Development of formal infrastructure for perception of intelligent agents as problem solvers

Ph. D. Marijan KUNSTIC, B. Sc. Marina BAGIC
Department of telecommunications, Faculty of electrical engineering and computing
Zagreb, Croatia
E-mail: marijan.kunstic@fer.hr, marina.bagic@fer.hr

ABSTRACT

There are few important reasons for introduction of intelligent mobile agents model in telecommunications. These are, but not limited to: transport platform development which provides us with a high level of flexibility during communication process and »softwaresation« of telecommunications. Introduction of mobile agents generates search for appropriate formal system to develop their efficient formal specification. Here, we present formal model for specification of mobile agents in telecommunication network management. Network management can be perceived as process of decomposite problem solving where each network problem is decomposed to smaller ones called primary problems. Primary problems are then objects of problem solvers. Two aspects of formal specification of network mangement system based on mobile agents are considered here: static and dynamic.

Keywords*:* Multiagent systems, Formal specification, Problem space, Solver space, Object-oriented techniques.

1. INTRODUCTION

Since intelligent mobile agents inherently solve problems, they are introduced here to solve netwok management tasks. Here we have tried to make their formal infrastructure based on their numerous features such as: schedule within the network environment, their dedicated priorities, thier probabilities of appearance at particular network locations and their interagent relations.

Static and dynamic aspects of formal specification of network mangement system based on mobile agents are considered here.

Analysis of statics has brought us to notion of *agent relational metalevel* based on cognitive maps to visualize and analyze benefits of such approach.

Discussing dynamic aspect of a system, we introduced here extended definitions of problem and solver space. We extend the analysis started in [4]. Problem and solver spaces are visualized here as vector spaces. Problem vector space is constituted of finite numeral of linear combinations of primary problem vectors. Specified linear oparators transform these vectors onto their codomain in solver vector space. So, solver vector space consists of primary vector solvers together with all of their combinations. Linear operator may transform origin vector space either to null or none-null vectors. Those transformed to null vectors constitutes *kernel* of origin vector space and the others create its *image*. We also reason on *isomorphic* linear operators and their effects on problem and solver vector spaces.
Since primary problems and solvers are vectors themselves, they are presented here as finite n-tuples with their n-features. Their features are explored in great detail here. They are also introduced here as group and ring elements

In this manner mobile agents as problem solvers are introduced. They are presented as specified linear combinations of primary solvers. These provides us with flexible mechanism for changing agent problem domain or increase of agent’s knowledge. We analyzed here agents behaviour with emphasis on their optimal cooperation with minimal communication in decision processes and level of generalization of model’s application in different network management domains.

2. DYNAMICS IN MULTIAGENT SYSTEMS

Dynamic aspect of a system is perceived here as extension to definitions of problem and solver space in [4]. Problem and solver spaces are visualized here as vector spaces.

2.1. Agents in vector space

Let us first define vector space, one of the main issues in this paper.

Definition 2.1. A set X is a **vector space** over a field F (e. g. field of real or complex numbers), if given an operation *vector addition* defined in X, denoted $\mathbf{x} + \mathbf{y}$ for all $\mathbf{x}$, $\mathbf{y}$ in X, and an operation *scalar multiplication* in X, denoted $a * \mathbf{x}$ for all $\mathbf{x}$ in X and a in F, the following properties hold for all a, b in F and $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ in X:

- $\mathbf{x} + \mathbf{y}$ belongs to X; X is closed under vector addition;
- $\mathbf{x} + (\mathbf{y} + \mathbf{z}) = (\mathbf{x} + \mathbf{y}) + \mathbf{z}$; associativity of vector addition in X;
- $\mathbf{x} + \mathbf{0} = \mathbf{x} + \mathbf{0} = \mathbf{0}$; $\forall \mathbf{x} \in X\ \exists\ \mathbf{0} \in X$, an additive identity element;

- $\mathbf{x} + (-\mathbf{x}) = (-\mathbf{x}) + \mathbf{x} = \mathbf{0}$; for all elements $\mathbf{x}$ in X exists additive inverse $-\mathbf{x}$ in X;
- $\mathbf{x} + \mathbf{y} = \mathbf{x} + \mathbf{y}$; commutativity of vector addition in X;
- $a * \mathbf{x}$ belongs to X; X is closed under scalar multiplication;
- $a * (b * \mathbf{x}) = (a * b) * \mathbf{x}$; associativity of scalar multiplication in X;
- $\mathbf{1} * \mathbf{x} = \mathbf{x} * \mathbf{1} = \mathbf{x}$; multiplicative identity 1 of the field F; neutrality of one;
- $a * (\mathbf{x} + \mathbf{y}) = a * \mathbf{x} + a * \mathbf{y}$; distributivity with respect to vector addition;
- $(a + b) * \mathbf{x} = a * \mathbf{x} + b * \mathbf{x}$; distributivity with respect to field addition.

