УДК 004.82 + 004.89

O.V. PROKHOROV, O.M. PAKHNINA

National Aerospace University named after M.Ye. Zhukovsky «Kharkiv Aviation Institute», Ukraine

THE APPROACH TO ORGANISING INFERENCE ON ONTOLOGIES IN MULTIAGENT SYSTEMS

It's provided to use multiagent system's technologies and ontologies for solving problems, directly relating to analyses, diagnostic, evaluation and identification of situations, forecasting of outcomes, information fusion, coordination of unmanned aerial vehicle's group operation. An approach is suggested, which allows analysing the correctness of ontology and its completeness, searching for disturbances in semantic links, agreement of representations about the subject area and the semantics of objects described in terms of different ontologies during their merging. Issues of mapping the elements of the OWL and SWRL-languages to a knowledge base of a decision support system based on the first-order predicate logic are discussed. The key principles of implementing an effective inference algorithm on ontologies for different strategies of search reduction are described.

Key words: *multiagent system, inference, knowledge base, decision support system, predicate logic, ontology, simulation modeling, knowledge-oriented system, agent approach.*

Introduction

The enhancement of scope of solvable tasks and its complication, substantial increase of autonomy and improving reliability of unmanned aerial vehicles (UAV) necessitate the development of aircraft control systems of new generation, based on multipurpose use of modern intelligent information technologies. Such systems make it possible to operate under the conditions when air situation is rapidly changing, if random environmental disturbances and another uncertainties appear. Viz multiagent system's technologies are used for solving problems, directly relating to analyses, diagnostic, evaluation and identification of situations, forecasting of outcomes, information fusion, coordination of unmanned aerial vehicle's group operation.

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. Ontologies are used as a tool for structuring and representing information in such systems [1].

Since an ontology defines the terms in a specific knowledge area, it should be described by a formal language based on mathematical logic principles. Then, clear, detailed and complete definitions can be formulated for classes of objects, their properties and interrelations. In turn, ontology processing tools can automatically infer certain conclusions by being based on mathematical logic principles.

Analysis of Recent Research and Publications

Plenty of papers are dedicated to intelligent aircraft control systems' development based on adaptive controllers with self-organization and predictive algorithms' implementation. However intelligent systems that actively use experts' knowledge and experience and focuse on immediate operation in UAV's and its subsystems' performance (based on analyses of ever changing external and internal environment model) are of special interest. The key tool for structuring and representing knowledge in multiagent systems are ontologies.

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 (meaningful interpretation of conclusions received by formal logic methods) and decidability (computations will be completed in finite time);

- *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 (asking, querying), and application of reasoning to available knowledge (reasoning, entailment).

To solve the first problem, the most widespread tool is *SPARQL* [3], a language for querying RDF, which accepts RDF-data as a set of statements or triplets *rdf(Subject, Predicate, Object)*.

The OWL DL level is the one that is focused to currently existing knowledge description systems, and logical programming and inference 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* [4] 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 inference 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*[5] (*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 inferring 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 3.3.1 [6], 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. From the viewpoint of structure, a matter of principle is separating the ontology into a terminological part (TBox) and data part (ABox, "A" means *assertions*) [8]. In the process of responding to queries, the T-part of the ontology is transformed to a logic-type program, which is then executed using the A-part (axioms) as data.

Thea is a library for the Prolog language [9], 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 [10] suggests mapping an ontological model on an AMN (*Abstract Machine Notation*) of the B-Toolkit system.

Objective of the Study

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 decision support system (DSS) on ontologies, that will be included in UAV software system structure, a challenge within the framework of this study.

Decision Support System on Ontologies

The system 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 system suggested has the following features:

- the decision support system shell is built using the ActiveX technology and it can easily interact in the client-server intranet environment;
- system performance is ensured by implementing

the deductive reasoning engine with different strategies of search reduction;

- the possibility to use facts received by inference;
- a built-in library of functions and an effective tool of connecting various computational program modules;
- a convenient graphical environment, providing different operation modes;
- ultimate automation of knowledge base adjustment, including hints and syntaxis check tools;
- dialog interaction and forming answers to questions is done in a natural language;
- various forms of generating answers: diagnoses, recommendations, and urgent messages;
- the possibility of building a chain of events, facts, criteria and rules for explaining the solutions offered.

