Agent-based Semantic Grid Service for Resource Management

A.V. Prokhorov, E.M. Pakhnina

National aerospace university named after M.Ye. Zhukovsky «Kharkiv Aviation Institute», 17, Chkalov Str., Kharkiv, Ukraine

avprohorov@yahoo.com, elena.pakhnina@khai.edu

Abstract. *Using semantic technologies in grid-systems is considered. We propose the agent-based semantic information service for grid-resource discovery. Ontology is a good approach to resource discovery by providing a formal conceptualization of a grid-system. The approach suggested in the paper allows mapping the elements of ontological languages to the semantic service knowledge base relying on the first-order predicate logic. The meta-model is presented describing the rules used for solving the problem of detecting discrepancies between the derived rules and facts of ontology. The issues of implementing an effective reasoning algorithm on ontologies for different strategies of search reduction are described.*

Keywords

Semantic grid service, resource discovery, ontology, intelligent agent, reasoning

1 Introduction

Nowadays, as the grid develops, it penetrates industry and business, claiming the role of a universal infrastructure for data processing, wherein a great number of services function, which not only allow solving specific application problems, but also offer services in search for the necessary resources, collection of information on their condition, data storage and delivery.

The grid source management is a process of determining requirements, comparing resources and applications, distribution of resources, as well as planning and monitoring. The grid efficiency depends on availability, accuracy and actuality of the information about all the resources connected their properties and state. Difficulties in resource management result from: a great number of participants, demanding for the multilevel resource planning; conflict of the interests of the users, resource owners and system administrators; the general lack of the system information, as well as inconsistency and unpredictability of the computing facilities of the resources available (individual resources do not have centralized management, that is why they can leave the grid system or join it any time); availability of own policies with the most resources that must be also taken into account etc.

Alternately, a considerable increase in the number of the Internet-related applied problems conditioned its development to the new level named the Semantic Web, intended for affording the opportunity of the effective data search and analysis both by a man and by program components. The multiagent system technologies and ontologies are used for solving these problems, directly associated with the automatic processing of the semantic contents of web-resources, acquisition and integration of knowledge from the distributed information sources in the Internet. The key element of a program agent in a multiagent system, allowing it to make decisions, plan actions, and interact with other agents is a knowledge base containing models of conceptual notions, relations and rules for analysis and situational orientation. As a tool for structuring and representing information in such systems just ontologies are used.

As the grids develop, more and more difficulties emerge in coordination and distribution of resources, which can be solved by the progressive combination of the Semantic Web and the grid [1]. Semantic technologies are capable of improving various aspects of the grid-system operation (Fig. 1): from detection and selection of computational resources and data storages, the scheduler operation, to the intellectual user interface. Therefore, the Semantic Grid, being a product of the Semantic Web and SOA combination, extends the existing grid-infrastructure so that to enrich the grid with semantics. Semantic annotations are created for each resource in the grid, processed by the intellectual

agents. In addition, the intellectual agents must implement the logic of the access to resources via the grid-services in order to arrange interaction among the applications, users or other agents, i.e. to support the semantic interoperability.

