GRANULAR COMPUTING APPLIED TO ONTOLOGIES

SILVIA CALEGARI AND DAVIDE CIUCCI

Abstract. Granular Computing is an emerging conceptual and computing paradigm of information processing. A central notion is an information-processing pyramid with different levels of clarifications. Each level is usually represented by 'chunks' of data or granules, also known as information granules. Rough Set Theory is one of the most widely used methodologies for handling or defining granules.
Ontologies are used to represent the knowledge of a domain for specific applications. A challenge is to define semantic knowledge at different levels of human-depending detail.

In this paper we prose four operations in order to have several granular perspectives for a specific ontological commitment. Then these operations are used to have various views of an ontology built with a rough-set approach. In particular, a rough methodology is introduced to construct a specific granular view of an ontology.
Keywords: granular computing, granular information, ontologies, description logic, rough set theory, ontologies editor.

1. Introduction

In the last decade the need to share different kinds of knowledge and/or information among different applications increased the interest in research on ontology. This term was originally used in Philosophy, where it indicated the systematic explanation of Existence. In Computer Science the term has been used in various areas, such as knowledge engineering, knowledge representation, qualitative modelling, database design, language engineering, information integration, information retrieval and extraction, knowledge management and organisation, agent-based system, e-commerce design [Guarino(1998)], and recently in computing with words [Reformat and Ly(2009)]. Ontologies also play a key role in one of the newest and heavily studied areas: the Semantic Web.

With the support of ontologies, users and systems can communicate with each other through an easy information integration [Soo and Lin(2001)]. Ontologies help people and machines to communicate concisely by supporting information exchange based on semantics rather than just syntax. They provide a semantic structure for sharing concepts across different applications in an unambiguous way. In the Semantic Web an ontology is defined as *a formal conceptualisation of a particular domain of interest shared among heterogeneous applications. It consists of entities, attributes, relationships and axioms* to provide a common understanding of the real world [Lammari and Mtais(2004), Gruber(1993),

TO APPEAR IN INTERNATIONAL JOURNAL OF APPROXIMATE REASONING - DOI:10.1016/J.IJAR.2009.11.006

Guarino and Giaretta(1995)]. Thus, they are useful in solving one of the key issues in the development of the Semantic Web: to enable machines to exchange meaningful knowledge across heterogeneous applications to reach the users' goals.

Granular Computing (GC) is a recent discipline, the name comes from [Lin(1997)], and it can be viewed as a unifying science of different fields of research which "will bring us closer to a new paradigm of human-inspired computing" [Yao(2008)]. The main concepts of GC are of course the one of *granule* and of multiple *levels of granularity*, which are closely linked together. A granule is a chunk of knowledge made of different objects "drawn together by indistinguishability, similarity, proximity or functionality" [Zadeh(2008)]. A level is just the collection of granules of similar nature.
According to the level of granularity taken into account, i.e., to the point of view, a granule "may be an element of another granule and is considered to be a part forming the other granule. It may also consist of a family of granules and is considered to be a whole" [Yao(2008)]. From one level we can pass to a lower level of granularity, i.e., decompose a whole into parts; this corresponds to "analytical thinking" whereas, going to an upper level merging parts into wholes, corresponds to "synthetic thinking" [Yao(2007)]. An example of activity which can be described by granular computing is structured writing [Yao(2007)]. Indeed, an article (or a book) can be viewed at different levels, for instance paragraph, section, chapter.

It is clear that a hierarchy of granules can be viewed as a taxonomy, and thus as a light ontology [Mizoguchi(2004)]. For instance, in [Qiu et al.(2007)] a hierarchy of this kind is built by an information table using rough-set techniques. What is more interesting is if in general, granular ideas can be of some help in the ontology field. What we are going to do is to give a positive answer to this question. Roughly speaking, we will outline four different operations inspired by granular computing which can be performed on an ontology and apply them to some cases of study. These operations have been developed in Protégé in order to obtain a granular perspective of an ontology in a semi-automatic way [Calegari and Ciucci(2009)].

2. Preliminary notions

In this Section, we give the basic notions we will encounter throughout the paper. At first, the modeling primitives used for building ontologies are reported. Then, rough sets and a granular interpretation of ontologies are defined.

2.1. **Ontologies.**

All ontology languages include the definition of modeling primitives, in particular, the conceptual primitives that can be used to formalise the knowledge about a domain. Usually they also include the formalism for representing algebraic properties of the relationships (e.g. transitive, symmetric, reflexive, inverse).

Some years ago, there was an important discussion to tackle the problem for finding a unique meaning of ontology. Indeed, the Computer Science community is divided among the researchers that insert *instances* into the logical structure that formalises the ontology and the ones that consider the *instances* outside the formal definition of ontology. Both these perspectives are correct because this disagreement is only due to the different use, i.e., to the specific application where the ontology has to be utilised. In this work we consider the first case and thus Gruber's formal definition of an ontology:

Definition 1. [Gruber(1993), Tamma(2001)] *An Ontology is defined as the quintuple:*

$$\mathbf{O} := \{I, \mathcal{C}, \mathcal{R}, \mathcal{F}, \mathcal{A}\}$$

where:

- I *is the set of individuals, that is the actual objects of the world. The individuals are also called* instances *of the concepts;*
- $\mathcal{C}$ *is the set of* concepts*, that is the set of the abstractions used to describe the objects of the world.* $\mathcal{E} = \mathcal{C} \cup I$ *is the set of the* entities *defined by the union of the set* C *and the set* I*;*
- $\mathcal{R}$ *is the set of relationships defined on the set* $\mathcal{E}$ *s.t. each* $r \in R$ *is an ordered n-ple* $r \subseteq \mathcal{E}^n$*;*
- $\mathcal{F}$ *is the set of functions defined on the set of concepts that return a concept. That is, each element* $f \in F$ *is a function* $f : \mathcal{E}^{n-1} \rightarrow \mathcal{E}$*;*
- $\mathcal{A}$ *is a set of* axioms*, that is first order logic predicates which constrain the meaning of concepts, relationships and functions.*

The concepts and instances are usually hierarchically organised. The most common and intuitive hierarchy is represented by IS-A relationship (this is also known in literature as a generalisation/specialisation or subsumption relationship). Generally, it is assumed that each subclass *inherits* data structure (e.g. instances) and behaviour (e.g. methods or properties) from the super-class, that is if A is an ancestor of B (denoted by $A \mapsto B$) and $B \mapsto C$ then, $A \mapsto C$.

Usually, ontologies are formally defined by a language from the Description Logic (DL) family. In particular, in the Semantic Web case, $\mathcal{SHOIN}(D)$ [Horrocks et al.(2003), Horrocks and Patel-Schneider(2004)] is used to give a semantic to OWL, the W3C (World Wide Web Consortium) [1] standard language. Here we are not interested in modifying the syntax and semantic of any DL, thus we do not enter into details about such a formalism. However, in Section 3, we will use some typical formalism of DL, which we refer to [Baader et al.(2003)].

2.2. **Rough Sets.** We now turn our attention to one of the most widely used approaches to granular computing: rough sets. The rough-set techniques can be used with two different aims, to handle uncertainty (by defining a lower and an upper approximation of a set) and to granulate information (usually, by means of binary relations). In the present work we only deal with this second aspect and we use equivalence and similarity relations to build granules from instances of an ontology. Typically in rough-set applications a granule is obtained by looking at the properties of objects: two objects belong to the same granule if they assume the same value for all the properties under investigation.

