Multilingual verbalization of ORM conceptual models and axiomatized ontologies

Mustafa Jarrar[1] **C. Maria Keet**[2] **Paolo Dongilli**[2]

[1]STARLab, Vrije Universiteit Brussel, mjarrar@vub.ac.be
[2]Faculty of Computer Science, Free University of Bozen-Bolzano, Italy {keet, dongilli}@inf.unibz.it

Abstract. *Verbalization is the process of writing the semantics captured in axioms into natural language sentences, which enables domain experts (who are not trained to understand technical/formal languages) to be able to participate in the modeling and validation processes of their domain knowledge. We present a novel approach to support multilingual verbalization of logical theories, axiomatizations, and other specifications such as business rules. This engineering solution is demonstrated with the Object Role Modeling language and the ontology engineering tool DogmaModeler, although its underlying principles can be reused with other conceptual models and formal languages, such as Description Logics, to improve its understandability and usability by the domain expert. Our engineering solution for multilingual verbalization is characterized by its flexibility, extensibility and maintainability of the verbalization templates, which allow for easy augmentation with other languages than the 10 currently supported.*

1. Introduction

Verbalization is the process of writing the semantics captured in a logical theory, i.e. the relations between the entities and their constraints, into pseudo-natural language (fixed-syntax) sentences. For example,

$$\forall x\,(\text{Account}(x) \rightarrow \exists y,z\,(\text{Person}(y) \wedge \text{Company}(z) \wedge (\text{OwnedBy}(x,y) \vee \text{OwnedBy}(x,z)))$$

can be verbalized into

Each Account must be OwnedBy a Person or a Company, or both.

Verbalization of a conceptual model is a well known, and by the domain expert highly appreciated, feature of the Object-Role Modeling (ORM) methodology. This pseudo-natural language is understandable for domain experts, in contradistinction to axioms familiar to logicians, thereby capable of bridging a gap in communication between modelers and domain experts and provides a common ground for mutual understanding of the semantics a logical theory captures – be it a conceptual model, business rules or a formal ontology. Currently, only ORM with application support of tools such as VisioModeler, DogmaModeler, and Microsoft Visio can automatically generate verbalizations in English. With the emerging Semantic Web and ontology development efforts in general, a usable and effective mode of communication with domain experts is essential if the model or ontology is to describe the subject domain accurately. Verbalization as a usability requirement is a beneficial mechanism for the logic and ontology development community to increase acceptance of their efforts among a wider base of potential users. For domain experts, such as lawyers, bio-ontologies developers, and the business rules community, it enables them to confirm for themselves that their knowledge is represented

accurately and that eventually its related software can be functioning according to their specifications.

We introduce a flexible, extensible, and easy to maintain engineering solution to verbalize logical theories. This solution does not only accommodate for English-language users, but recognizes the multilingual settings of international cooperation and therefore its methodology was developed to ensure providing a means for a quick and easy generation of verbalizations in other languages. At present, these include German, French, Italian, Spanish, Catalan, Dutch, Arabic, Russian, and Lithuanian; developing a verbalization template for a language requires about 2 hours, which then is used to automatically verbalise any model in that language.

In the remainder of the article, we briefly introduce ORM and our DogmaModeler ontology engineering tool in section 2 and in section 3 we present our verbalization template-approach. Extant research and useful techniques to move from understandable pseudo-natural language to grammatically correct sentences by using language generation engines is the topic of section 4. Section 5 concludes the paper and presents our future research directions.

2. ORM and DogmaModeler

ORM [1] is a conceptual modeling approach, which is the successor of NIAM that was developed in the early 70's. We have chosen ORM to demonstrate our approach as it is comprehensive in its treatment of many rules and constraint types (identity, mandatory, uniqueness, subsumption, subset, equality, exclusion, value, frequency, symmetric, intransitive, acyclic, etc.). Furthermore, ORM has an expressive and stable graphical notation, captures many constraints graphically and is attribute-free[1], thereby minimizing the impact of change on the models and facilitating model evolution with much less disruption compared with e.g. EER and UML. Moreover, it has greater expressivity than other graphical-based or graphic-supported ontology and conceptual modeling methods and ORM has been fully formalized in First Order Logic [2], such that one can consider both the visualization and verbalization as icing on the formal cake.

Although ORM was originally developed as a database modeling approach, it is used successfully also in other conceptual modeling scenarios, such as business rules modeling [3][4] ontology modeling [5][6], and XML-Schema conceptual design [7], which illustrates and emphasizes its design- and implementation-independence. We regard an ORM schema as a general conceptual model independently of a certain application or modeling scenario; and we interchange the term "ORM schema" with the term axiomatization, logical theory, and ontology to refer to the same artifact.

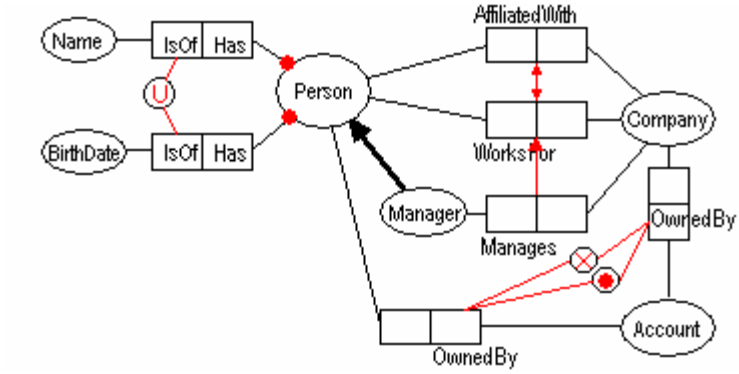
We illustrate the formalization, verbalization, and graphical notation of ORM in the example.

