An Analysis of Contextual Aspects of Conceptualization: A Case Study and Prospects

Krzysztof Goczyła, Aleksander Waloszek and Wojciech Waloszek

Abstract In this chapter we present a new approach to development of modularized knowledge bases. We argue that modularization should start from the very beginning of modeling, i.e. from the conceptualization stage. To make this feasible, we propose to exploit a context-oriented, semantic approach to modularization. This approach is based on the Structural Interpretation Model (SIM) presented earlier elsewhere. In the first part of thischapter we present a contextualized version of the SYNAT ontology developed using the SIM methodology. For the approach to be useful in practice, a set of tools is needed to enable a knowledge engineer to create, edit, store and perform reasoning over contextualized ontologies in a flexible and natural way. During our work on the SYNAT project, we developed such a set of tools that are based on a mathematical ground of tarset algebra (also introduced elsewhere). In the second part of this chapter we give a deeper insight into some aspects of using these tools, as well as into ideas underlying their construction. The work on contextualization of knowledge bases led us to further theoretical investigation of hierarchical structure of a knowledge base systems. Indeed, in a system of heterogeneous knowledge sources, each source (a knowledge base) can be seen in its own separate context, as being a part of a higher level contextual structure (a metastructure) with its own set of context parameters. Higher levels of the knowledge hierarchy do not substantially differ

This work was partially supported by the Polish National Centre for Research and Development (NCBiR) under Grant No. SP/I/1/77065/10 within the strategic scientific research and experimental development program: "SYNAT-Interdisciplinary System for Interactive Scientific and Scientific-Technical Information".

K. Goczyła (✉) · A. Waloszek (✉) · W. Waloszek (✉)
Gdańsk University of Technology, Gdańsk, Poland
e-mail: kris@eti.pg.gda.pl

A. Waloszek
e-mail: alwal@eti.pg.gda.pl

W. Waloszek
e-mail: wowal@eti.pg.gda.pl

R. Bembenik et al. (eds.), *Intelligent Tools for Building a Scientific Information Platform: From Research to Implementation*, Studies in Computational Intelligence 541,
DOI: 10.1007/978-3-319-04714-0_6,

from basic SIM structure, so that all context-aware services, including reasoning, can be performed. The theoretical background of this conception is presented in the third part of this chapter.

Keywords Knowledge base · Ontology · Modularization · Tarset · Contextualization

1 Introduction

1.1 Foreword

In this chapter we described the first systematic practical case study for deployment of our original methods of knowledge base modularization (the methods are described in details in [1]; they are tarset approach [2] and SIM [3]—Structured Interpretation Model—knowledge base organization; a short introduction to these methods is presented in Sects. 1.2 and 2.4). This case study allowed us to illustrate these methods and resulted in abundance of conclusions and recommendations for further development.

In the first part of this chapter (first part of Sect. 2) we report our work on contextualizing a fragment of SYNAT ontology, taking into account all aspects differentiating the process of conceptualization characteristic for contextual knowledge bases from the process for non-contextual ones. The main aspect we put stress on was the manner of implementing social roles and activities. To analyze the issue systematically, we distinguish the role- and concept-centric approaches, the former more often used during non-contextual, and the latter during contextual conceptualization. We illustrate the role-centric approach showing knowledge bases founded on the c.DnS ontology.

In the main part of Sect. 2 we describe our work on contextualizing a fragment of SYNAT ontology. This case study should be perceived as an experiment, whose goal was to gather as much experience as possible in order both to verify and in future develop modularization methods proposed by our group. Its results show that the proposed methods allow a knowledge engineer to build structures that significantly ease formulating queries and inserting data. These results are also very beneficial for our further research, because they let us to draw some important conclusions. Some of them we already managed to bring into life, e.g. optimization and improvement of the algorithm of reasoning. The others let us to formulate new ideas, for example, a proposal of controlled incomplete reasoning (see Sect. 3).

In our opinion, the most valuable result is the idea of a truly universal framework for all contextual approaches, where contexts would be substantial elements of designed models from the very commencement of the conceptualization stage. This framework would enable one to design knowledge bases in many levels with use of a recurrent mechanism embracing every layer of

knowledge representation structure. The last part of this chapter (Sect. 4) introduces the idea and provides evidence that it is feasible due to properties of our previous proposition.

1.2 Preliminaries

In our works in SYNAT project we concentrated on the *tarset theory of knowledge bases* and the *SIM* model. Both of them were applied in the practical tool called *CongloS* (see http://conglos.org) In this section we shortly describe these results. For more details we refer the Reader to our previous publications [1–3].

Tarset knowledge bases consist of modules called *tarsets* (originally named *s-modules*, see [2]). Every tarset T is a pair $(\mathbf{S}, \mathbf{W})$ where $\mathbf{S}$ is a *signature* (or a *vocabulary*, i.e. a set of names), and $\mathbf{W}$ is a set of Tarski style interpretations of the vocabulary $\mathbf{S}$.

To formally define a tarset knowledge base we introduce a *tarset variable* or *a module* (the notion similar to a *relation variable* in relational databases). Tarset variables are basically labels. We assign each tarset variable a value, i.e. a pair $(\mathbf{S}, \mathbf{W})$, this value may be perceived as a single point from the space of all possible tarsets $\mathbf{T}$.

In the space of tarset we define a set of operations, which together constitute tarset algebra (see [1, 2] for more details). Tarset algebra is a variant of a cylindric algebra, one of the simplest operations is intersection, defined as $(\mathbf{S}, \mathbf{W}_1) \cap (\mathbf{S}, \mathbf{W}_2) = (\mathbf{S}, \mathbf{W}_1 \cap \mathbf{W}_2)$; since this operation reduces the set of possible models, effectively all the conclusions from both the tarsets being intersected can be drawn from the intersection.

We can constrain values of tarset variables using a *coupler*. A coupler syntactically is an *inequality* of two tarset algebra expressions. The *arity* of a coupler is the number of tarset variables contained in both expressions of the inequality. For example, a coupler containing three tarset variables T_1, T_2 and T_3 (a *ternary coupler*) is the inequality: $T_1 \leqslant T_2 \cap T_3$ (this coupler means that all conclusions inferred from the tarsets T_2 and T_3 hold in the tarset T_1). We define an *instance of a tarset knowledge base* as a set of tarset variables (together with their assignments) and a set of couplers containing these variables.

Tarset variables and couplers are sufficient to express semantic properties of modular bases created according to existing modularization techniques, as for example DDL [4], but they still lack very vital information concerning admissible changes of the structure of a knowledge base. This gap is filled by the notion of a *schema of a tarset knowledge base*. A schema consists of definitions of *tarset types* and *coupler types*. To satisfy a schema, an instance of a knowledge base has to contain tarset variables and couplers such that every tarset variable has to be assigned a tarset type defined in the schema, and every coupler has to be assigned a coupler type defined in the schema.

Due to the notion of schema, tarset model gains flexibility which allows for describing within it also other models of modular knowledge bases. In our work we performed this task for SIM contextual knowledge bases [1], expressing its types of modules (context types and context instances) as tarset types and its three types of relationships between modules (instantiation, aggregation, and inheritance) as coupler types. (More information on SIM model is presented in Sect. 2.).

The CongloS system is a tool allowing a user to create tarset knowledge bases accordingly to the SIM method. Technically it is a set of plug-ins for Protégé—the most popular editor of ontologies in the OWL standard (http://protege.stanford.edu/). These plug-ins enrich the interface of the editor by adding elements necessary from the point of view of the SIM method and provide the system with mechanism for management of the structure of the knowledge base being edited. One of the most important plug-ins is an inference engine capable of reasoning from contextual knowledge bases.

While focused on Protégé, CongloS is a flexible tool that may be used in different configurations and consists of subsystems suitable also for handling general tarset knowledge bases (and for reasoning from them). CongloS can be adjusted to different target configurations by replacing specially distinguished adaptation layer.

2 Contextualizing SYNAT Ontology

2.1 SYNAT Ontology as a Case Study for Contextualization

Originally, our choice of SYNAT ontology (see [5]) as an object of the case study was simply justified by our participation in this project. However, SYNAT ontology turned out to have some features that make it a remarkable experimental material for such research. Firstly, it is an ontology with a very broad domain of interest which embraces different but strictly related problem sub-domains. Examples of the sub-domains are various aspects of organization of institutions, their structure, activities, effects of activities, events, work of people, their career paths, geographical locations, etc. Diversity of the domains alone suggests the possibility of distinguishing some subsystems, which, while being autonomous, are connected with others by various relationships.

Secondly, SYNAT ontology bears many characteristic features of constructive stance (discussed in Sect. 2.2, introduced in [6]), that is to say some specific set of solutions directed towards multi-aspectual description of diverse phenomena within a non-contextual ontology. Such a non-contextual ontology must simultaneously display attributes of high-level ontology and of operational knowledge base, so it must be able to be expanded by facts in order to make reasoning over them possible (according to Gruber [7], the former should be characterized by the weakest possible ontology commitment, while the latter should strengthen the

commitment, making it possible to particularize meaning of terms to one selected context). As a result predicates of high arity are used there to precisely capture all the contextual information needed.

The main purpose of the SYNAT ontology is to gather information about objects and events concerning academic and scientific society. It contains 472 classes and 296 properties.

The ontology is internally divided into five parts determined by five generic concepts (see Fig. 1):

1. *Agent*—the concept being an ancestor of human agents, organizations and groups.
2. *InformationResource*—the extension of this concept contains different carriers of information, mainly physical like documents, or electronic like audio and video resources.
3. *Event*—the concept represents all kinds of events associated with scientific and academic activity.
4. *Project*—concerns scientific projects.
5. *Characteristic*—this concept describes all kinds of abstract entities reifying properties of instances of all other concepts.

