Computer Science • Vol. 8 Special Edition • 2007

Kamil Szymański*, Grzegorz Dobrowolski*,
Jarosław Koźlak*, Anna Zygmunt*

A PROPOSITION OF KNOWLEDGE MANAGEMENT METHODOLOGY FOR THE PURPOSE OF REASONING WITH THE USE OF AN UPPER-ONTOLOGY

This article describes a proposition of knowledge organization for the purpose of reasoning using an upper-ontology. It presents a model of integrated ontologies architecture which consists of a domain ontologies layer with instances, a shared upper-ontology layer with additional rules and a layer of ontologies mapping concrete domain ontologies with the upper-ontology. Thanks to the upper-ontology, new facts were concluded from domain ontologies during the reasoning process. A practical realization proposition is given as well. It is based on some popular Semantic Web technologies and tools, such as OWL, SWRL, nRQL, Protégé and Racer.

Keywords: *ontology, knowledge management, ontology integration, upper-ontology, reasoning*

PROPOZYCJA METODOLOGII DO ZARZĄDZANIA WIEDZĄ NA CELE WNIOSKOWANIA Z WYKORZYSTANIEM ONTOLOGII WYŻSZEGO POZIOMU

Artykuł przedstawia propozycję organizacji wiedzy na cele wnioskowania z wykorzystaniem ontologii wyższego poziomu. Prezentuje model architektury zintegrowanych ontologii, składający się z ontologii domenowych z instancjami, współdzielonej ontologii wyższego poziomu z dodatkowymi regułami oraz z warstwy ontologii mapujących. Dzięki wiedzy wyższego poziomu uzyskano nowe fakty podczas wnioskowania. Zaprezentowano również propozycję praktycznej realizacji omawianego podejścia przy użyciu popularnych narzędzi i technologii dla Semantic Web, takich jak OWL, SWRL, nRQL, Protégé i Racer.

Słowa kluczowe: *ontologia, zarządzanie wiedzą, integracja ontologii, ontologia wyższego poziomu, wnioskowanie*

* Institute of Computer Science, AGH University of Science and Technology, Kraków, Poland, camel_sz@go2.pl; grzela@agh.edu.pl; kozlak@agh.edu.pl; azygmunt@agh.edu.pl

1. Introduction

Ontologies [4] are often used for knowledge representation in the Internet. One of the most popular ontology description languages is OWL [15] with semantics based on Description Logic [1]. Integrated ontologies [2, 19] enable us to share and reuse knowledge from many different sources. One of the integration methods is mapping [5, 10, 16, 18], which enable us to express equivalence or similarity between concepts of different ontologies. This way we can connect schemas of different ontologies, broadening the description range of our ontology.

It is quite probable that once created an ontology needs some revision or extension a little later. That modification can be caused by changing requirements or new kind of information, which was not available to us at the time of ontology specification. We need to integrate our ontology with another one to extend the description range of our ontology and obtain new information from instances. The latter is possible when the domains of integrated ontologies are similar or when we are integrating our ontology with an upper-ontology. The paper focuses on the latter case. The concept of an upper-ontology and its role in information integration and reuse is under research of Standard Upper Ontology Working Group [24]. One of the effects of their work is a general-purpose upper ontology, called SUMO [25, 17], which was successfully reused by many ontologies of different domains, like economy, finance, geography and more [26]. This proved that a properly layered knowledge structure can be beneficial to many other ontologies in terms of knowledge reuse. Let us use the following medical example.

Imagine that a certain hospital has a knowledge base in the form of an ontology. It contains the information about patients and their treatment. After some time, new diseases are described in some other ontology as well as new symptoms of illnesses already known. We would like to extend our knowledge base with these new facts. Of course it is better to reuse the knowledge from the other ontology, if possible, instead of writing the same from scratch. These two ontologies can be integrated after mappings between corresponding concepts are created. The ontology of a new disease will contain for example some rules about diagnosing that illness with methods already known. The ontology of the new disease should be a part of an upper-ontology, because there may be other hospitals which can benefit from new information contained in that ontology. The mapping process, which was mentioned earlier, will be one of the use cases in our knowledge management system. The second feature of the application is connected with the use of those integrated ontologies. Doctors are able to query the extended knowledge base and they can receive information whether their patients can suffer from the new illness.

One of the goals of Semantic Web [9] is knowledge integration and reuse. When we have integrated ontologies, we operate on a bigger knowledge. Moreover, new facts can be inferred during the reasoning phase [1]. Knowledge integration is a complex task because of the freedom of world modeling and different information range of

used concepts, naming just a few of the potential problems. That is why the process is believed to give correct results only when performed by a man [19, 20].

The paper focuses on a crucial issue in knowledge management like correct and clear data organization, which divides information of different kinds into separate modules. It should support knowledge reuse and inferring new facts during automated reasoning. A universal model for integrated ontologies is presented in the article. It consists of a layer of concrete domain ontologies with instances, an upper-ontology layer with additional rules and ontologies mapping knowledge from domain ontologies with corresponding upper-ontology concepts. There is obviously an interface layer to query integrated ontologies as well. Thanks to the ontology integration, new facts from domain ontologies could be concluded during reasoning. The article also gives answers to practical engineers' questions about technology and tools that can be used for the presented knowledge management methods, along with a short description as well as pros and cons in using them.

