Semantic Interoperability in Agentspace:

Proposal of agent interface to environment*

Stanislaw Ambroszkiewicz

Institute of Computer Science, Polish Academy of Sciences,
al. Ordona 21, 01-237 Warsaw. Poland
Email: sambrosz@ipipan.waw.pl, Web: www.ipipan.waw.pl/mas/

Received: September 4, 2000 / Revised version:

Abstract. Agentspace is an emerging environment resulting from process automation in the Web. Interoperability is crucial to assure meaningful interaction, communication and cooperation between heterogeneous agents and services. In the paper it is assumed that the interoperability limited to the basic agents/services interactions is already given. The interoperability is extended so that heterogeneous agents can exchange knowledge and understand each other about the interactions. To make understanding possible a new approach to concept meaning is introduced. The meaning of concepts is given explicitly in a constructive way contrary to the approach of Gruber and Guarino [11,12] where the meaning of a concept is constrained by logical axioms.

1 Introduction

Cyberspace, the emerging world created by the global information infrastructure and facilitated by the Internet and the Web, offers new application scenarios as well as new challenges. One of them is so called "semantic web", i.e. conceptual structuring of the Web in an explicit machine-readable way. The goal is to facilitate resource discovery, intelligent browsing, e- commerce, e-business, etc. These very tasks are supposed to be performed by autonomous software (mobile) agents on behalf of their users. A mobile software agent is an executing program that can migrate from host to host across a heterogeneous network under its own control and interact with other agents.

Since the software agents are supposed to "live" in the cyberspace, they must be intelligent, that is, they must efficiently realize the goals delegated to them by their human masters. To do so they must perceive the world, interact with the world, as well as with other agents and humans. It is clear that a single agent cannot perform efficiently its tasks in a large open world without cooperation with other agents. Sophisticated agent interaction mechanisms and services are needed. During interactions the agents communicate, negotiate, and form organization structures.

Hence, along the development of cyberspace the new world (called agentspace), inhabited by the new intelligent creatures called software agents, is being created. It seems that the process automation in the Web makes the development of agentspace inevitable.

Human users are situated at the border of the agentspace and can influence it only by their agents by delegating to them complex and time consuming tasks to perform. Since the Web is an open distributed and heterogeneous environment, agents and services can be created by different users according to different architectures. Interoperability is crucial to assure meaningful interaction, communication and cooperation between heterogeneous agents and services. The interoperability is not only restricted to interaction and communication, but it also comprises semantic interoperability. In order to use services established by different users working in heterogeneous domains, agents have to be capable of acquiring knowledge how to use those services and for what purposes. Hence, heterogeneous agents must exchange knowledge and understand each other.

The lack of interoperability is the main obstacle in development of the agentspace. Security is of great importance, however, it does not appear to be the main cause of absence of widespread applications of agent technology. To achieve interoperability certain standards should be introduced. As usual the fewer standards the better. Instead of a separate standard for any particular application domain, we are looking for a minimal standard sufficient to achieve a semantic interoperability.

* The work is done within the framework of ESPRIT project No. 20288 CRIT-2

How can the mobile agent technology help here? To realize the concept of mobile agents together with agents interactions and service infrastructure, a special middleware called "mobile agent platform" (MAP, for short) is needed. There is a number of platforms available over the Internet, for example IBM Aglets, Concordia, Grasshopper, Mole, Ajanta, and Voyager to mention only some of them. One of them is Pegaz [21] developed at our Institute. These platforms are for creating infrastructures on computer networks so that details of network functioning are hidden from the users as well as from the agents. This makes programming more easy. A programmer need not manually construct agent communication, nor agent transportation.

The most important MAP's qualities are independence from operation systems (due to Java) and abstraction from minor (from the point of view of agent activity) details typical for various hosts connected to the Web. Such abstraction means establishing a special "place" on the host open to the Web. This is important for security of the host and limits external access to resources and services only to those situated at this place.

Most of the existing mobile agent platforms create similar infrastructures. However, there is a problem of interoperability between different platforms if agents are to migrate from one platform to another.