The structure used to deal with these ideas is known as Information Table or Information System [Pawlak(1981)].

Definition 2. *An* Information Table *is a structure* $\mathcal{K}(X) = \langle X, Att(X), val(X), F \rangle$ *where: the universe* X *is a non empty set of* objects*;* $Att(X)$ *is a non empty set of* attributes*;* $val(X)$ *is the set of all* possible values *that can be observed for an attribute a from* $Att(X)$ *in the case of an object x from* X*;* F *(called the* information map*) is a mapping* $F : X \times Att(X) \rightarrow val(X)$ *which associates to*

[1] http://www.w3.org/

any pair, consisting of an object $x \in X$ and of an attribute $a \in Att(X)$, the value $F(x, a) \in val(X)$ assumed by a for the object x.

Given a set of attributes $D \in Att(X)$, two objects $x, y \in X$ are called *indiscernible* with respect to D, and we write xI_Dy, iff $\forall a \in D$, $F(a, x) = F(a, y)$.

It can be easily verified that I_D is an equivalence relation and so it partitions the universe X in disjoint classes (granules) $E_D(x)$ defined as $E_D(x) := \{y \in X : xI_Dy\}$.

A generalisation of this classical (Pawlak) approach is obtained by relaxing the requirement that the indiscernibility relation I_D of the Information System is an equivalence relation. The simplest of this generalisation makes it necessary to drop the transitivity condition. The result is a similarity (or tolerance), i.e., reflexive and symmetric, relation [Skowron and Stepaniuk(1996)]. The notion of granule is defined exactly as above but now, a granule contains the objects which are similar (and not necessarily equivalent) to a given one.

In our case study we will use a similarity relation obtained by a weakening of the usual indiscernibility one, which considers equivalent two objects if for all selected attributes they have the same values. If we relax this requirement, we can say that two objects are similar if they have the same values only for some attributes. Formally, let $D \subseteq Att(X)$, then x is similar to y with respect to D and ϵ, with $\epsilon \in [0, 1]$, and write $x\mathcal{R}_{D,\epsilon}y$, iff

$$\frac{|\{a_i \in D : F(x, a_i) = F(y, a_i)\}|}{|D|} \geq \epsilon \tag{2.1}$$

This relation says that two objects are similar if they have at least $\epsilon|D|$ attributes with the same value. It can be easily shown that this is a reflexive and symmetric relation, but non a transitive one. Let us note that this approach is very close to the original one of Poincaré in his introduction to *similarity* relations just by "distances" [Poincaré(1893)] where in the present case, the distance is $d(x, y) := |\{a_i \in D : F(x, a_i) = F(y, a_i)\}|$. This metric has been also considered in [Polkowski(2007), Polkowski(2008)] and used to define a rough inclusion function.

2.3. **Granular interpretations of ontologies.** Granularity is the act of representing and operating the information at different levels of detail. "*Granularity deals with organising data, information, and knowledge in greater or lesser detail that resides in a granular level or level of granularity and which is granulated according to certain criteria, which thereby give a perspective also called view, context, or dimension on the subject domain, $\mathcal{D}_G$, henceforth called* granular perspective" [Keet(2008)]. Following an ontology granular perspective, a granular level $\mathcal{L}_G$ contains one or more entities (here $\mathcal{E}$, see Section 2.1), that is, representations of concepts and their instances. However, the ideas about what granularity comprises can differ among research disciplines which tend to consider or emphasise diverse aspects of the same entity. Thus, several interpretations of granularity and graphical representations capturing differences in interpretation, representation, and/or emphasis can be obtained. Keet [Keet(2008)] defines a classification of these possibilities:

i Emphasis on entity types and their instances
ii Emphasis on relation between entities and levels
iii Emphasis on the perspective and criteria for granulation

iv Emphasis on formal representation

In this work, points (i) and (ii) are taken into account. The tree structure is chosen when it is necessary to define several levels. In this way the graphical representation is more readable than the circle one generally introduced for granular information. Figure 1 depicts the graphic structure taken into account. The circles are the concepts, and the squares are the instances. The numbers beside the tree indicate the corresponding granular level $\mathcal{L}_G$ where each concept and instance are collocated. The multi-classification has been studied by Keet [Keet(2008), Sec 2.3.2]. However, in our case we can have instances belonging to different granular levels and for a same $\mathcal{L}_G$ we can have different types of instances. In the general case, the rule to

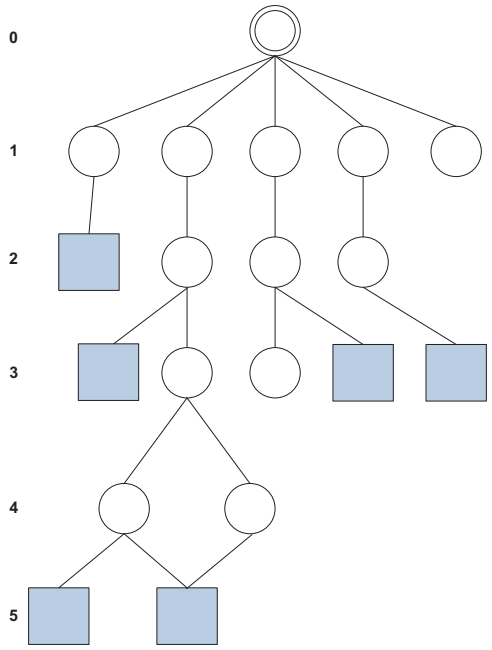


FIGURE 1. Granular classification considered.

select which are the relevant relations for granularity is not specified, both regarding the relation among the entities in different levels and regarding how granular levels relate to each other. In our specific case, let us suppose to have an ontological commitment where each branch of the tree defines a taxonomic relation. In this way, a layer in the tree structure with the same depth corresponds to a granular level $\mathcal{L}_G$. Thus, an ontology can be reformulated in granular levels by using a tree structure. In this approach, we define a semi-automatic methodology in order to obtain an ontological granular perspective of an existing ontology. To this aim, the instances are disposed in a new classification considering the new granules. In this perspective, it is very difficult to assign automatically a semantic relation among all the granular levels (e.g., the IS-A relation is not the unique possible relation). At the moment, our choice is not to investigate this problem and to insert *unknown* as label. Once obtained a granular perspective, the expert of domain will give the right semantic labels to each relation in order to change the *unknown* relation with *IS-A, part-of, instance-of, ...*

3. Granular operations

In this Section the ontology operations inspired by granular computing are presented. They can be considered as the basis of more complex algorithms, and we will show some simple examples in Section 4.2.

We distinguish two operations for lowering the level of granularity: *elimination and generalisation*, and two for rising it: *splitting and refinement*. They can be considered as pairs of inverse operations: elimination is the inverse of refinement and generalisation the inverse of splitting.

We will describe these operations with respect to a concept of an ontology C called the *focus* concept. In the examples the focus concept will be *feline*.

3.1. **Generalisation.** A generalisation consists in grouping together a set of different concepts C_i in a unique one C. The criterion adopted to grouping the concepts C_i is not specified here: it can depend on the context or on the application. For example, we can use rough sets or some clustering method, see for instance [Polkowski and Skowron(2001), Kreinovich(2008), Roychowdhury(2008), Lingras et al.(2008)]. Figure 2 represents a schematic view of this operation. All the subconcepts $C_1 \dots C_m$ of A, or a part of them $C_1 \dots C_n$, are collapsed in a unique concept C.

