An Abstract Representation Model for Evolutionary Analysis of Multi-Agent Interactions

Arash Shaban-Nejad
McGill Clinical and Health Informatics, Department of Epidemiology and Biostatistics, Faculty of Medicine
McGill University
Montreal, Canada
arash.shaban-nejad@mcgill.ca

Volker Haarslev
Department of Computer Science & Software Engineering, Faculty of Engineering & Computer Science
Concordia University
Montreal, Canada
haarslev@cs.concordia.ca

***Abstract*—Intelligent agents are able to assist human in managing highly dynamic and complex systems in various knowledge intensive domains. The communication between different agents interacting in an integrated multi-agents system can be managed through a set of steering rules, which together form interaction protocols. To support the negotiation, communication and interaction between different intelligent agents, using an appropriate knowledge representation formalism is crucial. This paper introduces the potential of category theory as a formal representation vehicle to facilitate evolutionary analysis of agent interaction and negotiation for managing evolving ontologies in the domain of biomedicine. Utilizing categories supports agents' communication, negotiation, state transitions, compositions and transformations in different levels of abstractions.**

Keywords-biomedical ontologies; evolutionary analysis, multi-agent system; agent interactions; category theory;

I. INTRODUCTION

One of the classic AI-complete (or AI-hard) [1] problems occurs when one needs to manage unexpected situations and deal with changes while planning for a real world critical system. Critical, in this case, means when the failure or malfunction of the system may result in severe loss [2]. One may find excellent examples of life support critical systems based on massive integrated/dynamic knowledge bases in health science, dealing with the health and life of a patient, the failure of which is intolerable.

In order to support and improve the level of automation for analyzing and managing evolving biomedical ontologies and knowledge bases, in our earlier works [3, 4], we have proposed the RLR agent-based framework for Representation, Legitimation and Reproduction of changes in distributed ontologies with minimum human intervention. To support the negotiation, communication and interaction between different intelligent agents, in an automatic manner, a mathematical knowledge representation formalism is necessary. As highlighted in [6], despite worldwide efforts in this domain, no proven formal frameworks, methods, and tools for modeling automatic agent interactions and argumentation yet exist. The interaction protocols, which consist of a set of steering rules to manage the interactions, are commonly represented using UML [7], Petri nets [8], State-charts [9], state-transition diagrams or finite state machines [10]. RLR employs categorical notions as a basis for modeling an agent communication language. The categorical framework is expressive enough to model the agents' behaviors, yet abstract enough to represent the generality of the protocols. RLR benefits from the algebraic power gained by using an abstract categorical representation of agents' interactions to increase the autonomy of argumentations in the change management framework. Category theoretical representation, with its ability to derive formal inference out of a diagrammatic representation, is independent of the type of interactions and their details, so its generality can be used to describe different types of protocols, study a multi-agent system (MAS) framework in different levels of abstraction, analyze rule transformations (yielding a practical image of adaptive learning agents and their semantics), and formalize dialectic trees for argumentation. In addition, we utilize the category theoretical distributed graph transformation techniques [41] for evolutionary analysis of concurrent model transitions and transformations in multi-agent systems using the specific conditions and transformation rules.

II. FORMAL REPRESENTATION MODEL

Category theory provides a universal algebra for the representation of highly abstract concepts. Categorical notations consist of diagrams with arrows. Each arrow $f: X \rightarrow Y$ represents a function. A Category C includes: A class of objects and a class of morphisms (“arrows”), and for each morphism f there exists one object (A) as the domain of f, and one object (B) as the codomain; For each object A, an identity morphism, which has domain A and codomain A (“ID_A”); and For each pair of morphisms f:A→B and g:B→C, (i.e., cod(f) = dom(g)), a *composite morphism*, $g \circ f$: A→C exists. Some of the primitive constructors of category theory that we use in our framework are as follows: Products, Coproducts, Functors, Natural Transformation, Pushout and Pullback. In order to orient the reader with a precise definition of categories and some important introductory definitions, we refer to [5] for additional information.

In the rule-based graph transformation, generally applying a transformation rule $L \rightarrow R$ denotes finding a proper match of L (Left hand side) in the source graph and replacing L by R (Right hand side), leading to the target graph of the graph transformation. Following the double-pushout approach

(DPO) [42], a transformation rule is defined [26] as a pair $t: L \leftarrow I \rightarrow R$ of morphisms $l: I \rightarrow L$ and $r: I \rightarrow R$ such that l is injective, where the graphs L and R are called the left and right-hand sides respectively, and I is called the interface or gluing graph. Further information on different notions and techniques for graph transformation can be obtained from [13, 14].

III. Categorical Analysis of the RLR Multi-Agent Framework

RLR is a cooperative Multi-agent framework, for capturing, representing, tracking and analyzing changes within ontological structures through a rule-based reactive and proactive behaviour acting along with an integrated argumentation framework. In this research we use categories to support agents' communication, negotiation, state transitions, compositions and transformations in different levels of abstractions.