2. The concept of a knowledge management system operating on integrated ontologies

Several knowledge elements were specified. Each of them has a corresponding user with a certain role. There are the following layers in the system:

- **An upper-ontology.** It defines universal rules with the use of which knowledge base queries are performed. The upper-ontology usually contains generalized domain-specific knowledge. It contains classes, properties and rules needed to obtain answers to predefined question templates. A person called an upper-ontology expert is associated with this ontology layer. It is a person who defines this ontology, namely its classes, properties, rules as well as the mentioned question templates. Then, the upper-ontology is used during the reasoning process.
- **A domain ontology.** It is an ordinary ontology with instances. This is the lowest layer which carry basic domain-specific data on which we reason. This ontology is created by a user called domain expert – a person who knows this domain well.
- **A mapping ontology.** It is an intermediate layer which binds upper-ontology with domain ontology. It is usually prepared by a domain expert who knows domain ontology very well. He obviously has to learn the upper-ontology structure in order to create correct mappings between proper classes and roles of both ontologies. He can also create additional mapping rules if roles and classes of the domain ontology do not have direct corresponding equivalent elements in the upper-ontology. This feature considerably extends mapping possibilities between the different ontologies.
- **A query interface layer.** It is used by an end-user who chooses upper-ontology – domain of interest, the user's question to knowledge base, selected from a list of predefined templates and concrete domain ontology from which the user wants to receive the answer.

The interface was mentioned only to give the reader an idea about the system from the perspective of an end-user. Other system components will not be described in details as it is beyond the scope of this paper. It concentrates on layered ontology model and technologies used in knowledge management. The system itself is an interface in the mapping process (which in fact needs to be performed by a domain expert himself) and in asking queries. It also helps changing query templates to real questions and covers all data exchange between the system and a reasoning engine.

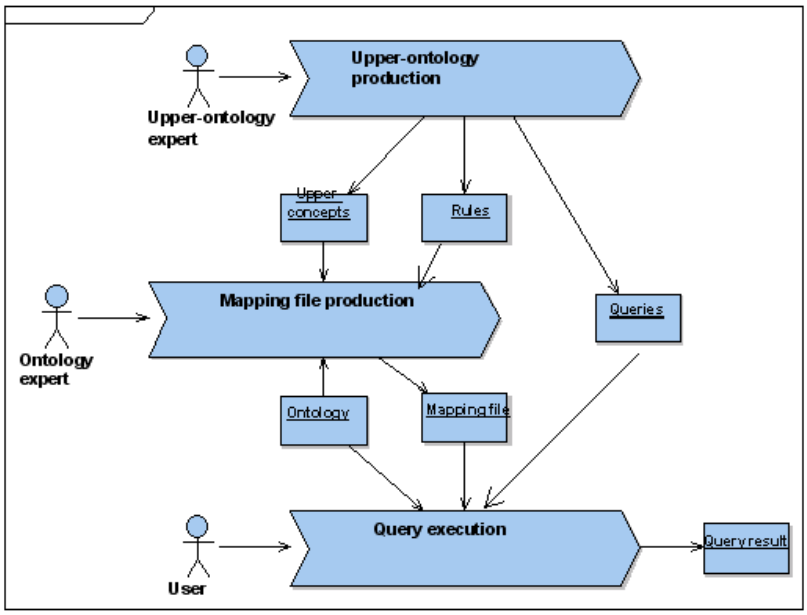

Fig. 1. System users, ontologies and dependencies

Figure 1 shows the users of the system and their specific roles associated with different ontologies. An upper-ontology expert creates the upper-ontology with question templates and rules to obtain new facts. A domain ontology expert creates the mapping ontology in which he associates proper classes and properties of a domain ontology with corresponding classes and properties of an upper-ontology. He can do it by defining direct equivalence mappings or additional mapping rules. An ordinary end-user performs questions to knowledge base. He receives information from the extended knowledge of the integrated ontologies.

3. Elements of implementation

This section is an extension of the presented methodology. It adds concrete technology and suggestions of tools along with their short description and presents the general system architecture.

3.1. Knowledge management

3.1.1. RacerPro – renamed abox and concept expression reasoner

Racer [21, 6] was the first reasoning engine with OWL support. It is one of the fastest reasoners available nowadays. It is based on description logic and can operate on ontologies. Racer has its own language, similar to LISP [13], which is used for facts and query definition. It reads OWL ontology and internally Racer sees it as a SHIQ description logic [1]. The communication with Racer system can be done either using the TCP or HTTP protocol. There are a few tools that can communicate with the reasoner, such as JRacer [21], written in Java, or LRacer [21], with LISP API. As for queries to the knowledge base, one can use RICE [23] which has a graphical interface.

3.1.2. Upper-ontology

An upper-ontology is an OWL ontology whose most important part are rules, thanks to which new facts can be inferred from the ontology. Rules are written in SWRL [29]. Each rule is an implication whose prerequisite is a conjunction of facts and conclusion is always formed of one single fact.