These five concepts are roots of separated trees of taxonomies. First four concepts embrace entities existing in the reality. The last one is the essence of the ontology, typical for the constructive approach. It is the root for the richest tree of classes defining reified properties of different kinds and arities for all the other classes of individuals. The best example is characteristic of information resources gathered under the class *InformationResourceCharacteristic*. While the class *Information-Resource* describes only physical carriers, its characteristic contains information about the content. First, there is a group of concepts describing content types, e.g. Catalogue, Guide, Encyclopedia, Norm, Patent, News, Report, etc. Then there are concepts for periodical resources, for describing structure of documents, for access conditions, or a very rich group of notions concerning web resources.

Another example is characteristic of persons. The main concept for this branch of conceptualization is *PersonCharacteristic*. It contains basic personal data, like names and addresses, as well as all data connected with the career in science and education. Of course, the organization of the data relies on reification of relations (see Fig. 2).

The ontology is divided into four modules, however this division is not realized as modularization in its common sense, i.e. as a way allowing to reason from smaller parts of knowledge. The purpose of the division is to separate information taken from two external ontologies, GIO (Geographic Information Objects, see [8]) and SCPL (Science Classification elaborated by the Polish government), and to adapt them to the needs of designed structure of concepts. This design decision allows a user to import the original ontologies without harming existing classification.

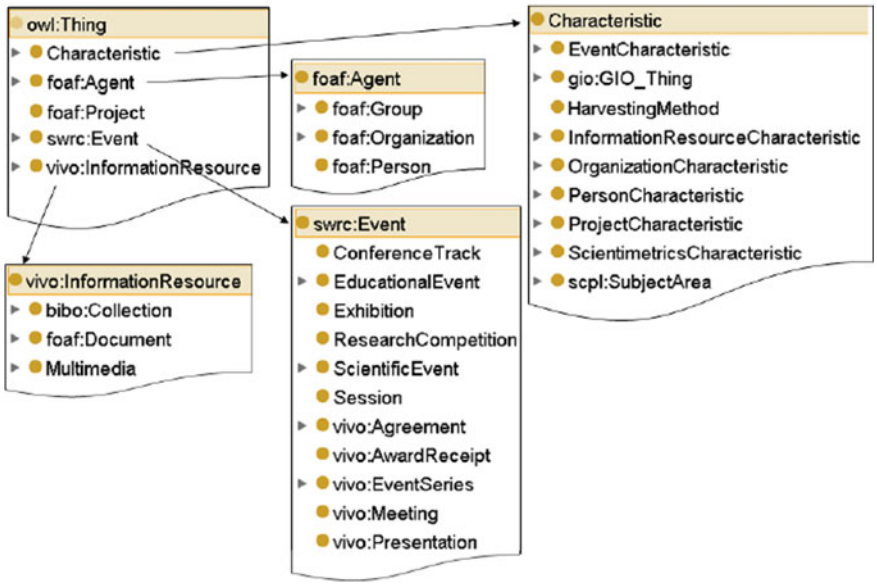

Fig. 1 The main branches of taxonomy in the SYNAT ontology (from [5])

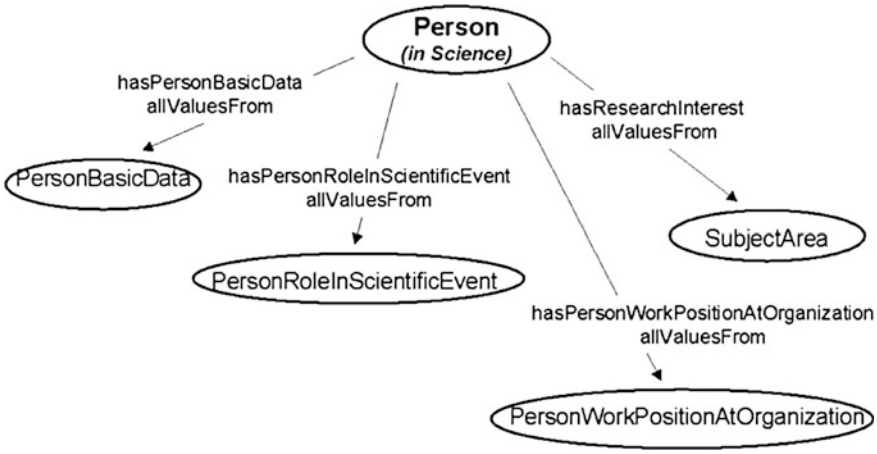

Fig. 2 Reified *n*-ary properties describing persons (from [5])

The SYNAT ontology is also ready to reuse some other globally known ontologies (available under "free" licenses, like CC and BSD), like for example FOAF [9], VIVO [10], SWRC [11] or BIBO [12]. This is realized by relating local terms to the terms defined by those ontologies with axioms (the ontologies are imported by the SYNAT ontology).

2.2 c.DnS as a Classical Example of Constructive Approach

Inclusion of contextual information within description of facts (added to an ontology in order to enable factual reasoning) is connected with constructive approach and is mainly due to the requirement of monotonicity of an ontology. This requirement prevents introduction to an ontology sentences that are in contradiction with its current contents. For example it is infeasible to state that "John Doe is a rector" and then "John Doe is not a rector" to model a situation when John Doe ends his term of office and is not elected a rector for the next period. This is the reason why an operational knowledge base cannot constrain the meaning of the concept *Rector* to "current rector".

Non-contextual knowledge bases must therefore conform to the requirement of monotonicity, but without constraining the meaning of the used terms. Consequently, descriptions of various phenomena within such bases have to contain large amounts of additional contextual information. For instance, the fact of employment of a person in some institution has to be expressed as a predicate of very high arity containing information about the time of the employment, work post, position etc.

c.DnS ontology is an extreme example of constructive approach. It was described in [6]. The abbreviation means Constructive Descriptions and Situations and it reflects the intension of its authors to describe situations in a constructive way. According to a philosophical stance called constructivism, reality is described as a mental structure depending on a context and for a specific reason with use of properly chosen system of notions. The authors, recalling this stance, propose a system containing very few classes describing the most general objects of a domain of interest (e.g. *Entity*, *Social Agent*, *Situation*, *Collection*, *Information Object*, *Time*), and also some meta-classes (e.g. *Description*, *Concept*) for objects that represent elements of a description of a domain, not a domain itself.

The meaning of the main classes is as follows:

1. *Description*—individuals of this class represent a kind of conceptualization, e.g. laws (government regulation between others), plans, projects, etc.; they are mainly related to individuals representing the classes *Concept* and *Entity*.
2. *Situation*—individuals represent things which have to be described, e.g. facts, legal cases, actions (realizing plans or projects), etc.; the relations connect this individuals with instances of *Description*, other situations (as sub- or super-situations) and entities.
3. *Concept*—individuals represent concepts defined by a description; concepts are related to collections and entities.
4. *Entity*—the extension represents anything what exists. This concept is a superclass of all others. The individuals are divided into two groups: schematic and non-schematic. They are related to instances of *Collection*, *Information Object* and other entities.

5. *Social Agent*—the only role played by an individual being an instance of this class is to describe situations or share it with other agents.
6. *Collection*—is the super-class for groups, teams, collectives, associations, collections, etc. A special case is a collective of agents sharing one description of a given situation.
7. *Information Object*—an information object may express a description and be about an entity.
8. *Time*—individuals represent time intervals and are properties of instances of the *Description* and *Situation* classes.

Every knowledge base is understood as a *c.DnS* relation, i.e. a set of tuples, each of which contains eight elements:

$$c.DnS(d, s, c^*, e^*, a^*, k^*, i^*, t^*) \rightarrow D(d) \wedge S(s) \wedge C(c^*) \wedge E(e^*) \wedge A(a^*) \wedge K(k^*) \wedge I(i^*) \wedge T(t^*)$$

The asterisk means that a given variable is an ordered list of values of this same type.

The meaning of a tuple is as follow: a social agent a ($A = Social\ Agent$) as a member of knowledge communities k^* ($K = Collection$) perceives a situation s ($S = Situation$) and describes it using a description d ($D = Description$) and assigning entities e^* ($E = Entity$) with concepts c^* ($C = Concept$). Information objects i^* ($I = Information\ Object$) are supposed to express the description d. Time intervals t^* ($T = Time$) play two roles: first, they are temporal attributes for s, informing when the situation occurred; secondly they should describe a time interval when the description was carried out.

An example of a c.DnS tuple is:

$$\begin{aligned} c.DnS(&KnowledgeOfPreviousCases\#1,\ KillingSituation\#1,\\ &\{Precedent,\ Killer,\ Tool,\ HypotheticalIntention\},\\ &\{Event\#1,\ PhysicalAgent\#1,\ Tool\#1,\ Plan\#1\},\\ &Detective\#1,\ InvestigationTeam\#1,\ CriminalCode\#1,\\ &\{TimeOfEvent\#1,\ TimeOfInterpretation\#1\}) \end{aligned}$$

The tuples may be *projected* on relations of lower arity, the main projections are depicted in Fig. 3. The bold gray arrows in the figure represent ternary relations that include time as an additional parameter. The dashed arrows represent derived relations (e.g. a description unifies a collection iff the description defines a concept which covers the collection).