Fig. 1 shows the structure of the ontology-based DSS being developed.

To transform from OWL and SWRL ontology formats based on the XML space of names, an extensible stylesheet transformation language XSLT [11] (*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 DSS 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 inference 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.

The systems functions in the following basic modes:

- adjustment and acquisition of the knowledge base;
- consulting or the question-answer mode (the system outputs answers – diagnoses, recommendations, messages – to specific questions in a natural language in real-time decisioning);
- automatic (formation and archiving of answers to previously formulated questions – parameter control with issuance of urgent messages and formation of recommendations); and
- explanation of results (exhaustive information is output about the reasons of receiving the given answer – facts and rules involved in the reasoning process).

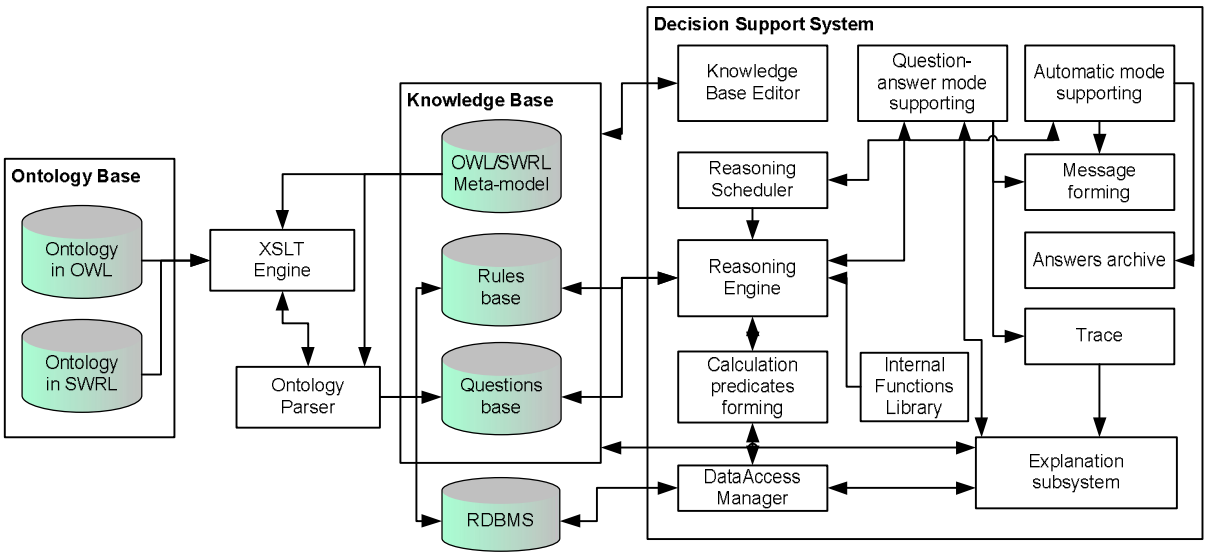

Fig. 1.Ontology-based DSS structure

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 inferencing algorithm after terms have been substituted with predicates of actual values.

The Features of Representation of OWL and SWRL Ontologies on the Knowledge Base of the System Suggested

From the viewpoint of OWL, ontology

O=<T, A, R, Dom, C, F>

is a description of concepts (classes) **T** *owl:Class* in the subject area being considered, and of properties *owl:ObjectProperty* for each concept, which describe different attributes **A** and relations **R** of the concept with account of a set of admissible values defined by domain **Dom** and restrictions **C** *owl:Restriction* imposed on properties. An arbitrary group of ontology elements can form fragment **F**. There are two a priori defined classes: *owl:Thing* – the most general class containing all concepts; and *owl:Nothing* – an empty class. Each class being created is a subclass of *owl:Thing* and a superclass *owl:Nothing*. OWL has two kinds of properties: *object properties* – link objects with other objects; *datatype properties* – link objects with data type values.