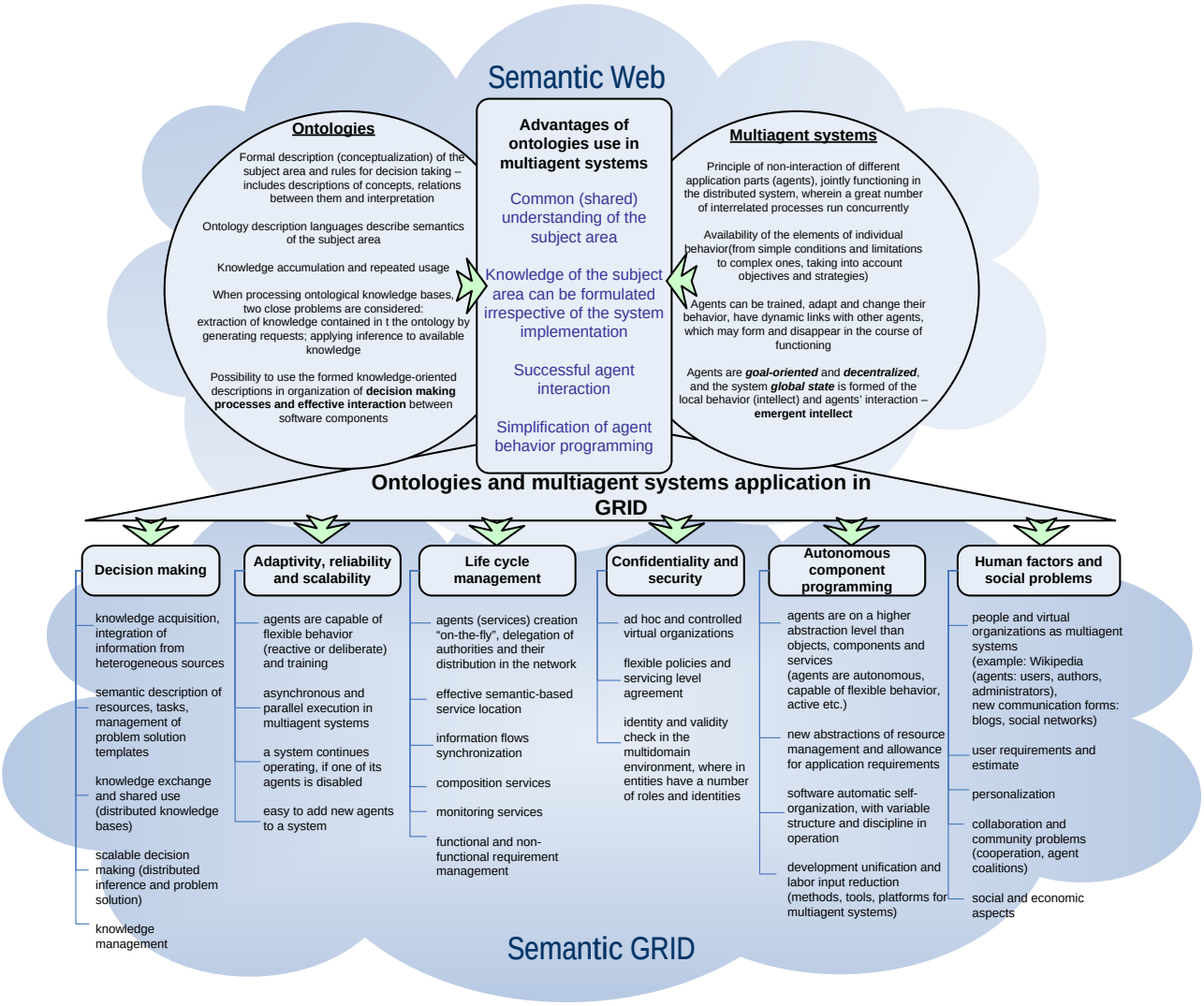

Fig.1. Using ontologies and multiagent systems in GRID

Therefore, the efficiency of the organization and search for the resources for the distributed computing systems can be increased by means of the introduction of the methods and means of the intellectual information technologies into the grid-system operation mechanisms. A semantic information service based on the grid-resource ontology [2] can be considered as a basic element of such intellectualization.

Presently, the most widespread ontological model recommended by the W3C consortium is the OWL language [2]. Three sublanguages are defined in this language: *OWL Lite* – a classification hierarchy of things and simple conditions of consistency of things; *OWL DL* (*Description Logic* – a decidable subset of first-order logic predicates) – maximum language expressiveness without loss of computational completeness and decidability; *OWL Full* – very high expressiveness (metaclasses, classes as values) and complete syntactical freedom RDF (*Resource Description Framework*) with a loss of guaranteed computational completeness and decidability.

When processing ontologies, two close problems are considered: knowledge elicitation in the ontology by formation of queries, and application of reasoning to available knowledge.

To solve the first problem, the most widespread tool is *SPARQL*, a language for querying RDF, which accepts RDF-data as a set of statements or triplets *rdf(Subject, Predicate, Object)*.

Today, there is a great number of studies aimed at creation of the means of search for the grid-resources (the first and important stage in the resource management), including those using semantic technologies.

Paper [3] deals with the basic principles of operation of the semantic service of the resource location, using the ontologies presented in the RDF format, and the rules based on fuzzy logics, however, the ontologies used here did not reflect the specificity of the grid-resources.

The information service of monitoring and search, implemented as a superstructure over the Globus Toolkit, is described in [4]. The aggregated resource data are saved in the Sesame storage of RDF triplets and are requested using the SPARQL request language. Using only SPARQL queries, without using the reasoning, does not provide sufficient advantages over the LDAP- and UDDI-based traditional technologies.

Research in the field of the application of ontologies to increase the resource selection accuracy in the gridis conducted at the National technical University of Ukraine «Kyiv Polytechnic Institute», among them – within the framework of the State Program of Implementation of Grid-Technologies for 2009-2013 in Ukraine. The ontology of the grid-resources for the semantic information grid-service is presented in [5].