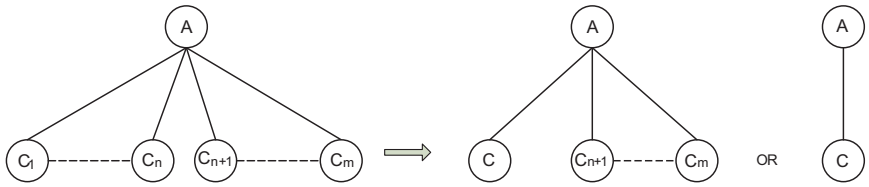
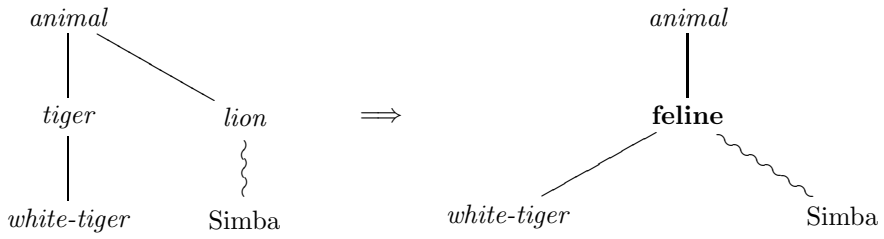


Figure 2. First Operation: generalisation.

Example 1. *As an example let us consider a partial ontology of animals. Here and in the following diagrams, a straight line represents an IS-A relation and a curve represents an INSTANCE-OF relation.*



The correspondence of concept C of the general schema is feline, which is obtained as the union of tiger and lion.

Let us note that in the above example, the subconcept white-tiger and the instance Simba are retained in the new ontology. Indeed, particular attention must be given to subconcepts and instances. Our choice in the realisation of all the granular operations is to retain as much information as possible. Thus, the generalisation operation has the following effect on the ontology:

- All subconcepts of C_i become subconcepts of the focus C. In our example "white-tiger" becomes a subconcept of "feline".
- All (direct) instances of C_i become instances of C. In our example "Simba" becomes an instance of "feline".
- The common ancestor of all C_i becomes a super-concept of C. In the example, this role is played by the concept "animal".
- the properties shared by all concepts C_i are assigned to C, all other properties of C_i are assigned to all subconcepts of C_j if any, deleted otherwise.

Thus, if our ontology is expressed in DL and we generalise concepts C_i to C, the operations corresponding to the above points are respectively the following:

(1) Replace all the axioms in the T-Box of the form $D \sqsubseteq C_i$ with axioms $D \sqsubseteq C$;
(2) Replace all the axioms in the A-Box of the form $a : C_i$ with axioms $a : C$;
(3) Find the concept D such that for all C_i, there holds $C_i \sqsubseteq^* D$ and add the axiom $C \sqsubseteq D$.
(4) If for all C_i there are axioms of the form $C_i \sqsubseteq F$ in the T-Box, replace them with axioms $C \sqsubseteq F$. Replace the remaining axioms $C_i \sqsubseteq F$ with axioms $C'_i \sqsubseteq F$ where C'_i is a subconcept of C_i.
(5) Delete all other axioms involving concepts C_i.

Let us note that point 4 also includes the conservation of properties shared by all the subconcepts of C_i, since in DL they are expressed through axioms of this form. For instance $C_i \sqsubseteq \exists HasColor.Red$ denotes that C_i is Red, and if all C_i are Red then it makes sense that also C has this property. About the properties not shared by all the C_i, we decided to assign them to the subconcepts of C_i if any, delete them on the contrary. This prevents having a non-monotonic form of reasoning. We note that this is different from [Lammari and Metais(2004)] where, when merging concepts C_i in a concept C, all properties of all C_i are assigned to C and this fact is acknowledged to the user.

3.2. **Elimination.** The difference with generalisation is that the focus concept C already exists in the ontology. Thus, elimination consists in deleting some or all of the direct subconcepts of the focus concept C, while retaining all the instances, direct and indirect.

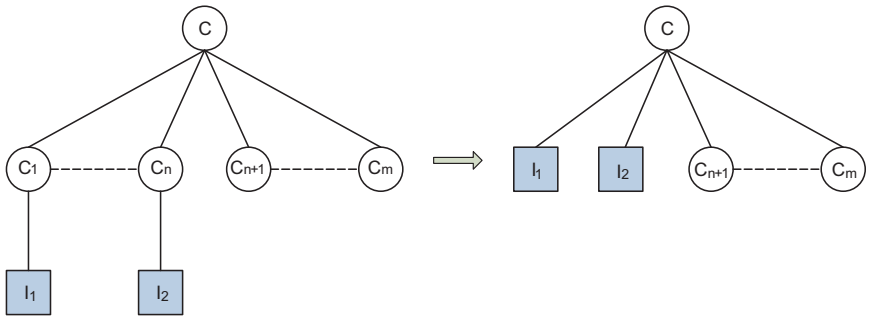
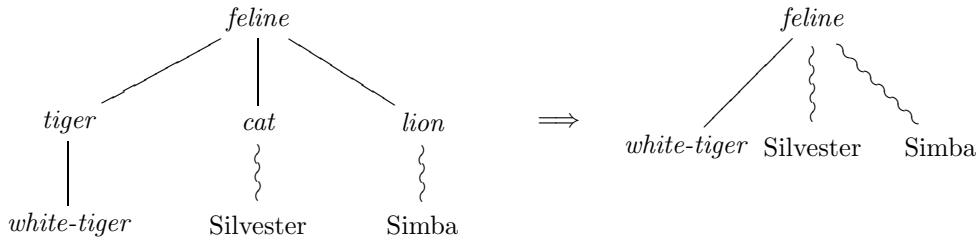


FIGURE 3. Second Operation: Elimination.

Example 2. *Considering our animal example, we can, for example, eliminate all the level below feline, that is the concepts tiger, cat and lion, while retaining all their instances and subconcepts.*



The operations to do are similar to the ones of generalisation except that we do not add a new concept. In particular, point (1), (2) and (5) are the same, point (3) is useless in this case, since the ancestor of all C_i is trivially C, and point (4) has to be changed in the following way.

4a. Compute the intersection of all C_i properties, call it P. If we are deleting only some and not all the subconcepts of C (as in Figure 3 with $C_1 \dots C_n$), delete from P the properties not shared by the remaining subconcepts of C ($C_{n+1} \dots C_m$ in Figure 3).
4b. Delete properties in P from all C_i and assign them to C.
4c. For any concept C_i, assign its (remaining) properties to all its subconcepts.

Thus, contrary to generalisation, only properties and non taxonomic axioms are considered. That is, in case of multiple inheritance we do not want to add taxonomic relations between concepts on different branches. For example, let us consider the following case: When eliminating C_1, C_2, we do not want to add the axiom $C \sqsubseteq F$.

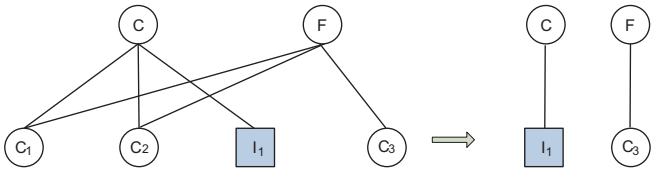


FIGURE 4. Elimination and multiple inheritance.

Moreover, in case not all the subconcepts of C are deleted (i.e., $C_1 \dots C_n$ in Figure 3) only the properties shared by all the subconcepts ($C_1 \dots C_m$) are assigned to C. Indeed, if some properties of $C_1 \dots C_n$ were assigned to C then they will be incorrectly inherited also by $C_{n+1} \dots C_m$.