The real power of cDnS ontology is *redescription*. Redescription allows us to describe this same situation from another points of view. For example, while the above tuple describes a situation of intentional murder, during an investigation the interpretation of the same event may be changed to self-defense:

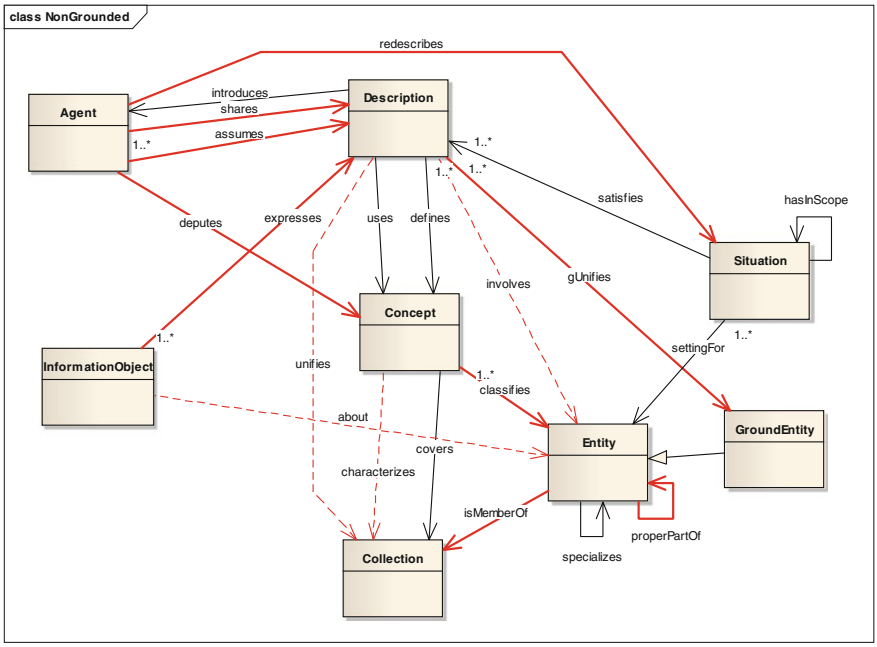

Fig. 3 The basic concepts and roles from c.DnS ontology in the form of UML diagram

$$
\begin{aligned}
&c.DnS(KnowledgeOfPreviousCases\#1,\ KillingSituation\#1,\\
&\quad \{Precedent,\ Killer,\ Tool,\ HypotheticalIntention\},\\
&\quad \{Event\#1,\ PhysicalAgent\#2,\ Tool\#1,\ SelfDefence\#1\},\\
&\quad Detective\#2,\ InvestigationTeam\#2,\ CriminalCode\#1,\\
&\quad \{TimeOfEvent\#1,\ TimeOfInterpretation\#2\})
\end{aligned}
$$

2.3 Role-Centric Versus Concept-Centric Approach

The presented extreme case of the constructive approach enables us to characterize it more precisely. For purpose of our argumentation we call it a *role-centric approach*. It is justified by the fact that the main property of this approach is cessation of designing hierarchic taxonomies of classes towards graph solutions. For example, if we want to model the property of *being a rector*, we introduce (in the simplest case) a binary predicate relating a given person to an abstract object called for example *rector's post* instead of declaring this person as an instance of a concept (e.g. *Rector*).

Assigning an individual to an extension of a concept is in accordance with specific character of DL languages. The grammar of these languages bases on a kind of descriptions, what therefore enables to formulate sentences similar to natural language utterances. Let such an approach be called a *concept-centric approach*.

Despite of the mentioned above arguments, the concept-centric approach is rarely used by knowledge engineers. The reason is, already recalled above, the monotonicity principle. This principle eliminates ability of modeling a situation when an object ceases to be an instance of a given class, for example, when someone ceases to act as rector. In other words, the concept *Rector* has too strong ontological commitment, because *being a rector* means someone who is taking a post of a rector in a certain place and time (i.e. for given values of context parameters).

The role-centric approach seems to have big expressive power but it reduces the power of DL understood as ability of terminological reasoning with the open world assumption (OWA). In case of the role-centric approach this power is being replaced by the power of reasoning similar to the one used in logical programming based on Horn rules—reasoning from facts based on the close world assumption (CWA). Such a kind of reasoning has *extensional character* rather than *intensional* one. In [13] Guarino has explained the difference between the extensional and intensional character of reasoning.

The extensional reasoning has its origin in the definition of conceptualization given in [14]. By Genesereth and Nilsson conceptualization is defined as a structure (Δ, R) where Δ is a domain and R is a set of relations on Δ. Guarino criticized this definition arguing that it is extensional while the process of conceptualization has intensional character. Calling this definition extensional Guarino meant the mathematical understanding of the set of relation R. Every element of this set is an n-ary relation on Δ, i.e. a subset of Δ^n. During conceptualization a human mind takes into account not only currently existing relationships, but also all possible relationships in all possible worlds.

Such kind of relations Guarino defines as *conceptual relations*. To consider this fact we should first define a *domain space* as a pair (Δ, W) where W is a set of all possible worlds. Formally, a conceptual relation ρ of arity n is a function $W \rightarrow \mathcal{P}(\Delta^n)$, from all possible worlds to the powerset of Δ^n. The image $\rho(W)$ is, by Guarino, the set of all *admittable extensions* of ρ, i.e. all possible (mathematical) relations of one kind in all possible worlds.

Finally, conceptualization is a triple $\mathbf{C} = (\Delta, W, \mathcal{R})$ where $\mathcal{R}$ is a set of conceptual relations defined in a domain space (Δ, W). A conceptualization is, according to this definition, much richer phenomenon because every ρ from the set $\mathcal{R}$ assigns to it a lot of its admittable extensions (mathematically defined relations).

The concept-centric approach gives us much better ability to implement intensional reasoning in the sense given by Guarino. Describing someone's

profession by a concept, i.e. unary predicate, allows us to utilize all advantages of DL languages in describing relations between extensions of given concepts. For example, it is easy to say that someone who is a *Rector* has always to be a *Professor* using a construct stating subsumption between them. This statement restricts the set of possible interpretations to only those corresponding to the set of possible worlds. Formulating statements uttering similar meanings in role-centric approach is much more difficult.

Entering data is also much more complicated. As an example let us say, according to the SYNAT ontology, that an individual *johnDoe* is a Rector. To do it we have to state the following statements:

$$Person(johnDoe)$$
$$PersonWorkPositionAtOrganization(personPositionAtOrganization\_JD)$$
$$Rector(rector\_JD)$$
$$hasPersonPositionAtOrganization(johnDoe, personPositionAtOrganization\_JD)$$
$$hasRoleAtOrganization(personPositionAtOrganization\_JD, rector\_JD)$$

All these sentences replace one simple statement *Rector*(*johnDoe*) used in the concept-centric approach. In consequence, the role-centric approach produces a lot of additional individuals, that have no equivalent in the reality, and a lot of sentences relating these individuals to each other.

Sometimes ontology designers try to reduce the aforesaid faults. The good example is the ontology of fraudulent disbursement described in [15, 16]. Its architecture is depicted in Fig. 4. This is a modular ontology, and the only mapping used to connect modules is the OWL import statement. In the modular structure the cDnS ontology is used as a top level ontology.

The core layer of the structure is formed by four modules: $\text{c.DnS}_{\text{Crime}}$ contains description of crime situations, $\text{c.DnS}_{\text{Inquiry}}$ contains information about inquiries managed by public prosecutors, $\text{c.DnS}_{\text{Investigation}}$ describes investigations conducted by the police. The last module, $\text{c.DnS}_{\text{Workflow}}$, contains description of situations that cannot be classified to any of the first three modules. The lowest layer contains modules specializing $\text{c.DnS}_{\text{Crime}}$.

Of course, every information inserted into the ontology makes a proper c.DnS tuple. However, the authors created a parallel taxonomic structure of domain concepts inheriting from the c.DnS concept *Entity*. A set of specifically defined axioms make a reasoner to infer extensions of these concepts basing of binary relations they are involved in. As a result we can easier find a given individual and its properties. This parallel structure mitigates some of the drawbacks of the role-centric approach, however at the cost of making the ontology somewhat more complex. In addition, modularization of the ontology divides information into smaller portions. This division helps to keep better control over quickly growing amount of data and to avoid reasoning from the entire ontology.

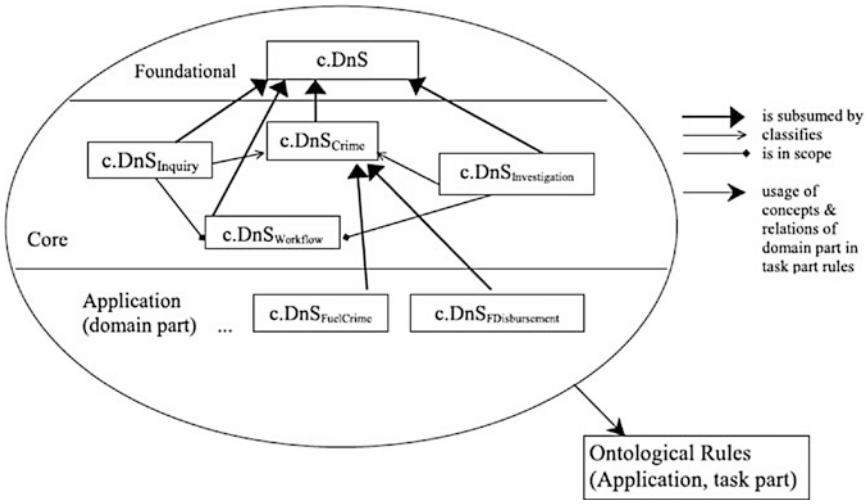

Fig. 4 The architecture of the fraudulent disbursement ontology (from [16])

2.4 SIM Model as a Contextual Approach

SIM model gives hope for more extensive use of concept-centric approach without losing the operability of a knowledge base. It is possible due to the fact that a SIM base is divided into modules-contexts, and every context is characterized by its own set of contextual parameters (not necessarily made explicit). As a result, the meaning of each concept can be particularized without restriction simply by using such a term in a proper place within knowledge base structure.

Since SIM model has been thoroughly described in other publications (e.g. [1, 3]), here we only very briefly review the basics of the method.

According to the SIM method TBox and ABox of an ontology constitute two structures organized in different manners. Division of TBox is based on a relation very similar to OWL *import*. Contextualized TBox $T = T(\{T_i\}_{i\in I}, \trianglelefteq)$ is a set of *context types* $\{T_i\}$ (partial TBoxes) connected with *inheritance* relation $\trianglelefteq$ established on a set of indexes I. $\trianglelefteq$ is a poset with a minimal element m, T_m being a *top context type*.