Each part of the conjunction in prerequisite can be one of the two following kinds. The fist one is a statement saying that a certain instance belongs to some class, e.g. $C(?x)$ – an instance, represented by a x variable, belongs to class C. The second possibility is a statement saying that two instances are connected with some role, e.g. $p(?x, ?y)$ – x and y are connected by a role p. When SWRL rules are used in ontology, we define some additional conditional relationships between instances, e.g.

$$hasParent(?x, ?y) \wedge hasBrother(?y, ?z) \rightarrow hasUncle(?x, ?z)$$

which means that if y is a parent of x and z is a brother of y, then z is an uncle of x. Developers can freely operate on the ontology structure and SWRL rules using the Protégé API [30, 11]. Expression parser with validation module can be especially useful during system interface implementation.

As an upper-ontology does not depend on any domain ontology, it must contain its own classes and properties used in rules. These classes and properties are the elements to which classes and properties of a domain ontology are mapped.

The third element connected with the upper-ontology layer is a set of prepared questions or templates of them. They can only use elements of the schema contained in the upper-ontology. Questions to knowledge base are realized using nRQL [28]. The system interface has an editor for easier operations on questions.

nRQL (new Racer Query Language), a query language used in the Racer engine, is designed to query ontological knowledge bases. Both the language syntax and internal Racer's knowledge representation are very similar to LISP language.

Several constructions are of our particular interest:

- Instances retrieval, e.g.

$$(retrieve\ (?x)\ (?x\ Woman))$$

which retrieves all women from the ontology.

- Instance type checking, e.g.

$$(retrieve\ (\ )\ (betty\ Woman))$$

which gives T if the instance *betty* is of type $Woman$ and *NIL* otherwise.

- Retrieval of pairs of instances, connected with a specified role, e.g.

$$(retrieve\ (?mother\ ?child)\ (?mother\ ?child\ hasChild))$$

which gives all that instance pairs that the second instance is a child of the first one.

Several constructions can be combined together forming the question we are interested to obtain the answer to.

In Figure 2 we can see relationships among upper-ontology elements.

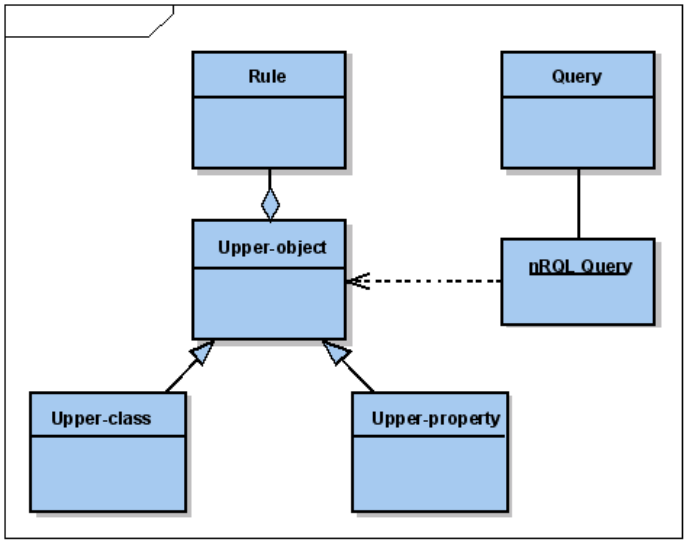

Fig. 2. Upper-ontology elements

In the center there are the abstract entity called the upper-object and its subclasses – the upper-class (a class in an upper-ontology) and the upper-property (a property in an upper-ontology). Each upper-object may have some rules (in the form of SWRL) of its creation from other upper-objects. Queries, implemented with nRQL, make use of the upper-objects as well.

3.1.3. Mapping and mapping ontology

A mapping ontology binds a domain ontology with an upper-ontology. It is created for the domain ontology to map, with equivalence or generalization/specialization, its concepts and roles with the ones from the upper-ontology. For domain ontologies and upper-ontologies are considered to be independent and self-contained, any change in their structures, like adding import statements, is forbidden. That is why mapping information need to be written in a separate ontology and that ontology can import the domain ontology and the upper-ontology. Then it can use any ontology elements from both of the imported ontologies.

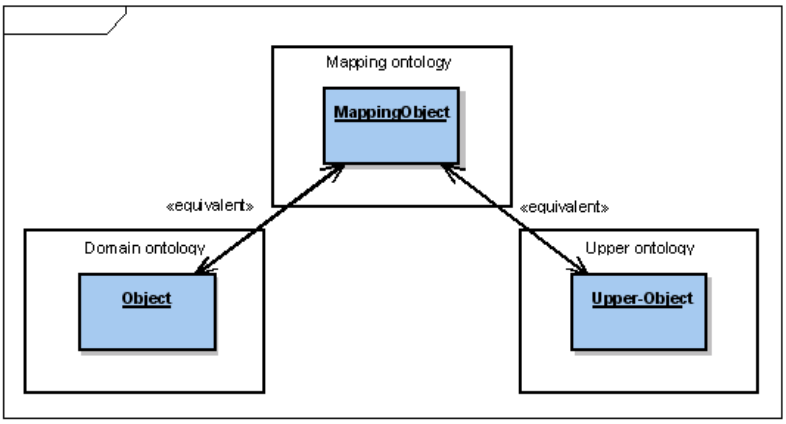

Fig. 3. Direct mapping structure

Figure 3 shows one of two kinds of possible mappings, namely a direct mapping, without additional mapping rule. To map an object from domain ontology with corresponding upper-object from an upper-ontology, a mapping object must be created in the mapping ontology. This object, a class or a property, has equivalence bindings to both domain ontology class or property and upper-ontology object. We can formalize it this way ...

$$D\#Class1 \equiv M\#ClassI \equiv U\#ClassA$$

where:

$Class1$ – a class in the domain ontology with namespace D,

$ClassI$ – a class in the mapping ontology with namespace M,

$ClassA$ – a class in the upper-ontology (with namespace U) we want to map $Class1$ to,

$\#$ – namespace/local name separator.

