Fuzzy Learning and Behavior in Multi-Agents Based Interactive Systems

Dorian Gorgan
Computer Science Department, Technical University of Cluj-Napoca
26, G. Baritiu St., RO-3400 Cluj-Napoca, Romania,
tel: (+40)-64-194835, fax: (+40)-64-194491, E-mail: gorgan@utcluj.ro

Abstract

The structure and the functionality of a multi-agents based model is presented. The model requirements imposed by interactive techniques are highlighted. The model consists of active entities called agents, and passive entities such as behaviors, trajectories, actions, and conditions. The agents have explicit defined behaviors as spatial and temporal evolutions. The agents use fuzzy rules for an intelligent and adaptive behavior and a run-time learning. A consistent set of actions and fuzzy rules for a dynamic learning are defined. The model evolution is a combination of agent behaviors. The agents communicate by messages to change information and to cooperate throughout their activities. The evolution is connected to a trajectory that is defined by parameters, positions, rules, and actions. The trajectory provides the possibility for direct manipulation on the model entities, on their elements and on the abstract notions. That means a proper technique for applications such as visual programming, courseware authoring, rapid prototyping, and so on. The paper emphasizes on a proper structure of a multi-agent model having direct manipulation facilities.

Key words: multi-agents, fuzzy rules, cooperative learning, trajectory, direct manipulation, message communication.

1 Introduction

The interactive applications use direct manipulation to provide to user the possibility to operate on application entities. In applications such as visual programming, courseware authoring, dynamic simulation or rapid prototyping, the application developers can use direct manipulation technique to define the model evolution.

The paper attempts to find answers to the following questions:

1. what is the proper distributed structure for a dynamic model with a parallel, concurrent and adaptive evolution ?
2. how intelligent multi-agent model provides behavior definition using direct manipulation and a dynamic behavior execution ? What is the significance of the trajectory notion in the behavior definition and execution ?
3. how fuzzy logic could improve the intelligence of agents in interactive systems ?
4. what is the consistent and minimal set of actions need for an agent to execute a cooperative task ?
5. how can be used fuzzy knowledge in distributed and cooperative learning ?

2 Multi-Agent Model

The distributed and dynamic model consists of active entities which are *agents* and passive entities such as *behaviors, trajectories, rules, conditions*, and *actions.*

2.1 Agent Structure

Agents are active entities with private behavior. The evolution implies the execution of an associated behavior on a specific trajectory with a well defined state and parameters. The formal structure of an agent, is the following:

```
agent (
        name,
        position,
        parent_agent,
        son_agents,
        attributes,
        active_flag,
        visibility_flag,
        behavior_name,
        processors (
                behavior,
                server,
                display,
                interactors
        )
)
```

The agent name is unique defined. Every agent may be a prototype for an instance agent which has a different name and a private position. The position denotes the explicit position in a virtual space. When the agent moves on the trajectory its current position is relative to the start position or to the parent-agent position. An agent having an associated behavior executes its evolution only if the active_flag is true. The visibility_flag specifies the agent is visible or not during its evolution.

An agent can be simple agent or a complex agent called aggregate. For simplicity we will use for both types the same common name agent. An agent may be a component of another complex agent called parent_agent or parent. The components are called son_agents. An agent inherits from its parent all the characteristics - attributes, active_flag, visibility_flag, behavior, if it does not have another explicit definitions for them.

For the reason of execution speed there are two types of attributes: graphics and application oriented. Graphics attributes may be for instance: color, line pattern, dimensions, orientation. The application oriented attributes designate for instance: weight, age, temperature, nominal current, knowledge. An attribute is defined by three elements: name, value type and value. The type specifies the type of attribute value: *integer, float, string*, and *fuzzy*. Any attribute and its elements may be inquired or modified.

The evolution of an agent is performed by its four type associated processors: *behavior, server, display* and *interactor*. The behavior processor performs the private agent behavior. The server processor performs the delegated actions requested by own behavior processor or by other active entities. The display processor creates the presentation (sound, graphics, images) of the owner agent. The interactor processors perform the interface between agent and external modules like user, application program, or graphical systems. Interactors access the agent state according to the value of the interface signals.

2.2 Behaviors

Every active agent has an associated behavior. The same behavior may be associated to one or to a set of agents. A behavior is a model entity with a name as an unique identifier. Its structure provides elements for direct manipulation such as name, and position inside an virtual coordinates system. The behavior is characterized by parameters such as the number of moving steps among two explicit positions on the trajectory, other application parameters (delay time, blinking rate, etc.), and a set of trajectory positions with associated rules.

The behavior is dynamic entity that may be operated by model developer or by the model agents through their actions. A behavior may be created, instanciated, destroyed, associated to an agent, and modified. The behavior can be linked to different sets of trajectory positions. Two behaviors having the same trajectory set may differ by behavior parameters.