Since most of today's mobile agent platforms are implemented in Java, it is postulated in [15] to create a new OMG standard to define implementation-specific (i.e. Java-specific) convention that allows a mobile agent to migrate to any standard-compilant platform.

The OMG MASIF [16] and FIPA Mobility Support [9] standards and generic MAP architectures (e.g. Pegaz [21] and Grasshopper [10], Mole [17], Ajanta[2]) are examples of attempts to provide a basic standard for agentspace infrastructure.

These efforts are trying to assure interoperability, limited to "core or basic functionality" concerning agent and service interactions, and called interaction interoperability. However, it is not enough; semantic interoperability is needed to achieve the perfect interoperability.

Although the MAP technology was designed to provide mobility to the software agents, it seems that the standards MASIF / FIPA may be seen as efforts towards building uniform Web infrastructure analogously to traffic and communication infrastructure e.g. highroads, airports, telecommunication in the real world. Introducing uniform infrastructure does not exclude autonomy and heterogeneity of the environments but it supports their development by the ability to cooperation.

Since the technology of mobile agent is not matured, it is still far from achieving agreement for the standards that could assure interaction interoperability. These very standards are necessary to define structure of basic agents / services interactions in the agentspace.

Our approach to semantic interoperability presupposes the existence of commonly accepted standards that give interaction interoperability. We propose a formal specification of the interaction structure (called *generic structure of MAP environment*) that may suggest some ideas how to construct such standards. Formal specification of the interaction structure is necessary for heterogeneous agents to exchange knowledge and understand each other, cooperate, form new services and organizations. However, this very understanding presupposes that the meaning of primitive resources and primitive services is already given and is common. So that the proposed semantic interoperability is not completed.

Although the partial semantic interoperability proposed for the agentspace is relatively easy to achieve, it may be viewed as a contribution towards the global semantic interoperability for the Web where the key issue is to represent the meaning of resources and services in a uniform machine - readable way. The semantic interoperability concerning the meaning of resources on the Web is a subject of current research, see OIL - Ontology Interchange Language [19], DAML - DARPA Agent Markup Language [6], SUO - Standard for Upper Ontology of IEEE [23].

Our contribution to the problem of semantic interoperability consists in a specification of common interface to environment for heterogeneous agents. The interface may serve as means to achieve semantic interoperabilty between these agents. The rest of the paper is devoted to presentation of the interface and our approach to semantic interoperability.

The research presented in the paper is based on our experiences gained during realization of Pegaz - our mobile agent platform [21], and modeling agent virtual organizations as sophisticated agent interaction mechanisms [5].

2 Our methodology

Let us suppose that there are several communities of agents living in a world. Each community consists of homogeneous agents that can interact, speak their own language, and understand each other.

What does it mean the "language" and "understanding" inside a community?

Usually, language is means of asking for / passing knowledge about the world. Knowledge is expressed by concepts, like for example: *pencil*, *color*, etc. *Color* is an attribute of *pencil*, so that the sentence: *my pencil is green* expresses a part of knowledge about the world.

Let us fix for a moment a single community A1 speaking language L1. Let us notice that speaking is possible if there is interaction interoperability between them. Since the agents are homogeneous, they have the same collections of concepts with the same names and use them in the same way, so that also semantic interoperability (understanding) between them is not a problem. Explicit

meaning of the concepts, or ontology (i.e. formal specification of conceptualization) that is associated with the language is not necessary for the agents of the same community to understand each other.

Let us consider another agent community, say A2 speaking language L2, where the concepts are the same as in A1 however, their names are different. Let it be the only difference between A2 and A1. To achieve understanding between agents from A1 and A2, only a mapping between vocabularies of L1 and L2 is needed.

The real problem begins when there is a community, say A3 speaking L3, where the concepts are different. How to achieve semantic interoperability between agents of A1 and A3? Let us suppose that there is interaction interoperability between them, i.e. they can communicate. In order to understand each other, the agents must, first of all, exchange the meaning of concepts they use. The meaning must be given explicitly. Suppose it is given in the form of ontologies O1, and O3.