A formula for properties mapping would look very similar. A generalization/specialization relationship can also be applied. In that case one of the relations

would be generalization/specialization. When there is need for more sophisticated mappings, one can introduce a mapping rule, like in the following SWRL example ...

$$M\#ruleI : D\#role1(?x, ?y) \wedge D\#role2(?y, ?z) \rightarrow U\#roleA(?x, ?z)$$

where:

- $ruleI$ – a definition of a SWRL rule in the mapping ontology with namespace M,
- $role1(?x, ?y)$ – a binary role between instances in the domain ontology (with namespace D),
- $role2(?x, ?y)$ – a binary role between instances in the domain ontology (with namespace D),
- $roleA(?x, ?z)$ – a reference to the role in the upper-ontology with namespace U,
- $\#$ – namespace/local name separator.

This way the indirect relation between part of the knowledge from domain ontology and upper-ontology can be expressed.

3.1.4. Reasoning

In our application a reasoning process is performed by the Racer system [6]. A user chooses a mapping ontology and a set of domain questions. In fact the whole integrated ontology is read into the system while picking the mapping ontology because of proper import statements in it. Applying all rules to knowledge base only once can not be enough.

RacerPro does not apply SWRL rules for newly inferred facts during execution of other rules. New statements can be used in other rules, so we must force the reasoner to repeat the process by sending it the command $(reexexute - all - rules)$.

To be sure that all possible facts are inferred, we apply all rules x times, where x is the number of rules. For the re-execution of a rule does not apply to the facts that were previously used by that rule, the processing time does not lengthen so considerably.

3.2. General Architecture

In Figure 4 the general system architecture and data flow is presented.

A domain ontology and an upper-ontology are integrated by a mapping ontology. All is read by Racer which generates new facts about domain ontology while performing reasoning, for example by applying SWRL rules. A user chooses his question associated with the upper-ontology and sends the nRQL query to the reasoner to get the answer.

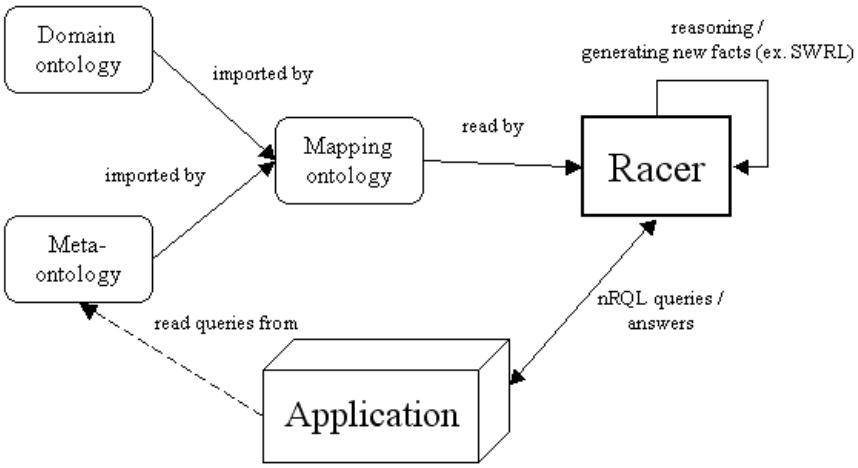

Fig. 4. General system architecture and data flow

4. Tests

To integrate ontologies it is necessary to create mappings between classes, properties and sometimes even between instances. It is natural that these connections can be made only when ontology domains are similar. As reasoning rules use both classes and properties, lack of mappings between properties of mapped classes make the class mapping useless in term of reasoning on instances of those mapped classes.

In our tests we want to show that integrating domain ontologies with proper upper-ontology leads to domain knowledge extension. Answers to questions associated with the upper-ontology can be obtained from domain knowledge. They could not be received basing only on domain ontology schema. Moreover, the main advantage of an upper-ontology will be shown in the "shared knowledge of the upper-ontology" section, namely that the same upper knowledge can be used in many domain ontologies, which goes well with the knowledge reuse principle. Thanks to the upper-ontology and questions concerning it, the same questions can be posted to many ontologies when these ontologies are supplied with correct mappings. Real existing ontologies are used in tests to avoid the risk of adapting of the ontologies for the purpose of methodologies described in the paper. Ontologies of finite state machine and Petri Net are the subject of our tests, but it is worth mentioning that the methods presented in the article do not depend on any particular domain, so they could be applied to the medical example from the introduction as well. What is important in the tests is the fact of extending the knowledge and inferring new information. We would like to get new facts about cycles from our test ontologies. The interpretation of a cycle depends on the chosen domain. If we talk about cycles in a road system of a city, the cycle would be a sequence of roads that allows us to drive in circles. A state

machine has a cycle when there is a possibility of repeatedly switching between the same sequence of states. Information about cycle, its general definition, will be a part of our upper-ontology. Referring to the medical example, it will be the same kind of information as new knowledge about lately discovered disease.