Example. We show a simple example of an ORM Schema in Fig.1 (the running example, depicted in Fig. 2, includes more sophisticated constraints that will be verbalized below). ORM's prime components are object-types, i.e. universals (e.g. Person, Company, etc.), and roles, (relations such as Has, OwnedBy, etc.). ORM constraints are denoted by icons such as (U) for inter-uniqueness, (•) for mandatory, (X) for exclusion, (→) for subset, and (↔) for equality. The underpinning logics of the ORM schema in Fig.1A are presented in Fig.1B, and Fig.1C presents the verbalization of these axioms. Many other kinds of rules are supported in ORM, but are not illustrated here for the sake of brevity. See [2] for more about ORM and its complete formalization, which we follow in the paper. ♦

[1] Attribute-free means that there are just objects types and roles in an ORM model, without the 'prioritization' of one object type (universal) over another as is done in other conceptual modeling languages, like (E)ER and UML.

The above example is developed using our DogmaModeler tool [6]. DogmaModeler is an ontology engineering tool based on ORM principles, using double articulation of an ontology into an ontology base and axiomatized commitment layers [5]. It supports the development of ontologies and business rules using ORM and fully implements the multilingual verbalization approach presented in this paper. DogmaModeler also supports the generation of ORM-ML (which is an ORM mark-up language, i.e. XML serialization of the ORM notation) [5], modularization and automatic composition of ORM modules through a well-defined composition operator [8]; the incorporating of linguistic resources in ontology engineering [9] and automatic mapping of ORM schemes into X-Forms and HTML forms [6]. Its design and implementation of the multilingual verbalization is the topic of the next section.

A



B

- *Subsumption:* $\forall x\ (Manager(x) \rightarrow Person(x))$
- *Mandatory:* $\forall x\ (Person(x) \rightarrow \exists y\ (Name(y) \wedge Has(x,y)))$
- *Mandatory:* $\forall x\ (Person(x) \rightarrow \exists y\ (BirthDate(y) \wedge Has(x,y)))$
- *InterUniqueness:* $\forall x_1 x_2, y,z\ (Person(x) \wedge Name(y) \wedge BirthDate(z) \rightarrow (IsOf\_N(x_1,y) \wedge IsOf\_B(x_1,z) \wedge IsOf\_N(x_2,y) \wedge IsOf\_B(x_2,z) \rightarrow x_1 = x_2))$
- *Equality:* $\forall x, y\ ((Person(x) \wedge Company(y) \wedge WorksFor(x,y)) \leftrightarrow AffiliatedWith(x,y))$
- *Subset:* $\forall x, y\ ((Person(x) \wedge Company(y) \wedge Manages(x,y)) \rightarrow WorksFor(x,y))$
- *ExMandatory:* $\forall x\ (Account(x) \rightarrow \exists y,z\ (Person(y) \wedge Company(z) \wedge (OwnedBy\_P(x,y) \vee OwnedBy\_C(x,z))))$
- *Exclusion:* $\forall x,y,z\ (Account(x) \wedge Person(y) \wedge Company(z) \rightarrow \neg(OwnedBy\_P(x,y) \wedge OwnedBy\_C(x,z)))$
- …

C

- *Subsumption:* Each Manager must be a type of Person.
- *Mandatory:* Each Person must Has at least one Name.
- *Mandatory:* Each Person must Has at least one BirthDate.
- *InterUniqueness:* The combination of {BirthDate, Name} must refer to at most one Person.
- *Equality:* Each Person WorksFor a Company must AffliatedWith that Company, and the other way around.
- *Subset:* Each Manager who Manages a Company must WorksFor that Company.
- *ExMandatory:* Each Account OwnedBy Person or OwnedBy Company, or both.
- *Exclusion:* No Account can OwnedBy a Company and OwnedBy a Person.

Fig. 1. A: Example of ORM schema. B: the formal semantics of the ORM schema example; C: its corresponding verbalization.