A. *Analyzing a Multi-agent Framework in Different Levels of Abstraction*

In our method, we define different types of categories based on different local and global perspectives. Each agent can be considered a category, with states of the agent as objects and the actions that cause an agent to change its state as morphisms. More generally, we can define a category of agents, with agents as objects and the different types of communication and interaction channels between agents as (functor) morphisms. In the same way, one can for example define the services given by agents as a category, with agents as objects and the composition relations between the agents (representing different interactions, communications, message passing, or sharing attributes between agents) as morphisms, or, alternatively, the category of services, with agents' services as objects and the mapping between the services as (functor) morphisms. Moreover, by changing the level of abstraction, we define a multi-agent system as a category[1] consisting of services as objects and the relations between them as morphisms, as well as the category of multi-agent systems, with each system composed of several agents providing different services as an object and the different communication channels between two or more distributed multi-agent systems as (functor) morphisms. This viewpoint about the categorical conceptualization of MAS structures in different levels of abstractions leads us towards defining a formal semantic for various interactions occurring between agents in evolving environments.

In an integrated multi-agent-based framework such as RLR, functors (morphisms in the category of all small categories) and their compositions are powerful abstraction mechanisms for analyzing the relations between different categories (e.g., relations between categories of agents, relations between a category of agents and a category of services, or a category of multi-agent systems and a category of states, etc.). As an example, consider a scenario for the alignment of two ontologies, by considering state as a category with the different states of an agent (i.e., initial state (Ini_S); requesting merge (Req_M); receiving the merge result (Rec_M) and checking for validity (Chk_V)) as objects and the message passing between the states (i.e., issuing alert, change notice, ontology ID, and so on) as morphisms (Figure 1).

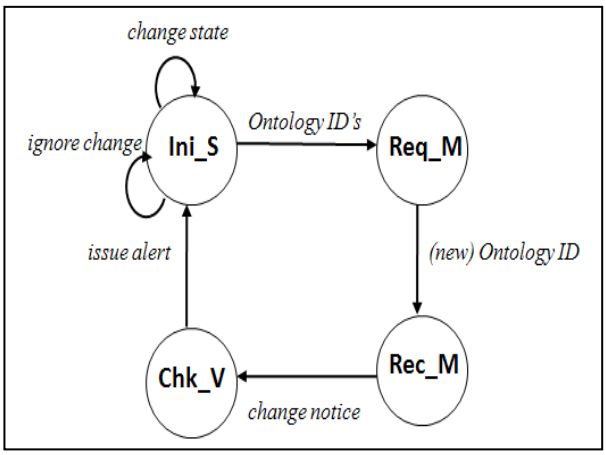

Figure 1. The categorical illustration of states for the ontology merging scenario

The above categorical diagram might be changed to demonstrate different options for performing ontology merging. For example one may want to check whether two ontologies are from the same domain or not (Chk_D) and if they are not from the same domain, cancel the merging and move back to the initial state (Figure 2).

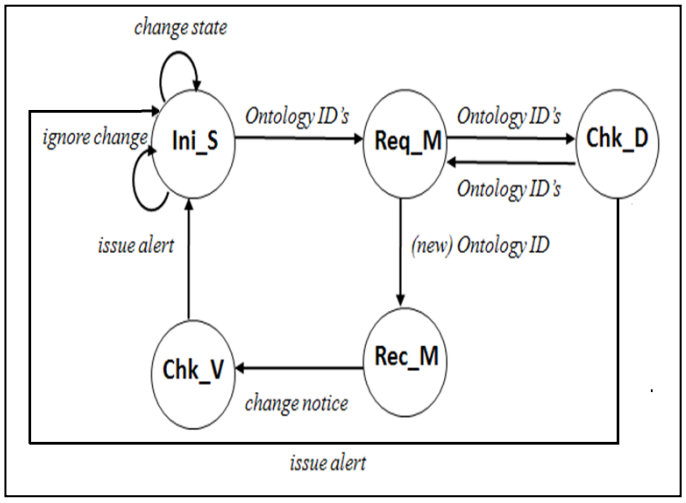

Figure 2. The categorical illustration of states for an alternative ontology merging scenario.

Considering the first model represented in Figure 1 as Category $O\mathrm{St}_1$ and the modified version represented in Figure 2 as category $O\mathrm{St}_2$ in the category of states using the composition law and a functor, we are able to represent the

[1] Based on different conceptualizations, one may consider a multi-agent system to be a category with agents as objects and the relations between them as the morphisms.

transition between St_1 and St_2 through the functor F representing "Check Domain": F: $St_1 \rightarrow St_2$.

In a similar way different associations between different types of objects (e.g., various cognitive units [11] described in the categorical sense) can be modeled. For example[2], one can describe a set of prepositions as objects within the category of prepositions and the relations between them as the morphisms in this category. Figure 3 represents a typical diagrammatic representation of such interactions for the following prepositions:

1. Agent **AG_1** received a message.
2. Agent **AG_1** has perceived a change request through the message.
3. The perceived change request is a delete request.
4. The delete request is issued to be performed on ontology $\mathbf{O_1}$.
5. The target for the deletion is concept $\mathbf{C_x}$ within Ontology $\mathbf{O_1}$.
6. Ontology $\mathbf{O_1}$ is currently being used by $\mathbf{KB_1}$.
7. The concept $\mathbf{C_x}$ is being reused in a process **Pr_1.**
8. The concept $\mathbf{C_x}$ has three sub concepts $\mathbf{C_{x1}}$, $\mathbf{C_{x2}}$, $\mathbf{C_{x3}}$.
9. Two concepts $\mathbf{C_{x1}}$ and $\mathbf{C_{x2}}$ are currently being used in a process **Pr_1.**
10. The controller agent of $\mathbf{KB_1}$ should be notified about the request.
11. The negotiation for loss/benefit has been performed.
12. Based on the negotiation outcome the delete request is postponed
13. The notification to the agent **AG_1** is sent.
14. **AG_1** ignored the change request.