[6] deals with the tool for working with ontologies, G-Ontology, used by the authors to solve the problem of orientation in the information array, search optimization, at that, the authors selected their own special format to describe ontologies, however, OWL-compatible.

Still, when using semantic technologies in the grid, questions arise, related to the necessity to population and keeping in the actual status the ontological knowledge base, improving the existing and search for the new effective solutions for software implementation of the output machines etc.

Since ontology defines the terms of a specific area of knowledge, it must be described with a formal knowledge based on the principles of the mathematical logic. Then clear, detailed and integral definitions can be formulated for classes of objects, their properties and relations among them. In their turn, the ontology processing means will be able to automatically derive certain conclusions, based on the principles of the mathematical logic.

The OWL DL level is the one that is focused to currently existing knowledge description systems, and logical programming and reasoning systems. They solve the following problems: check ontology correctness, process queries in terms of ontology, and map and integrate ontologies.

The open applied programming interface *API Jena* is used to work with OWL ontologies. However, neither of the reasoning engines existing in Jena (*Transitive reasoner, RDFS rule reasoner, OWL Mini, OWL Micro Reasoners*) completely support OWL DL.

For complete OWL DL reasoning support, external reasoning engines are required, such as *FaCT* (an open-source DL reasoning engine developed by Prof. Ian Horrocks, with the Manchester University), *RacerPro* (a commercial development of *Racer Systems GmbH & Co*., Germany), *Pellet* (an open-source reasoning engine developed by the *MIND LAB* of the Maryland University) and other ones supporting the *DIG* (*DL Implementation Group*) interface.

Besides working directly with OWL, an approach based on extending or transforming the OWL ontology into other languages and systems is often used.

There are other more complex reasoning engines using such languages as *RuleML* (*Rule Markup Language*), which is a subset of the declaration language *Datalog*, and *SWRL* (*Semantic Web Rule Language*), which combines OWL DL and *RuleML*. Owing to such an extension, *SWRL* acquires the capability of adding and using Horn disjunctions (*Horn-like rules*) for explicit indication of the method for reasoning new facts from RDF statements. Rules in SWRL can be written as part of an ontology. The delivery package of one of the widely used tools for creating and editing ontologies, Protégé-OWL, includes as a component the plug-in SWRLTab for working with rules in this language. Language SWRL has certain constraints, among which is the impossibility to use negations and disjunctions.

Transformers are available, allowing to transform ontologies in OWL and SWRL to a knowledge base of a shell for developing expert systems JESS [7] (*Java Expert System Shell*), which uses the language of scenarios consistent with the CLIPS knowledge representation language.

KAON2 (*The KArlsruhe ONtology and Semantic Web tool suite*) is a development of the Karlsruhe University in collaboration with the Manchester University. It is a system for management of ontologies. KAON2 is capable of importing ontologies in the OWL DL language and organizing reasoning with output of responses to queries on composition and properties using its own internal language based on Horn clauses.

Thea is a library for the Prolog language [8], which provides generation and processing of OWL ontologies. It includes the OWL-parser, OWL-generator, SQL-to-OWL translator and reasoning engine Thea OWL for translating OWL ontologies to Prolog based on the DLP (*Description Logic Programs*) concept. The OWL ontology processed as RDF triplets is used in the SWI-Prolog environment.

Paper [9] suggests mapping an ontological model on an AMN (*Abstract Machine Notation*) of the B-Toolkit system.

Hence, analysis has shown that though some developers have their own reasoning engines operating in various subsets of *First-Order Logic Predicates*, among which are Prolog, DL and others, the most widespread approach is the one that involves transforming ontologies to the internal formats of these systems. The limitation of some of these ontological projects, the impossibility of their embedding in other applications (for instance, stand-alone usage of the reasoning engine in intellectual agent structures) and absence of convenient visualising tools for working with reasoning make the development and usage of an in-house semantic information grid-service a challenge within the framework of this study.

2 Semantic grid-service

2.1 Fundamentals

The semantic grid-service is a multiagent system, wherein the following agents operate: *ResourceDiscoveryAgent* – the mobile resource discovery agent, *RequestHandlingAgent* – the request agent, *OntologyUpdateAgent* – the ontology update agent (solves the problem of updating the information on resources in ontology, it receives over LDAP or UDDI from the information service of the grid-system), *OntologyMatchmakingAgent* – the agent for resource matchmaking based on semantic descriptions of resources and requests.