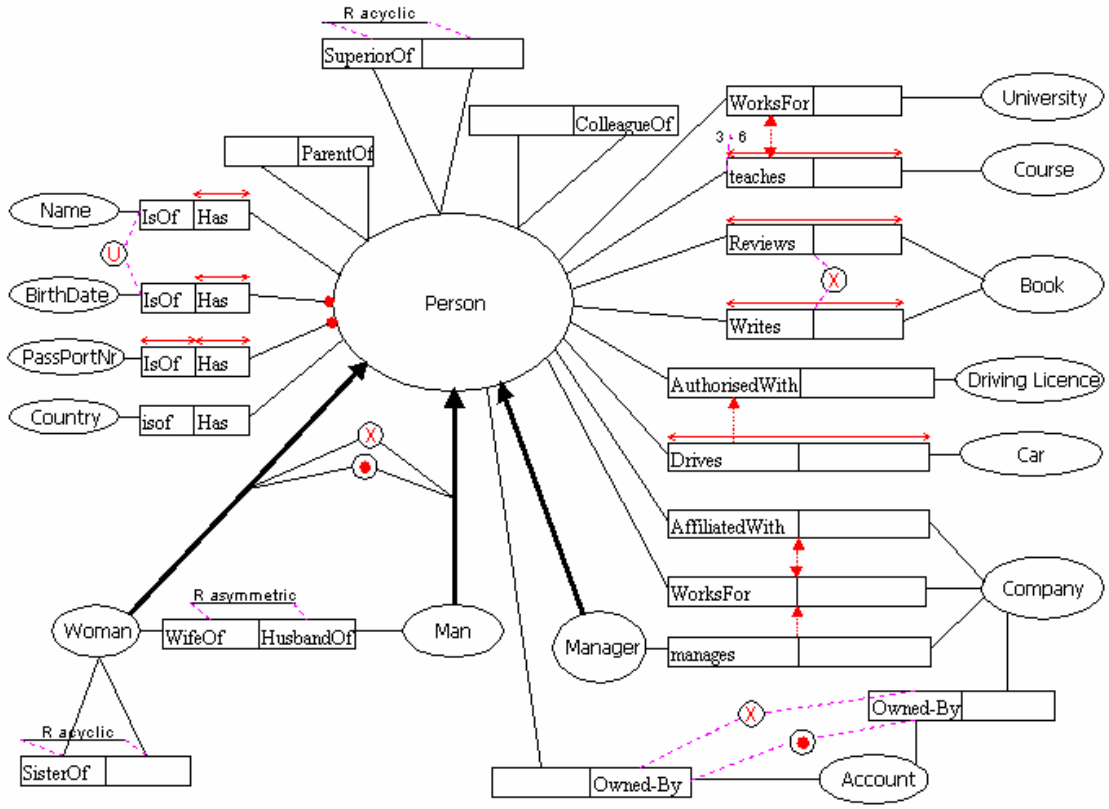


Fig. 2. ORM diagram (or axiomatized ontology commitment layer) used as running example.

3. Multilingual verbalization

We describe the software engineering approach first in section 3.1 and subsequently proceed to the explanation of the verbalization templates in 3.2. In the last section we discuss further customization options to meet different software and user requirements.

3.1 Approach

The flexible approach and extendable and easy to maintain implementation of the multilingual verbalization is shown in Fig.3. Based on the semantics of each type of constraint that is natural language independent, a verbalization template file is created for some natural language *x* such that it contains a fixed-syntax structure for each constraint. Each structure contains tags to refer to object types and roles that take part in the constraint and is appended with natural language text within text tags to glue together the components of the constraint to create near-natural language sentences. The verbalization template files are attached to DogmaModeler as "setting-files". Notice that the files for each language are not intended to be customized by "normal" ontology builders. Rather, the idea is to equip ontology engineers and experts with an easy to translate (or improve) verbalization mechanism. At run-time, the appropriate predefined verbalization template file is applied to an ORM-ML document that encodes an axiomatized ORM schema to automatically verbalize its contents. One can simply click on the "Pseudo NL" tab to view its verbalization; this is shown in Fig. 4 for Italian using the running example as depicted in Fig. 2.

To avoid creating a verbalization template file from scratch for each language, one can take a template file from a grammatically related language *y* and use this as template for the new language in order to speed up its creation and translation. This new template file is applied to an

ORM-ML document containing terms of the object types and roles in language *y* to also automatically generate pseudo-natural language sentences from an ORM diagram in that language. In the next section, we illustrate the verbalization templates and how to create such a file for one's desired language.

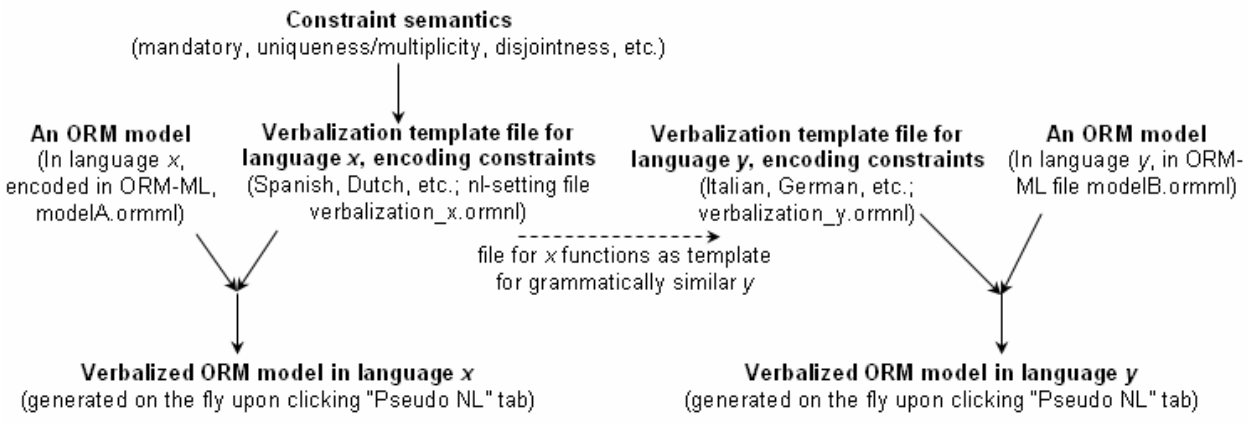


Fig. 3. High-level view of the multilingual verbalization strategy implemented in DogmaModeler.