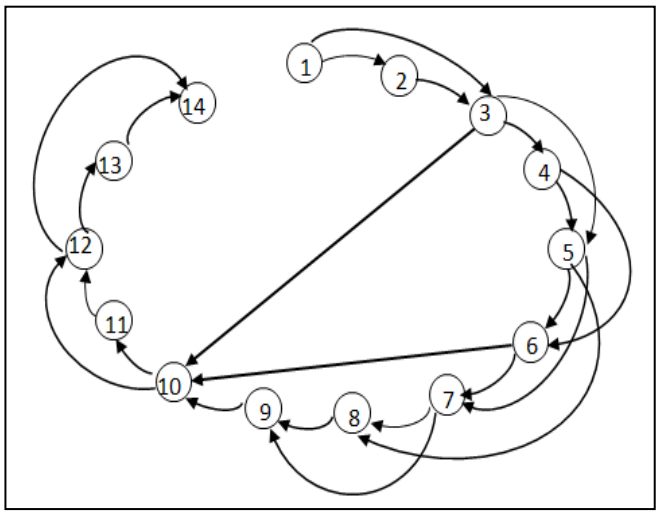

Figure 3. A generic categorical representation of different prepositions in an agent based framework dealing with a "delete request" message.

As can be seen in Figure 3 several concurrent interactions may be performed through the compositions between the morphisms. Also several inferred knowledge can be gained[3] through this categorical approach, which can later on be used in the learning phase in the RLR framework. Upon successful completion of the negotiation process in RLR, the ontology will either remain unchanged or be modified to convey the new knowledge based on the outcome.

B. *Analyzing a Representation of Agents' Rule Compositions and Transformations*

Intelligent agents perform actions in a context by using rules that guide interactions. In order to perform an action, which may lead to a state transition, often two or more sets of rules may be combined and integrated, ideally in an automatic fashion. The manual combination of rules is neither desirable nor feasible in many circumstances (e.g., when dealing with large sets of rules). The mathematical power of categories can deliver a formal guidance for combining these sets of rules, which are usually described in a diagrammatic representation[4]. For example, in the RLR framework, the agents follow certain rules, some simple and some complex ones (in the case of multiple options leading to different decision points, e.g., adding concepts, which needs the combination of several rules to find a place, check the validity, and so on).

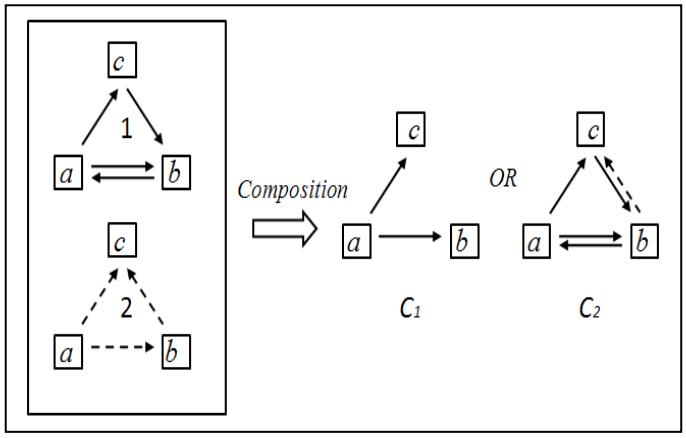

Figure 4. The composition of two initial agents' action graphs through conjunction (C_1) or adjunction (C_2). As can be seen in C_1 emphasizes are on common paths within the two action graphs, while in C2 the focus is on sum of the available paths.

As shown in Figure 4, the two graphs 1 and 2, respectively denoting the (partial) state diagrams of agents A_1 and A_2 with nodes, represent the state and edges symbolizing the transitions. These two agents have their own opinions about the set of states in a change management process, which may differ with each other in some particular cases. To achieve the compositions of the two agents' views on performing a task, one can follow several options including conjunction or adjunction [12]. As the above figure shows in these types of compositions, the origin of transitions (arrows) might not be preserved. This approach can be generalized to support composition of more than two action graphs with more complex structures.

[2] Our example is inspired from the communication between the cognitive units presented in [11].

[3] As an example of this inferred knowledge, one may notice the simple composition of morphisms in Figure 3 such that for instance: $1 \rightarrow 2 \rightarrow 3$ implies $1 \rightarrow 3$.

[4] They may be represented by UML, state transition diagrams, Petri nets, or finite state machines, to name a few possibilities.

Using categories to enhance the learning process has been also addressed in [15] by measuring and comparing the relative sizes of classes of inferable sets of functions based on inductive inference. To define the semantics of agents' protocols, we describe a set of pre- and post-conditions that need to be satisfied before/after the occurrence of a particular action or actions. Then the categorical semantics can be used to model different interaction protocols within a general dialectic framework. Currently, there are a few approaches for using categorical semantics for describing agent interactions. One of these approaches [16] has focused on denotational semantics, considering the protocols abstracted away from the type and the nature of the interaction results. Pfalzgraf [17] has proposed a distributed logical ground based on category theory, the concept of logical fiberings [18], and many-valued logics [19] for modeling multi-agent communications. In summary, the idea in this approach is to allocate a local logic (logical fiber) to each agent and make the fibering (global logical state space) out of the group of all the fibers over the base space of agents.