When a resource is requested from the grid-service, first, it sends the request to the request agent. If the *RequestHandlingAgent*'s database contains the requested information, the *RequestHandlingAgent* returns the response. Otherwise, the *RequestHandlingAgent* redirects the request to the *OntologyMatchmakingAgent* (i.e., the resource search will demand for the ontology reasoning). The *OntologyMatchmakingAgent* also can generate requests for the resources for applications, acting on behalf of the ontology user.

All the service agents use the developed base ontology of grid resources, comprising grid-specific fundamental concepts and relations (*VirtualOrganization*, *GridMiddleware*, *GridResourse*, *GridUser*, *GridApplication*, *GridService* etc.). I.e., the objective of the base ontology creation is to provide flexibility and adaptability of a semantic information service and the possibility to use it in various grid-architectures and middleware.

The semantic grid-service is based on logical calculus of first-order predicates. When creating knowledge models, a special internal expert knowledge description language is used. The reasoning engine is a modified resolution method for predicate calculus.

The grid-service suggested and Decision Support System functioning in the last one have the following features:
- the service is built using the Web Service technology and it can easily interact in the system's service oriented architecture;
- system performance is ensured by implementing the deductive reasoning engine with different strategies of search reduction;
- the possibility to use facts received by reasoning;
- a built-in library of functions and an effective tool of connecting various computational software components;
- a convenient graphical environment, providing different operation modes;
- ultimate automation of knowledge base adjustment, including hints and syntaxis check tools;
- dialog user interaction and forming answers to questions is done in a natural language;
- the possibility of building a chain of events, facts, criteria and rules for explaining the solutions offered.

Fig. 2 shows the structure of the semantic grid-service being developed.

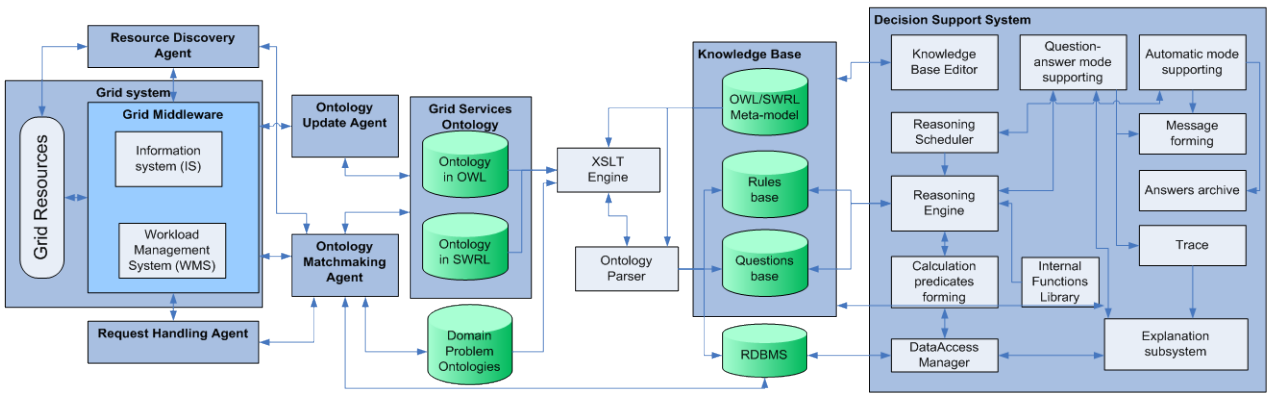

Fig.2. Semantic grid-service structure

To transform from OWL and SWRL ontology formats based on the XML space of names, an extensible style sheet transformation language XSLT (*eXtensible Stylesheet Language for Transformations*) is used. The ontology parser transforms directly to the internal format of the system knowledge base in the predicate calculus language.

This module uses a special meta-model (metaknowledge about rules of selection, detecting syntactical and semantic errors, and using and transforming ontological knowledge) for transformation. It includes rules for transferring OWL and SWRL ontological knowledge to grid-service rules and facts. One of the tasks of the parser is detecting inconsistency of elicited rules and facts.

The system knowledge base includes the rule base and the question base. The rule base serves for formalising the description of logical problems in the system knowledge representation language. The questions base serves for acquisition, storage and use of lists of possible questions for each logical problem.

The key component of the system is the reasoning module intended for reasoning of conclusions (answers) from a set of rules stored in the KB using the modified resolution method.