In general, element *owl:Restriction* contains element *owl:onProperty* and one or more declarations of restrictions. The following types of restrictions exist: restrictions on the type of property values: *owl:allValuesFrom* (describes the class of possible values, which can be taken by a property specified by element *owl:onProperty*), *owl:hasValue* (a property specified by element *owl:onProperty* should have a concrete value) and *owl:someValuesFrom* (describes a class of possible values, which at least one of the properties can take); and cardinality restrictions (value numbers) [1].

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 DSS 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.

Table 3 shows an illustrative variant of questions to the DSS. When specifying a constant, the name of the respective variable is enclosed by the symbol «*?*». To receive all possible answers, indicate the keyword *all* in the system; otherwise, the first answer found will be returned.

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

Let the following rule be written in the knowledge base, area of which is related to land object recognition by UAV:

- internal representation: **ACTION(V1)**,
- formula:

OBJECT(V1,V2) ∧ TYPE(V1,V3) ∧
∧ EQUAL(V3,100) → ACTION(V2).

The base of facts received from the ontology contains the following primary predicates:

OBJECT(5,TP), TYPE(5,100).

The question to the system: **ACTION(VX)**.

Table 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	$C:=\exists R.I$	`Property-Inst(v1,v2,v3)`	Property v1 of the member of class v2 has value v3	Property-Inst(*rdf:ID*,*rdf:resource*,*rdf: resource*)
intersectionOf	$C3:=C1 \cap C2$	`Intersection-Class(v1,v2,..,vN)`	Class v1 is an intersection of classes v2 and … vN	IntersectionClass(*rdf:ID*, *rdf:resource*,…)
unionOf	$C3:=C1 \cup C2$	`Union-Class(v1,v2,…,vN)`	Class v1 is a merger of classes v2 and … vN	UnionClass(*rdf:ID*, *rdf:resource*,…)
subPropertyOf	$R1 \subseteq R2$	`hasSubProp-erty(v1,v2)`	Property v1 is a subproperty of v2	hasSubProperty(*rdf:ID*, *rdf:resource*)
complementOf	$C2:=\neg C1$	`Complement-Class(v1,v2)`	Class v1 does not belong to v2	ComplementClass(*rdf:ID*, *rdf:resource*)
equivalentClass	$C1 \equiv C2 \equiv \ldots$	`Equivalent-Class(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	$R1 \equiv R2 \equiv \ldots$	`Equivalent-Prop(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*)
Functional-Property	$\top \subseteq \leq R$	`FunctionalProp (v1)`	Property v1 is functional	FunctionalProp(*rdf:ID*)
InverseFunctionalProperty	$\top \subseteq \leq R^-$	`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	$C2:=\forall R.C1$	`AllValues-From(v1,v2,v3)`	In class v1, property v2 takes the values of class v3	AllValuesFrom(*rdf:ID*, *rdf:resource*, *rdf:resource*)
someValuesFrom	$C2:=\exists R.C1$	`SomeValues-From(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)*

Table 2

Computational predicates reflecting the logic of ontological concepts and relations

Description	Formula
Transitivityof classes	`isSubClassOf(v1,v2)& isSubClassOf(v2,v3)-> isSubClassOf(v1,v3)`
Transitivityof properties	`hasSubProperty(v1,v2)&hasSubProperty(v2,v3)-> hasSubProperty(v1,v3)`
Transitivityof a property	`TransitiveProp(v1)& PropertyInst(v1,v2,v3)& Property-Inst(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)`
Transitivityof a domain	`hasSubProperty(v1,v2)&hasDomain(v2,v3)&~hasDomain(v1,v3)-> ->hasDomain(v1,v3)`
Transitivityof 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)`

Table 3

Variants of questions to the inference system on ontologies

Semantic content of the question	Internal representation
Is the given object an instance of a class?	`all isMemberOf(?vx1?,?vx2?)`
What objects are instances of the given class?	`all isMemberOf(vx1,?vx2?)`
In what classes is the given object an instance?	`all isMemberOf(?vx1?,vx2)`
Is this class a subclass of the given class?	`all isSubClassOf(?vx1?,?vx2?)`
What subclasses does the given class have?	`all isSubClassOf(vx1,?vx2?)`

After being transformed to the clausal form (required for the resolution method),we receive the following set of disjuncts:

~ACTION(V1,V2)∨~TYPE(V1,V3)∨
∨EQUAL(V3,100) ∨ ACTION(V2),
OBJECT(5,TP),
TYPE(5,100),
~ ACTION(VX).