According to definition above we introduce here vector problem space and vector solver space.

Definition 2.2. Problem vector space VP is space of primary vector problems and all of their linear combinations.

$$VP := \wp_1\mathbf{p_1} + \wp_2\mathbf{p_2} + \ldots \wp_i\mathbf{p_i} + \ldots + \wp_{n-1}\mathbf{p_{n-1}} + \wp_n\mathbf{p_n}$$

where $\wp_{i,}$ is scalar determining some feature of vector it is assigned to; p_i, primary vectors of space basis which generate the space. Vectors p_i are ordered n-tuples where each component determines one feature of particular vector. Therefore, it is possible to define n features with values under some field, e.g. field of real numbers, ($\aleph_1$, …, $\aleph_i$, $\aleph_n$). Primary vectors are considered nondecomposite to smaller ones. Their related n-tuple features are not considered vectors.

Problem vector problem space is generated by environment of specific domain. For simplicity here are considered merely finite vector spaces. Each vector space has an arbitrary number of basis (minimum number of linearnly independent vectors) which are all equally numbered and determine dimension n of particular space.

Vector problem space needs solutions to its problems. Therefore, it implies existance of another space connected to it solving problems of solver space, therefore it is called solver vector space.

Definition 2.3. Solver vector space VS is a space of primary vector solvers with all of their linear combinations;

$$VS := \Im_1\mathbf{s_1} + \Im_2\mathbf{s_2} + \ldots \Im_i\mathbf{s_i} + \ldots + \Im_{n-1}\mathbf{s_{n-1}} + \Im_n\mathbf{s_n}$$

where $\Im_{i,}$ denotes scalar that determine a particular vector feature; s_i, primary vector of space basis. Here are examined merely finite vector spaces with different space basis. Ordered n-tuples (η_1, η_i, …, η_n) determine n features of primary problem solvers assigned to problem s_i. This defined space is generated by telecommunication network, infrastrucutre of nodes and available resources.

Each element of solver space is considered agent. Therefore agent system is defined as set of solvers with exactly defined relations between them. Or in other words, agent system is ordered relational state within collection or subset members of solver space

2.2. An agent in Group theory

Monoids characterise data types with an associative binary operation and a neutral element under the operation. Inverse elements as required for groups are not so common, so we introduce here analysis of possible approach to inverse group elements with respect to technology of mobile telecommunications agents.

Definition 2.4. A group G is defined as a finite or infinite set of operands (called "elements") that may be combined or "multiplied" via a binary operator to form well-defined products and which furthermore satisfy the following conditions:

- **Closure**; $\forall a, b \in$ G in G, the product $ab \in G$;
- **Associativity**; $\forall a, b, c \in$ G multiplication is associative: $a(bc) = (ab)c$;
- **Identity**; $\exists e$ an identity element such that $ea = ae = a$, $\forall a \in G$;
- **Inverse**: $\exists\ a^{-1}$, an inverse of each element such that $aa^{-1} = a^{-1}a = e, \forall a \in G.$

Agents have already been defined as vectors, e.g. a set of primary vector problems as their linear combination. Since primary problems are not only formal variables to represent agents, i.e. they have associated components themselves, primary problems can be perceived as group G. (G, ·) satisfies then all the definitional conditions for groups (2.4.) with the + operation inherited from set of real numbers R. Operation · in a particular situation can be addition, composition of functions, etc. If Abel's (comutative) group is represented in additive manner as in (G, +), than the group is additive. Identity element of additive group is called null–element (represented "0") and inverse element is denoted $-a$.