The data access module or the universal attribution procedure serves for forming a base of facts (the source and intermediate data of the logical problem being currently solved) for the selected system of axioms by the primary (base) relations to the data in the KB, which are partial instances of respective ontology concepts.

The explanation subsystem provides exhaustive information about the reasons of receiving a specific answer (facts and rules) involved in reasoning. This facilitates system testing and increases confidence in the result received.

In the knowledge base, each predicate shall have an indication of its type (primary is attributed with facts, the seondary is defined by a formula, and the computational is attributed with facts recieved by reasoning), an internal representation in a special formalised language, a semantic content, the predicate computation formula (for the secondary type), the fact determination formula (for the computational type), and the semantic content of recommendations. The semantic content of the predicate is a text of the answer to the question. The semantic content shall contain variables (which will be substituted with values found by reasoning).Besides these predicates, there is one more type of predicates, viz. order predicates (GREATER THAN, LESS THAN, EQUAL TO and others), the trueness of which is calculated by the internal program module of the reasoning algorithm after terms have been substituted with predicates of actual values.

2.2 Specific features of mapping OWL ontology and SWRL rules to service knowledge base

The elements of languages OWL and SWRL were mapped. Table 1 shows the elements of the OWL language, their respective constructs of the *SHIQ* descriptive logic, the internal representation of primary predicates and their semantic content in the KB, as well as facts to be written in the KB system. For the key elements of the SWRL language, the meta-model also includes the rules of representation in the DSS knowledge model.

Table 2 shows the predicate formulas of the meta-model, which are formed based on logic and assertions about ontological concepts, and properties and relations in OWL. The result of reasoning by these rules will be the formation of new facts used in subsequent queries (computational predicates). Besides, the meta-model describes rules, using which the problem of detecting inconsistencies of rules and facts elicited from the ontology can be solved.

Among them the following can be found: a class cannot be antithetical to itself, two classes cannot be antithetical and equivalent at the same time, not all instances have been found, not all links between instances have been confirmed, and others. Hence, during reasoning, the system allows establishing the following: ontology correctness, whether it is complete and are there any disturbances in semantic links, i.e. whether all elicited facts are confirmed by facts from the ontology.

Such a problem occurs often when it is necessary to align different representations about the subject area and semantics of objects described in terms of different ontologies. When translating ontologies to the service knowledge base, relations between ontological concepts of different ontological contexts are searched for; it is ascertained which concepts of another ontology can be equivalent, and which are superconcepts or subconcepts of an ontology that has been loaded a priori into the KB. If a conflict is identified, recommendations are given for feasible options of resolving it.

The secondary predicates in the DSS knowledge base indicate the necessity to solve reallogical-analytical problems on the ontology of a concrete subject area.

The complexity of existing proof methods in predicate calculus consists in their undecideability and the necessity of exhausting a big number of variants of proofs when searching for a solution. Hence, the time consumption in certain implementations of these methods can cel their practical value. The system suggested has implemented an effective reasoning algorithm for different strategies of reducing exhaustion. Its flow chart is shown in Fig. 3.