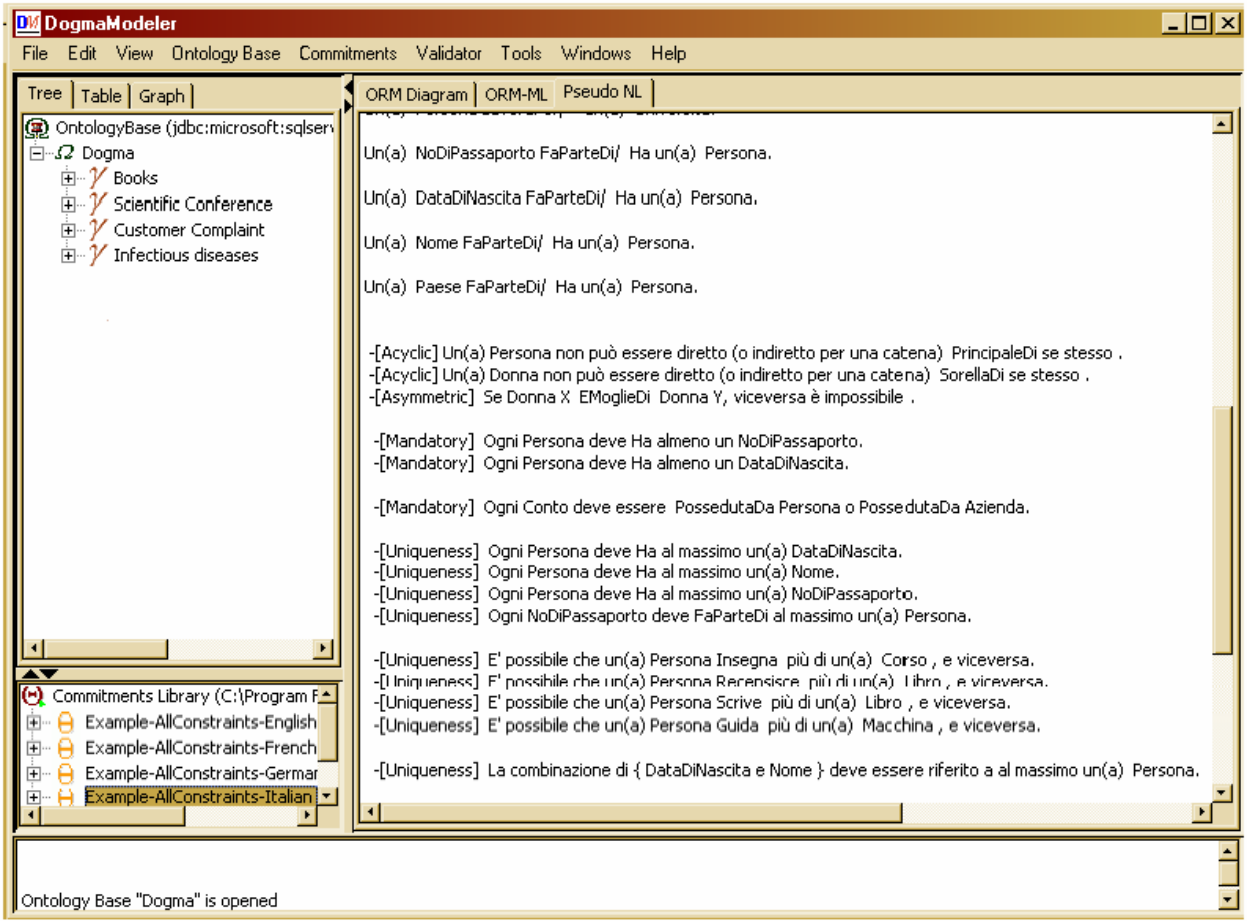


Fig. 4. Pseudo NL window with a section of the Italian verbalization of the ORM model in Fig. 2.

3.2 Verbalization templates

Each language has its own verbalization template, but all adhere to a meta-level structure of the particular type of constraint to generate the verbalization. For instance, to verbalize a mandatory constraint, one needs two objects and the role connecting the two as base ingredients, then text to include the modal verb "must" (respectively its equivalent in the desired language) that refers to the mandatory participation of the relevant object, which subsequently is appended with any other text to make the pseudo-natural language a human understandable sentence. Fig.5 shows the encodings for the mandatory constraint in English and Arabic, and Fig.6 the exclusive constrain with an example from the model depicted in Fig.2. With the subset constraint, one has e.g. ...that Person... in English whereas in Italian ...il/lo/la Persona in questione... is the preferred reading while maintaining the same meaning. Such grammatical differences are adjusted for each language in the template file by varying the text in the text tags and their position (like before or after the object or role). Hence, to create a new verbalization template for one's desired language one may change:

1) The text in-between the text tags <Text>…</Text>, where the text in the desired language goes,
2) Adding or removing text tags to make the sentence closer to natural language,
3) The order of the text, object, and role tags, provided the semantics of the constraint is preserved.

```
<Constraint xsi:type="Mandatory">
 <Text> -[Mandatory]  Each</Text>
 <Object index="0"/>
 <Text>must</Text>
 <Role index="0"/>
 <Text>at least one</Text>
 <Object index="1"/>
</Constraint>
```

Each ObjectA **must** roleR **at least one** ObjectB

```
<Constraint xsi:type="Mandatory">
 <Text> كل </Text>
 <Object index="0"  />
 <Role index="0"  />
 <Object index="1"  />
 <Text> واحد على الاقل</Text>
</Constraint>
```

كل ObjectA roleR ObjectB واحد على الاقل

Fig. 5. English and Arabic verbalization templates for the Mandatory constraint.

```
<Constraint xsi:type="Exclusive">
<Text> -[Exclusive]  Each</Text>
<Object index="0"/>
<Text>should be either</Text>
<Object index="1"/>
<Loop index="1">
  <Text>or</Text>
  <Object index="n"/>
</Loop>
</Constraint>
```

```
<Constraint xsi:type="Exclusive" >
<Text> -[Exclusive] Jeder/s</Text>
<Object index="0"/>
<Text> kann entweder ein </Text>
<Object index="1"/>
<Loop index="1">
<Text>oder ein</Text>
<Object index="n" />
</Loop>
<Text>sein</Text>
</Constraint>
```

Each Person **should be either** Woman **or** Man. **Jede/r/s** Person **kann entweder ein/e** Frau **oder ein/e** Mann **sein**.