The inference process continues until the graph root yields a statement containing constants as terms, the constants being the solution of the problem (Fig. 2).

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 inferencing algorithm for different strategies of reducing exhaustion. Its flow chart is shown in Fig. 3.

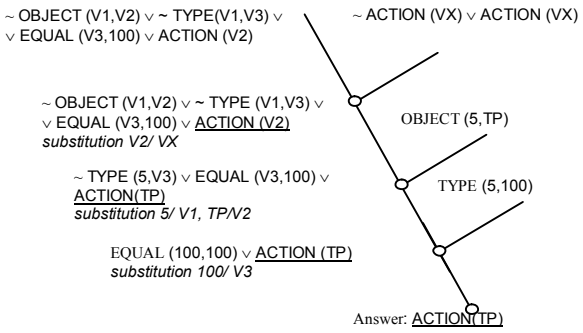

Fig. 2.Example of inference tree

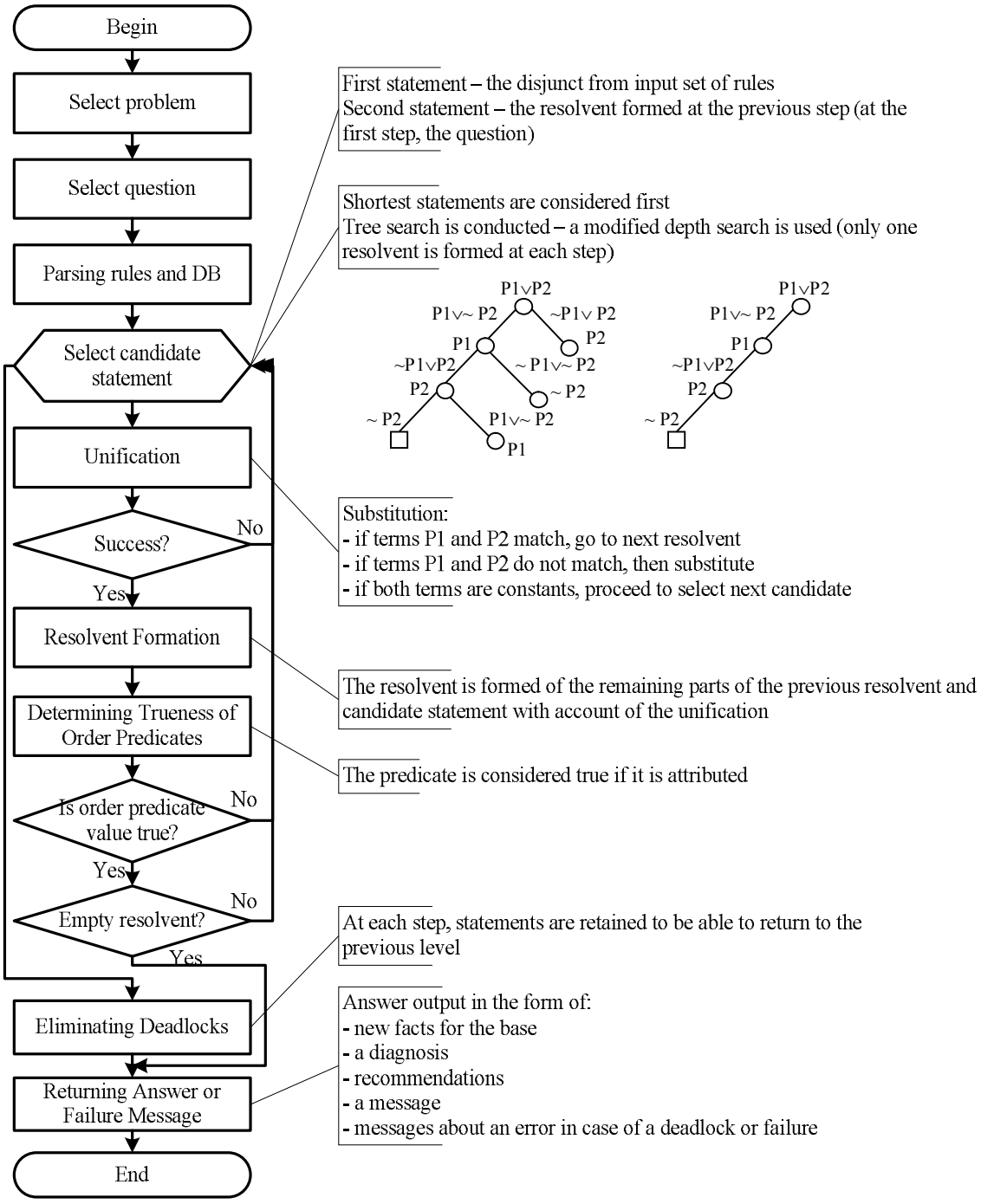

Fig. 3.Outline flowchart of the logic inference algorithm

Basic DSS modules (in particular, the reasoning module) without the graphic environment are 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.

Conclusion

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 logical inference rules, when working with ontologies, allow manipulating concepts and data far more effectively for elicitation of new knowledge. Practical implementation of the ontological approach is designed as a prototype of a knowledge-oriented system for simulation of production processes based on an agent approach on a JADE platform. It includes functioning intellectual agents, which make decisions and interact with the help of an ontological knowledge base and a reasoning engine of the system suggested. This allows solving the problem of joint use of ontologies by program agents for accumulation and repeated use of knowledge, and creation of simulation models and programs, which operate with ontologies, rather than with rigidly specified data structures for analysing knowledge in a subject area.

Developed integrated framework provides support of all stages for intelligent decision support systems' implementation and their adaptation for handling of applied problems for the purposes of UAVs.

References