Tab.1. Primary predicates for the OWL ontology

OWL element	*SHIQ* description logic	Predicate internal representation	Semantic content	Facts
class	C	`isClass(v1)`	v1 is a class	isClass(*rdf:ID*)
Thing	$\top$	-	Thing is a class	isClass(*Thing*)
Nothing	$\bot$	-	Nothing is a class	isClass(*Nothing*)
subClassOf	$C1 \subseteq C2$	`isSubClassOf(v1,v2)`	v1 is a subclass of v2	isSubClassOf (*rdf:ID, rdf:resource*)
type	I:C	`isMemberOf(v1,v2)`	v1 is a member of class v2	isMemberOf(*rdf:ID,rdf:resource*)
oneOf	$C:=I1 \cup I2 \cup \ldots$	`isMemberOf(v1,v2)`	v1 is a member of class v2	isMemberOf(*rdf:ID,rdf:resource*)
domain	$\leq R \subseteq C$	`hasDomain(v1,v2)`	Property v1 has a domain restricted by class v2	hasDomain(*rdf:ID, rdf: resource*)
range	$\top \subseteq \forall R.C$	`hasRange(v1,v2)`	Property v1 is s elected from the range of class v2	hasRange(*rdf:ID, rdf: resource*)
ObjectProperty	R	`hasProperty(v1,v2)`	Class v1 has property v2	hasProperty(*rdf:ID, rdf:resource*)
hasValue	$C:=\exists R.I$	`PropertyInst(v1,v2,v3)`	Property v1 of the member of class v2 has value v3	PropertyInst(*rdf:ID,rdf:resource,rdf :resource*)
intersectionOf	$C3:=C1 \cap C 2$	`IntersectionClass(v1, v2, .., vN)`	Class v1 is an intersection of classes v2 and … vN	IntersectionClass(*rdf:ID, rdf:resource,…*)
unionOf	$C3:=C1 \cup C 2$	`UnionClass(v1,v2,…,vN)`	Class v1 is a merger of classes v2 and … vN	UnionClass(*rdf:ID, rdf:resource,…*)
subPropertyOf	$R1 \subseteq R2$	`hasSubProperty(v1,v2)`	Property v1 is a subproperty of v2	hasSubProperty(*rdf:ID, rdf:resource*)
complementOf	$C2:=\neg C1$	`ComplementClass(v1,v2)`	Class v1 does not belong to v2	ComplementClass(*rdf:ID, rdf:resource*)
equivalentClass	$C1 \equiv C2 \equiv \ldots$	`EquivalentClass(v1,v2)`	Class v1 is equivalent to class v2	EquivalentClass(*rdf:ID, rdf:resource*)
disjointWith	$C1 \subseteq \neg C2$ $C1 \cap C2 \equiv \bot$	`DisjointClass(v1,v2)`	Classes v1 and v2 are non-intersecting ones	DisjointClass (*rdf:ID,rdf:resource*)
equivalentProperty	$R1 \equiv R2 \equiv \ldots$	`EquivalentProp(v1,v2)`	Property v1 is equivalent to property v2	EquivalentProp(*rdf:ID, rdf:resource*)
inverseOf	$R1 \equiv R2^-$	`InverseProp(v1,v2)`	Property v1 is opposite to property v2	InverseProp(*rdf:ID, rdf:resource*)
TransitiveProperty	$R^+ \subseteq R$	`TransitiveProp(v1)`	Property v1 is transitive	TransitiveProp(*rdf:ID*)
SymmetricProperty	$R \equiv R^-$	`SymmetricProp(v1,v2)`	Property v1 is symmetrical to property v2	SymmetricProp(*rdf:ID,rdf:resource*)
FunctionalProperty	$\top \subseteq \leq R$	`FunctionalProp (v1)`	Property v1 is functional	FunctionalProp(*rdf:ID*)
InverseFunctional Property	$\top \subseteq \leq R^-$	`IFunctionalProp (v1)`	Property v1 is inversely functional	IFunctionalProp(*rdf:ID*)
sameAs	$I1=I2=\ldots$	`isSameAs(v1,v2)`	Instance v1 is identical to v2	isSameAs(*rdf:ID,rdf:resource*)
differentFrom, AllDifferent	$I1 \neq I2 \neq \ldots$	`isDifferent(v1,v2)`	Instance v1 differs from v2	isDifferent(*rdf:ID,rdf:resource*)
allValuesFrom	$C2:=\forall R.C 1$	`AllValuesFrom(v1,v2, v3)`	In class v1, property v2 takes the values of class v3	AllValuesFrom(*rdf:ID, rdf:resource, rdf:resource*)
someValuesFrom	$C2:=\exists R.C1$	`SomeValuesFrom(v1,v2, v3)`	In one of the members of class v1, property v2 takes the value from class v3	SomeValuesFrom(*rdf:ID, rdf:resource, rdf:resource)*

Service is adapted for being including in the intelligent agents construct that functions based on the JADE (*Java Agent Development Framework*) platform. Using ontologies during interaction of agents allows transferring highly structured information and dramatically simplifying operations with agents and programming them.

Tab.2. Computational predicates reflecting the logic of ontological concepts and relations