Fig. 6. Example of the Exclusive constraint verbalization templates for English and German, and the resulting verbalizations for the Scientific Conference example model.

```
<Constraint xsi:type="Subset">
 <Text> -[Subset]  If</Text>
 <Object index="0"/>
 <Role index="child"/>
 <Object index="child"/>
 <Text>then this</Text>
 <Object index="0"/>
 <Role index="parent"/>
 <Object index="parent"/>
</Constraint>
```

```
<Constraint xsi:type="Subset">
 <Text> -[Subset]  Se un(a)</Text>
 <Object index="0"/>
 <Role index="child"/>
 <Text>un(a)</Text>
 <Object index="child"/>
 <Text>, en tal caso esto(a)</Text>
 <Object index="0"/>
 <Role index="parent"/>
 <Text>un(a)</Text>
 <Object index="parent"/>
</Constraint>
```

Fig. 7. English and Spanish verbalization template for the ORM Subset constraint.

To structure and simplify this process, it is advisable to take as template for a translation a verbalization template of a language that is grammatically closely related. This is the 'path of least amount of changes' that have to be applied. For instance, both German and Dutch are known to have the infinitive form of verbs at the end of a sentence where the verb phrase includes also a modal verb at its head, but not English (see the Exclusive constraint in Fig. 6). Therefore, creating the German verbalization template was done based on the Dutch version instead of reinventing the wheel to achieve the optimal sequence of objects, roles, and text tags, because then a word-by-word translation sufficed compared to starting from scratch. (Of course, we could have started with German and base the Dutch version on the German one; the choice was made arbitrarily). Analogously, the verbalization templates for the Roman languages group required only one 'base' translation with a detailed analysis of Italian grammar, and subsequently the verbalization templates for Spanish, Catalan, and French were based successfully on this Italian template file. A suggested approach to create verbalization templates for the Slavic languages is to take the existing Russian verbalization template, make a translation to Bulgarian and subsequently to e.g. Czech and Polish.

As third example, given the verbalization template of the Subset constraint in Fig. 7 and the running example, the verbalization of the constraint between driving a car and having a driving license reads in the currently supported languages as follows

English: If a Person Drives a Car then this Person must be AuthorizedWith a DrivingLicense.
Dutch: Als een Persoon RijdtMet een Wagen dan moet ook dit/deze Persoon BeschiktOver een Rijbewijs.
German: Wenn ein/e Person ein Auto Fahrt, dann muss diese Person auch ein/e/en Führerschein Hat.
Italian: Se un(a) Persona Guida un(a) Macchina, in questo caso il/lo/la Persona in questione Ha un(a) Patente.
Spanish: Se un(a) Persona Conduce un(a) Coche, en tal caso esto(a) Persona A un(a) PermisoDeConducir.
Catalan: Si Persona Condueix Cotxe llavors aquest(a) Persona Té PermísDeConducir.
French: Si un(e) Personne Conduit un(e) Voiture, alors ce(tte) Personne A un(e) PermisDeConduire.
Lithuanian: Jei Asmuo Vairuoja Automobilis, tai Asmuo Autorizuotas VairuotojoPažymėjimas.
Russian: Если Человек Водит Автомобиль , то Человек Авторизирован ВодительскиеПрава.
Arabic: اذا انسان يقود سيارة فان هذا الانسان مخول ب رخصة سياقة

The complete verbalization templates of all 22 ORM constraints are illustrated with the running example for each of the 10 languages and presented in separate technical reports available online at http://www.starlab. vub.ac.be/staff/mustafa/orm/verbalization/.

3.3 Application scenario options for verbalization

Depending on the type of user and intended use of verbalization – e.g. to verbalize a section of an ontology, database constraint, derivation, or business rule – one may prefer one fixed-syntax structure over another as preferred reading of the constraints in the model. We illustrate this for modality, which refers to the way a verbalized statement express action, necessity, or (im)possibility. The axiom, taken from the running example,

$$\forall x,y\ (\text{Manager}(x) \wedge \text{Company}(y) \wedge (\text{Manages}(x,y) \rightarrow \text{WorksFor}(x,y))$$

can be verbalised as:

- It is mandatory for a Manager who Manages a Company to also WorksFor that Company.
- If a Manager who Manages a Company is not a Person who WorksFor that Company, then ...
- If a Manager Manages a Company, then this Manager is a Person who WorksFor that Company.
- Each Manager who Manages a Company must WorksFor that Company.

Although the above statements verbalize the same formal axiom differently, such as also using the supertype of Manager, the difference between them is not merely syntactic or linguistic. The difference between their modality depends on the application and reasoning scenarios, i.e. whether the verbalization expresses an integrity constraint, derivation or inference rule, business rule or policy rule. If the above axiom is used to represent a constraint that (eventually) will be used by an ontology-inspired database management system, it will be seen as integrity constraint with a mandatory modality like the first sentence. If the same rule will be executed by a business rule engine, a certain action could be implemented in case the rule is violated (the second sentence). On the other hand, if it is a derivation/inference rule executed by a DL reasoner like FaCT or RACER, the reasoner will retrieve all managers of a company as its employees (sentence 3). Depending on the application and reasoning scenario, the verbalization template can be adapted to suit its preferred modality, or a separate template could be released for each scenario in addition to the present default verbalization. The main feature of the template-approach presented in the paper is that it is *scenario-independent*, hence can be adapted in the same manner as demonstrated for the language-specific adjustments in section 3.2 and suit different preferences and requirements from various domain experts.