4.1. Finite State Machine ontology (FSM)

This ontology (`http://www.learninglab.de/~dolog/fsm/fsm.owl`) describes the structure of a finite state machine (Fig. 5) and contains a simple example.

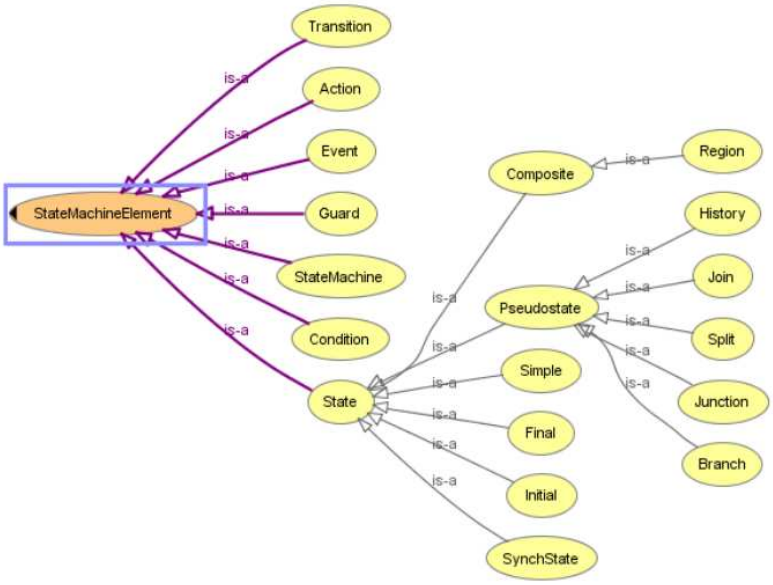

Fig. 5. FSM ontology

An ontology describing a structure of a simple graph was created (Fig. 6). It only contains concepts like a node, an edge and connection between starting and ending node.

Fig. 6. Upper-ontology elements for FSM

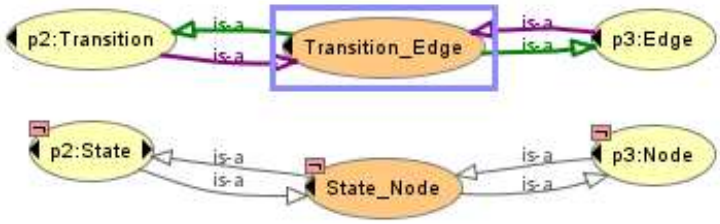

Fig. 7. Mappings for FSM

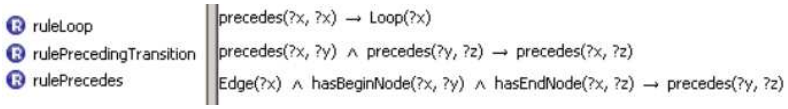

Fig. 8. Cycles detection rules

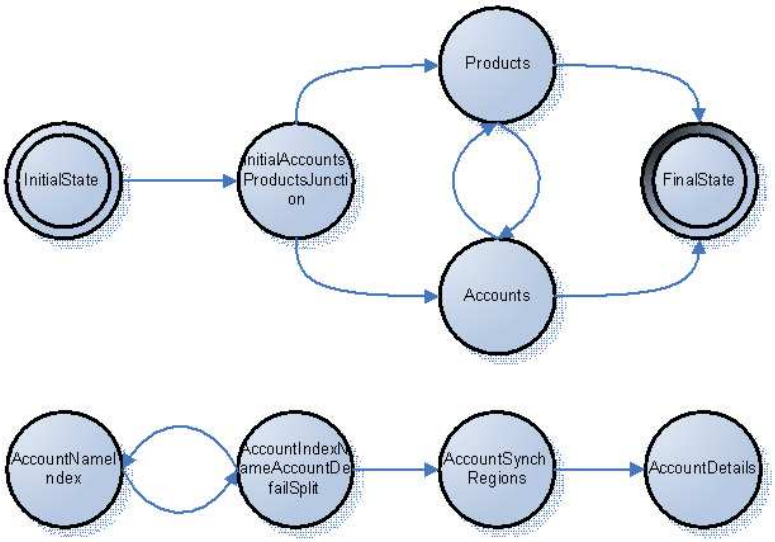

Fig. 9. An example of a state machine

Proper knowledge elements of the upper-ontology were mapped to domain ontology concepts and roles (Fig. 7).

Then, several rules detecting cycles in graphs were created in the upper-ontology basing on classes and properties of that ontology (Fig. 8).

After applying rules during reasoning, new information about cycles in the state machine from Figure 9 was gained.

Nodes Products, Accounts, AccountNameIndex and AccountIndexName-AccountDetailSplit were associated with the Loop class (Fig. 10) and we can see that each of them is indeed part of some loop.