Description	Formula
Transitivity of classes	`isSubClassOf(v1,v2)& isSubClassOf(v2,v3)-> isSubClassOf(v1,v3)`
Transitivity of properties	`hasSubProperty(v1,v2)&hasSubProperty(v2,v3)-> hasSubProperty(v1,v3)`
Transitivity of a property	`TransitiveProp(v1)& PropertyInst(v1,v2,v3)& PropertyInst(v1,v3,v5)&~PropertyInst(v1,v2,v5)-> PropertyInst(v1,v2,v5)`
Inheritance of instances of classes	`isMemberOf(v1,v2)& isSubClassOf(v2,v3)&~ isMemberOf(v1,v3)-> isMemberOf(v1,v3)`
Inheritance of instances of properties	`PropertyInst(v1,v2,v3)&hasSubProperty(v1,v4)&~PropertyInst(v4,v2,v3)->PropertyInst(v4,v2,v3)`
Transitivity of a domain	`hasSubProperty(v1,v2)&hasDomain(v2,v3)&~hasDomain(v1,v3)->hasDomain(v1,v3)`
Transitivity of a range	`hasSubProperty(v1,v2)&hasRange(v2,v3)&~hasRange(v1,v3)->hasRange(v1,v3)`
Symmetry of properties	`SymmetricProp(v1,v2)&~SymmetricProp(v2,v1)->SymmetricProp(v2,v1)`
Opposition of properties	`InverseProp(v1,v2)&~ InverseProp(v2,v1)-> InverseProp(v2,v1)`
Inversely functional property	`InverseProp(v1,v2)& FunctionalProp (v1)&~ IFunctionalProp(v2)-> IFunctionalProp(v2)`
Functional property	`InverseProp(v1,v2)& IFunctionalProp (v1)&~ FunctionalProp(v2)-> FunctionalProp(v2)`
Instances of equivalent classes	`EquivalentClass(v1,v2)&isMemberOf(v3,v1)&~isMemberOf(v3,v2)-> isMemberOf(v3,v2)`
Instances of nonintersecting classes are different	`DisjointClass(v1,v2)& isMemberOf(v3,v1)& isMemberOf(v4,v2)&~ isDifferent(v3,v4)-> isDifferent(v3,v4)`
Two classes including each other are equivalent	`isSubClassOf(v1,v2)&isSubClassOf(v2,v1)&~EquivalentClass(v1,v2)->EquivalentClass(v1,v2)`
Properties of equivalent classes	`isSameAs(v1,v2)& PropertyInst(v3,v1,v4)&~PropertyInst(v3,v2,v4)->PropertyInst(v3,v2,v4)`
Instances of equivalent properties	`EquivalentProp(v1,v2)&PropertyInst(v1,v3,v4)&~PropertyInst(v2,v3,v4)->PropertyInst(v1,v3,v4)`
Two properties including each other are equivalent	`isSubPropertyOf(v1,v2)&isSubPropertyOf(v2,v1)&~EquivalentProp(v1,v2)->EquivalentProp(v1,v2)`
Equivalency of instances with identical functional properties	`FunctionalProp(v1)& PropertyInst(v1,v2,v4)& PropertyInst(v1,v3,v4)&~ isSameAs(v2,v3)-> isSameAs(v2,v3)`
Restriction of class properties	`PropertyInst(v1,v2,v3)&isMemberOf(v2,v4)&AllValuesFrom(v4,v1,v5)->isMemberOf(v3,v5)`
Relation of instances by a domain-restricted property	`hasDomain(v1,v2)& PropertyInst(v2,v3,v4)&~isMemberOf(v3,v1) ->isMemberOf(v3,v1)`
Relation of instances by a range-restricted property	`hasRange(v1,v2)& PropertyInst(v2,v3,v4)&~isMemberOf(v3,v1) ->isMemberOf(v3,v1)`
Each definite class if a subclass of owl:Thing	`isClass(v1)&~isSubClassOf(v1,Thing)-> isSubClassOf(v1,Thing)`

3 Conclusion

Therefore, the application of semantic technologies opens wide prospects for further improvement of base elements of grid-systems, makes the task of search for, coordination and distribution of resources in grid-systems less labor-consuming, and its solution – more efficient.

The paper suggests an approach for mapping elements of ontological languages OWL and SWRL on a knowledge base in a decision support system based on first-order predicate logic. It has several advantages that distinguish it from existing ontology reasoning engines. The grid-service is practically implemented in the form of the prototype of a knowledge-oriented system relying on the agent approach, based on the JADE platform, which comprises intellectual agents functioning in it, that take decisions and interact with the with the help of an ontological knowledge base and an reasoning engine.

The author's researches are also connected with solution of the tasks of the load balancing and effective algorithms o resource management in the grid, extraction of knowledge from the textual unstructured information in the Internet by the system agents, as well as ontology building automation.