3.4 Experimental results

We have tested the automatically generated verbalizations with about 40 lawyers during the development of the Customer Complaint management ontology[2], which had to facilitate cross-language communication to support cross-border e-commerce. The subject domain covers a semantic description of complaints that could be issued by any legal person against any other legal person (NGO, company, natural person, etc.) and several of the salient characterizations involve complaint problems and resolutions, rules of complaint and so forth, totaling to about 220 concepts, their definitions, and 300 relations between them. The monolingual ontology was lexicalized in multiple languages, thereby capable of providing an interface in a user's preferred language and maximizing both understandability and consequently its usability. [6]. Further, our experience during building the Customer Complaint ontology in cooperation with many domain experts is that verbalizations greatly assists the experts unfamiliar with ontologies in building and validating axiomatizations. It is indeed an easily understood language for domain experts,

[2] The CContology was developed as part of the CCFORM thematic-network project IST-2001-34908.

especially those who are not trained to understand technical or formal languages. This is applicable across subject domains: quite distinct from law and e-commerce is biology, where verbalization enables verification of semantics encoded in an ORM model as well and can be used to improve both formal bio-ontologies and the process to formalise them. Domain experts can check themselves that the model indeed captures their knowledge accurately, resulting in a better representation an increased level of trust that there is a mutual understanding between domain expert and modeler about the subject matter; an example is shown in Fig. 8, where a domain expert can check the facts and verbalized constraints one sentence at a time[3] [10].

4. Related work and Discussion

Although the pseudo-natural language verbalization template is not a formal language, the templates are unambiguous and well structured. However, natural language sentences are more flexible than what is currently captured with the fixed-syntax sentences in the verbalization template files. To make the verbalized sentences grammatically correct in any natural language, a full machinery with an automated morphological analysis for each language is required. This is an active research area in the natural language generation (NLG) research community. Reiter and Dale [11] and give a clear thorough description of how NLG systems are built, while a complete and up-to-date list of existing systems can be found in [12]. Hovy [13] distinguishes in particular four main approaches for text generation: *canned text*, *template*, *phrase-based*, and *feature-based*, sorted by increasing complexity and decreasing suitability for real-time applications. Canned texts have been used for a long time in every kind of application to generate messages for users, and obviously for the present context this approach is not applicable.

- **[Mandatory]** **Each** Bacteriocin **must** ProducedBy **at least one** Bacterium.
- **[Mandatory]** **Each** Bacteriocin **must** Inhibits **at least one** Bacterium.
- **[Uniqueness]** **Each** Bacterium **must** Produces **at most one** Bacteriocin.
- **[Uniqueness]** **It is possible that** Bacterium InhibitedBy **more than one** Bacteriocin , **and vice versa**.
- **[Exclusion]** **No** Bacterium Produces Bacteriocin **and also** InhibitedBy **that** Bacteriocin.

Fig. 8. Exclusion, Mandatory and Uniqueness constraints for the two facts, and the automatically generated verbalization that easily can be verified by domain experts.

As explained above, our template approach as implemented in DogmaModeler ensures the creation of a fixed-syntax sentence structure, which contains a main clause, optionally a subordinate clause, and/or one or more coordinate clauses. What it lacks, however, is the correct inflection of words according to person, number, gender, case, and in particular for verbs, their tense and mode. This gap can be filled going a step further, by means of a phrase-based realization where a small set of grammatical rules are set-up to drive the correct morphological choice of phrase entities that need to be filled in the templates. A first analysis step can be

[3] During validation, it appeared that the 1:n uniqueness constraint between Bacterium and Bacteriocin might need to be relaxed to m:n in the future. Thus far, no bacteria have been found to produce more than one type of bacteriocin, but with the current state of knowledge about bacteriocins, this is theoretically not impossible (i.e.: there is no known, confirmed, reason why it cannot happen).

lexicalization, where each term (concept/entity label) of an object is looked up in a computer-usable dictionary to determine the gender of the word and, at run-time, generates the correct article for the term (el Coche, das Buch, la Persona, etc.). For German this can be used in conjunction with a string analysis of the role name to check for included prepositions, such as ArbeitetFür, that change the declination of the article. More challenging is role analysis concerning verb use. With the uniqueness constraint

Each Person must Has at most one BirthDate.

the Has is taken directly from the ORM model (more precisely: its ORM-ML document), but a grammatically correct sentence should have the software automatically change the 3rd person singular Has into the infinitive Have. For languages grammatically richer than English, like Italian and Spanish, it requires a careful conjugation of verbs to determine e.g. that in the uniqueness constraint

E' possible che un(a) Persona Insegna più di un Corso, e viceversa.
("It is possible that a Person Teaches more than one Course, and vice versa")

the present indicative Insegna should be substituted for the subjunctive Insegni in order to have a grammatically correct sentence. On the other hand, software implementations of morphological analyzers to remedy these mild imperfections of the verbalizations for the languages DogmaModeler supports will take considerable more investigation by (computational) linguists. Considering the experimental results, the engineering solution with verbalization templates has proven to be an effective, and by domain experts much appreciated, feature in the conceptual analysis stage.