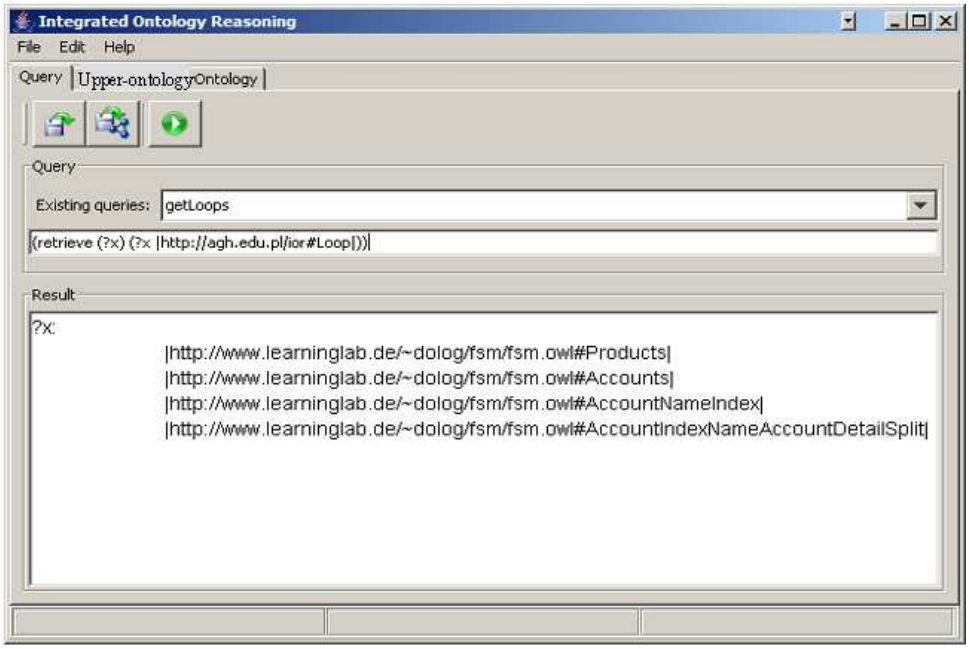

Fig. 10. Cycles in the FSM example

4.2. Petri Net ontology

This ontology (`http://www.aifb.uni-karlsruhe.de/Forschungsgruppen/BIK/wi2007/PNOntolgy.owl`) describes a Petri Net. It defines a Petri Net structure with places, transitions, arcs and markers (Fig. 11).

We created an upper-ontology for the Petri Net ontology. There was knowledge about graphs in the upper-ontology, but what is more important is the ability of a node to contain a marker now (Fig. 12).

Then, in the upper-ontology, some rules defining active transitions were created ...

- $ActivePlaceRule : hasMarking(?x, ?y) \wedge Place(?x) \wedge Mark(?y) \rightarrow ActivePlace(?x)$
- $ActiveTransitionRule : ActivePlace(?x) \wedge connectsNode(?x, ?y) \wedge Arc(?y) \wedge hasArc(?z, ?y) \wedge Transition(?z) \rightarrow ActiveTransition(?z)$

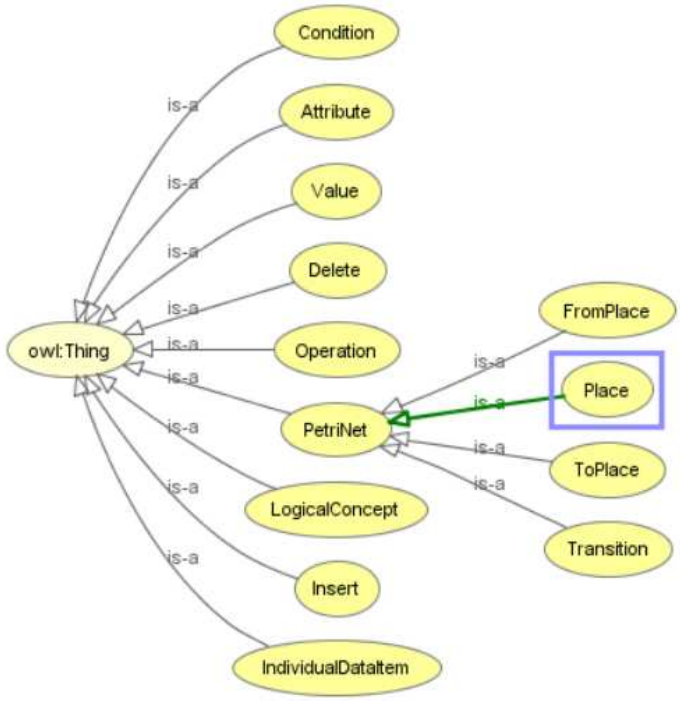

Fig. 11. The Petri Net ontology

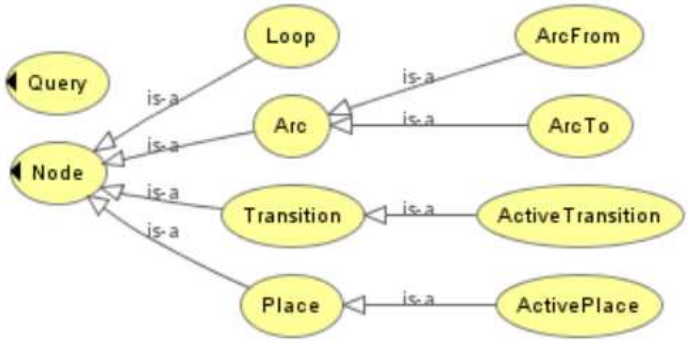

Fig. 12. The upper-ontology for the Petri Net ontology

After proper mapping creation (Fig. 13) for ontologies from Figures 11 and 12, we asked for active transitions in the example net shown in Figure 14. The system returned transitions T1, T2 and T4, which is correct.