Ontologies O1 and O3 are formal theories, so that a language for ontology exchange, like OIL [19] is needed as a common standard. This, however, is not enough. To achieve understanding, a translation from O1 to O3 and vice versa is needed. Since O1 and O3 are formal theories, the translation is nothing but an interpretation of theory O1 in O3 such that the axioms of O1 are theorems in O3. The problem is that the translation can not be done in an automatic way.

The conclusion is that in order to understand each other, the agents must have something in common. If the concept meaning is represented by ontologies, then a language for ontology interchange must be a common standard.

Our approach to semantic interoperability is different. We assume the this very "something in common" is generic representation of the world structure, and perception mechanism. This assures that meaning of concepts can be reduced to the generic representation. Since the generic representation of the world structure is supposed to be common, the exchange of concept meaning is possible between heterogeneous agents. The rest of the paper is devoted to an application of our approach to the agentspace.

3 Outline of agent interface to environment

Usually a user of a MAP can construct his/her own agent/service internal architecture using a special "frame" provided by the MAP developer. The frame is called "agent class", and consists of Java classes whose methods represent (see for example [10]) "the essential interfaces between agents and their environment". The frame gives access to the core functionality of a MAP and communication service, so that it makes possible for an agent to interact with the environment and determines what is visible for the agent during these interactions, e.g. whether the agent can see the resources (data) of another interacting agent, its name, interaction place, etc..

The access should be standardized if we want agents to migrate from one platform to another.

In order to reason and exchange knowledge about interactions and understand each other, the agents must have common representation of interaction structure (i.e. generic structure of MAP environment) and perceive the environment in a uniform way. We propose to separate and standardize agent interface to environment in the following way. The interface should consists of three layers. Firstly, to standardize the access to the core functionality, i.e. the basic interaction types. It constitutes the first layer of the interface, called **functionality layer**. Secondly, to specify formally the interaction and perception structure determined by the first layer. It constitutes the second layer of the interface, called **representation layer**.
Thirdly, to construct, on the basis of the second layer, meaning interchange language. It constitutes the third layer of the interface, called **language layer**.

Let us notice that the proposed agent (service) interface to environment does not apply to internal agent (service) architecture. Users have full freedom of designing and developing the architectures.

The generic structure of MAP environment as well as perception structure are determined by the functionality layer. The MAP environment specification concerns the structure of the world that includes: primitive entities (like agents, places, services, etc.), primitive actions (like migration, communication, using a service, etc.), and events describing local interaction between agents, services, resources and places. The events form a partial order according to the causal relation between them. The perception structure determines what can be perceived by an agent participating in an interaction, e.g. during moving to a host the agent can perceive the host name. The representation layer is also called "ontology core" that formalizes the structure of the world realized by the functionality layer. The ontology core is the basis for designing knowledge to be implemented into agent architecture. It is up to a designer how to built knowledge structure consisting of interrelated concepts. The crucial point is that the meaning of these concepts can be reduced to the ontology core. Since the ontology core is supposed to be a common standard, this gives rise to create language to interchange concept meaning between heterogeneous agents. Meaning Interchange Language constitutes the language layer. A detailed description of representation layer is presented in Section 3 whereas in Section 5 we present an idea how to construct the language layer.

4 Representation layer - generic MAP environment and perception

It is presupposed that the functionality layer is built according to commonly accepted standards (unfortunately not existing yet) that could assure the interaction interoperability.

The representation layer of the proposed interface consists of specification of generic MAP environment and perception. As it was already mentioned above, these two parts, i.e. the structure of the world and what can be "seen" during local interactions, should be determined by the functionality layer. So that our proposal of the representation layer may be useful for constructing also the functionality layer.

4.1 Generic structure of MAP environment

The main components of the structure (created by a platform like Grasshopper or Pegaz) are places, services located at the places, and agents that can move from place to place. The agents use services, communicate, and exchange data and resources with other agents. The agents can also collect, store, exchange, and transport the resources.