Finally, we note that both generalisation and elimination have the following peculiarity: some or all the direct subconcepts of the focus concept are deleted but the instances are retained. Let us note that this is different from the usual behaviour implemented in ontology editors (for example, in Protégé) where it is not possible to delete a concept which has direct or indirect instances.

3.3. **Refinement.** The inverse of elimination is refinement, where the focus concept C is detailed adding new sub-concepts to it. The problem here is to pay attention to instances of C. We have two choices: leave them as instances of C or if some further knowledge about the domain described by the ontology is available, assign them to a proper subconcept. These two options are drawn in the diagram of Figure 5.

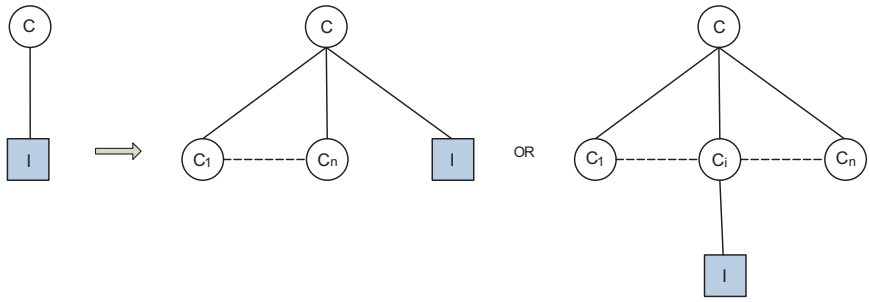
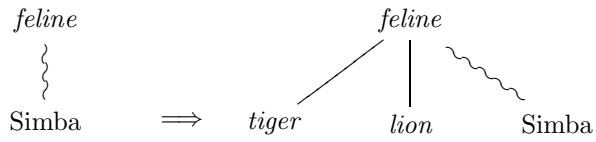
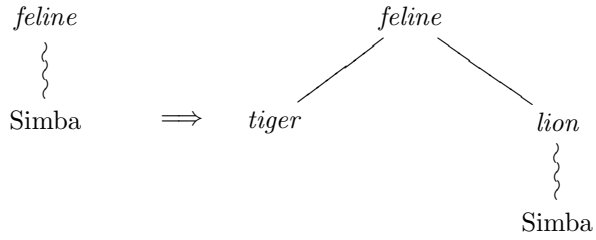


FIGURE 5. Third Operation: Refinement.

Example 3. *As an example of the two cases, we first consider the situation where no additional knowledge is available. Thus, when refining the concept feline, its instance Simba remains an instance of feline.*



Otherwise, if it is known that Simba is a lion the situation can be described by the following diagram.



The operations needed by a refinement are simple.

(1) Add the axioms corresponding to the fact that the new concepts C_i are subsets of the focus concept C: $C_i \sqsubseteq C$.
(2) If possible, re-assign the individuals a_j belonging to C to the proper (new) subconcepts C_i. This implies deleting the axioms $a_j : C$ in the A-Box and adding the axioms $a_j : C_i$.

Of course, the last point requires having some external knowledge which can be given by a domain expert, for instance.

3.4. **Splitting.** The difference with respect to refinements is that in the present case, the focus concept C is not retained, but substituted by more detailed concepts C_i (See Figure 6). Of course, here the problem of how to manage the subconcepts

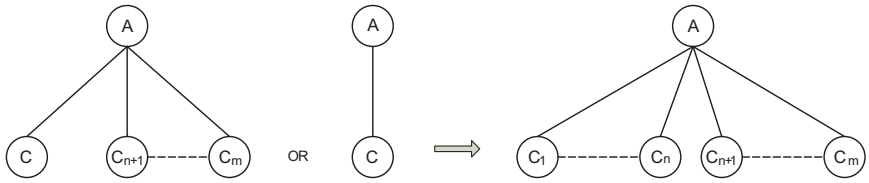
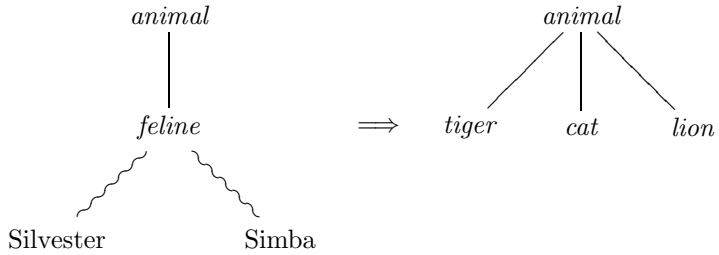


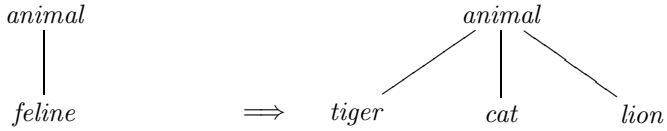
FIGURE 6. Fourth Operation: Splitting.

and instances of C is more serious. In this case we introduce the constraint that it is not possible to delete a concept having instances *unless* the instances are reassigned to one of the new introduced concepts.

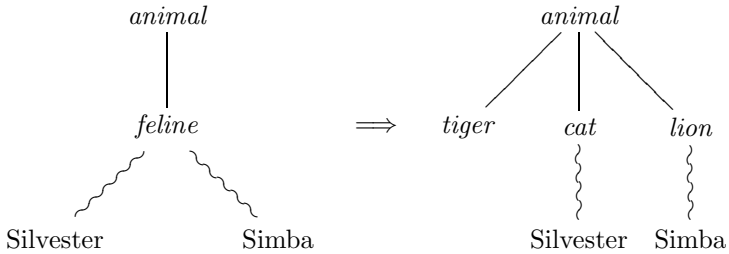
Example 4. *According to the previous constraint, it is not possible to perform a splitting like the following one, since the information about Silvester and Simba is lost.*



On the other hand, it is possible to split feline if it has no instances:



or, alternatively, if the instances are re-assigned to new concepts:

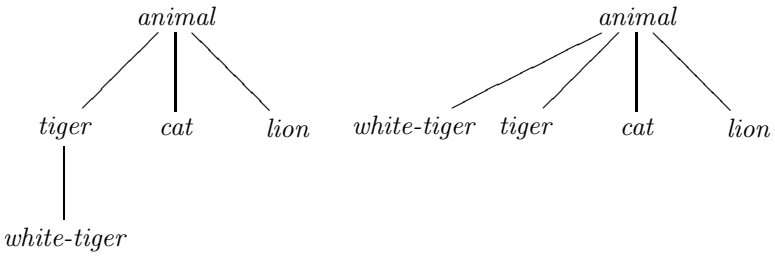


Of course, only the solution which respects the constraint is feasible in an automatic way, the other one requires the intervention of an expert.

In case the focus concept C has some subconcepts, they can either become subconcepts of a superconcept of C or be defined by the user as subconcepts of the new introduced ones.

Example 5. *In case of the following ontology*

a splitting of the concept feline can be performed in one of the following two ways



With respect to the knowledge base, to perform this operation in the simplest case where the focus C has no instances and no sub-concepts we have to:

(1) Delete axioms of the form $C \sqsubseteq D$ where D is any superconcept of the focus C and introduce axioms $C_i \sqsubseteq D$, for any new concept C_i;

If C also has some instances or sub-concepts, then, they can be managed as described above and the changes in DL-axioms are straightforward. Two further steps are needed:

2a. For any instance a_j of C find a proper concept C_i and substitute the axiom $a_j : C$ in the A-Box with $a_j : C_i$.