2.3 Trajectories

The trajectory defines the evolution of an associated agent within virtual space. The main goal of trajectory is to provide a concrete representation and direct manipulation on abstract notions. Using the space representation the notion of successive moments of time are indirect translated into successive explicit positions. Consequently, the synchronization between agents in different moments of their evolution is similar with relations between agents on specified positions. A formal definition of trajectory is as following:

```
trajectory (
        position_name,
        position,
        rule_set
)
```

Each rule incorporates conditions and associated actions. The rules execution from an explicit trajectory position means all the associated actions fire if the condition is true and the current agent there is on that position. Between two explicit position the agent has a smooth motion and a speed defined by its behavior. An agent accepts requests as delegated actions on any intermediate position, but performs its private rules only on explicit positions.

2.4 Rules

The experimented multi-agent model is a rule based model. A rule has a name and consists of a condition and a set of actions. If the evaluated condition is true the actions are performed in a sequential order. In the user interface environment the rule can be identified by its name.

2.5 Conditions and Expressions

Conditions are logical expressions which reflect the state of the dynamic model. A condition may be defined in terms of agents, behaviors, positions, and their elements which have name, position and graphical presentation. Therefore the whole syntactic form of a condition may be built using the direct manipulation technique.

2.6 Actions

A minimal set of 12 actions have been experimented: *create, delete, instanciate, append, get, put, activate, show, jump, rotate, translate,* and *scale*. The first four actions control the creation and destruction of model entities. *Get* and *put* actions inquire and modify the content of the model entities. The *activate* and *show* actions control the motion and the visibility of the model entities. The action *jump* controls the sequence of trajectory positions. There are also three graphical actions such as *rotate, translate* and *scale* which control the graphical attributes of agents and their presentation.

Every action is an entity defined by the following elements: operator, input_entities, output_entities, and error. The operator acts on the input_entities and provides the output_entities. Input and output entities are defined in terms of model entities and their elements. For the reason of a pessimistic synchronization and for respecting the property of serializability, any operation on the model elements must be performed only by delegation.

3 Fuzzy Knowledge

The model behavior is related to the state of model entities and their defined evolution. The state of agents are codified in attributes and their values. There are two types of attributes: *crisp* and *fuzzy*. Attributes denote the knowledge an agent has about its universe. The presented work highlights the properties of fuzzy knowledge based agents. The possibility of fuzzy learning is also investigated.

An attribute has a formal structure such as (id, type, val). id denotes the attribute name, type may be integer, real, string, and fuzzy. The element val specifies the attribute value according to its type.

The element val for a fuzzy attribute is a *membership function*. The membership function denotes what is the *degree of truth* of the knowledge K, described by the attribute id. Let define the notions we use.

Let be U the set of values for attribute id and A subset of U. According to a given knowledge K, let be a set of ordered pair (ai, mai), where ai is an element of the set A and mai is a real value in the interval [0, 1]. For a specific knowledge K the mapping of the set U to the interval [0, 1] is described as a function called the *membership function* of A. The fuzzy subset A is a *linguistic variable* which represents the cognitive category of knowledge K for a given agent, and the set U is the *universe of discourse*. For each value ui of atribute id, the fuzzy value mui means the degree of truth of the statement "ui is A", or "id with value ui is A". The elements of U are ordered using a K knowledge based metrics.

Example: Let be four agents M1, M2, M3, and M4 placed on the vertices of a square. Each of them has the knowledge about the ratio of distances to the rest of agents, and about the angles made by the rest of agents and having the corner in the owner agent.

For distance ratio dr the universe of discourse is U1 = [0, 2], and the linguistic variable A1 = {1, sqrt(2)}. For angle a the universe of discourse is U2 = [0, 180], and A2 = {45, 90}.

The ordering concept of elements from universe of discourse has a great significance in the knowledge propagation between neighbor elements.

4 Cooperative Learning

Model agents achieve their evolution through a cooperative behavior execution. To evaluate a rule condition an agent needs knowledge from other agents. It requires knowledge by the action get. Also, an agent may improve the knowledge of another agent sending to it knowledge obtained from its experience and evolution. To send knowledge an agent uses the action put.

4.1 Knowledge Requiring

During model execution the agents need knowledge from other agents. For example, a fuzzy expression such as:

x is A1 and y is A2

returns a degree of truth that is the MIN operator on mx and my. Let assume the agent M3 needs the knowledge 'x is A1' from agent M1 and the knowledge 'y is A2' from agent M2. M3 evaluates the expression executing two actions get:

```
get     (agent M1), (attr dr(x) is A1)
get     (agent M2), (attr a(y) is A2)
```

A fuzzy expression such as:

x is A1 and y is A2 then z is A3

may be translated in three actions get. Let suppose the knowledge 'z is A3' is located in agent M4. The fuzzy value mz = MIN(mx, my) will be sent to M4 to request the crisp z. The fuzzy value mz will clip the membership function of A3 from M4, computes z by defuzzyfication (i.e. the method centroid), and returns the crisp z.