In our formal description the primitives are: **places, agents, actions, services,** and **resources**. Places are composed into a graph. An edge between two nodes in the graph expresses the fact that there is an immediate connection between the corresponding places. Resources are primitives grouped into types, analogously as types of data, and objects in programming languages. Services are represented as operations on resources and are of the form: $Op : C \rightarrow B$, producing object of type B from object of type C. Primitive types of resources and services are specific for application domain.

The infrastructure is developed by constructing sophisticated services (i.e., enterprises, organizations) by the agents. To do so, the agents should be equipped with appropriate mechanisms that are called *routines*. Routines are composed of the following primitive actions: **move** to another place, **use a service**, **agent cloning** (i.e. agent creates a copy of itself), **get a resource** from other agent, **give a resource** to other agent, **communicate** with other agent, **finishing activity** of an agent.

The routines for forming and reconfiguring agent organizations are quite sophisticated, see for example our concept of team formation [5] for simplified version of the environment.

Service may be understood as a composition of routines (forming a workflow) performed by a group of agents. In order to manage the workflow resulted from the service, the agents should form an organization structure.

Let A denote the set of agents' primitive actions. Some actions belonging to the core functionality of a MAP are not considered here like agent naming, registration, and actions concerning the security. They are hidden from the user as well as from the agents. In order to have the environment structure as simple as possible we should abstract from those details that are not essential for the agents to realize their goals.

It is a subject of discussion whether the set of primitive actions proposed above is comprehensive enough to define all the actions that can be useful for the agents.

Joint action a (for example a communication action) can be executed if for all the agents needed for an execution, the action a is enabled and selected for execution, i.e., intuitively all the agents "can" and "do want" to participate in the execution of this action. If one of the agents cannot or doesn't want to participate in the execution, then the attempt to execute action a fails. For this reason we assume that for any joint action a, and any agent i, needed to execute a, there is a local action $fail(a, i)$ of agent i that corresponds to agent i's failure to execute joint action a.

The crucial notion needed to define a representation of the "world", presented intuitively above, is the notion of "event". Any event corresponds to an action execution, so that it "describes" an occurrence of a local interaction of the agents participating in the execution. The term "local" is of great importance here. Usually, local interaction concerns only few agents so that the event associated with this local interaction describes only the involved agents, services, resources, and places.

The events form a structure, that expresses their causal relations.

Having primitive concepts and setting initial conditions for agents, places, and distribution of primitive resources and services, the set E, of all events that can occur in the environment, can be defined. It is important to notice that due to the local interaction, it is easy to extend the set E if new agents, places, resources, and services are added. Hence, the representation is appropriate for open environments. It is also generic in the sense that in order to construct the representation an agent need not know all agents, places, resources, and services in the world. In the course of learning about them the agent can extend the representation or shrink it if it is necessary to forget about some agents, places, etc..

Let $E_i \subseteq E$ denote the subset of events in which the agent i does participate. Event e is called *joint* if it belongs to at least two different sets E_i. Event structures have been successfully applied in the theory of distributed systems [24] and several temporal logics have adopted them as frames [14].

Now, we present a formal definition of event structure.

A **labeled prime event structure** is a 5-tuple $\mathcal{ES} = (E, A, \rightarrow, \#, l)$, where

1. E is a set of *events* or *action occurrences*,
2. A is a finite set, called a set of *actions*,
3. $\rightarrow$ a subset of $E \times E$ is an irreflexive, acyclic relation, called the *immediate causality relation between the*

events, i.e. $e \rightarrow e'$ means that the event e is one of the immediate causes of the event e'. The causality relation satisfies the following condition: $\downarrow e \stackrel{def}{=} \{e' \in E \mid e' \rightarrow^* e\}$ is finite for each $e \in E$, where $\rightarrow^*$ is the reflexive and transitive closure of $\rightarrow$. This means that the past of any event is finite.

