*Computationally Intelligent Hybrid systems: The fusion of Soft and hard computing*, Wiley, IEEE Press.
[6] D.L.Hall and J.Llinas (Eds)(2001), *Handbook of Multisensor data fusion* ,CRC Press, Boca Raton, Fl.
[7] M.Woolridge (2002), *Introduction to Multi-Agent systems*, John Wiley, New York.
[8] E.Shakshuki and Y. Jun (2004), *Multi-agent development toolkits: An Evaluation*, Lecture Notes in Artificial intelligence, 3029, Springer Verlag, New York, pp.209-218.
[9] I,Gorton,(2004) *Evaluating agent Architectures: Cougaar, Aglets and AAA*, Lecture Notes in Computer Science,Vol.2940, Springer Verlag, New York,pp.264-274,
[10] M.J.North, M.J andE.J Burton (2006), *Escaping the accidents of History: An overview of Artificial life modeling with Repast*, in [10], pp. 115-142..
[11] A.Adamsky and M.Komosinsky (2006), *Artificial life Models in Software*, Springer, New York.
[12] V. Balzani., et al, *Molecular Nanotechnology: Towards Artificial Molecular machines and motors*, see Web.
[13]V.Balzani et al(2007), ,*Molecular devices and Machines , Nanotoday*,Vol.2, pp.18-25, April.

Professor E.V. Krishnamurthy is with the Computer Sciences Laboratory, Australian National University. He is the author of several books and papers in Computer Science and Information technology.
Address:
Computer Sciences Laboratory,
Research school of Information Sciences and Engineering,
Bldg.115 Australian National University, Canberra, ACT 0200, Australia.
Email: Evk.Krishnamurthy@anu.edu.au