1. Prokhorov, A.V. Features of using ontologies during interaction of agents in a production process simulation system [Text] / A.V. Prohorov, O.M. Pakhnina // Proceedings of the Kharkiv Air Force University: Sci. Periodical Edition. – Kh.: I. Kozhedub Kharkiv Air Force University. – Kharkiv (Ukraine). – 2008. – Issue 3 (18). – P. 164-170.

2. McGuinness, D.L. OWL web ontology language overview, W3C Recommendation. [Electronic resource] / D.L. McGuinness. – 2004. – Access mode: http://www.w3.org/TR/owl-features/ – 16.06.2013.

3. Prudhommeaux, E. SPARQL Query Language for RDF, W3C Candidate Recommendation [Electronic resource] / E. Prudhommeaux. – 2007. – Access mode: http://www.w3.org/TR/rdf-sparql-query/ – 16.06.2013.

4. McBride, B. An Introduction to RDF and the Jena RDF API. [Electronic resource] / B. McBride. – 2008. – Access mode: http://jena.sourceforge.net/tutorial/RDF_API. – 16.06.2013.

5. Horrocks, I. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. [Electronic resource] / I. Horrocks. – 2004. – Access mode: http://www.daml.org/rules/proposal/ – 16.06.2013.

6. Ovdey, O.M. Review of ontology engineering tools [Electronic resource] / O.M. Ovdey // Proc. 6th All-Russian Scientific Conference Electronic Libraries: Perspective Methods and Technologies, Electronic Collections - RCDL'2004. – Pushchino (Russia). – 2004. – Access mode: http://www.impb.ru/~rcdl2004/ – 16.06.2013.

7. Friedman-Hill, E. Jess in Action: Java Rule-Based Systems [Text] / E. Friedman-Hill// Manning Publications. – Greenwich, CT (USA). – 2003. – P. 356-410.

8. Motik, B. A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes [Text] / B. Motik // Proc. of the 13th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2006). – Phnom Penh (Cambodia). – 2006. – P. 236-247.

9. Vassiliadis, V. Thea: A Web Ontology Language – OWL Parser for SWI-Prolog. [Electronic resource] / V. Vassiliadis. – 2005. – Access mode: http://www.semanticweb.gr/TheaOWLParser/ – 16.06.2013.