4. $\#$ a subset of $E \times E$ is a symmetric, irreflexive relation, called *conflict relation*, such that $\# \circ \rightarrow^*$ is a subset of $\#$ (called *conflict preservation condition*). Two events are in conflict if they cannot occur in the same run. One of the reasons for conflict occurrence is agent choice of action, i.e. which action an agent chooses to execute.
5. $l : E \longrightarrow A$ is a labeling function that indicates for each event which action is executed at that event, i.e., $l(e) = a$ means that event e is an occurrence of action a.

Two events e, e' are in immediate causality relation, i.e. $e \rightarrow e'$ if the same agent participates in them, and the agent has executed the action $l(e')$ immediately after the execution of action $l(e)$.

4.2 Perception structure

It is clear that agent participating in an event should have a possibility to observe some essential features of the event like the name of action executed at this event, other agents, places, services and resources involved in the interaction corresponding to the event.

We choose a set $P = \{p_1, p_2, ..., p_l\}$ of propositions of the Propositional Calculus to be the set of common observables. The propositions characterize agent locations, resources, services available on places, and so on. They are evaluated at events, and should be specified so that they can provide a uniform way to describe essential features of the events. The value of a proposition at an event is either true or false, or undefined. Since each event is local, values of some propositions (describing agents or places not participating in the execution) cannot be determined and are left undefined.

Although the propositions are supposed to describe the events completely, a particular agent may have no access to some details, for example concerning the resources possessed by another agent.

Let $\Gamma = \{t, f, u\}^l$ (l is the number of propositions in the set P) be the set of all possible strings of length l over the set $\{t, f, u\}$ coding all the valuations of the propositions. Here t stands for **true**, f stands for **false**, whereas u stands for **undefined**. For example, for $l = 6$, the string $o = ttfutf \in O$ corresponds to valuation at some event where propositions p_1, p_2 and p_5 are true, propositions p_3, p_6 are false, whereas the value of proposition p_4 is undefined. Hence, the set Γ is the collection of all possible valuations of propositions from the set P.

Perception Π is defined as a function of the form: $\Pi : E \rightarrow \Gamma$. For any event $e \in E$ perception Π determines which proposition is true, which one is false, and which one is undefined.

Ontology core, understood as the minimum needed to achieve semantic interoperability, is constituted by specification of generic MAP environment and the perception structure.

5 Knowledge and meaning of concepts

In this section we present an approach to knowledge and concept meaning.

In order to construct efficient agents, their architecture must be simple. Therefore, we need simple, but still flexible, notion of knowledge that can be implemented into agent architecture. Hence, we face the following problems:
How to represent, store, pass, and reason about knowledge in an efficient way, so that it can be implemented into simple agent architecture ?

Classical definitions of knowledge are built on global states and global time, see [7]. The consequence of that definition is logical omniscience of the agents and an arbitrarily deep nesting of knowledge operators, that is, formulas under consideration are of the form: *agent* i_1 *knows that agent* i_2 *knows that ... that agent* i_n *knows that event* e *occurred.* This very omniscience is frequently regarded as a drawback especially if an agent is to take decisions in real time. Moreover, when modeling such an agent, it turns out that the representation of the whole world must to be put into the "brain" of the agent, see the concept of BDI-agent [22]. This is acceptable if the world is small, say up to a few agents, but if the world is getting larger, then it is computationally unrealistic do deal with such a model, see [5]. Hence, if the world is large and/or what is even worse, the world is "open," then the classical notion of knowledge remains only an elegant theoretical notion.

Our alternative proposal to the classical notion of knowledge consists in assuming that initially agents know almost nothing and acquire, update, and revise knowledge during local interactions by perception and communication with other agents.

We assume that knowledge of agent i is stored in the variables $v_1^i, \ldots, v_K^i$ ranging over the sets $W_1, \ldots, W_K$. These variables are individual variables of the agent i. The sets W_k can be of any sort including: char, integers, reals, etc.