Comparing the multilingual verbalization approach with other solutions, to our knowledge, no extensive multilingual support for verbalizing logical theories and conceptual models exist. An attempt is described in [14] under the name ModelExplainer, which is a feature-based system (see distinctions above) able to generate textual summaries of object-oriented data modeling (OODM) diagrams, using information from data modeling tools such as Rational Rose and Visio. This solution uses a full-fledged generation grammar and a lexicon only for English. Another system that is noteworthy is YAG [15], an efficient real-time text-realization system which employs an approach that lies between template-based and phrase-based generators. On the multilingual side, among the existing feature-based systems available, KPML [16] is worth mentioning. Derived from the previous Penman NLG system [17], it covers the whole natural language generation process starting from discourse planning to text realization by using, among other linguistic resources, a generation grammar which is implementation of systemic-functional linguistics theories [18]. At the time of writing, there are grammars available for English (Nigel grammar), Czech and Spanish; the ones for Bulgarian, Chinese, Dutch, French, German, Greek, Japanese, and Russian are still under construction. KPML, and feature-based generation systems in general, would be too complex for the kind of verbalizations needed by DogmaModeler and they are not well suited in terms of processing speed and efforts needed to create the linguistic resources for many different languages. KPML is being used instead in other research efforts, as e.g. in [19] where the authors describe how they are leveraging the high quality generative features of KPML and the results achieved to date by applied Systemic-Functional Linguistics to enhance the user-friendliness of an intelligent query interface.

The use of a phrase-based generation approach is still the best choice even if we consider that Halpin's [20] implementation of verbalization in the NORMA tool (under development) might be more sophisticated than the here introduced template solution with respect to verbalization of complex facts, i.e. to string together facts into longer pseudo-natural language sentences (see [3]), than is supported now with DogmaModeler. However, the software implementation is not as

flexible and extensible as the verbalization templates and limits itself to the English language only. If possible, using the best of both approaches will benefit domain experts and eases the modelers' effort to build good conceptual models and ontologies. A new initiative on the verbalization of formal languages has been started recently within the REWERSE Project (IST-2004-506779, EU FP6 Network of Excellence). They, e.g. Wagner et al [21], also use the template-approach for verbalization and have some initial examples for rules verbalization, but is limited to English and focused on the (officially) not formalized UML, which is a less expressive language than ORM.

The multilingual verbalization solution described here has been developed for ORM in general and implemented in DogmaModeler in particular, but its underlying principle is equally usable, and useful, for other axiomatizations and logical theories. For instance, the ORM Exclusive and Totality constraints for Person, Man, and Woman in the running example corresponds to the completeness and disjointness constraint of the *Phone* hierarchy with its two subtypes *CellPhone* and *FixedPhone* in [22]. Berardi *et al*. [22] present its representation in UML, FOL, and the DL version DLR_{ifd} (DLR_{ifd} supports n-ary *R*elations, *i*dentification constraints, and *f*unctional *d*ependency constraints) as depicted in Fig. 9-A. In order to communicate this knowledge quickly and effectively with non-technical and non-formal domain experts, one should be able to generate a run-time verbalization of the represented knowledge. This we present in Fig. 9-B with the English verbalization template according to our methodology. One can add the here introduced methodology for multilingual verbalization equally effectively to ontology development tools like Protégé [23] and iCOM [24].

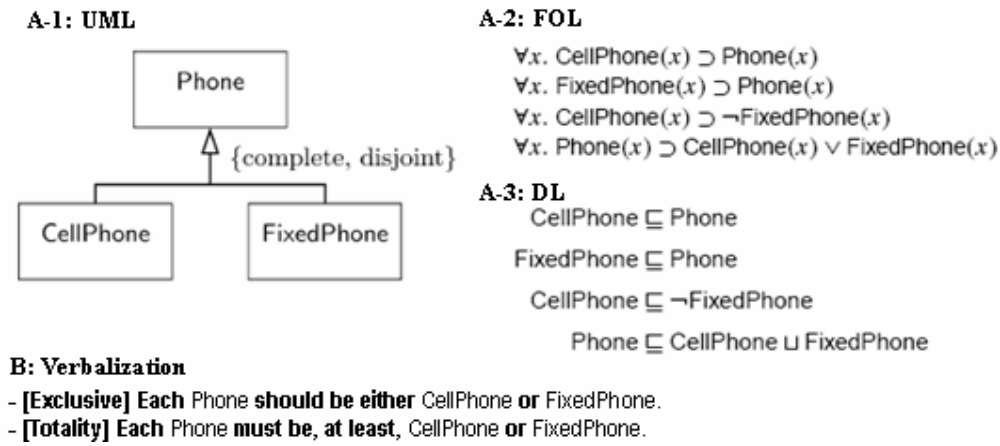


Fig. 9. A: the Phone hierarchy in UML, the Description Logic language DLR_{ifd}, and FOL (source: [22]); B: its corresponding verbalization in English.

5. Conclusions

We have developed and described a novel approach to for automated multilingual verbalization of logical theories, axiomatizations, and other specifications such as business rules. This engineering solution was demonstrated with Object Role Modeling and the DogmaModeler ontology engineering application. The easy-to-customize verbalization template is at run-time applied to an in ORM-ML encoded ORM model or ontology commitment layer to generate the verbalization in one of the supported languages. We have created templates for English, Dutch, German, Arabic, Russian, Italian, Spanish, Catalan, French, and Lithuanian. The approach and implementation for multilingual verbalization is characterized by its flexibility, extensibility and maintainability of the verbalization templates, which allow for easy augmentation with other languages within

DogmaModeler, as well as use of the same methodology for verbalizing other types of representation of logical theories, such as Description Logics, to improve understandability and usability by domain experts.

Future work includes the implementation of lightweight morphological functions as those included in the YAG System [15] or more sophisticated morphological realization constraints as the ones used by the KPML generation module [16]. Furthermore, we will investigate the verbalization of longer sentences, more complex fact types and investigation into its options with the DL-based iCOM tool that already has a DL reasoner and graphical support for ontology modelling.