In the RLR framework the semantics of an evolving agent-based system can be captured through a category of states and a set of operational transitions *Op:* $St_m \rightarrow St_n$, representing that the state St_m can change into St_n by performing an operation *Op*. As illustrated in Figure 5 each individual agent (e.g., $A_1 \ldots A_n$) can make a transition using a function (e.g., f_n, g_n, …), which force the transition of MAS to the new states through the operation arrows (e.g., *f* or *g*).

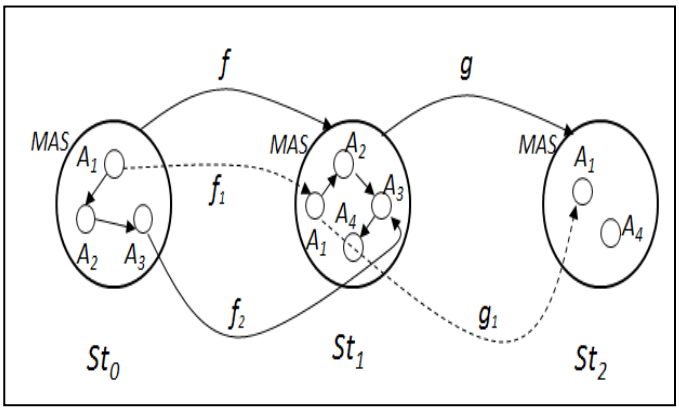

Figure 5. The representation of a multi-agent system (MAS) transitions to different states using different operations and in different levels of abstractions.

The interactions in RLR can be studied through a category with a set of states (***St***) denoting the points, a set *M* of possible message expressions, and a transition morphism T (product of states and massages). The current existing formalisms seem sufficient to model the interaction protocols for a relatively small set of interactions, but as the number of messages, exchanged expressions and potential interactions between multiple levels of nesting increase, it is far from trivial to manage all the prospective arrangements.

Categories support the agents' rule interactions with no need for deep architectural and procedural nesting. As a simple example, let us look at the composition operation (○), which can be used to formalize the declarative rule interactions for agents. For instance, one may need to define a situation in which an agent should decide about the deletion of a node in an ontology. Since the rules are not isolated in RLR by using the composition operation (○), we can represent: $R_S \circ R_P \circ R_R \circ R_D$, where R_S is a morphism denoting "select node command", R_P is "parent checking condition", R_R is "remove child morphism", and R_D is the action (e.g., deletion) to be taken in the next move.

C. *Modeling Argument Trees*

Analyzing the dependencies and legitimacy of a claim in an argument should be performed within a logical structure. Toulmin [20] described an argument based on the Claim and Data supporting this Claim, a Warrant to infer the Claim from the Data and Backing to support the materials that support the Warrant, a Qualifier to represent the soundness of an argument with uncertainty, and a Rebuttal (Reservations) to represent the exceptional cases (Figure 6).

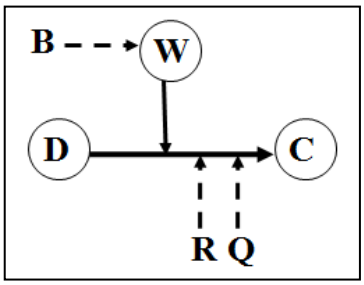

Figure 6. The Toulmin's layout for argumentation, with C, D, W, B, R and Q denote respectively Claim, Data, Warrant, Backing materials, Rebuttal, and Qualifiers[5].

Toulmin's description of argumentation trees have been adopted as one of the preferred vehicles for representing an argumentation framework through two or more contradicting structures where the roots, the nodes, and the edges respectively denote a claim, the grounds (supporting information), and the warrants (rules). Many of the uncertain and arguable grounds can be considered sub-claims, which are supported by a set of nodes (grounds). Figure 7 shows an example of such a tree.

As it can be seen in the tree shown in Figure 7, each branch has been associated with an argument about the claim (root) and its interactions with other branches (other arguments) form the argumentation structure. Currently several tools are available for creating such argument diagrams (e.g., Araucaria [21]). Toulmin argumentation diagrams mainly focus on the static representation of arguments, but they have been also extended to reflect the evolving nature of argumentations in various domains (e.g., the dialogue game [22, 23]).

[5] Adapted from: "A Description of Toulmin's Layout of Argumentation" http://www.unl.edu/speech/comm109/Toulmin/layout.htm

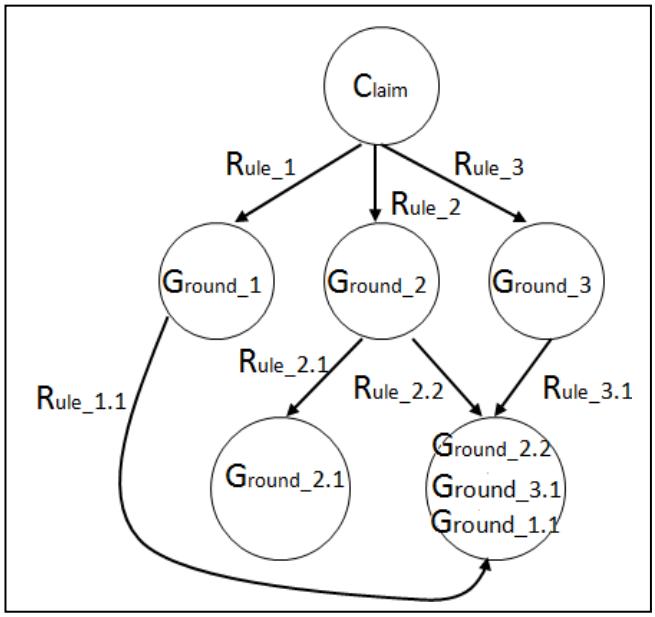

Figure 7. A partial representation of a tree-like dialectically grounded argumentation structure. In this structure C represents the claims (e.g., Ontology O is a formal Ontology); G denotes grounds (e.g., Ontology O is written in OWL-DL); and R represents the warrants or rules (e.g., OWL-DL ontologies are formal ontologies).