2b. For any subconcept C_j of C find a proper concept C_i and substitute the axiom $C_j \sqsubseteq C$ with $C_j \sqsubseteq C_i$ in the T-Box.

In any case as a last step we have to

(3) Delete all other axioms involving concept C.

4. Case Study

In this Section, we apply our operations to the "WINE ONTOLOGY" defined in the Protégé platform. By starting from the classical ontology and by using the rough set methodology of Section 2.2, a granular perspective of the ontology is built. A comparison between the classical ontology and the granular one is performed. Let us note that our methodology can be applied to any ontology, and we do not discuss how the expert has defined this ontological knowledge.

We remark that the four granular operations of Section 3 have been developed in an ad-hoc Protégé-OWL Editor plugin, thus allowing semi-automatic management of ontological knowledge [Calegari and Ciucci(2009)].

4.1. **Protégé-OWL Editor.** As previously stated, ontologies play a central role in the Semantic Web: they provide a formal model of the knowledge of a domain that can be exploited by intelligent agents.
A development tool for Semantic Web applications should provide services to access,

visualise, edit and use ontologies. Furthermore, a tool should include a *reasoner* in order to provide intelligent assistance during the definition and evolution of an ontology. It should also be customisable, extensible and scalable. A final issue for the Semantic Web development tools is that they should be *open source*.

Protégé [Noy et al.(2000), Noy et al.(2001), Gennari et al.(2003)] is a platform for ontology modelling and knowledge acquisition, and it is an open-source tool developed at Stanford Medical Informatics. Although the development of Protégé was historically driven by biomedical applications [Gennari et al.(2003)], this system is domain-independent and has been successfully used for many application areas like the Semantic Web. The architecture of Protégé consists of two parts, named "model" and "view". The Protégé model is the internal representation mechanism for ontologies and knowledge bases. The view components of Protégé provide a user interface to display and manipulate the underlying model. A *Protégé model* can represent ontologies consisting of classes, properties (slots), property characteristics (facets and constraints), and instances.

The Protégé-OWL editor [Horridge et al.(2004)] is an extension of the Protégé core system that supports OWL (in [Gennari et al.(2003)] details on the new architecture of the system are reported). In particular, the Protégé-OWL editor is the "OWL Plugin" [Knublauch et al.(2004a), Knublauch et al.(2004b)] that is a large Protégé plugin with support for OWL. It can be used to load and save OWL files in various formats, to edit OWL ontologies with custom-tailored graphical widgets, and to perform intelligent reasoning based on DLs. In particular, a reasoner called RACER (Renamed ABox and Concept Expression Reasoner) [7] is used. One of its main services is to test if a class is a subclass of another class, namely to calculate the *inferred* ontology class hierarchy.

The OWL Plugin user interface provides various default tabs. The most important ones are: "OWLClasses tab", "Properties tab" and "Individuals tab".
The "OWLClasses tab" displays the ontology class hierarchy, allows developers to create and edit classes, and displays the result of the classification. The "Properties tab" can be used to create and edit the properties in the ontology. The "Individuals tab" can be used to create and edit individuals, and to acquire Semantic Web contents.

4.2. **The "WINE ONTOLOGY".** The case study advised to have a granular perspective of an ontology is the context of the *wines*. Figure 7 shows a screen-shot of this ontology. Super-concepts and sub-concepts are visualised by using a tree structure, and beside each concept the number of instances own by the concept is indicated. In detail, the concept "Port" is selected and its unique instance "Taylor Port" with all its properties. In particular, we have loaded the wine project (i.e., wines.pprj) and not its owl version (i.e., wines.owl). This is due to the fact that in the owl version, for each instance, only the properties where a value has been assigned are visualised, i.e., for "Taylor Port" only $Color$ and $Sugar$ properties. On the contrary, in this analysis we need to have an exhaustive vision of the domain, so we preferred to upload the ".pprj" format (see Figure 10(a)).

In the Protégé editor (as in Protégé-OWL editor) the concepts are visualised by using a JTree Java component. Figure 8 shows the mapping from the Protégé ontology to the tree structure relative to Figure 1. As defined in Section 2.3 the

[7] http://www.sts.tu-harburg.de/projects/entry.html#Racer

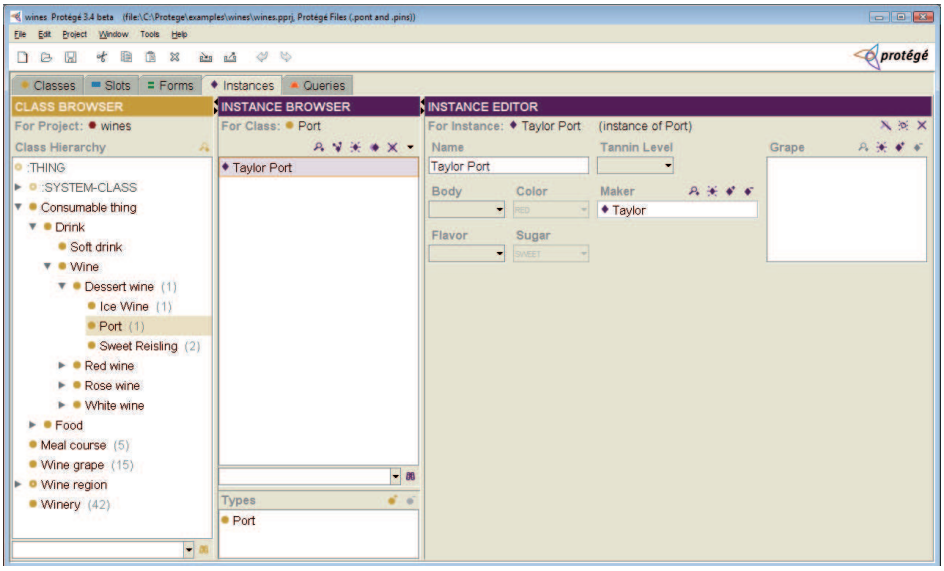


FIGURE 7. Wine Ontology.