10. Skvortsov, N.A.. Использование системы интерактивного доказательства для отображения онтологий (Using interactive proof systems for ontology mapping) [Text] / N.A. Skvortsov // Proc. 8th All-Russian Scientific Conference Electronic Libraries: Perspective Methods and Technologies - RCDL'2006. – Suzdal (Russia). – 2006. – P. 268-273.

11. Clark, J. XSL Transformations (XSLT), W3C Recommendation. [Electronic resource] / J. Clark. – 1999. – Access mode: http://www.w3.org/TR/ – 16.06.2013.

Поступила в редакцию 6.06.2013, рассмотрена на редколлегии 12.06.2013

Рецензент: д-р техн. наук, проф., зав. каф. информационных технологий проектирования ЛА Е.А. Дружинин, Национальный аэрокосмический университет им. Н.Е.Жуковского «ХАИ», Харьков.

ПІДХІД ДО ОРГАНІЗАЦІЇ ЛОГІЧНОГО ВИВЕДЕННЯ НА ОНТОЛОГІЯХ В МУЛЬТИАГЕНТНИХ СИСТЕМАХ

О.В. Прохоров, О.М. Пахніна

Для вирішення завдань, безпосередньо пов'язаних з аналізом, діагностикою, оцінюванням та розпізнаванням ситуацій, прогнозуванням розвитку подій, узагальненням інформації, координацією та узгодженням сумісних дій групи безпілотних літальних апаратів (БПЛА) запропоновано використовувати технології мультиагентних систем та онтології. Запропоновано підхід, що дозволяє проводити аналіз коректності онтології та її повноти, пошук порушень у семантичних зв'язках, узгодження уявлень про предметну область і семантику об'єктів, що описані у термінах різних онтологій при їх об'єднанні. Розглянуто питання відображення елементів мов OWL та SWRL у базі знань системи підтримки прийняття рішень, що базується на логіці предикатів першого порядку. Описано основні принципи реалізації ефективного алгоритму логічного виведення на онтологіях при різних стратегіях скорочення перебору.

Ключові слова: мультиагентна система, база знань, система підтримки прийняття рішень, логіка предикатів, онтологія, імітаційне моделювання, знання-орієнтована система, агентний підхід.

ПОДХОД К ОРГАНИЗАЦИИ ЛОГИЧЕСКОГО ВЫВОДА НА ОНТОЛОГИЯХ В МУЛЬТИАГЕНТНЫХ СИСТЕМАХ

А.В. Прохоров, Е.М. Пахнина

Для решения задач, непосредственно связанных с анализом, диагностикой, оцениванием и распознаванием ситуаций, прогнозированием развития событий, обобщением информации, координацией и согласованием совместных действий группы беспилотных летательных аппаратов (БПЛА) предлагается использовать технологии мультиагентных систем и онтологии. Предложен подход, позволяющий проводить анализ корректности онтологии и ее полноты, поиск нарушений в семантических связях, согласование представлений о предметной области и семантику объектов, описанных в терминах разных онтологий при их объединении. Рассмотрены вопросы отображения элементов языков OWL и SWRL в базе знаний системы поддержки принятия решений, основанной на логике предикатов первого порядка. Описаны основные принципы реализации эффективного алгоритма логического вывода на онтологиях при различных стратегиях сокращения перебора.

Ключевые слова: мультиагентная система, база знаний, система поддержки принятия решений, логика предикатов, онтология, имитационное моделирование, знаниеориентированная система, агентный подход.

Прохоров Александр Валерьевич – канд. техн. наук, доцент, доцент кафедры информационных управляющих систем, Национальный аэрокосмический университет им. Н.Е. Жуковского «Харьковский авиационный институт», Харьков, Украина, e-mail: avprohorov@yahoo.com.

Пахнина Елена Михайловна – инженер кафедры информационных управляющих систем, Национальный аэрокосмический университет, им. Н.Е. Жуковского «Харьковский авиационный институт», Харьков, Украина, e-mail: elena.pakhnina@khai.edu.