Acknowledgements. Our special thanks to Andriy Lisovoy who helped in the implementation of DogmaModeler, and to Hai Nguyen Hoang who helped in the first implementation of the verbalization component during his Master thesis. We are grateful to Andriy Lisovoy, Núria Casellas, Juozas Gordevicius, and Pieter Verheyden for their help in translating the Russian, Spanish, Lithuanian, and Dutch translations.

Technical reports in the supported languages that contain an example ORM model with all constraints, their verbalization, and the verbalization template file are available online from http://www.starlab.vub.ac.be/staff/mustafa/orm/verbalization/.

References

[1] Halpin, T.A. Information modeling and relational databases. (3rd ed.). San Francisco: Morgan Kaufmann Publishers, 2001.

[2] Halpin, T.A. *A logical analysis of information systems: static aspects of the data-oriented perspective*. PhD Thesis, University of Queensland, Australia. 1989.

[3] Halpin, T..A. Business Rule Verbalization. *3rd International Conference on Information Systems Technologies and Applications (ISTA 2004)*. Salt lake City, USA, 15-17 July 2004.

[4] Demey, J., Jarrar, M., Meersman, R. A Conceptual Markup Language that supports interoperability between Business Rule modeling systems. In: *Proceedings of the Tenth International Conference on Cooperative Information Systems (CoopIS 02)*. Springer Verlag LNCS 2519. (2002) pp. 19–35

[5] Jarrar, M., Demy, J., Meersman, R. "On Using Conceptual Data Modeling for Ontology Engineering". *Journal on Data Semantics: Special issue on Best papers from the ER/ODBASE/COOPIS 2002 Conferences*, 2003, 1(1): 185-207.

[6] Jarrar, M. *Towards methodological principles for ontology engineering*. PhD Thesis, STARLab, Vrije Universiteit Brussel, Belgium. 2005.

[7] Bird, L., Goodchild, A., Halpin, T.A. Object Role Modelling and XML-Schema. In: *Proceedings of the 19th International Conference on Conceptual Modeling (ER'00)*, Laender, A., Liddle, S., Storey, V. (eds.). LNCS, Springer Verlag. 1999.

[8] Jarrar, M. Modularization and automatic composition of Object-Role Modeling (ORM) Schemes. In: Halpin, T., Meersman, R. (eds): *Proceeding of the International Workshop on Object-Role Modeling (ORM'05)*, OTM 2005 Workshops. LNCS 3762, pp613-625.

[9] Jarrar, M. Towards the notion of gloss, and reusing linguistic recourses in ontology engineering. *3rd Int. Wordnet Conference*, Jeju Island, Korea. 2006.

[10] Keet, C.M. "Biological Data and Conceptual Modelling Methods". *Journal of Conceptual Modeling*, Issue 29, October 2003. http://www.inconcept.com/jcm.

[11] Reiter, E., Dale, R. *Building Natural-Language Generation Systems*. Cambridge University Press, 2000.

[12] Bateman, J. and M. Zock's List of Natural Language Generation Systems, http://www.fb10.uni-bremen.de /anglistik/langpro/NLG-table/NLG-table-root.htm.

[13] Hovy, E. Language Generation: Overview (Ch. 4.1). In: *Survey of the State of the Art in Human Language Technology, Studies in Natural Language Processing*. Cambridge University Press, pp139-146. 1997.

[14] Lavoie, B., Rambow, O., Reiter, E. Customizable descriptions of object-oriented models. In: *Proceedings of the Fifth Conference on Applied Natural-Language Processing (ANLP-1997)*, 1997, pp253-256.

[15] McRoy, S.W., Channarukul, S., and Ali, S.S. Text realization for dialog. In: *Proceedings of the Int. Conf. on Intelligent Technologies*, Bangkok, 2000.
[16] Bateman, J.A. Enabling technology for multilingual natural language generation: the KPML development environment. *Journal of Natural Language Engineering*, 1997, 3(1):15-55.
[17] Mann, W.C. An overview of the Penman text generation system. In: *Proceedings of the Third National Conference on Artificial Intelligence* (Washington, August 22-26, 1983), pp261-265.
[18] Halliday, M.A.K., Matthiessen, C.M. *An Introduction to Functional Grammar*. London: Edward Arnold, 3rd ed.. 2004.
[19] Dongilli, P., Franconi, E. An Intelligent Query Interface with Natural Language Support. In: *Proceedings of the 19th International FLAIRS Conference FLAIRS-2006*, Melbourne Beach, Florida, May 2006.
[20] Halpin, T.A. ORM 2. *International Workshop on Object-Role Modeling (ORM'05).* In: *OTM Workshops 2005*. Halpin, T., Meersman, R. (eds.), LNCS 3762. Berlin: Springer-Verlag. pp676-687. 2005.
[21] Wagner, G., Lukichev, S., Fuchs, N.E., Spreeuwenberg, S. *First-Version Controlled English Rule Language*. REWERSE Deliverable IST506779/Eindhoven/I1-D2/D/PU/b1. 47p.
[22] Berardi, D., Calvanese, D., De Giacomo, G. "Reasoning on UML class diagrams". *Artificial Intelligence*, 2005, 168(1-2): 70-118.
[23] Protégé. http://protege.stanford.edu/.
[24] Franconi, E., Ng, G. The iCom tool for intelligent conceptual modelling. *7th Intl. Workshop on Knowledge Representation meets Databases (KRDB'00)*, Berlin, Germany. 2000.