A contextualized TBox cannot have a model by itself. What determines the structure of the interpretation is a division of ABox. Contextualized ABox is defined as $A = (\{A_j\}_{j\in J}, inst, \ll)$ and consists of *context instances* $\{A_j\}$ (partial ABoxes), function $inst$: $J \to I$ (called *instantiation function*) assigning each partial ABox to its partial TBox and the *aggregation relation* $\ll$ established on a set of indexes J; $\ll$ is a poset with a minimal element n, A_n being a *top context instance*, $inst(n) = m$. Each partial ABox has its local interpretation defined as $\mathcal{I}_j = (\Delta^{\mathcal{I}j}, \bullet^{\mathcal{I}j})$, $\bullet^{\mathcal{I}j}$ assigns elements of the domain to concepts and roles defined in $T_{i:\ inst(j)\ =\ i}$ and its ancestors. Contextualized interpretation $\mathcal{I} = (\{\mathcal{I}_j\}_{j\in J}, \ll)$ is a

set of local interpretations connected with the relation $\ll$ (the same as in the case of ABoxes). The flow of conclusions is assured by aggregation conformance rules obligatory for all models:

(1) $\bigcup_{k\in\{k:j\ll k\}} \Delta^{\mathcal{I}k} \subseteq \Delta^{\mathcal{I}j}$,
(2) $\bigcup_{k\in\{k:j\ll k\}} C^{\mathcal{I}k} \subseteq C^{\mathcal{I}k}$,
(3) $\bigcup_{k\in\{k:j\ll k\}} R^{\mathcal{I}k} \subseteq R^{\mathcal{I}k}$,
(4) $a^{\mathcal{I}k} = a^{\mathcal{I}j}$ for $j \ll k$.

Generally, the nodes of a SIM knowledge base structure may be perceived as tarsets, and the relationships are couplers. The transformation is straightforward. There are only two tarset types predefined: the first one for context types and the second one for context instances. And, correspondingly, there are three coupler types: the first one for expressing inheritance, the second one for instantiation, and the third one for aggregation relationships.

2.5 Contextualization Process

During our work we conducted a case study: contextualization, in the form of SIM knowledge base of a fragment of the domain of interest covered by SYNAT ontology. Within the case study we set ourselves a goal to exploit concept-centric approach and to investigate whether its use allows for preserving operability of a knowledge base.

As the fragment of the domain we picked subjects covered by the factual part (ABox) of SYNAT ontology. These subjects embrace publishing of articles, their editors, and employment of persons in institutions.

As an indicator of operability we chose conformance to the two sets of requirements:

1. Ability to answer the set of 21 competency queries, examples of which are presented below:
   (a) Which articles were written by person x?
   (b) By which institutions person x was employed?
   (c) Which persons were co-authors of the articles written by person x?
2. Ability to perform the set of 18 update operations to the base, examples of which are presented below:
   (a) addition of a new person,
   (b) addition of a new working post for a person,
   (c) addition of a new publisher.

As mentioned above, we took as a starting point the exemplary ABox provided with SYNAT ontology. A major part of the ABox is presented in Fig. 5. For the

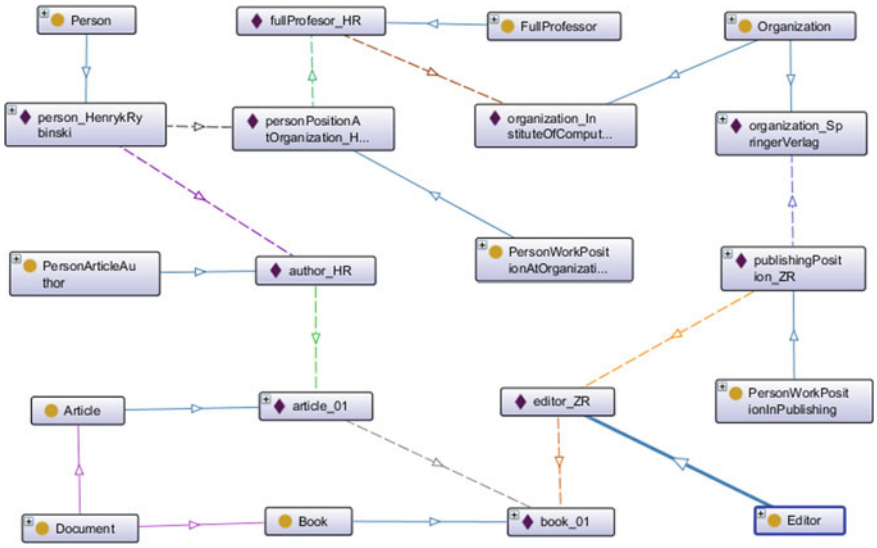

Fig. 5 A fragment of an exemplary ABox provided with SYNAT ontology

sake of this presentation we chose a fragment which allows for an answer to a query about which employees of Institute of Computer Science published articles in Springer-Verlag (in the further part of this chapter this query is denoted as Q). The picture was generated from *OntoGraf* plug-into Protégé editor.

The graph structure presented in Fig. 5 is typical for role-centric approach. It gives an ontology user considerable flexibility of specifying complex contextual information, however at the cost of complication of the structures that need to be inserted to the base. Ground objects (Agents and Information Resources) in the picture are *person_henrykRybinski*, *organization_InstituteOfComputerScience*, *organization_SpringerVerlag*, article_*01*, *book_01*. The remaining individuals are reifications of characteristics of the objects and specify relationships between them. For instance individual *author_HR* describes the authorship of the *article_01* by *person_henrykRybinski* (and can be associated with attributes one would like to relate to this authorship—e.g. its date). One of the cost of this flexibility is complexity of queries: the query Q is in fact query about instances of the following concept:

$$\exists hasPersonWorkPositionAtOrganization.\exists hasRoleAtOrganization.\exists holdsPersonRoleAtOrganization.\{organization\_InstituteOfComputerScience\} \sqcap \exists hasPersonAuthorship.\exists isAuthorOfArticle.\exists isIncludedIn.\exists isEditorOf^{-}\exists hasPersonRoleInPublishing^{-}.\exists hasWorkPositionInPublishingOrganization.\{organization\_SpringerVerlag\}$$

As a first step to contextualization of the base we divided the domain of interest of the ontology into two fragments: objects we wanted to model as individuals (like in the original ontology) and objects we wanted to reflect as elements of

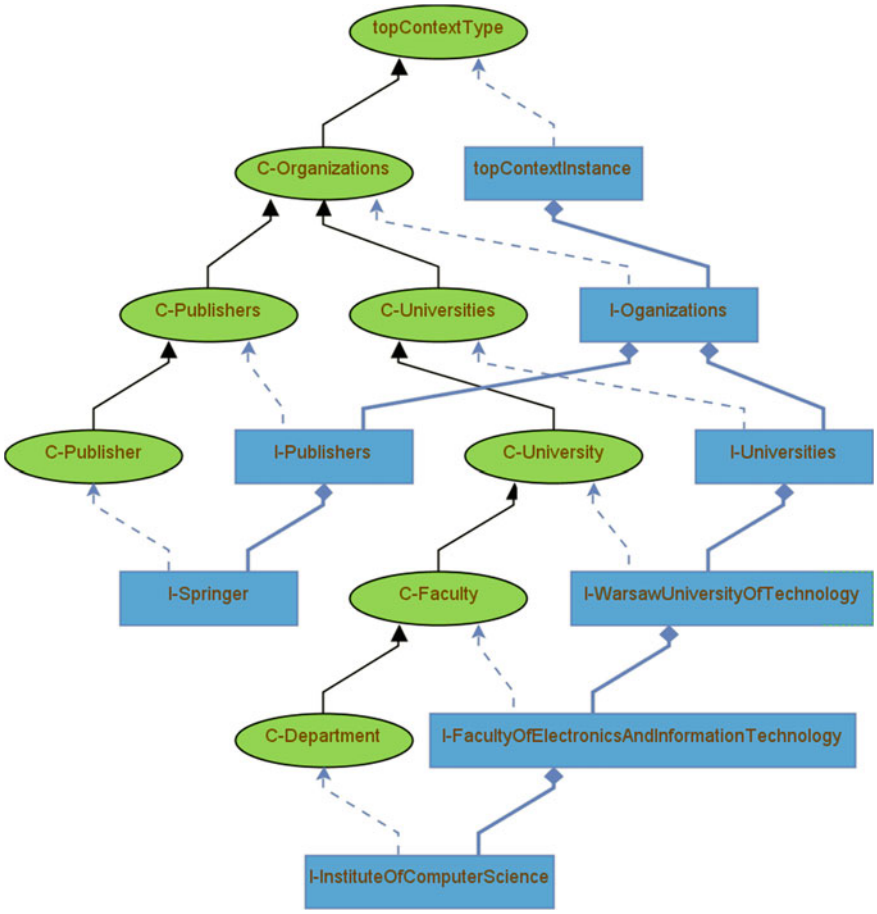

Fig. 6 Structure of contextualized SYNAT ontology

contextual structure. This decision has been made on the basis of analysis of competency queries and updates. As a result we decided to reflect in the contextual structure institutions like publishers, faculties, and institutes.

In the effect we obtained the modular structure of the knowledge base depicted in Fig. 6. In the figure context types are denoted with light ovals, and context instances with darker rectangles. The names of the context types are prefixed with *C-* and should be interpreted as referring to the contents of their context instances, e.g. the name *C-Universities* means that each instance of this context type represents a group of universities, similarly, the name *C-University* indicates that each context instance of this type represents a single university. The names of context instances are prefixed with *I-* and may carry additional information, e.g. about which university is represented by this instance.

As it can be seen in Fig. 6, the hierarchy of context types is divided into two major branches. At the top of this tree there is context type *C-Organizations*. In this contexts only very general vocabulary is introduced, namely concepts: *Person*, *Employee* (subsumed by *Person*), and *Product*. (It is worth noting, that the vocabulary does not concern organizations themselves—they are represented not as individuals but as modules—other entities important from the point of view of an organization.) The vocabulary is generally taken from SYNAT ontology, though sometimes with some major changes in meaning, as *Employee* in SYNAT ontology denotes a social role, while in the contextualized version it relates to a person.