Let us consider additive group (G, +) with respect to addition operator + in problem solving domain. Operator + is introduced here as "solver operator". The main issue here is the meaning of inverse element. Therefore, we merge problem space with the solver space into one space by group theory. If there is a problem a (e.g.primary) defined as an entity in problem space, let $-a$ be its solver. This than implies also the meaning of identity element. Equation $a + (-a) = (-a) + a = e$; denotes solving of problem a by its solver $-a$ as comutative action no matter the order of problem and solver in equation. When problem a interacts with its solution $-a$ (e.g. solves the problem), it leaves no side effects at the place of interaction. Problem is removed and identity element e is left at particular position which has no meaning from network domain perspective. [2] proves that each group element has its own unique inverse element so the assumption is accepted here.

Applied on any group element by solver operator identity element e does not change either problem or solver added to it. Notion of null-agent is introduced as agent with no side effects.

Closure means creation or merging primary problems and solvers into bigger ones. This feature denotes also an increase of network problem complexity, network size or other network demands. Increasing complexity might introduce new primary problem. Exceptional closure feature denotes that group already contains new problems and solvers although they did not phisically appear in network yet. It can be perceived as built in mechanism of growth. It is also important to preserve meaning in actions with solver operator, e.g. if problem *a* has its own unique solver *–a*, there is no point in reasoning on interaction between problem *a* and solver of problem *b* although the very structure exists in group.

Associativity is not of a special importance for model of agents, but is denoted here for complete compatibility with group definition. It is therefore correct : $a + (b + c) = (a + b) + c$ because agents do not change their features or their internal structure moving primary solvers.

2.3. Extension to ring structure

Let us further extend group structure (G, +) to ring structure by applying · *composition* operator on the very G structure. Than, (G, ·) became semigroup. According to semigroup definition features of group closure and group associativity are valid the only valid ones here. We put emphasis on mechanism of composition and its interactions with solver operator on group elements. Composition creates composite problems and agents, e.g. collection of primary solvers solving particular problem appearing at particular network place. For instance, $a{\cdot}b{\cdot}c$ (or *abc*, for short) denotes network problem which can be decomposed to primary problems *a, b* and *c*. On the other hand, $(-a)(-b)(-c)$ denotes solver to solve specified problems *a, b* and *c* , respectively. Another example $abc + (-b)(-c)$ denotes also possible situation where solver $(-b)(-c)$ can partially solve *abc* problem. This expression $abc + (-b)(-c) = a$ leaves *abc* problem not completely solved, i.e. *a* is left unsolved. Another solver can be applied here, e.g. $(-a)(-d)$ which can solve either *a* or *d* primary problem.

If composition is percieved as association of agents into bigger entities, new dimension of system reasoning is introduced here. New structure formed by more than two agents can solve more problems than each agent respectively. But composition implies that overall problem solving ability remains constant. No matter the external manner of binding structures, their internal structure remains unchanged.

Merging (G, +) with (G, ·) ring structure (R, +, ·) is achieved where R is equivalent to G and operators + and are defined according to the above definitions as problem and solver manipulators. Complete ring structure also allows + and · to be distributed over group elements, i.e. for all a, b, c ∈ R there is:

$$a (b + c) = ab + ac, (a+b) c = ac + bc.$$

Problems and solvers are further divided into composition of equivalence classes. Each class contains elements with the same specified feature, e.g. i-class contains p_i as its representative so each i-class element contains p_i. Therefore we have new group with equipotent classes as its elements. If classes would have not been disjunctive, we would have partition based on Lagrange's group theorem; $|G|=|G/N||N|$, where $|G|$ denotes the order of a group, $|G/N|$ number of elements in each class (i.e. group element order) and $|N|$ is a number of classes within the group (e.g. here a number of primary problems). Also, elements of our group are normal and are denoted G/N (read "*G* modulo *N*") as elements of quotient group since they share some common feature on which group is constituted. In this manner agents are implicitely constructed as elements of these classes, e.g. p_1 and $p_1p_2p_4$ are in the same $<p_1>$ class as they both contain p_1. Since classes are not disjunctive there could occur existence of the same element in two or more classes, e.g. p_1p_4 could appear either in class $<p_1>$ or in class $<p_4>$. Normally, they already exist there by definition. In a particular case it might not be common but is feasible.

2.4. Operations on problem space

Since we reasoned on problems' and solvers' main features and operators under them at lower level of abstraction (e.g. group and ring perspective), we return out attention to vector spaces, i.e. to higher level of abstraction. In vector spaces operator + is not defined as solving operator but inherited from its definition in set of real numbers R.