In RLR, we also define categories of arguments with each category including the arguments as objects interacting within an argumentation framework and the interactions between them as morphisms. Our initial plan to design a categorical model for RLR agent protocols starts with creating a graph for potential messages exchanged by the agents. Consider the category $C_{Communication}$ with a set of *time points* as objects and *message expressions*, usually placed in argumentation protocols, as morphisms. The morphisms represent the expression needs for argumentation between two time points (simply denoting the start and end of an argumentation). Thus, a communication for a protocol in the argumentation framework can be simply modeled by a sequence of morphisms and their compositions. Functors and natural transformations can be used to define different assignments between various categories. Here we are also able to generalize the communications between different protocols (e.g., two categories C_{Comm}, $D_{Comm.}$) using functors (i.e., F: $C_{Comm.} \rightarrow D_{Comm}$). In existing agent languages such as FIPA-ACL [6], the messages exchanged between the agents may consist of requests and notifications, for example, without the possibility to define any combination rules; while in our approach we can define the rules' compositions in various levels of abstractions. This observation has been also studied in [24] from a different perspective [7], where every MAS diagrammatic topology has been interpreted as a category PATH where the nodes are the objects and every sequence of consecutive arrows (a path which may include more than one single arrow) in the diagram is a morphism[8]. Based on [24], a base diagram, which is a category PATH, has been associated with each MAS to represent the general attributes and organization of related communication channels (arrows) for that MAS.

As mentioned in Section III.A, the arrows in the category of agents (morphisms) convey a communicative operation of forwarding a message from one agent to another. A category of such arrows together implies an argument framework starting with an initial action and ending with a final decision (i.e., one may consider the classical example of auctioning). The set of rules provides sufficient expressiveness for the argument framework (e.g., winning_bid ≥ starting_bid). Each communication protocol can be considered a reusable pattern, which is "formally defined and abstracted away from any particular sequence of execution steps" [25], and can be applied to other frameworks with different purposes. The categorical representations along with the graphical transformation greatly resemble UML diagrams (specifically state and activity diagrams) while providing more expressivity in terms of the underlying semantics.

IV. Category Theoretical Distributed Graph Transformation

The transformation rules can be used to determine and model agents' behaviors and operations in MAS [40, 41]. They also capture the effects of different agents' actions and operations on local or network levels, thus as a representation method, these rules enable modeling the agents' cooperation and interactions. When we consider graph transformation for formalizing agents' interactions and cooperation by means of communication with the other agents within a specific MAS or between different MASs and with their environment, it can be used for representing the transformation of the agents' communication network. To analyze changes in relations between a set of cooperative agents within a generic multi-agent system (MAS), considering the category of MAS, a transformation mechanism based on DPO can be defined by finding a pushout complement for a particular state through examining the gluing condition, consisting of dangling condition and identification condition [14].

Considering the challenges for modeling changes in distributed systems, which involve several issues including traceability and synchronization, RLR utilizes a distributed graph transformation technique, which explicitly supports the synchronization and concurrency processes. For the sake of consistent change management, a process within the RLR model needs to be synchronized with its adjacent processes in order to evolve coherently. For example, if two operations want to act on a common ontology through specific actions and conditions, these actions should act under a consensus agreement so they can both perform and evolve coherently. To coordinate the potential changes in the processes, a set of synchronization requests are issued at each abstraction level. These requests need to follow certain transformation rules and

[6] Standardized by the Foundation for Intelligent Physical Agents: http://www.fipa.org/repository/aclspecs.html

[7] In the study performed in [24] the authors focused on the communications between different MASs rather than the dialogues between individual agents.

[8] This actually implies the composition of morphisms.

conditions, which are compatible with each other[9], in such a way that they support the concurrent evolutions of different parts of the system autonomously based on the consensus agreement. To ensure the consistency of the transformations, we enforce certain types of reactions and behaviors (preferably among several options) for agents in certain states, when the conditions are applicable (determined by L in the rule $L \leftarrow I \rightarrow R$). The overall effect of an action within a scenario (e.g., select a node to be deleted) is described by a pair of instance diagrams[10], modeling the before/after states [30]. Sequences of transformations represent the changes in the states' agents and their behavior, and model their interactions within the communication channels in a MAS[11]. For example using the method presented in [30], we consider the communications between the Explorer Agent (EA) and the Log-Reading Agent (LRA), which together capture changes by processing and reading change logs in parallel, in a specified time range, in RLR (described in [3]) to capture the type of change operation (Figure 8).