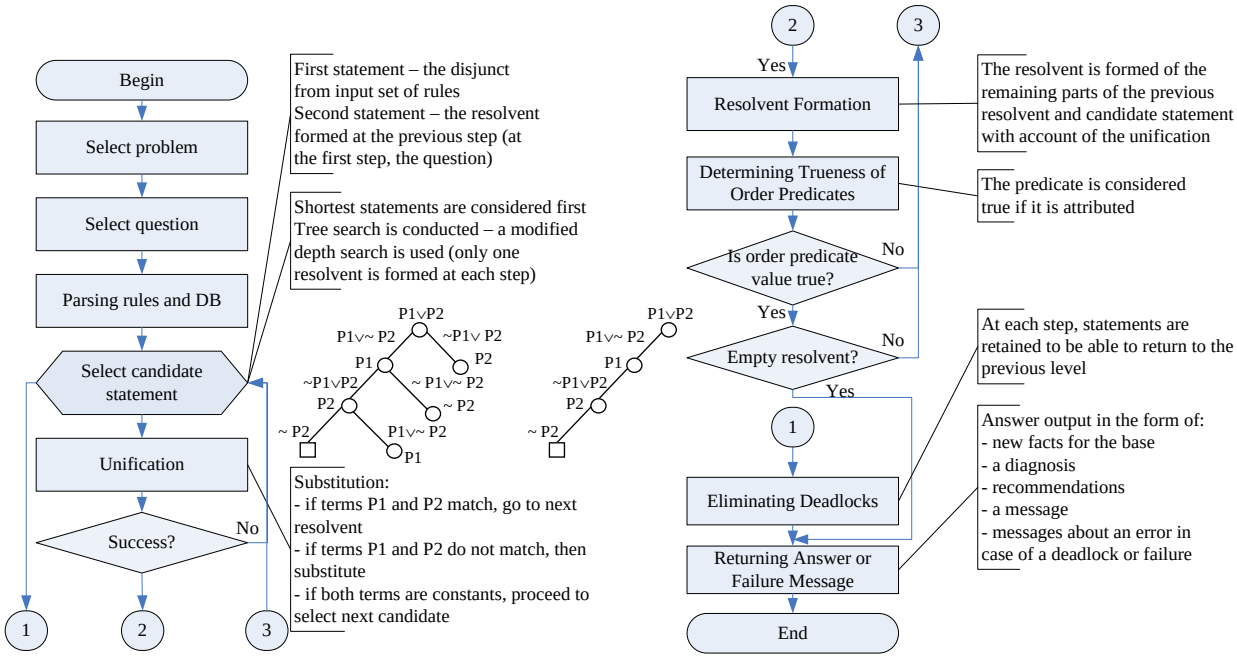

Fig. 3.Outline flowchart of the logic reasoning algorithm

References

[1] Xiong Zenggand, Zhang Xuemin, Liu Li: Resource management model and resource discovery algorithm for P2PGrid. *Journal of networks*, Vol.6, No. 8, 2011. – pp. 1187-1194.

[2] Mahamat Hassan, Azween Abdullah: A new grid resource discovery framework. *The International Arab Journal of Information Technology*, Vol. 8, No. 1, 2011. – pp. 99-107.

[3] Tangmunarunkit H., Decker S., Kesselman C.: Ontology-based resource matching in the Grid – the Gridmeets the semantic web. *The SemanticWeb-ISWC2003*. – 2003. – pp. 706-721.

[4] Said M., Kojima I. S-MDS: Semantic Monitoring and Discovery System for the Grid. *Journal of Grid Computing*. — 2008. —Vol. 7, N 2. —P. 205—224.

[5] A.S. Pospeshniy, S.G. Stirenko: Онтология ресурсов для семантического информационного сервиса грид. *Электронное моделирование*// Ontology of Resources for the Semantic Information Grid-Service. *Electronic Simulation*. – 2011. – V.33. No.4. – PP. 35-47.

[6] A.V. Zhuchkov, S.A. Arnautov, N.V. Tverdokhlebov, S.V. Golitsyn, I.G. Strizh: Использование онтологий при работе с гетерогенными федеративными массивами данных в распределенных информационных системах// Use of ontolgies in Operating Heterogeneous Federative Data Arrays in Distributed Information Systems. *Collected Papers "Distributed Computations and Grid-Technologies in Science and Education"*. - Dubna, 2004. - PP. 99-103.

[7] Friedman-Hill E. Jess in Action: Java Rule-Based Systems. *Manning Publications*, 2003, 480p.

[8] Vassiliadis V.: Thea: A Web Ontology Language – OWL Parser for SWI-Prolog. 2005, http://www.semanticweb.gr/TheaOWLParser/

[9] N.A. Skvortsov: Использование системы интерактивного доказательства для отображения онтологий// Using Interactive Proof Systems For Ontology Mapping // Proc. 8th All-Russian Scientific Conference *Electronic Libraries: Perspective Methods and Technologies, Electronic Collections*, Suzdal, Russia, 2006.