Vector spaces reasoning enrich problems and solvers with additional features. Each primary member has its own weight which is denoted ($\aleph_1, \dots, \aleph_i, \aleph_n$) and ($\eta_1, \eta_i, \dots, \eta_n$) in II.A section. These attributes are useful guides in choosing particular agent from group of agents to solve particular problem.

Problem vectors which cannot be assigned to solver vector form a *kernel* of problem vector space. Others form its *image*. Out special interest here are *isomorphic* translations of problem vectors onto solver vectors. Isomorphic operator is bijective homomorphism between the two spaces. Homomorphism is $f : VP \rightarrow VS$ operator where there is:

$$f(ab)=f(a)f(b)$$

for each a, b ∈ VP under composition operation. Interpretation of equation is such that composition of primary problems needs composition of respective solvers. Bijection implies that f is also injection and surjection.

3. STATICS IN MULTIAGENT SYSTEMS

Statics in multiagent systems are inspired here with reasoning based on cognitive maps as in [5]. Causal maps model interrelationships among a variety of concepts.

Concepts' domains are not necessarily defined precisely because there are no obvious scales for measuring.

Static aspect is perceived here through interagent relations which do not change no matter how often overall numeral of system elements change. Relation only disappears if all instances of some system element involved in this particular relation disappear. In other case, meaning non-disappearance of system elements, relation change is equivalent to introduction of new resources in telecommunication system, but it is not supposed to happen often. But is feasible. So, statics is not absolute here. We therefore may more precisely say we are dealing with *metastatics* here. To support this metastatics feature we introduced notion of *agent relational metalevel* based on cognitive maps to visualize and analyze benefits of such approach.

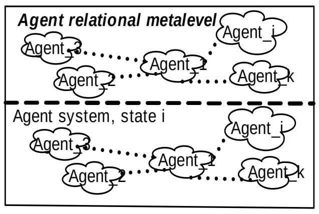

Figure 1. Agent system with associated relational agent metalevel

Although causal maps with given semantics make language with dynamics as its inherent feature, they are useful tool to represent metastatic formal specification of entities involved in system. Causal maps are focused on relations among entities (agents, in our case) which are strongly defined, no matter if they are mobile or not. Precisely, if two agents change their position within the network, their relation remains the same. Positions have no impact on relations (Figure 1 and Figure 2).

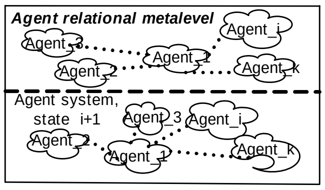

Figure 2. Changed agent system state with associated relational agent metalevel which remains the same

Analysis of relational agent metalevel bring conclusions on prediction of system behaviour and future events, explanation of past events and decision making.

4. Problem solving

Encapsulation in object-oriented programming languages can be used as translation mechanism on collection of agents as mathematical group structure to objects. Since group is consisted of elements under some specified operator, encapsulation translates these data and operations onto variables and methods in classes.

Inheritance in object-oriented programming can be thought of as a process of modeling agents, e.g. specialization of inherited classes. If primary vectors are modeled as classes, then all of their n-features are translated onto classes attributes. Since agents are formed as primary vectors collection, they are modeled as composition of primary vector classes. Each primary problem can enrich its features with the mechanism of inheritance. By this mechanism also growth of problems in particular problem domain is visualized. According to reasoning in 2.3. section problem solvers are divided into equivalence classes where each class preserves some common features, i.e. each element of a equivalence class has at least one feature in common with every other element of the same class. Therefore, for implementation purposes, we define here interface template to be used by all agents needing it. Since there is no general problem solver, but instanciated different solver types, it is realistic to use *generic inteface* to be applied on all solver classes according to their needs (Figure 3). Each problem solver can implement its own way of manipulation on specified problem (`public abstract double PrimaryProblem1()`).

```
public interface ProblemSolver {
  public abstract double PrimaryProblem1();
  public abstract double PrimaryProblem2();
                        . . .
  public abstract double PrimaryProblemN();
}
```

Figure 3. Agent system generic interface in Java

In the same manner composite problems can be implemented.