To compute a set of rules such as above, the defuzzification becomes a distributed operation. The crisp z will be computed from a few membership functions from different agents. That is the reason for requiring by operations get two centroid coordinations (zi, mzi) to compute the centroid of z. For example, in the case of two rules, the crisp z is: z = (z1 * mz1 + z2 * mz2) / (mz1 + mz2)

4.2 Knowledge Improvement

An agent may improve knowledge of another agent sending a knowledge to it. The sent *knowledge quantum* is an elementary membership function which is added to the membership function of a target attribute from an agent. The maximum fuzzy value w of the quantum means the possibility of agent to assimilate new knowledge. The width of elementary triangle describes the propagation of a new information between elements of universe of discourse.

An agent sends a knowledge quantum by action put:

```
put     (agent M1), (attr dr(x))
```

The degree of truth for an attribute and specified value may be improve repeating the same action put.

The significance and the possibilities of knowledge quantum modeling have to be explored by other many examples.

4.3 Fuzzy Learning

The experiments emphasize on a cooperative learning *with examples* and a *run-time learning*. For instance, the above-mentioned agents can learn the shape of a square throughout a learning phase, when the model developer draws a lot of square as examples. The agents collect knowledge about their universe and build the membership functions. The run-time learning builds the membership functions by the execution of agent behavior, and the execution of actions put.

The proposed model investigates two type of run-time learning: *implicit* and *explicit*. The implicit learning executes every operation get by a pair of operation consisting of get and put. The knowledge is improved by reading. The natural significance of implicit learning concerns on the mental process of knowledge refreshing by recapitulation. Explicit learning is accomplished by explicit actions put throughout the agent behavior execution. The knowledge improvement is an explicit defined operation.

The experiments highlight also, the significance of *saturation* process within a membership function improvement. The saturation is processed by *normalization* or by *high-end clipping* technique.

5 Implementation

We have used C++ Language and MS-Windows to implement the multi-agent model. Communication mechanism is implemented by messages transmitted through agent private queues.

We are currently building more complex interactive techniques for assisting users in performing tedious tasks. We are also developing the cooperative and distributed learning in multi-agent model used in graphical user interfaces.

6 Conclusions

The experiments highlights the significant potential of fuzzy logic in adaptive and distributed behavior definition.

The multi-agent model reveals the usefulness of topological elements in interactive applications such as visual programming, multimedia, courseware authoring, and rapid prototyping.

A great significance is assigned to the speed of model execution, related to the knowledge and behavior modeling.

7 References

[1] Arbab, F., Herman, I., Reynolds, G.J.: 'An Object Model for Multimedia Programming', *Computer Graphics Forum*, Vol. 12, No. 3, pp. C-101 - 113, 1993.

[2] Chin, R.S., Chanson, S.T.: 'Distributed Object-Based Programming Systems', *ACM Computing Surveys*, Vol. 23, No. 1, pp. 91-124, March 1991.

[3] Duke, D.J., Harrison, M.D.: 'Abstract Interaction Objects', *Computer Graphics Forum*, 12(3), pp. 25 - 36, 1993.

[4] Gorgan, D.: 'Object Behavior Modelling in Graphical Systems'. *Research Report*. Rutherford Appleton Laboratory, UK, May 1995.

[5] Gorgan, D.: 'Dynamic Model Functionality'. *Research Report*. Technical University of Cluj-Napoca, August, 1995.

[6] Mullender, S.J., ...: 'Amoeba A Distributed Operating System for the 1990s'. *Computer*, Vol. 23, No. 5, pp. 44-53, May 1990.

[7] Tanenbaum, A.S.: '*Modern Operating Systems*'.Prentice-Hall, Inc. 1992.

[8] Yager, R.R., and Zadeh, L.A.: '*An Introduction to Fuzzy Logic Applications in Intelligent Systems*', Kluwer Academic Publishers, 1991.

[9] Yamakawa, T.: '*Stabilization of an Inverted Pendulum by a High-Speed Fuzzy Logic Controller Hardware System*', Fuzzy Sets and Systems 32(1989) pp. 161-180, North-Holland Publ. Comp., Amsterdam, 1989.

[10] Zadeh, Lotfi: '*Fuzzy Sets*', Information and Control, 8:338-353, 1965.

[11] Zadeh, Lotfi: '*The Calculus of Fuzzy Restrictions*', in Fuzzy Sets and Applications to Cognitive and Decision Making Process, edited by L.A. Zadeh et. al., Academic Press, pp. 1-39, New-York, 1975.