The left branch concerns publishers: in the context type *C-Publishers* we introduce a set of new terms connected with publishing, among others concepts: *Book*, *Article* (both being subsumed by *Product*), *Editor* (subsumed by *Employee*) and roles: *hasAuthor*, *hasEditor*, *includesArticle*. (The same remark as before also applies to roles: their meaning is very similar to that in SYNAT ontology, but e.g.: while in SYNAT ontology role *hasAuthor* connects a social role—authorship—to an article, in the contextualized version, accordingly to the concept-centric approach, it directly connects a person with an article.)

The context type *C-Publisher* is intended to embrace context instances representing single publishers. The most important concepts introduced here are *LocalArticle* (subsumed by *Article*), *LocalAuthor* (subsumed by *Author*), *LocalEmployee* (subsumed by *Employee*), *LocalEditor* (subsumed by *LocalEmployee* and *Editor*), denoting respectively articles published by the specific publisher, their authors, and employees and editors of the specific publisher. Defining these concepts allows a user to state within a single sentence that an entity is bound somehow with the publisher. Moreover, these concepts are defined in such a place in the hierarchy that the knowledge about individuals being their instances does not flow between context instances representing single publishers.

The right branch of the context types tree is organized analogously. The only difference is that also some aspects of organizational structure are reflected in the tree: as a consequence, here a university is understood as both a single university and a group of faculties it includes (analogous relation takes place between a faculty and its institutes).

At the top level of the branch, in the context type *C-Universities*, several concepts are introduced, however, for our discussion the important one is *Professor*, subsumed by *Employee*. At the level of *C-University* new concepts are introduced: *UniversityEmployee* (subsumed by *Employee*) and *UniversityProfessor* (subsumed by *UniversityEmployee* and *Professor*). The concepts are analogous to concepts *LocalEmployee* and *LocalEditor* defined for publishers, and have their counterparts in the lower levels of the hierarchy: *C-Faculty* introduces *FacultyEmployee* and *FacultyProfessor*, and *C-Institute* introduces *InstituteEmployee* and *Institute Professor*.

In order to reflect the ABox fragment from Fig. 5 in this contextual structure, following assertions should be formulated in the context instance *I-Springer*:

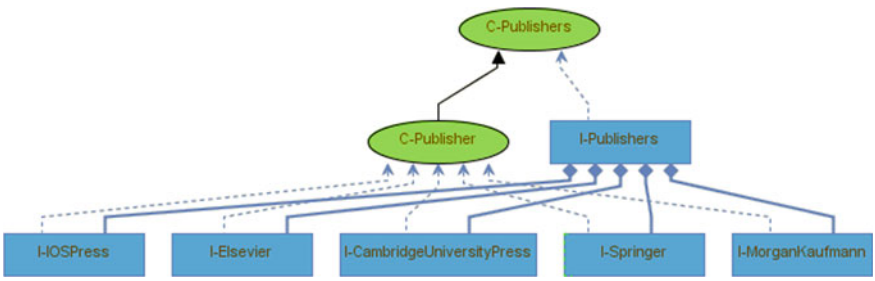

Fig. 7 A fragment of contextualized SYNAT ontology after addition of new publishers

$$
\begin{aligned}
&\mathit{LocalAuthor}(\mathit{person\_henrykRybinski})\\
&\mathit{LocalArticle}(\mathit{article\_}01)\\
&\mathit{hasAuthor}(\mathit{article\_}01, \mathit{person\_henrykRybinski})\\
&\mathit{LocalBook}(\mathit{book\_}01)\\
&\mathit{includesArticle}(\mathit{book\_}01, \mathit{article\_}01)\\
&\mathit{LocalEditor}(\mathit{person\_ZbigniewRas})\\
&\mathit{hasEditor}(\mathit{person\_ZbigniewRas})
\end{aligned}
$$

Additionally, in the context instance *I-InstituteOfComputerScience*, the single assertion has to be formulated: $\mathit{InstituteProfessor}(\mathit{person\_henrykRybinski})$.

This information suffices e.g. to answer query Q, which in the contextualized knowledge base has to be issued in two steps: first, one has to ask tarset-algebraic query for the module $\mathit{I\text{-}InstituteOfComputerScience} \cap \mathit{I\text{-}Springer}$, and then ask the resulting module about instances of concept $\mathit{InstituteEmployee} \sqcap \mathit{LocalAuthor}$.

The resulting contextual knowledge base has been verified against all the competency queries and updates. All of the queries were possible to be issued to the base, although some of them have to be formulated as queries about a set of context instances. An example of such a query is "Which publishers published articles written by person x?", which has be expressed as a query about a list of context instances in which it is true that $\mathit{LocalAuthor}(x)$ (such a form of query is covered by KQL language we use in CongloS).

Similarly, all the updates were possible to be performed, and some of them needed to be done by altering the contextual structure of the knowledge base. For instance, adding new publishers requires to insert new context instances: Fig. 7 presents an appropriate fragment of the knowledge base structure after such an update.

To sum up, the case study showed that it is possible to create a contextualized knowledge base in SIM model with the assumption of concept-centric approach which conforms to the specified requirements. This approach may lead to substantial simplification of inserting new knowledge and querying a base.

Fig. 8 A fragment of contextualized SYNAT ontology after addition of a new context type

The study was also an opportunity for observing a worrisome effect that may constitute an obstacle with use of the methods in more complex engineering solutions, namely the necessity of redefining *Professor* at each level of context hierarchy, which we plan to alleviate by introducing a new kind of coupler.

2.6 Conclusions

Analysis of the results obtained during the work allows us to draw some conclusions. The introduced structure is strictly related to the specified list of requirements. Extension of the list may lead to rendering the structure infeasible. In such a situation, a user of the knowledge base would find himself lacking of some important information, including perhaps implicit contextual parameters intended by a designer of the base.

Naturally, lack of this knowledge is simply a consequence of an incompatibility between specified requirements and user expectations, and as such might be neglected. Nonetheless, the ability to adjust the base to changing user needs is an important feature influencing the operability of the base. During the course of contextualization we identified the kind of changes in requirements that cause difficulties with the adjustment. Specifically, these are the changes that force to assign some attributes to the fragments of the original domain that have been modeled as context instances (or more generally, elements of the structure of the base) rather than individual objects.

An example of such a situation may be an addition of a new competency question about articles published by publishers from Berlin. It requires us to extend stored information by geographical locations of publishers, however there seems to be no natural way of doing this.

One of the possibilities here is to introduce a new context type *C-LocationHolders* containing a concept like *GeographicalLocation* and to inherit *C-Publisher* from this context type (see Fig. 8). A user may then assign each publisher a location by specifying an assertion like *GeographicalLocation*(*Berlin*) in *I-Springer*. However effective, this solution may easily lead to perplexity, as it mixes different levels of descriptions.

Analysis of this example leads to the conclusion that the most natural course of action is to assign attributes directly to publishers, which means directly to context instances. It in turn shows the direction of refining the description of contextual structure towards a manner similar to describing individual objects of the basic domain of interest. In the case of SIM method such a refinement might consist in treating context types in an analogous way to concepts and context instances to individuals. As a result we could gain a possibility of flexible adjustments of the knowledge base structure to changing user needs, without necessity of utilizing unnatural constructs.

Moreover, introduction of this analogy would open new possibilities for reasoning. The description of knowledge base structure could be then treated as conforming to OWA (Open World Assumption) and the relationships between modules could be inferred from some characteristic of the structure. Further elaboration of this idea led us to formulation of postulates for a new theory of information contexts described in Sect. 4.

Another issue identified during the experiment was the problem of complexity of reasoning in the relation to both the size of the structure and the contents of a knowledge base. The conducted case study embraces only a small fragment of real-world domain of interest. It is not easy to notice that the size of the base, assuming it indeed covers the majority of universities and their employees in Poland, would grow enormously.

It should be stressed that this problem also exists for non-contextual ontologies. Reasoning with OWA is very expensive and reasoning engines conforming to the assumption have difficulties with efficient handling of relatively small numbers of individuals (some attempts to overcome these limitations have been undertaken in [17], however, current trend involves rather sacrificing OWA, like e.g. in [18]).

Deployment of contextual approach should improve the effectiveness of reasoning: after all, contextualization means constraining only to relevant information, and as such should result in faster inferences. Unfortunately, with currently exploited algorithms and methods of knowledge representation, we do not observe the improvement of reasoning speed. This problem is discussed in much more details in Sect. 3, however we may briefly recall here the main reason of its occurrence: the manner of representing knowledge with the requirement of maintaining completeness of reasoning require to gather all of the sentences from the whole knowledge base in order to perform inferences from any point of view.

There are two possible ways of solving this problem. The first one consists in optimizing algorithms and improving the methods of representing contextual knowledge. Though we recorded some successes along this path, we find the second way much more promising. This way consists in supplying the inference engine with knowledge about elements of the structure of a base, embracing contextual parameters important for the process of reasoning. Such knowledge would allow the engine to neglect some of the portions of the base in some kinds of inferences (we call this *controlled incompleteness of reasoning*). This idea harmonizes very well with the aforementioned theory of information contexts and is elaborated upon in Sects. 3 and 4.

3 Reasoning Over Contextualized KB

Commencement of practical work over contextualizing SYNAT ontology was an opportunity to review and assess the principles of work of internal mechanisms which comprise CongloS system. One of the most important mechanisms to assess was the contextual inference engine.

In general, contextual approach was assumed to reduce the complexity of reasoning. Contexts are created to ease the process of inference by constraining the domain and the level of details of its description only to interesting aspects. However, sound and complete reasoning in full accordance with SIM definitions requires that the information from all the contexts in the base should be used.

The currently utilized methods employ description logics sentences (axioms and assertions) to represent contents of contexts. With this assumption, the only way to consider in the target context (the one representing the point of view from which we reason) conclusions from other contexts is by transferring sentences from one representation to another. Such a transfer is conducted basically by executing operations of tarset algebra described by couplers defined between modules of a knowledge base.