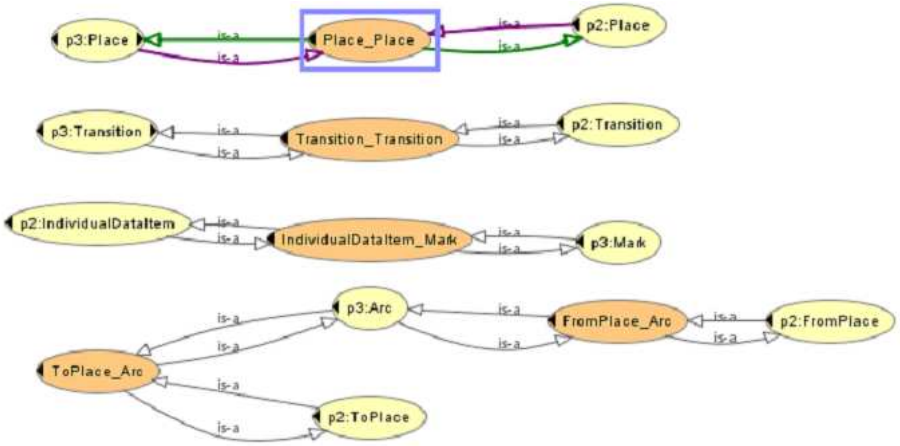

Fig. 13. The mapping ontology for the Petri Net ontology (classes only)

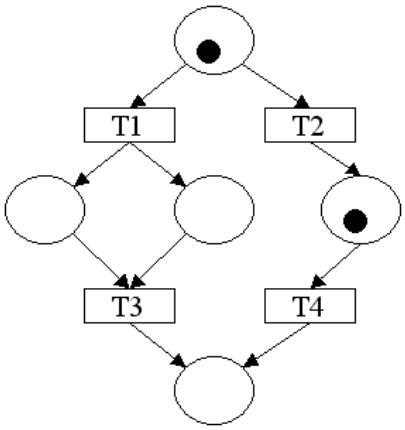

Fig. 14. An example of a Petri Net

To sum up, we gained new knowledge about active transitions from Petri Net ontology thanks to the upper-ontology and its rules. The Petri Net ontology did not have an active transition concept, so naturally asking that knowledge base to give us active transitions would be pointless.

4.3. Shared knowledge of an upper-ontology

The examples described above showed gaining of new information from domain ontology using upper-ontology rules. The situation where every domain ontology uses a separate upper knowledge would not be very interesting. That kind of upper-ontology

would be just a specific ontology extension with new schema or rules. This would not be real universal knowledge. So now we will use the same upper-ontology for two previously shown domain ontologies – FSM and Petri Net, because both of them can have the cycle feature introduced. The knowledge for describing the cycles will be of course in the upper-ontology (Figs. 6, 8). So we have two domain ontologies and one common upper-ontology. Now we need to specify mapping ontologies for both domain ontologies so that knowledge about cycles could be used in domain ontologies. A mapping ontology for FSM ontology does not change – it is shown in the Figure 7. As for Petri Net ontology, the following mappings need to be created ...

- $MappingOntology\#ArcTo \equiv PetriNet\#ToPlace$
- $MappingOntology\#ArcFrom \equiv PetriNet\#FromPlace$
- $MappingOntology\#ruleBeginNode : PetriNet\#Transition(?x) \wedge PetriNet\#hasArc(?x, ?y) \wedge MappingOntology\#ArcFrom(?y) \wedge PetriNet\#connectsNode(?y, ?z) \rightarrow UpperOntology\#hasBeginNode(?x, ?z)$
- $MappingOntology\#ruleEndNode : PetriNet\#Transition(?x) \wedge PetriNet\#hasArc(?x, ?y) \wedge MappingOntology\#ArcTo(?y) \wedge PetriNet\#connectsNode(?y, ?z) \rightarrow UpperOntology\#hasEndNode(?x, ?z)$
- $PetriNet\#Transition \equiv MappingOntology\#Transition \equiv UpperOntology\#Edge$

The first two mappings are just aliases for two Petri Net classes to be used in the following rules. Third and forth mappings are mapping rules. They describe a begin node (the starting point for some edge) and an end node (the ending point for some edge) respectively. The last mapping says that a transition concept from the Petri Net ontology should be treated like an edge concept from the upper-ontology with graph and cycle knowledge.