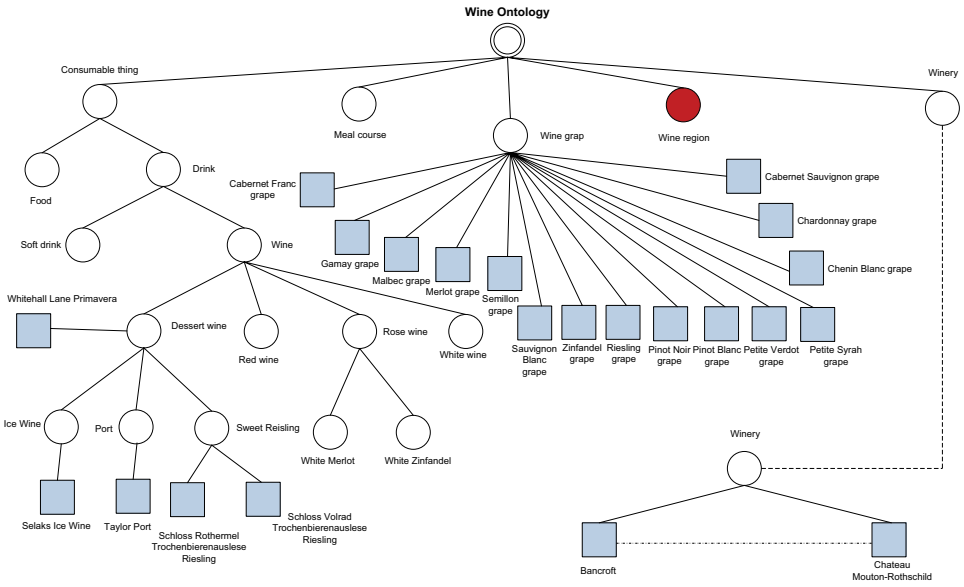
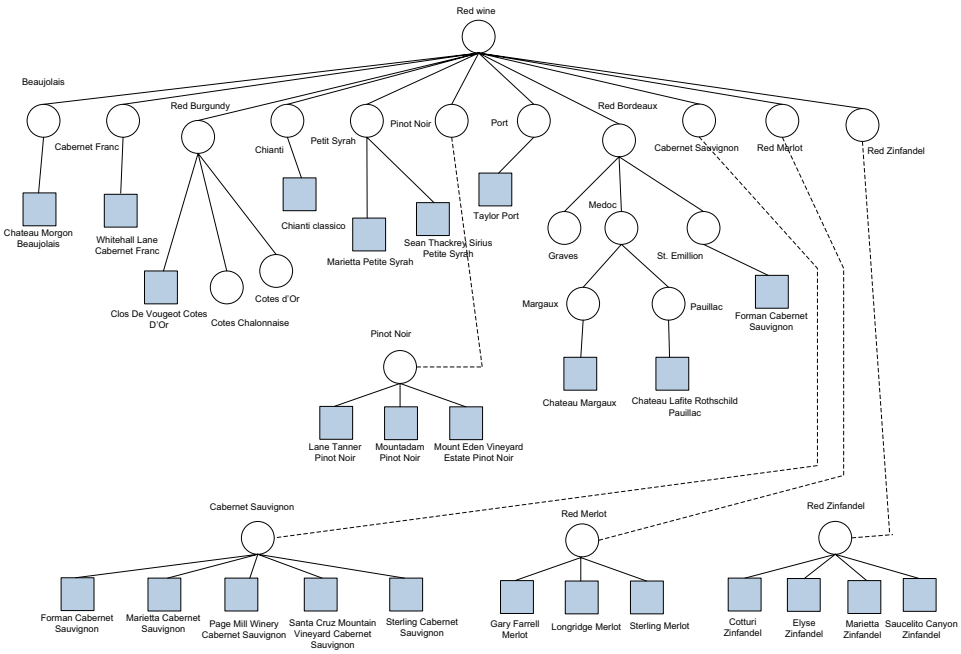


FIGURE 8. Wine Ontology project converted in a tree structure.

white circles are the concepts (or classes, by following the Protégé editor's point of view) and the squares are the instances. Furthermore, the colored concepts are abstract classes whereas the others are concrete classes. For the sake of readability, by considering the concept "Winery", we have not drawn all 42 instances, and we have not inserted the instances of the concept "Meal course" since they are called with un-meaningful names, i.e., the set of instances for "Meal course"



(a)

FIGURE 9. Ontology definition of the red wines converted in a tree structure.

is $\{wines - current00029, wines - current00031, wines - current00032, wines - current00033, wines - current00035\}$.

Figure 8 reports only a partial tree structure of the ontology. The focus has been set on the definition of red wines (see Figure 9). Thus, in our analysis we have applied granular approaches to this part.

4.3. **Granular Technique.** In order to have a granular perspective by following a tree structure of an ontology, two main aspects have to be taken into account: (1) how to choose the granular levels (2) what the relation is among all the levels. We will show how the methodology based on rough sets of Section 2.3 has been used in order to answer these questions and construct an ontology.

Rough Methodology. The granular perspective of the wines ontology has been built starting from the set of instances I of the classical ontology; in this way, the classes without individuals have been dropped out. Then the instances are clustered into granules, analysing the values of their properties assigned in the standard ontology definition. Consequently, the levels represent the properties and the visit from a granule to another is performed following the values of the properties.

The first observation is that the instances can be partitioned in disjoint sets according to the properties which they are defined on. These sets constitute the macro-granules of knowledge in the granular perspective of the ontology. So, the first level of the ontology is made up of the granules $food$, $tanninlevel$, $location$ and $wine - grape$, see Figure 11. Indeed, any of these macro-granules is characterized by having a different set of properties. For example, all the instances in the macro-granule $location$ are defined *only* on the set of properties $P_1 :=$

$\{location, produces\}$, whereas the instances of the macro granule *tanninlevel* on the set of properties $P_2 := \{color, grape, tanninlevel, body, flavor, sugar\}$ (see Figure 10(a)).

Remark 1. *We observe that the property maker is the inverse relation of the property producer which, in this granular perspective, has been collocated as a sub-granule of location. Therefore, in order to not have a duplicate information, the property maker is not considered as a property of wine (i.e., maker* $\notin P_2$*).*

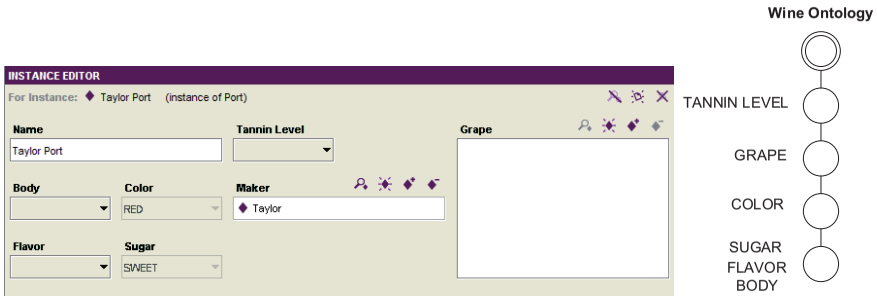
In this work the focus is on the granular perspective concerning the definitions of wines. Thus, the whole screen-shot of the ontology is not proposed, i.e., the definition of the granule *location* is skipped and only some instances of $food$ and $wine - grape$ are shown.

Now, considering only the properties P_2 of the granule *tanninlevel*, we can induce the Information System $\langle I, P_2, Val(I), F\rangle$ where $Val(I)$ is the set of all the values assumed by properties P_2 and F, as usual, the function assigning to a pair (x, p) the value assumed by the instance x on the property p. Thus, given a set of properties $D \subseteq P_2$ we can define an equivalence relation and consequently a partition (granulation) of the set of instances (see Section 2.2). According to the chosen set D we obtain a different granulation which is used as a different level of our ontology. Indeed, given $D_1 \subseteq D_2$, it is clear that the partition generated by D_2 is finer than the one obtained by D_1. Thus, if we start using only one property to generate the partition, we can obtain a sequence of refinements (levels of the ontology) adding attributes.

The issue is now to understand the order in which we have to consider the properties. The first two granular levels have been chosen by considering the properties having the greatest number of instances with the same value.