During our work over contextualizing SYNAT ontology we identified two fundamental problems related to this approach. The first problem consists in the fact that the transfer of sentences has to be all-embracing, i.e. has to embrace all the sentences from all the contexts, which is the consequence of the inference engine lacking information about relationships between knowledge in different contexts. The second problem stems from complicated structure of connections between modules (e.g. aggregation relation is represented by two cyclic couplers, i.e. the presence of aggregation between two modules M_1 and M_2 requires transfer of the knowledge—in the form of sentences—both: from M_1 to M_2 and from M_1 to M_2). Consequently, very often redundant knowledge was being transferred to the target module representation, resulting in vast growth of the number of sentences that had to be considered during reasoning.

The second problem is mainly of technical nature, and has already been mostly alleviated by introducing an optimization to the process of reasoning. The optimization consists in identification of redundant subsets of sentences. This task is carried out by performing graph simulations [19]. This solution is described in more details in Sect. 3.1.

The first problem, however, is much more significant. It can be partially solved by abandoning the sentential representation in favour of some other approach. Other kind of representation may be better suited for expressing knowledge at different levels of detail and its use may result in reducing the amount of information being transferred. This idea is illustrated by use of *cartographic representation* in Sect. 3.2.

Nonetheless, it seems that much more natural way to solve the first problem is to provide a user with a means of describing contents of specific contexts in intensional manner, thus specifying their influence on inference process. This information may

be provided in various forms: by simple heuristics or by more systematic ontological description of the knowledge base structure. This family of solutions is discussed in Sect. 3.3.

3.1 Optimizations to Sentential Reasoning

In this subsection we describe the optimization to reasoning over tarsets represented in sentential manner. This optimization is crucial for knowledge bases with complicated structure of couplers between modules (tarset variables).

First, we have to briefly recall the principles of sentential representation of tarset contents, included in the theory of $\mathcal{L}$-(2)-**D** representability. Given the set of *dummy* names **D**, a tarset T is $\mathcal{L}$-(2)-**D**-*representable* iff there exists a (finite) set of sets of sentences $\mathcal{S} = \{\mathcal{O}_i\}_{i\in[1..k]}, k \in N$, being its $\mathcal{L}$-(2)-**D** *representation* (denoted $T \sim \mathcal{S}$), which means that all the conclusions that can be drawn from T can also be drawn from $\mathcal{S}$ and, conversely, all the conclusions that can be drawn from $\mathcal{S}$ which do not concern terms from **D**, can also be drawn from T.

The contents of modules of SIM knowledge base can be easily shown to be $\mathcal{L}$-(2)-**D** representable, since (1) the user initially fills the modules with sentences, and (2) the SIM couplers consist only of three tarset algebra operations: intersection, rename and projection, and all of them preserve $\mathcal{L}$-(2)-**D** representability. A constructive algorithm for calculation of $\mathcal{L}$-(2)-**D** representation (Algorithm A_1) for the results of these operations is sketched below:

1. If $T_1 \sim \mathcal{S}_1, \mathcal{S}_1 = \{\mathcal{O}_{1:i}\}_{i\in[1..k]}$ and $T_2 \sim \mathcal{S}_2, \mathcal{S}_2 = \{\mathcal{O}_{2:j}\}_{j\in[1..m]}$, then $T_1 \cap T_2 \sim \{\mathcal{O}_{1:i} \cup \mathcal{O}_{2:i} : i \in [1..k], j \in [1..m]\}$.
2. If $T \sim \mathcal{S}, \mathcal{S} = \{\mathcal{O}_i\}_{i\in[1..k]}$, then $\rho_\gamma(M) \sim \{\gamma\mathcal{O}_i : i \in [1..k]\}$.
3. If $T \sim \mathcal{S}, \mathcal{S} = \{\mathcal{O}_i\}_{i\in[1..k]}$, then $\pi_{\mathbf{S}}(T) \sim \{\gamma_{T,\mathbf{S},\mathbf{D}}(\mathcal{O}_i) : i \in [1..k]\}$; where by $\gamma_{T,\mathbf{S},\mathbf{D}}$ we understand a function renaming terms from T not included in **S** to terms from **D**.

In the case when no non-standard couplers with union have been defined by the user, the above formulas can be greatly simplified, by the observation that all the representations may contain only a single set of sentences:

1. If $T_1 \sim \{\mathcal{O}_1\}$ and $T_2 \sim \{\mathcal{O}_2\}$, then $T_1 \cap T_2 \sim \{\mathcal{O}_1 \cup \mathcal{O}_2\}$.
2. If $T \sim \{\mathcal{O}\}$, then $\rho_\gamma(M) \sim \{\gamma(\mathcal{O})\}$.
3. If $T \sim \{\mathcal{O}\}$, then $\pi_{\mathbf{S}}(T) \sim \{\gamma_{T,\mathbf{S},\mathbf{D}}(\mathcal{O})\}$; $\gamma_{T,\mathbf{S},\mathbf{D}}$ is defined as above.

In the case of a standard SIM knowledge base, the source of complexity of reasoning over sentential representation is twofold: it is a consequence of cyclic couplers and of projection operation (formula 3 in the above list) which produces new sentences.

Let us illustrate this effect with a simple example. We consider a knowledge base K with four modules (see Fig. 9): context types M_1 and M_2, and context

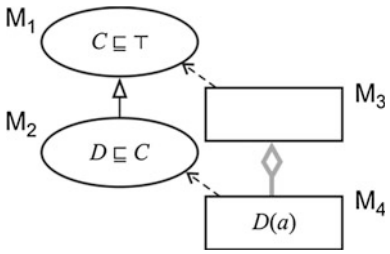

Fig. 9 An exemplary knowledge base K

instances M_3 (the only instance of M_1) and M_4 (the only instance of M_2). M_2 inherits from M_1, M_3 aggregates M_4, and initial contents of modules is: $M_1 := \{C \sqsubseteq \top\}$, $M_2 := \{D \sqsubseteq C\}$, $M_3 := \{\}$, $M_4 := \{D(a)\}$. Let us moreover assume that the set of dum*X*my names **D** consists of infinite (but countable) number of names of the form $_i$ (for concepts) and x_i (for individuals), $i \in N$.

Execution of a naïve implementation of algorithm A_1 for K would result in infinite growth of representations of modules M_3 and M_4. Below we present the results of some first iterations of the algorithm for these modules:

1. $M_4' \sim \{\{D(a), D \sqsubseteq C\}\}$
2. $M_3' \sim \{\{X_1(a), X_1 \sqsubseteq C\}\}$
3. $M_4'' \sim \{\{D(a), D \sqsubseteq C, X_1(a), X_1 \sqsubseteq C\}\}$
4. $M_3'' \sim \{\{X_1(a), X_1 \sqsubseteq C, X_2(a), X_2 \sqsubseteq C\}\}$

Numerous strategies exist that can be used to eliminate the above effect. One of the simplest ideas involves remembering which sentences, after projection, have already been transferred to other modules. This projection prevention strategy can be included into a category of pre-execution strategies as it prevents some of the superfluous transfers of sentences. However, the strategy requires a very careful control policy, as every projection has to be applied to a whole set of sentences (like in the above example, transfer of the conclusion that $C(a)$ to M_3 requires us to translate and transfer the whole set of sentences $\{X_1(a), X_1 \sqsubseteq C\}$).

After several experiments we found that the most effective solution is deployment of a simple post-execution strategy (hence removal strategy), which consists in removal of redundant subsets of sentences containing dummy terms. Post-execution strategies generally have an advantage over pre-execution ones, as they are not dependent on the set of tarset algebra operations.

The removal strategy consists of two major stages. In the first stage potentially redundant subsets are identified. To identify the subsets we used a procedure for calculating graph simulation [19], after changing the sentences in module representation to a graph form.

A simulation relation is defined for a labeled directed graph G understood as a quadruple (V, E, L, l) where V is a set of vertices, E set of edges, L set of labels and l is a function which assigns each vertex from V to a label from L. A relation $\leqslant \subseteq V \times V$ is a simulation iff for every $u \leqslant v$ it follows that (1) $l(u) = l(v)$ and (2)

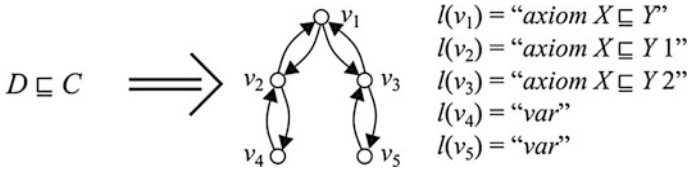

Fig. 10 An illustration of converting a sentence to a graph (removal strategy)

for all vertices u' such that $(u, u') \in E$, there exists a vertex v' such that $(v, v') \in E$ and $u' \leqslant v'$.

A simulation calculation algorithm utilized for the first stage of removal strategy is an elimination algorithm, i.e. it calculates the next approximation $\leqslant^{i+1}$ of the $\leqslant$ on the basis of the current approximation $\leqslant^{i}$ by eliminating from $\leqslant^{i}$ pairs which do not conform to the conditions (1) and (2) from the previous paragraph, and stops after reaching a fixpoint. In the original algorithm the first approximation $\leqslant^{1}$ was calculated as $\{(u, v)\colon l(u) = l(v)\}$.

In order to use the simulation calculation algorithm for identifying redundant subsets of sentences, the $\mathcal{L}$-(2)-**D** representation is transformed into a tri-partite graph, where labels are strings of characters. The first partition consists of vertices representing axioms. Only axioms of exactly the same grammatical structure (e.g. $C \sqcap D \sqsubseteq E$ and $E \sqcap F \sqsubseteq D$) are assigned the same labels. Vertices in the second partition represent possible reassignments, i.e. i-th position in the grammar structure of each axiom. Their label is the concatenation of the label of the axiom and the number of the position. The third partition embraces vertices representing actual terms used in grammar structure, and are all assigned the same label "*var*". Edges in the graph are set as follows: we connect bi-directly a vertex for every axiom with vertices for each of its assignments and, also bi-directly a vertex for every assignment (see Fig. 10 for the illustration of the process).