This variable collection defines **knowledge support.** One can think about it as an abstraction of a relational database. It is important to note that for homogeneous agents, the number of variables and their ranges are the same. So that, any variable stands for a common concept for a community of homogeneous agents.

The profile of current values of agent's variables is considered as the current agent's mental state. All possible agents' mental states are collected in the set $M =$

$\prod_{k\in K} W_k$. For example, $m_i = (w_1, w_2, \ldots, w_K)$ corresponds to agent i's mental state where $v_1^i = w_1, v_2^i = w_2, \ldots, v_K^i = w_K$.

Values of the variables $(v_1^i, \ldots, v_k^i)$ are updated by agent i's perception and revised by knowledge acquired from other agents via explicit communication.

5.1 Agent perception

At event e any agent participating in this event can evaluate any $p \in P$ and come up with the conclusion whether p is true or false or undefined or unknown. The value "unknown" refers to the propositions that cannot be evaluated by the agent because they are hidden from the agent, for example, propositions concerning the storage contents of another agent participating in the event e.

Let $O = \{t, f, u, *\}^l$ (l is the number of propositions in the set P, and $*$ refers to **unknown**) be the set of all possible strings of length l over the set $\{t, f, u, *\}$ coding all possible results of agent observations. For example, for $l = 6$, the string $o = ttf * tu \in O$ corresponds to agent's observation at some event that propositions p_1, p_2 and p_5 are true, proposition p_3 is false, the value of p_4 is unknown, whereas the value of proposition p_6 is undefined. Hence, the set O is the collection of all possible observations.

Agent i's perception is defined as a function $\pi_i : E_i \longrightarrow O$. An intuitive meaning of $\pi_i(e) = ttf * tu$ is that at event e agent i evaluates the propositions in the way described above.

5.2 Meaning of concepts and simple query language

Updating mechanism is defined as a function $Mem : M \times O \longrightarrow M$. An intuitive meaning of $Mem(m, o) = m'$ is that mental state m is updated to m' after observation o has been made by an agent.

Let $Q = \{*, ?\}^K$ be the set of *queries*. The interpretation of query $q = ***?*?$ is: **tell me the value of variable v_4 and v_6 in your memory (database)**. Let us note that such query is meaningful only for homogeneous agents. The set of all possible answers to the queries Q is defined as follows.

$Val(Q) \stackrel{df}{=} \prod_{k\in K}(W_k \cup \{*\})$.

Let query $q = q_1 \ldots q_K$ be sent to agent i, whose memory state is $m_i = (w_1, \ldots, w_K)$. The answer of agent i is the string $(x_1 \ldots x_K)$, where $x_j = w_j$, for $q_j =?$, and $x_j = *$ for $q_j = *$. For example, agent i's answer to query $***?*?$ can be $***2*9$.

The set of queries defines a simple language. Of course, in order to be a communication language it must be built in KQML [8] or ACL [1].

In the set of actions A we identify a subset of *communication actions*. For each query $q \in Q$ we have a joint action iqj of agents i and j with the intended meaning of sending query q to agent j by agent i. If agent j does agree to answer the query, then the action iqj is successfully executed, otherwise it fails.

Revision mechanism is defined as a function: $\Xi : M \times Val(Q) \longrightarrow M$. An intuitive meaning of $\Xi(m, x_1 \ldots x_K) = m'$ is that mental state m is revised to m' after receiving answer $x_1 \ldots x_K$ to its query sent to another agent.

The famous gossip problem associated with the revision mechanism occurs when the knowledge is exchanged between agents. Suppose that an agents received via communication from another agent a different information that he posses on the same subject. The agent must decide which one is the most recent one. The way of solving this kind of problems is by encoding an additional information, which is sufficient for deciding which agent has got the most recent information about the subject. One of the possible solutions is so called secondary information defined for gossip automata [18].

Let us note that these two "reasoning" mechanisms Mem and Ξ define agent knowledge, i.e. define how the concepts stored in variables $v_1^i, \ldots, v_K^i$ are used. The first mechanism updates agent mental state if a new observation has been made by the agent. The second one revises the current mental state if the agent has received an answer to its query.