Remark 2. *As already noticed, not all the instances in the ontology have assigned a value for all the properties. In the rough set terminology we can say that the information system has some* missing values [Grzymala-Busse and Grzymala-Busse(2005), Stefanowski and Tsoukias(2001), Latkowski(2005)]. *In our analysis for all the instances all the undefined properties are assigned the default value* 1. *Let us consider the previous example "Taylor Port" and all its properties (see Figure 10(a)). In this case the set of properties is* (tannin level==1),(body==1),(color==red),(flavor==1),(sugar==sweet),(grape
*Here the value "*1*" is considered at the same semantic level of all the other ones, that is, no special treatment is given for undefined values.*

By evaluating all the instances of the wines the property having the greatest number of instances with the same value is *tanninlevel*, where the value *1* has been defined 49 times. The second property chosen is *grape* with value *1* on 44 instances. However, this criterion is not always the best one since it can give an unbalanced classification. Thus, for the next level an "average" behaviour has been taken into account. Thus, *color* is the third property chosen because it gives a more balanced classification than the other properties. In detail, *red* is defined 28 times for this property. On the contrary, according to the "majority" criterium, we should choose *sugar* where $dry = 36$, but $sweet = 4$, $off - dry = 2$ and *1*= 9. The next level has been built applying the tolerance relation (2.1), where the similarity between instances by analysing their properties is considered. At this point, the set of properties considered for the similarity is $D_2 := \{sugar, flavor, body\}$.



(a) (b)

FIGURE 10. Definition of the granular perspective structure.

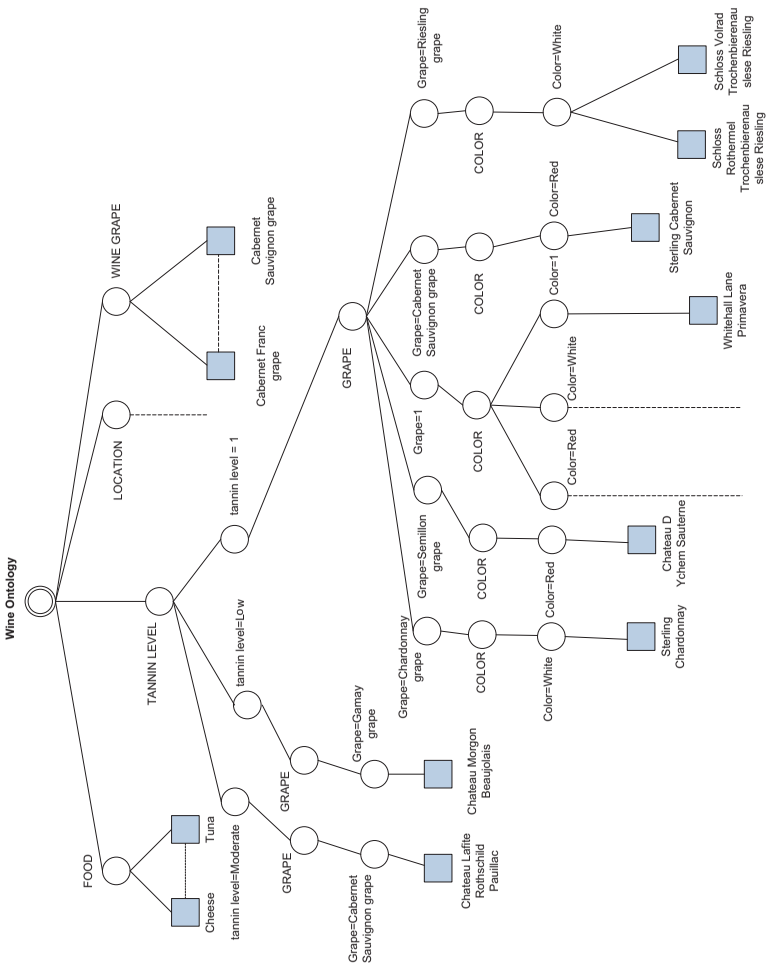


FIGURE 11. Wine Ontology project after the application of the rough methodology, only for three levels.

Formally, let $D_2 \subseteq P_2$, then given two instances $i_1, i_2 \in I$, i_1 is similar to i_2 with respect to D_2 and ϵ, with $\epsilon \in [0,1]$ iff

$$\frac{|\{d_j \in D_2 : F(i_1, d_j) = F(i_2, d_j)\}|}{|D_2|} \geq \epsilon \tag{4.1}$$

This relation says that two instances are similar if they have at least $\epsilon|D_2|$ properties with the same value. In our case, two instances belong to the same granule, if they have at least $|(D_2 - 1)|$ properties with the same value, i.e. $\epsilon := \frac{|(D_2-1)|}{|D_2|} = \frac{2}{3}$. For example, *Lane Tanner Pinot Noir* and *Marietta Zinfandel* belong to the same granule having 2 properties with the same value, i.e., ($flavor$ == *moderate*) and (*sugar* == *dry*).

Of course, due to the non transitivity of relation 4.1, we obtain a covering of the instances and not a partition. Thus, an instance can belong to more granules at the same time. For example, *Marietta Zinfandel* belongs to the same granule of *Forman Cabernet Sauvignon* since they both have (*body* == *medium*) and (*sugar* == *dry*). However, as said before, *Marietta Zinfandel* also belongs to the same granule of *Lane Tanner Pinot Noir* when considering $flavor$ and *sugar*. Finally, *Forman Cabernet Sauvignon* and *Lane Tanner Pinot Noir* are not similar; thus they do not belong to a common granule. Figure 12 shows how the wine instances are classified by considering their similarity on the set of properties D_2.

This way, the first aspect reported at the beginning of Section 4.3 is considered. Concerning the second question the answer is less specified than the first one. As stated in Section 2.3 we did not investigate the problem of semantic assignment to the relations obtained for this (and in general) granular perspective. Our choice is to insert *unknown* as a label. Most of the procedures for ontologies are semi-automatic approaches; so that once the granular levels have been chosen, the expert of domain will give the right name for each relation. For this specific domain (i.e., wine ontology) *IS-A* and *instance-of* are the considered names.

A small example. In this paragraph, a very small and illustrative example is reported in order to clarify how the tree structure is obtained through the application of the rough methodology. The goal is to cluster instances in granules by analysing the values of their properties defined in the standard ontology.

As explained at the beginning of this section, the first step is to identify the set of instances I. We have built a table where the rows are the instances of the ontology, and the columns are all the properties defined in the ontology. Let us consider a very small Wine Ontology having 3 instances and 4 properties. Table 1 reports all the details (i.e., which are the instances and the properties with their values). The second step is to analyse this tabular ontology in order to identify

TABLE 1. A tabular version of a small Wine Ontology.

Instances	**Color**	**Flavor**	**Body**	**Location**
Longridge Merlot	Red	Moderate	Medium	*Undefined*
Marietta Zinfandel	Red	Moderate	Light	*Undefined*
Chateau-D-Ychem	*Undefined*	*Undefined*	*Undefined*	Bordeaux region

the macro-granules for the definition of the first granular level. We can observe