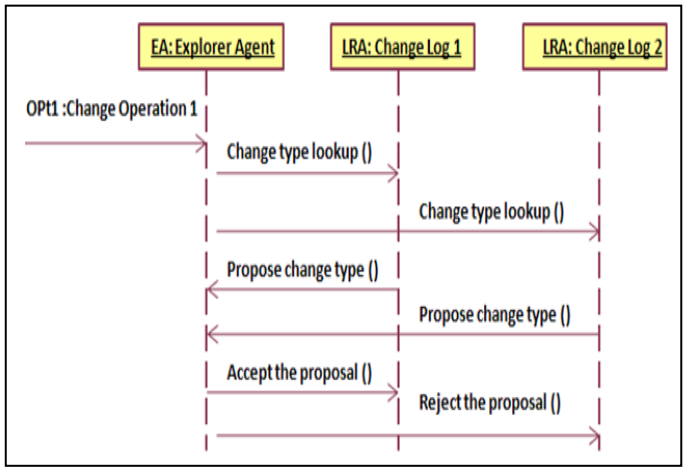

Figure 8. The communication between explorer and log reading agents to specify the type of a change operation. These communications can be placed during the negotiation phase.

Due to the ability of graph transformation for handling temporal representation [31], we have used the rule-based graph transformation [30] to describe the pre- and post-states of an agent-based model (Figure 9), grounded on the communication diagram demonstrated in Figure 8. In this figure, the Explorer Agent (EA) reacts to alterations that appear in the environment (e.g., a change operation) and tries to affect the environment by locating the change and determining its type based on different proposals. In the same way, one can define other rules for rejecting the proposals, storing the proposals for future decisions, or aborting all the communications.

[9] The compatibility here refers to this fact that the combinations of these transformation rules should keep the entire system in a consistent state.

[10] In UML, instance diagrams (object diagrams), are useful for exploring "real world" examples of objects and the relationships between them, while the type diagram reflects a given Use Case [27].

[11] In agile object oriented modeling, this usually is represented by UML sequence diagrams [28].

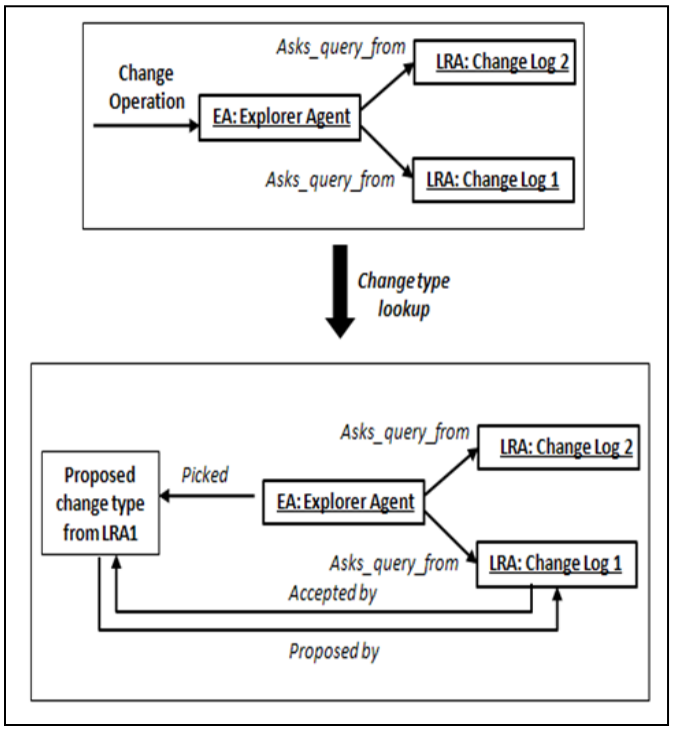

Figure 9. A generic transformation rule for describing the pre- and post states in an agent-based model transformation based on the communication diagram demonstrated in Figure 8.

By noticing the fact that many of the current dynamic agent models are represented by sequence and state diagrams, which have been studied here under a graph-oriented approach as well, we can extend our approach to study agents' model transformations in more complex situations. For example, by following some of the object-oriented principles like differentiation between instance and type graphs (diagrams)[12] [32, 33], we can model the typed graph transformations [34, 35] by means of refactoring [36, 37, 38] the state diagrams (adding/removing, merging, or decomposing the states) and conceptual models of ontological structures.

As an example, following the approaches presented in [39] and [38], we may merge the two states St_3 and St_4, which respectively represent the state of the RLR system after querying to determine the type of change and receiving the proposed answers, into one merged state $St_{3,4}$ (Figure 10 (a)). As another example, Figure 10 (b) demonstrates the transformation of a state diagram to the new diagram, representing the concurrencies between the two states St_5 (validation of the accepted response), and state St_6 (ask permission to put the results into an action). In fact, we would be able to model various aspects of agents in the RLR model, such as agents' networks, topology, properties, interactions, and cooperation based on the agreed goal in the negotiation process. Also, due to the rule-based nature of this framework, we can formally model the structure and behavior of an

[12] In conceptual modeling a type graph models a class diagram and an instance graph models the objects (instance) diagram.

evolving system and anticipate certain types of transformations and re-configurations upon future changes.