Election of particular agents is indirectly done by problem environment within domain of problem space. It is reasonable that agents will be “united” on the task of goal reaching; they will make agent system. Even if agents will have goals that are in conflict with each other, it is necessary to define minimal amount of communication between them. Protocol defines agent schedule and conversation rules between agents. Hence, it is necessary to be focused on: common goals and tasks recognition, avoiding of undesired conflicts, putting knowledge and resources “together” in order to solve the problem optimally, minimization of “undesired” communication (moving of agents between network nodes which can be avoided by optimal solvers' schedule), tracing and updating data, which means re-grouping of agents within the network.

Definitions that are recommended here can be efficiently verificated by programming simulation. To reach that goal it is necessary to develop application with following features;

Parameters are given in advance, or can be changed during runtime, but results are presented according to parameters. Results are:

- Problem vector space and problems' sets created from it;
- Solver vector space and solvers' sets with features and specifications which solver has

resources to solve every problem or set of problems that arises;
- Location list within the network with predefined relations (connectivity, distance, etc.)
- Probabilities of problems arising at locations, for every problem and every location;
- Algorithms to apply in conflict cases where more than one agent can solve particular problem.

Every management resource contains knowledge that controls management decision-making process. Every decision controls only one small segment of distributed system; hence it is necessary to communicate.

There are several performances that can be measured by simulation depending on solving problem way and interagents communication, as well as agents' schedule within the network: average solving time, maximum/minimum solving time, number of maximum time violations, amount of communication.

5. Conclusion

In this paper we have offered formal model for specification of mobile agents in telecommunications systems. It offers mathematical approach and is founded on group and vector definitions where their features are interpreted in agents terms.

We've presented the "infrastructure" for formal presentation of agent model as problem solving agent. Hence we've defined problem environment, problem space and problem set as well as solver vector space and set of problem solvers.

We have also research on mechanism of translation from specification to implementation of agent system and object-oriented techniques were found suitable to easily transform vectors to objects.

Every segment of network management process contains only segment of knowledge required for management decision-making, so it is necessary to ensure mechanisms of communication and knowledge (information) transfer. It is also necessary to define optimal way of communication and knowledge transfer depending on predefined parameters. One way of solution is programming simulation that will be presented in future works in more details.

6. REFERENCES

[1] Mardesic, S., "Matematicka analiza u n-dimenzionalnom realnom prostoru", 1. dio, Skolska knjiga, Zagreb, 1988.

[2] Zubrinic, D., "Diskretna matematika", Element, Zagreb, 2001.

[3] Elezovic, N., "Linearna algebra", Element, Zagreb, 1995.

[4] Kunstic, M.; Jukic, O.; Bagic, M.; "Defnition of formal infrastructure for perception of intelligent agents as »problem solvers«**",** Softcom 2002, Split, Venecija, Ankona, Dubrovnik, 2002

[5] Chaib-draa, B., "Causal Maps: Theory, Implementation, and practical Applications in Multiagent Environments", IEEE transactions on knowledge and data enigineering, Vol. 14, No. 6, 2002.

[6] P. Hughes, A.; Pahl, C., "A Generic Model for State-based Agent Systems", Proceedings of the 1st Irish Workshop on Formal Methods, Dublin, 1997.

[7] Tkalcic, H.: "Modeli pokretljivosti agenata u procesima upravljanja mrezom", diplomski rad, april 2002, Fakultet elektrotehnike i racunarstva Sveucilista u Zagrebu.

[8] Lindemann-von Trzebiatowski; Münch, I.; "The Role Concept for Agents in MultiAgent Systems", Humboldt University of Berlin Department of Computer Science, Artificial Intelligence

[9] Satoh, I., "Network Processing of Mobile Agents, by Mobile Agents, for Mobile Agents", National Institute of Informatics / Japan Science and Technology Corporation, Hitotsubashi, Chiyoda-ku, Tokyo, Japan.

[10] Bohoris, C., Pavlou, G., Cruickshank H., "Using Mobile Agents for Network Performance Management", Center for Communication Systems Research, School of Electronic Engineering and Information Technology, University of Surrey, Guildford, UK.

[11] Minar, N.; Hultman Kramer, K.; Maes, P.; "Cooperating Mobile Agents for Dynamic Network Routing,",MIT Media Lab, E15-305 20 Ames St, Cambridge MA 02139, USA