5. Conclusions

The paper presents a multi-layered model of integrated ontologies for the use of reasoning and knowledge management methodology, from integration to using. Each layer has different, specific functions. A domain ontology is the description of some reality and it is the source of instances on which we want to reason. An upper-ontology contains more general knowledge, usually from the same domain. A set of question templates is associated with that ontology to which the answer comes from integrated domain ontology. There is also a middle layer of a mapping ontology, which integrates both ontology elements. The mappings may be direct, when there are corresponding elements in both ontologies, or indirect. The latter case is implemented by additional mapping of SWRL rules. The upper-ontology can and should be used in many different ontologies so that we can obtain answers to the same questions from different domain ontologies. That kind of well organized model supports knowledge reuse.

The tests show that it is possible to gain new information from existing knowledge by integrating it with an upper-ontology. Our system helps in knowledge management

during the integration phase and can be used to query integrated knowledge. It supports reasoning so it can return inferred facts as well. The application has a couple of limitations because of some functionality features of Racer. The current version of the reasoner does not support Datatype Properties and transitive properties. If an ontology contains one of them, Racer will not be able to perform reasoning on that ontology.

Acknowledgements

We would like to thank Tomasz Liptak and Szymon Natanek for their help in writing this paper and system implementation.

References

[1] Baader F. *et al.*: *The Description Logic Handbook*. Cambridge University Press, 2003

[2] Euzenat J., Le Bach T., Barrasa J., Bouquet P., De Bo J., Dieng R., Ehrig M., Hauswirth M., Jarrar M., Lara R., Maynard D., Napoli A., Stamou G., Stuckenschmidt H., Shvaiko P., Tessaris S., Van Acker S., Zaihrayeu I.: *State of the art on ontology alignment*. Knowledge Web Deliverable, Technical Report, INRIA, 2004

[3] Fridman N., Musen M.: *SMART: Automated Support for Ontology Merging and Alignment*. Twelth Workshop on Knowledge Acquisition, Modeling and Management, Banff, Canada, 1999

[4] Gruber T.: *What is an Ontology*. `http://www-ksl.stanford.edu/kst/what-is-an-ontology.html`

[5] Haase P., Motik B.: *A Mapping System for the Integration of OWL-DL Ontologies*. IHIS'05, Bremen, November 2005

[6] Haarslev V., Moller R.: *RACER User's Guide and Reference Manual*. 2004

[7] Horridge M., Knublauch H. et al.: *A Practical Guide To Building Ontologies Using The Protege-OWL Plugin and CO-ODE Tools*, 1st ed., University of Manchester, 2004

[8] Horrocks I., van Harmelen F., Patel-Schneider P. et al.: *DAML+OIL*. `http://www.daml.org/2001/03/daml+oil-index.html`, 2001

[9] Hunter J.: *The Semantic Web*. 2002

[10] Kalfoglou Y., Schorlemmer M.: *Ontology Mapping: The State of the Art*. The Knowledge Engineering Review, Vol. 18:1, 2003, 1–31

[11] Knublauch H.: *The Protégé-OWL API – Programmer's Guide*. Stanford Medical Informatics, 2005

[12] Lee T. B., Hendler J., Lassila O.: *Semantic Web*. 2001

[13] *LISP*. `http://www.lisp.org/alu/home`

[14] Łuszpaj A., Szymański K., Zygmunt A., Koźlak J.: *The Process of Integrating Ontologies for Knowledge Base Systems*. 7th Software Engineering Conference, Cracow 2005

[15] McGuinness D. L., van Harmelen F.: *OWL Web Ontology Language Overview*. http://www.w3.org/TR/owl-features/, W3C Recommendation, 2004

[16] Namyoun C., Il-Yeol S., Hyoil H.: *A Survey on Ontology Mapping*. SIGMOD Record, Vol. 35, No. 3, Sep. 2006

[17] Niles I., Pease A.: *Towards a Standard Upper Ontology*. In Proceedings of the 2nd International Conference on Formal Ontology in Information Systems. FOIS-2001

[18] Pazienza M. T., Stellato A. et al.: *Ontology Mapping to support ontology-based question answering*. 4th International Semantic Web Conference (ISWC-2005) Galway, Ireland, November, 2005

[19] Pinto H. S., Martins J. P.: *Ontology Integration – How to perform the process*. Portugal 2001

[20] Pinto H. S., Martins J. P.: *Some Issues on Ontology Integration*. Portugal 2001

[21] *Racer Systems*. http://www.racer-systems.com/

[22] *Resource Description Framework (RDF)*. http://www.w3.org/RDF

[23] *RICE*. http://www.ronaldcornet.nl/rice/

[24] *Standard Upper Ontology Working Group (SUO WG)*. http://suo.ieee.org/index.html

[25] Standard Upper Ontology Working Group (SUO WG) *Suggested Upper Merged Ontology*. http://suo.ieee.org/SUO/SUMO/index.html

[26] *SUMO Ontology*. http://ontology.teknowledge.com/

[27] Tamma V.: *An Ontology Model Supporting Multiple Ontologies for Knowledge Sharing*. Thesis of University of Liverpool, 2001

[28] *The New Racer Query Language*. http://www.cs.concordia.ca/~haarslev/racer/racer-queries.pdf

[29] W3C: *SWRL – A Semantic Web Rule Language*. 2004

[30] *Welcome to the Protégé Project*. http://protege.stanford.edu/