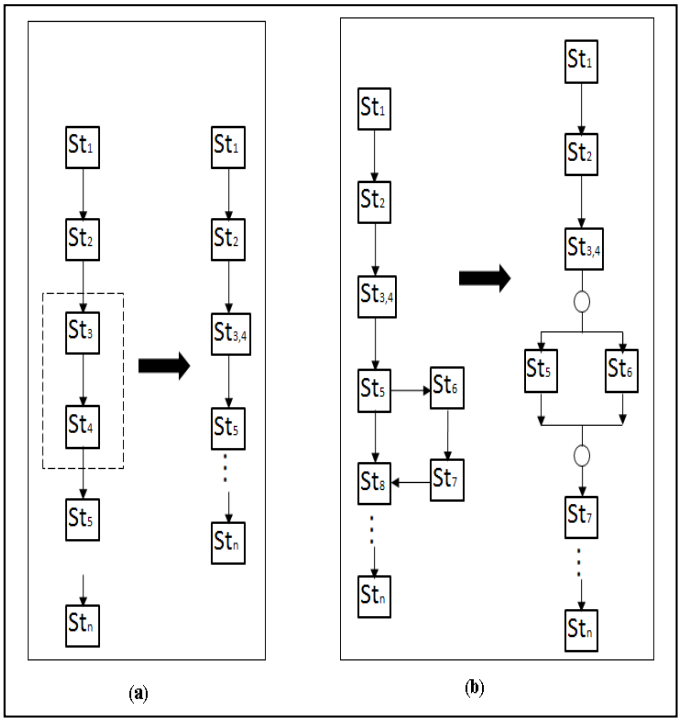

Figure 10. (a) pre/post state representation before/after merging two states; (b) the representation of concurrency of two parallel states.

V. DISCUSSION

Categories have been extensively used in mathematics and theoretical computer science to assist in separating the levels of abstractions and integration of generic components. The categorical method to study and measure changes and to test several hypotheses and certain effects on developmental change between two time points has been applied in biological and social analysis as well as in psychological [29] domains. However, the applications of category theory to formalize agent-based biomedical ontology change management are extremely rare. Our major contribution for this paper is extending the semantics for change management process within the RLR agent-based framework, by defining a categorical framework to support agents' communication, negotiation (e.g., formalizing dialectic trees), state transitions, compositions and transformations (e.g., rule transformation) in different levels of abstractions (e.g., agents and MAS). We have also employed graph transformation to facilitate concurrency and synchronized analysis of evolving ontological structures. For this purpose we have defined several categories including categories of agents, multi-agents systems (MASs), services, states, rules and prepositions.

REFERENCES

[1] D. Shahaf, and E. Amir,"Towards a theory of AI completeness," In Proc. of 8th Int'l Sympo. on Logical Formalizations of Commonsense Reasoning (Commonsense'07), in AAAI Spring Sympo., California, USA, 2007.

[2] T. Aven, "Identification of safety and security critical systems and activities," Reliability Engineering & System Safety, 94(2): 404–411, 2009.

[3] A. Shaban-Nejad, and V. Haarslev, "Incremental biomedical ontology change management through learning agents," In Proc. of the 2nd KES Intl. Symposium on Agent and Multi-Agent Systems (KES-AMSTA 08), March 27–28, Incheon, Korea, Springer Vol. 4953/ 2008: 526–535.

[4] A. Shaban-Nejad, and V. Haarslev, "Strategic health information management and forecast: the birdwatching approach,". In proc. of 2nd Intl. Conference on Computational Collective Intelligence (ICCCI'10), Taiwan, Springer LNCS 6423, 2010: 457-468.

[5] A. Asperti, and G. Longo, "Categories, types, and structures: an introduction to category theory for the working computer scientist," The MIT Press, 1991.

[6] I. Rahwan, and P. McBurney, "Guest editors' introduction: argumentation technology,". IEEE Intelligent Systems 22(6): 21–23, 2007.

[7] J. Lind, "Specifying agent interaction protocols with standard UML," In Proc. of AOSE'01, LNCS 2222, Springer, pp.136–147, 2001.

[8] M. Purvis, S. Cranefield, M. Nowostawski, and M. Purvis, "Multi-agent system interaction protocols in a dynamically changing environment," An Application Science for Multi-Agent Systems, Springer, pp. 95–111, 2004.

[9] H.R. Dunn-Davies, J. Cunningham, and S. Paurobally, "Propositional state charts for agent interaction protocols," Electr. Notes Theor. Comput. Sci. 134: 55-75, 2005.

[10] N. Fornara, and M. Colombetti, "Defining interaction protocols using a commitment based agent communication language," In Proc. of AAMAS'03, ACM Press, pp. 520–527, 2003

[11] J.R. Anderson, "Concepts, propositions, and schemata: what are the cognitive units?" In J. Flowers (edi.) Nebraska Symposium on Motivation. Lincoln, Nebraska: Uni. of Nebraska. 1981.

[12] M. Chechik, and S. Easterbrook, "Reasoning about compositions of concerns," In Proc. of the Workshop on Advanced Separation of Concerns in Software Eng. at ICSE'01, 2001.

[13] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer, "Fundamentals of algebraic graph transformation," Monographs in Theoretical Computer Science," An EATCS Series, Springer, 2006.

[14] H. Ehrig, M. Korff, and M., Löwe, "Tutorial Introduction to the Algebraic Approach of Graph Grammars Based on Double and Single Pushouts". In Proc. of 4th Int'l Workshop Graph-Grammars and Their App. to Comp. Sci., Bremen, Germany, LNCS 532, Springer, 1990 pp. 24-37.

[15] L. Fortnow, R. Freivalds, W.I. Gasarch, and M. Kummer et al., "Measure, category and learning theory," Proc. of the 22nd Colloq. on Automata, Languages and Programming (ICALP'95), Szeged, Hungary, LNCS 944, Springer, 1995, pp. 558–569.