The way the mechanisms define knowledge is highly abstract. Description Logic or Frame Logic can be used for representation, and OKBC [20] for implementation. However, our goal is to present the essence without going into specific representation and implementation details.

These two mechanisms define also the meaning of concepts $v_1^i, \ldots, v_K^i$. The meaning of a concept is determined by the way the concept is used. Let us stress that for any concept its meaning is defined within the community of homogeneous agents that use the concept.

Our approach to concept meaning follows the definition of "meaning" from Wittgenstein [25]: *"For a large class of cases - though not for all - in which we employ the word 'meaning' it can be defined thus: the meaning of a word is its use in the language."*

Our definition of knowledge and meaning is abstract and not constructive. On one hand it may be seen as an advantage, because usually concept meaning is defined in formal theories; see the approach of Gruber, et al. [11] and Guarino [12] where the interpretation (meaning) of concepts is constrained by logical axioms. On the other hand our definition gives an idea of what the knowledge and concept meaning is however, it does not help us to understand how the concepts are created and interrelated.

Concept formation is one of the crucial aspects of semantic interoperability, see for example Kangassalo [13].

6 Language layer - Meaning Interchange Language (MIL)

Homogeneous agents have the same primitive concepts and use them in the same way. This makes passing knowledge (via queries) between homogeneous agents meaningful. However, homogeneous agents are not identical. They are differentiated by their perceptions, and histories, i.e., by what they have observed and learned from other agents in the past.

The simplest way to achieve semantic interoperability is to assume that all agents are homogeneous. However, the Web is a heterogeneous environment, so that the assumption that all agents are homogeneous is too restrictive.

The problem is to design a generic language for interchanging concept meaning between heterogeneous agents. The language would give rise to achieving understanding between heterogeneous agents.

It is obvious that the understanding is impossible if the agents have nothing in common. We suppose that this very something in common is the first and the second layer of the proposed interface. The crucial point is the observation that any concept meaning (defined within homogeneous agent community) can be reduced to the ontology core that constitutes the second layer. Hence, in order to design Meaning Interchange Language we should only specify the reduction procedure in a generic way.

However, our approach to concept meaning is abstract and not constructive. In order to convey the meaning of a single concept (variable) the functions Mem and Ξ (at least a part of them) must be conveyed. The functions are given in an extentional, not constructive form, i.e. in a set - theoretic manner as sets of pairs.

It seems that generic rules for constructive formation of concepts are essential for MIL design.

Our approach to constructive concept formation is based on our work on constructive type theory [3], where we have introduced primitive constructors that are sufficient to construct all recursive functions. The limit of space does not allow to present details here.

MIL must contain the simple query language defined in the previous section. To be an agent communication language, MIL must also be built into existing standard language frame like ACL [1].

7 Discussion and preliminary conclusions

We have presented outlines of our idea of achieving partial semantic interoperability in agentspace. So that the paper should not be seen as a presentation of a completed work but rather as a proposal how to solve the hard problem.

The simplicity of the introduced representation of generic MAP environment may be considered as an advantage of our approach. Actually this very simplicity is necessary because of the implementation reasons. If we want agents to be efficient their architecture and interface to environment must be simple. Of course, our interface is merely a proposal so that it should be discussed and evaluated.

In the paper the notion of service was only mentioned. However, in the development of agentspace as well as of a MAP, services play a crucial role. They are needed for agents to use resources, facilitate access to information stored in heterogeneous databases, enable complex agent interactions, and perform sophisticated workflows. Agent virtual organizations in general and enterprises in particular may serve as examples of such sophisticated services. Our aim of future work is to propose a service interface to environment based on the ontology core introduced in the paper.