Such manner of constructing the graph allows us to almost directly use the simulation calculation algorithm to identify similar sets of axioms and terms, i.e. axioms of the same grammatical structure which hold similar terms in the same positions, and terms which occur in the same positions in similar axioms of the same grammatical structure. The only (small) modification to the algorithm is a slight change of the first approximation $\leqslant^{1}$, from which there are removed edges directed from vertices representing non-dummy names to vertices representing dummy names, to reflect the assumption that we cannot replace a non-dummy name with dummy one.

In the second stage of the algorithm, the resulting simulation $\leqslant$ is being analyzed, and the subsets of axioms that can be reduced to other subsets by replacing assignments to those of similar terms are calculated. A simple heuristic is used here, according to which more frequently used terms are replaced with similar terms in the first place (in a greedy fashion). After these replacements the set of axioms is reduced.

In the example with the base K application of the strategy modifies the 4th step of algorithm execution. The representation $\{\{D(a), D \sqsubseteq C, X_1(a), X_1 \sqsubseteq C\}\}$ of M_4'' is reduced to $\{\{D(a), D \sqsubseteq C\}\}$ on the basis of similarity between X_1 and D. X_1 is identified as a term used in exactly the same positions of similar sentences as D and is therefore replaced with D, which results in the reduction. Further consequence of this reduction is reaching the fixpoint in the 4th step and termination of the algorithm.

Ideologically, the removal strategy can be easily explained by the observation that dummy terms can be interpreted as "some examples" of concepts, individuals etc. So, for instance, the representation $\{\{X_1(a), X_1 \sqsubseteq C\}\}$ of M_3' can be read as "there exists some concept being subsumed by C, and a is a member of the concept." Such interpretation allows us to draw a (correct) conclusion that a is also a member of C. Thus sentences $X_1(a)$ and $X_1 \sqsubseteq C$ are clearly superfluous in the representation $\{\{D(a), D \sqsubseteq C, X_1(a), X_1 \sqsubseteq C\}\}$ of M_4'' as there already exists such a concept, and it is D.

The deployment of the removal strategy allowed for faster identification of fixpoints and for vast decrease of the time of reasoning in comparison with projection prevention strategy. (Preliminary tests indicate that the decrease is the higher the more complicated is the structure of the knowledge base—reaching two orders of magnitude in the case of knowledge bases comprising of more than 20 modules; more systematic experiments are planned to be executed in the nearest future.)

3.2 Non-sentential Representation of Tarsets

Another approach that may be assumed in order to avoid problems with transferring vast numbers of sentences is a change of the method of representing tarsets. New representation should be more deeply rooted in semantics and allow for easier change of the level of details of expressing knowledge (to facilitate movement of knowledge between different levels of SIM structure).

A good candidate for the new representation is knowledge cartography. Knowledge cartography (hence KC) is a technique developed by authors in their earlier works. It focuses on simplified description of the space of ontology models, yet aiming at capturing the largest number of possibly useful conclusions.

The key notion in KC is a *map of concepts*. A map of concepts m for ontology $\mathcal{O}$ is a function which assigns concepts from a chosen set $\mathbf{C}$ a subset of *set of regions* $\mathbf{R} = \{r_1, r_2, \ldots r_n\}$ ($n \in N$ is a *size of the map*) in such a way that $\forall C, D \in \mathbf{C} : m(C) \subseteq m(D) \Leftrightarrow \mathcal{O} \models C \sqsubseteq D$. Putting it in other words, relations between sets of regions assigned to concepts reflect interrelationships (like inclusion or disjointness) between interpretation of such concepts in ontology models.

An extension of the notion of map of concepts is an *individual placement*, a function r which assigns each individual from an ontology (let us denote their set $\mathbf{I}$) a subset of $\mathbf{R}$ in such a way that $\forall C \in \mathbf{C}, a \in \mathbf{I} : r(a) \subseteq m(C) \Leftrightarrow \mathcal{O} \models C(a)$.

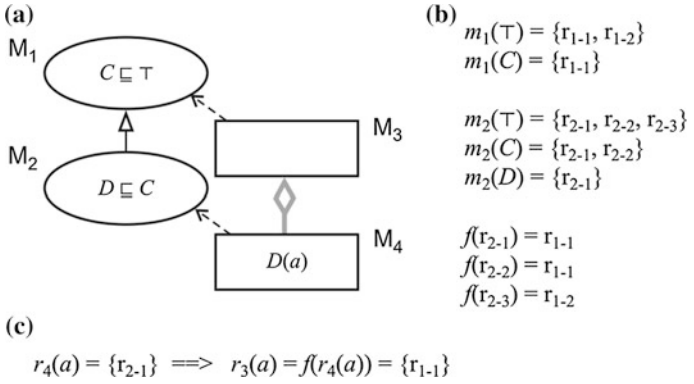

Fig. 11 Use of KC for inferences: **a** shows the KB structure, **b** maps m_1 and m_2 for M_1 and M_2 resp. and mapping f between them, **c** placements r_4 and r_3 for M_3 and M_4 and transfer of knowledge from M_4 and M_3 with use of the function f

The main strength of KC is its ability to unify different syntactic forms of the same concepts, e.g. concepts like $C \sqcap D$, $C \sqcap D \sqcap D$ and $\neg(\neg C \sqcup \neg D)$ have to be assigned the same set regions (by definition of the map). Moreover, this unification goes further to considering relations between concepts introduced by the ontology itself: e.g. if $C \sqsubseteq D$ is an axiom in $\mathcal{O}$ it has its consequences for instance in $m(C \sqcap D) = m(C)$.

Due to this feature KC can be used to transfer conclusion in the unified form. The idea here is to create a map of concepts for each context type and to establish a mapping between regions of different maps. The mapping function f may be then used to recalculate placement of individuals in different context instances. The idea of such a mapping and KC-based transfer of conclusions is illustrated in Fig. 11 (it is based on the example of the knowledge base K from Fig. 9).

The idea of using KC seems very appealing, and has vast potential to reduce the amount of information necessary to be transferred between contexts. However, KC also has drawbacks, one of the main being lack of full OWL 2 expressiveness. Moreover, use of KC does not eliminate the necessity of examination of contents of all the context from a KB in order to answer a query issued toward any target context.

3.3 Controlled Incomplete Reasoning Strategies

While the technique proposed in Sect. 3.2 has its potential in reducing necessary transfers of knowledge between modules of a SIM knowledge base, it still requires to carefully check all the contexts in search of all the relevant information. Though such kind of reasoning has its uses, it seems more natural for contextual approach

to be able to abstract from information stored in some of the contexts at least in some of inference tasks.

This direction of development seems to be very appealing, however only a very introductory conceptual research in this area has been undertaken by our group. The simplest strategy that can be exploited here is *user-guided* query answering. With use of this strategy during solving reasoning problem inference engine asks the user to explicitly decide upon the set of context considered during reasoning (knowledge from other contexts is not analysed).

The drawback of user-guided approach is that its effectiveness relies on the familiarity of the user with advanced issues, like the structure of a knowledge base. A viable alternative may be to use a *heuristic-guided* strategy. This strategy assumes that a set of heuristics is employed to determine the set of relevant contexts. Heuristics may vary in complication: the simplest ones may put a threshold on the distance between modules (understood as a path distance in a graph of knowledge base structure), the more sophisticated may count the numbers of common terms within two modules.

Both of the strategies introduced above bring in the serious danger of omitting important knowledge in the process of reasoning, for critical queries. To eliminate this danger it seems necessary to introduce a strategy which involves explicit specification of the knowledge about meaning of specific modules (contexts and context instances). Such idea of specifying "semantics of structure" harmonizes very well with the theory of information contexts described in the next section. This semantics may be used to perform a preliminary inferences whose result will determine the set of contexts to be used in answering a query. However, this would require major development of both knowledge base structure and contextual theory, according to the assumptions outlined in Sect. 4.

4 Hierarchical Structure of Knowledge Bases

The conducted experiments give us valuable hints and argument for the support of our main goal, which is to develop a novel, universal and holistic theory of knowledge representation. This theory of information contexts (TIC) is focused on contexts perceived as one of the main kinds of elements forming the structure of knowledge, and is not constrained in its possible range of application to knowledge bases.

What we perceive an important argument in favor of undertaking this task is that knowledge management techniques developed so far are becoming more and more inadequate for handling enormously growing heterogenic repositories available on the Internet. In the situation when the size of the repositories reaches petabytes and above, retaining ACID properties becomes impossible and relational databases gradually lose their status of a primary information sources to emerging NoSQL databases. Research on new methods and tools for managing large information sets (*big data*) is becoming necessity.

Our concept lies in engaging contexts as a model for substantial elements from the beginning of the conceptualization stage of a designing process. Conceptualization and contextualization should be realized together and affect each other. Both levels of abstraction contain mutually related fragments of a single model. The contextual level delivers elements allowing to create types and instances of contexts. Similarly the conceptual level delivers elements allowing to create classes of objects and their instances.

This similarity of the elements allows us to believe that there exists a way of unification of rules that order their creation and usage. The unification gives a basis for elaboration of a recurrent mechanism embracing every layer of knowledge representation structure. We propose to distinguish the following four layers in the structure.

Domain layer embraces ontological description of the universe of discourse. Assuming Description Logics as the basis, we can notice the well-known TBox/ABox distinction: terminological (general) knowledge is expressed with *concepts* and *roles*, while factual knowledge assigns individual objects to concepts, and pairs of objects to roles. Contextual aspects (assumed point of views) are most often neglected in this layer and need some external description (like OMV metadata [20]).