[16] M.W. Johnson, P. McBurney, and S. Parsons, "A mathematical model of dialog," Electr. Notes Theor. Comput. Sci. 141(5): 33-48, 2005.

[17] J. Pfalzgraf, "On categorical and logical modeling in multiagent systems," in proc. Of InterSymp-2004, Baden-Baden, Germany, 2005.

[18] J. Pfalzgraf, "On logical fiberings and automated deduction in many-valued logics using gröbner bases," RACSAM, Rev. Real. Acad. Ciencias, Ser. A. Mat., 98(1): 213–227, 2005.

[19] S. Gottwald, "Many-valued logics," In: Handbook of the Philosophy of Sciences. Vol. 5: Philosophy of Logic (D. Jacquette ed.), North-Holland: Amsterdam, 2007, pp. 545-592. available at: http://www.uni-leipzig.de/~logik/gottwald/SGforDJ.pdf

[20] S. Toulmin, "The uses of argument," Cambridge University Press, 1958.

[21] C. Reed, and G. Rowe, "Araucaria: software for argument analysis, diagramming and representation," International Journal on Artificial Intelligence Tools 13(4): 961-979, 2004.

[22] T.J.M. Bench-Capon, "Secification and implementation of Toulmin dialogue game," In Proc. of Legal Knowledge-Based Systems. JURIX: The 11th Conference, 1998, pp. 5–19.

[23] T.J.M. Bench-Capon, T. Geldard, and P.H. Leng, "A method for the computational modelling of dialectical argument with dialogue games. Artificial Intelligence and Law, 8(2-3): 233–254, 2000.

[24] J. Pfalzgraf, and T. Soboll, “The base diagram of a multiagent system: a categorical model of the general communication structure,” In (proc. Intersymp2007) Laker., G.E., and Pfalzgraf, J. (editors) Advances in Multiagent Systems, Robotics, and Cybernetics: Theory and Practice. (Vol. II) Published by IIAS, ISBN 978-1-897233-61-0. 2008.

[25] M. Wooldridge, N.R. Jennings, and D. Kinny, “The Gaia methodology for agent-oriented analysis and design,” Autonomous Agents and Multi-Agent Systems 3(3): 285–312, 2000.

[26] F. Drewes, B. Hoffmann, and D. Plump, “Hierarchical graph transformation,” *J. Comput. Syst. Sci.* 64(2):249-283, 2002.

[27] UML 2 Object Diagrams

http://www.agilemodeling.com/artifacts/objectDiagram.htm

[28] UML basics: The sequence diagram

http://www.ibm.com/developerworks/rational/library/3101.html

[29] M. Ato, E. Ato, and J. Gómez, “Analyzing change among developmental stages with categorical models,” Quality and Quantity 39(1): 87–108, 2005.

[30] R. Depke, R. Heckel, and J.M. Küster, "Formal agent-oriented modeling with UML and graph transformation,” Sci. Comput. Program. 44(2): 229-252, 2002.

[31] S. Gyapay, D. Varró, and R. Heckel, “Graph transformation with time,” Fundam. Inform. 58(1): 1–22, 2003.

[32] A. Corradini, U. Montanari, and F. Rossi, “Graph processes,” Fund. Inform. 26(3,4): 241–266, 1996.

[33] L. Baresi, R. Heckel, “Tutorial introduction to graph transformation: a software engineering perspective,” In Proc. of the 2nd Int'l Conference on Graph Transformations (ICGT'04), Rome, Italy, LNCS 3256, Springer, 2004, pp. 431–433.

[34] R. Heckel, A. Corradini, H. Ehrig, and M. Löwe, “Horizontal and vertical structuring of typed graph transformation systems,” Mathematical Structures in Comp. Science 6(6): 613–648, 1996.

[35] M. Große-Rhode, F. Parisi-Presicce, and M. Simeoni, “Spatial and temporal refinement of typed graph transformation systems,” In Proc. of MFCS'98, LNCS 1450, Springer, pp. 553–561, 1998.

[36] G. Sunyé, D. Pollet, Y. Le Traon, and J.M. Jézéquel, “Refactoring UML models,” in: «UML» 2001 — The Unified Modeling Language. Modeling Languages, Concepts, and Tools. LNCS 2185, Springer, 2001, pp. 134-148.

[37] T. Mens, N. van Eetvelde, S. Demeyer, and D. Janssens, “Formalizing refactorings with graph transformations,” Journal of Software Maintenance 17(4): 247–276, 2005.

[38] T. Mens, “On the use of graph transformations for model refactoring,” In: Proc. of GTTSE'05, LNCS 4143, Springer, 2005, pp.219-257.

[39] M. Boger, T. Sturm, and P. Fragemann, “Refactoring browser for UML,” In Proc. of NetObjectDays'02, LNCS 2591, 2002, pp. 366–377.

[40] P. Knirsch, and H.J. Kreowski, “A note on modeling agent systems by graph transformation,” In Proc. of AGTIVE'99, LNCS 1779, Springer, 1999, pp. 79–86.

[41] R. Depke, R. Heckel, and J.M. Küster, “Agent-oriented modeling with graph transformation,” In Proc. of AOSE'00, LNCS 1957, Springer, 2000, pp. 105-120.

[42] H. Ehrig, M. Pfender, and H.J. Schneider, “Graph grammars: an algebraic approach,” In: *Proc. of 14th Symposium on Foundations of Comp. Science*, IEEE, 1973, 167-180.