References

1. ACL-Agent Communication language. FIPA 97 specification part 2, Technical report, October 1997.
2. Ajanta a MAP developed at Dep. Computer Science, Univ. of Minnesota: www.cs.umn.edu/Ajanta
3. S. Ambroszkiewicz. Another Semantics for Typed Lambda Calculus: a simple example of functional language. ICS PAS Reports 689. Institue of Computer Science Polish Academy of Siences. Warsaw, September 1990.
4. S. Ambroszkiewicz, W. Penczek, and T. Nowak. Towards Formal Specification and Verification in Cyberspace. Presented at Goddard Workshop on Formal Approaches to Agent-Based Systems, 5 - 7 April 2000, NASA Goddard Space Flight Center, Greenbelt, Maryland, USA. To appear in Springer LNCS.
5. S. Ambroszkiewicz, O. Matyja, and W. Penczek. "Team Formation by Self-Interested Mobile Agents." In Proc. 4-th Australian DAI-Workshop, Brisbane, Australia, July 13, 1998. Published in Springer LNAI 1544.
6. DARPA Agent Markup Language (DAML) http://www.daml.org/
7. R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about knowledge,* MIT Press, 1995.
8. T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In J. Bradshaw Ed., *Software Agents*, MIT Press, 1997.
9. The Foundation for Intelligent Physical Agents (FIPA), FIPA 98 specification: http://www.cselt.it/fipa/spec/fipa98/fipa98.htm
10. Grasshopper http://www.ikv.de/products/grasshopper and http://www.ikv.de/download/
11. T. R. Gruber. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *In Formal Ontology Analysis and Knowledge Representation,* (N. Guarino and R. Poli Eds.) Kluwer Academic Publishers 1994.
12. N. Guarino. The Ontological level. In R. Casi, B. Smith and G. White (Eds.), *Philosophy and the Cognitive Science.* Hölder-Pichler-Tempsky, Vienna, 1994.

13. H. Kangassalo. Are Global Understanding, Communication, and Information Management in Information Systems Possible? - A Conceptual Modeling View - Problems and Proposals for Solutions. In Chen, P.P., Akoka, J., Kangassalo, H., Thalheim, B., (Eds.), *Conceptual Modeling: Current Issues and Future Directions.* Springer-Verlag, 1999.
14. K. Lodaya, R. Ramanujam, P.S. Thiagarajan, "Temporal logic for communicating sequential agents: I", *Int. J. Found. Comp. Sci.,* vol. 3(2), 1992, pp. 117–159.
15. T. Magedanz, M. Breugst, I. Busse, S. Covaci. Integrating Mobile Agent Technology and CORBA Middleware. *AgentLink Newsletter* 1, Nov. 1998, http://www.agentlink.org
16. OMG MASIF, Mobile Agent Systems Interoperability Facility Specification, OMG TC Document orbos/97-10-05
http://www.omg.org/cgi-bin/doc?orbos/97-10-05
17. Mole - MAP developed by Distributed Systems group at the IPVR of the University of Stuttgart http://mole.informatik.uni-stuttgart.de/
18. M. Mukund, and M. Sohoni. Keeping track of the latest gossip: Bounded time-stamps suffice, *FST&TCS'93, LNCS* **761**, 1993, 388-199.
19. OIL - the Ontology Interchange Language: http://www.ontoknowledge.org/oil/oilhome.html
20. OKBC is an API and reference implementation that allows representation system, platform and language-independent knowledge-level communication. http://www.ksl.stanford.edu/software/OKBC/
21. Pegaz our MAP - http://www.ipipan.waw.pl/mas/pegaz/
22. A. S. Rao and M. P. Georgeff. Modelling rational agents within a BDI–architecture. In Proc. KR'91, pp. 473-484, Cambridge, Mass., 1991, Morgan Kaufmann.
23. SUO - IEEE Computer Society, Standards Activity Board, Standard Upper Ontology, IEEE Study Group. http://ltsc.ieee.org/suo/
24. Winskel, G., *An Introduction to Event Structures,* LNCS 354, Springer - Verlag, pp. 364-397, 1989.
25. L. Wittgenstein. *Philosophical Investigations.* Basil Blackwell, Oxford 1958, pp.20-21.