Module arrangement layer concerns modular knowledge bases, though we may also treat standard import OWL relation as inter-modular. The essential function of the layer is to determine a role of each module in the structure and its contribution to the process of inference. One of the major problems within this layer is version management: possible influence of updating one module on other modules.

System management layer describes relationships between whole knowledge bases. Due to this specificity, the main problem within the layer (apart from inherited version management) is knowledge integration.

Paradigm layer concerns different methods of encoding knowledge and relationships between modules. It embraces problems of transformation between diverse methods of modularization of knowledge bases (like e.g. DDL [4], $\mathcal{E}$-connections [21]).

One may notice the internal similarity of different layers: in each of the layers we may see a dichotomy very similar to well-known TBox/ABox relation. This observation, supported by the encouraging results of the case study, led us to formulation of a hypothesis that the use of the hierarchical structure of knowledge may be beneficial at every layer. Moreover, the possibility of treating various elements of such structures alternatively and simultaneously as the elements of conceptualization (objects and classes) or contextualization (context instances and context types) would open significant possibilities of improvement of managing knowledge.

As a consequence we see substantial reason in creation of hierarchical and recurrent model of knowledge organization. If constructed properly, such a model can offer unified view on many important problems of knowledge engineering, thus broadening the range of use of many existing techniques.

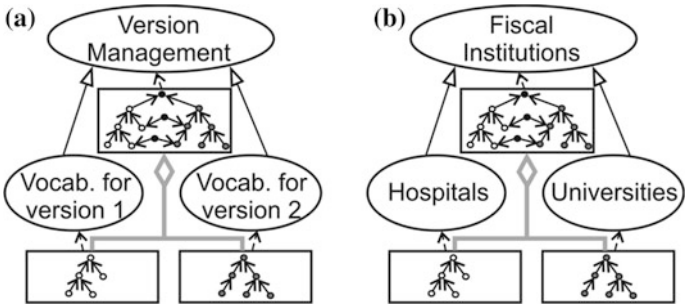

Fig. 12 Version management (**a**) and knowledge integration (**b**) in recursive knowledge model

An example of use of such a model covers a very important problem of *version management*. The issue here is the possibility that ontologies that import the one being updated may easily become unsatisfiable or incoherent, which is the consequence of the fact that the author of the original ontology being imported cannot easily foresee all of its uses.

In the recursive knowledge model this problem has an interesting solution. For two versions of a knowledge base we can create two descriptions in the form of two context instances in the system management layer. These two instances can be then aggregated in an instance which gathers knowledge about both versions. By placing appropriate couplers in the aggregating instance, we can establish a flow of facts and conclusions between the two versions. Consequently, it is possible to maintain only the newer version and to assure that only "safe" conclusions are transferred to the older version, which prevents the importing knowledge bases from becoming unsatisfiable.

The illustration of such a situation is presented in Fig. 12a. The structure of the older version is described with use of individuals depicted with white center, and the newer version with gray center. The aggregating context instance builds a new structure by adding a new top context (black individual at the top) and new couplers between chosen elements of both structures (black individuals below).

The couplers are basically algebraic operation on tarsets. Within version management projection and selection seem particularly well fit for the task. The former allows to ignore new terms and thus reduce the risk of name conflicts, while the latter allows to focus only on a subset of facts (e.g. from a specified date range). The end users, though, do not have to worry about mathematical formalities, as instead of algebraic terms, they use vocabulary defined in Version Management context type.

A remarkable feature of the model is the fact that exactly the same course of action can be undertaken to integrate knowledge from two knowledge bases. In Fig. 12b we can see exploitation of the same pattern for connecting two institutions. Lower instances represent individual knowledge bases of a hospital and a university, for the sake of discussion we may assume Medical University Hospital in Gdańsk, and Gdańsk University of Technology. Since both institutions are fiscal, we may aggregate their bases at the higher level. In such an approach the

fact that *Fiscal institutions* context type provides vocabulary plays two roles: facilitates building more specific contexts (like *Universities*), and allows for integration of knowledge bases in the whole sub-tree with use of common terms.

Lifting the same technique further allows for describing the way of *migration* from one representation model (like DDL) to another (like SIM). The aggregating context instance has to be placed on the highest level of system management layer and specify the connections between contents of appropriate module in the basic language of couplers and tarsets.

5 Summary

In this chapter we described our experiences and conclusions from performing contextualization of SYNAT ontology. This contextualization is a first systematic practical case study for deployment of our original methods of knowledge base modularization.

Contemporary knowledge bases grow larger and larger. The vast amount of information that has to be stored and managed in an intelligent way motivated intensive research in the field of management of large knowledge bases, especially their modularization.

In our research we consequently assume the stance that contextualization (decomposition of knowledge in accordance with various points of view) should be considered an integral part of conceptualization (the mental process which leads to creation of a description of some domain of interest). Participation in SYNAT project gave us the opportunity to practically verify this thesis.

The course of the experiments confirmed the usefulness and generality of the proposed approaches and allowed us to make many valuable observations. The main conclusion is that by further development of the contextual methods, it is possible to create a truly universal framework for capturing and effective management of knowledge contained on the Internet, which would cover aspects of Big Data management and Context-Oriented computing, consequently allowing for intelligent handling of vast amount of information and data currently produced on everyday basis.

Acknowledgments We want to thank Professor Wierzbicki for his comment to our presentation during the last SYNAT Workshop, which became a valuable inspiration for our further work.

References

1. Goczyla, K., Waloszek, A., Waloszek, W., Zawadzka, T.: OWL API-based architectural framework for contextual knowledge bases. In: Bembenik, R., Skonieczny, Ł., Rybiński, H., Niezgódka, M. (eds.) Intelligent Tools for Building a Scientific Information Platform: Advanced Architectures and Solutions. Springer, Berlin (2013)

2. Goczyła, K., Waloszek, A., Waloszek, W.: A semantic algebra for modularized description logics knowledge bases. In: Proceedings of DL 2009, Oxford, United Kingdom (2009)
3. Goczyła, K., Waloszek, A., Waloszek, W.: Contextualization of a DL knowledge base. In: Proceedings of DL Workshop DL'2007, pp. 291–299 (2007)
4. Borgida, A., Serafini, L.: Distributed description logics: assimilating information from peer sources. J. Data Semantics **1**, 153–184 (2003)
5. Wróblewska, A., Podsiadły-Marczykowska, T., Bembenik, R., Rybiński, H., Protaziuk, G.: SYNAT system ontology: design patterns applied to modeling of scientific community, preliminary model evaluation. In: Bembenik, R., Skonieczny, Ł., Rybiński, H., Niezgódka, M. (eds.) Intelligent Tools for Building a Scientific Information Platform: Advanced Architectures and Solutions. Springer, Berlin (2013)
6. Gangemi, A., Lehmann, J., Catenacci, C.: Norms and plans as unification criteria for social collectives. In: Proceedings of Dagstuhl Sem. 07122, vol. II, pp. 48–87 (2007)
7. Gruber, T.R.: Towards principles for the design of ontologies used for knowledge sharing. Int. J. Hum. Comput. Stud. **43**, 907–928 (1995)
8. Paliouras, G., Spyropoulos, C. D., Tsatsaronis, G. (eds.): Knowledge-driven multimedia information extraction and ontology evolution: bridging the semantic gap. Series: LNCS, vol. 6050 (2011)
9. FOAF Vocabulary Specification 0.98. Namespace Document 9 August 2010 - Marco Polo Edition. Available at: http://xmlns.com/foaf/spec/
10. Krafft D.B., Cappadona N.A., Caruso B., Corson-Rikert J., Devare M., Lowe B.J., VIVO Collaboration: VIVO: Enabling National Networking of Scientists, Web-Sci10: Extending the Frontiers of Society On-Line. (Raleigh, NC, April 26-27, 2010). VIVO model available at: http://vivoweb.org/download
11. Sure, Y., Bloehdorn, S., Haase, P., Jens Hartmann, J., Oberle, D.: The SWRC ontology—semantic web for research communities. In: 12th Portuguese Conference on Artificial Intelligence-Progress in Artificial Intelligence EPIA 2005. LNCS, vol. 3803, pp. 218 – 231. Springer, Heidelberg (2005). SWRC model available at: http://ontoware.org/swrc/
12. BIBO Ontology, http://bibotools.googlecode.com/svn/bibo-ontology/trunk/doc/index.html
13. Guarino, N.: Formal ontology and information systems. In: Proceedings of the 1st International Conference on Formal Ontologies in Information Systems FOIS'98. Trento, Italy, IOS Press, pp. 3–15 (1998)
14. Genesereth, M.R., Nilsson, N.J.: Logical Foundation of Artificial Intelligence. Morgan Kaufmann, Los Altos (1987)
15. Cybulka, J.: Applying the c.DnS design pattern to obtain an ontology for investigation management system. In: Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems. LNCS, vol. 5796, pp. 516–527 (2009)
16. Jędrzejek, C., Cybulka, J., Bąk, J.: Towards ontology of fraudulent disbursement. In: Agent and Multi-Agent Systems: Technologies and Applications. LNCS, vol. 6682, pp. 301–310 (2011)
17. Fokoue, A., Kershenbaum, A., Li Ma, Schonberg, E., Srinivas, K., Williams, R.: SHIN ABox reduction. In: Proceedings of DL Workshop DL'2006, pp. 135–142 (2006)
18. Horst, H.J.: Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. J. Web Semantics **3**(2–3), 79–115 (2005)
19. Henzinger, M.R., Henzinger, T.A., Kopke, P.W.: Computing simulations on finite and infinite graphs. In: Proceedings of the 36th Annual Symposium on Foundations of Computer Science, pp. 453–462 (1995)
20. Hartmann J., Palma R., Sure Y.: OMV—Ontology Metadata Vocabulary, ISWC 2005
21. Kutz, O., Lutz, C., Wolter, F., Zakharyaschev, M.: $\mathscr{E}$-connections of abstract description systems. Artif. Intell. **156**(1), 1–73 (2004)