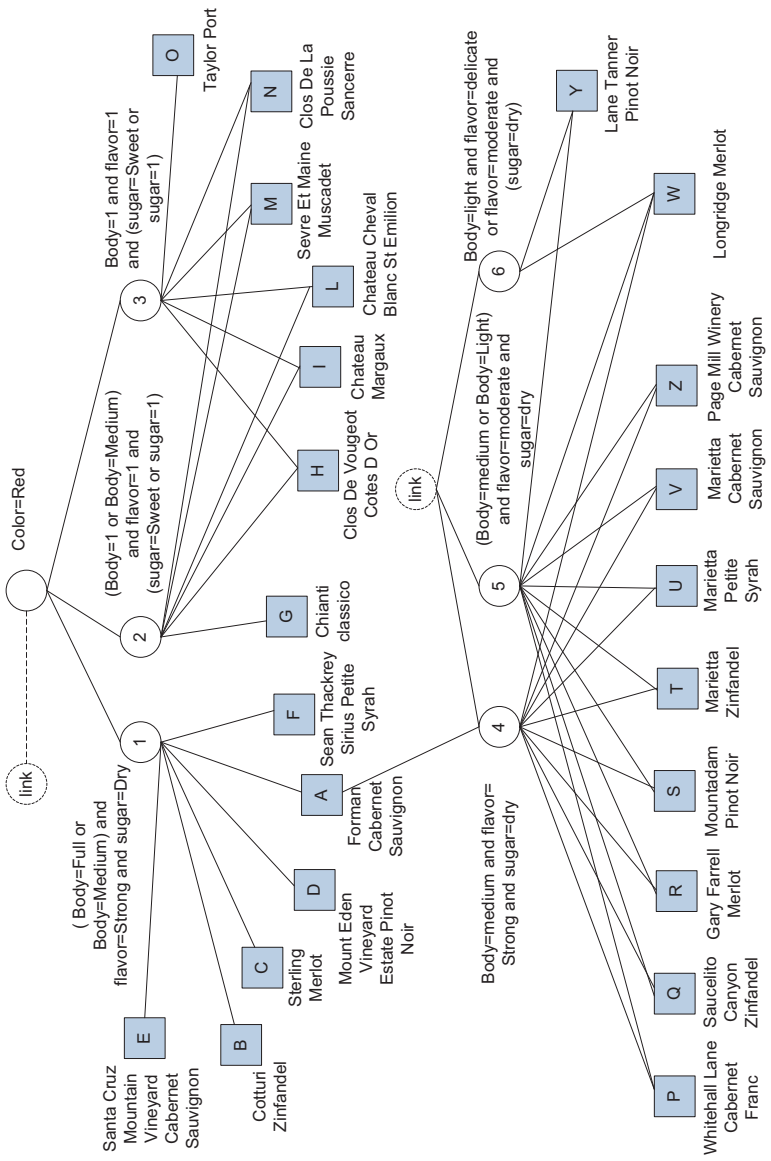


FIGURE 12. Granular perspective of the red wine ontology after the application of the rough methodology, the last granular level.

that instances are defined on two disjoint sets of properties: $P_1 := \{Location\}$ and $P_2 := \{Color, Flavor, Body\}$. Thus, $Location$ belongs to the first level with its instance $Chateau - D - Ychem$ at the second granular level. Whereas for P_2 the choice of the property to define the first level can be made arbitrarily between $Color$ and $Flavor$ since they both assume the same values for all their instances. We have considered $Color$ at the first granular level. The next level has been built applying the tolerance relation to the $D_2 := \{Flavor, Body\}$ set. Let us remember that two instances are similar if they have at least $\epsilon|D_2|$ properties with the same value. In

this illustrative example $\epsilon := \frac{1}{2}$, that is, two instances belong to the same granule having one out of two properties with the same value. Indeed, *Longridge Merlot* and *Marietta Zinfandel* can be clustered to the same granule by having the same value for the properties *Flavor* (i.e., $Flavor == Moderate$). Figure 13 depicts the final granular perspective obtained after the rough methodology adopted. At

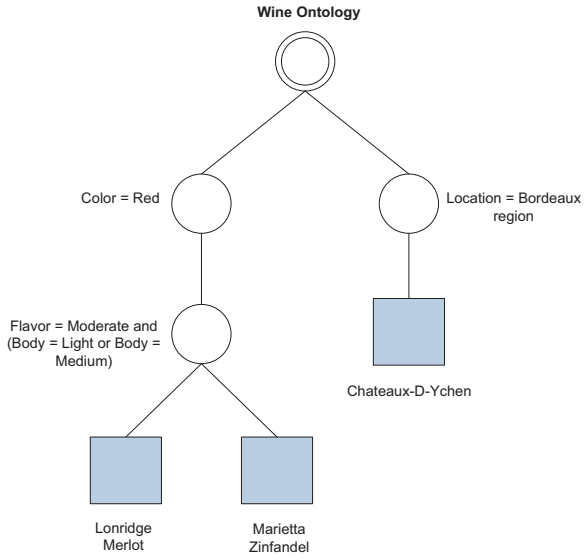


FIGURE 13. A small Wine Ontology after the application of the rough methodology.

this point it is necessary to apply this granular view to the standard ontology, and a semi-automatic approach is considered. We have developed an ad hoc plug-in in the Protégé-OWL editor ([Calegari and Ciucci(2009)]). This plug-in applies the four granular operations proposed in Section 3 in order to manage the ontology and build the view obtained in Figure 13.

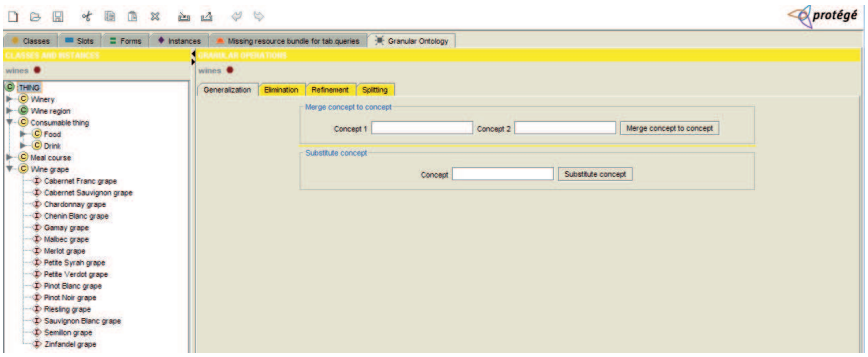
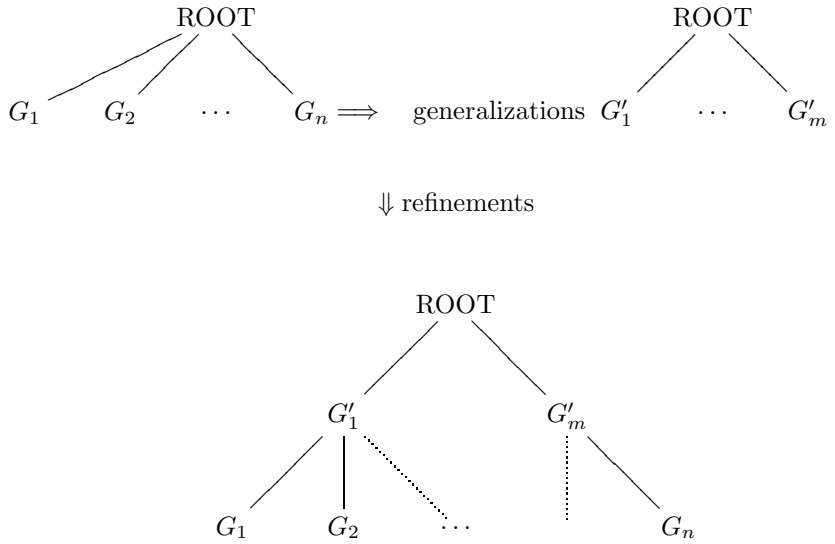


FIGURE 14. The plug-in developed for the granular operations.

Rough Methodology: Some considerations. Figure 10(b) depicts the top-down order obtained after all these considerations. In all the figures the names of these levels are reported in the tree structure only to increase its readability (e.g., the granule *color* can be eliminated). Furthermore, the labels beside the empty granules define their semantics and not their names which must be assigned by a domain expert. With respect to the granular operations of Section 3, we can see that the final ontology has been obtained through a sequence of refinements. With each refinement a new layer in the ontology is added, using the partition obtained by considering a new property. At each stage the instances are correctly assigned to the newly introduced concepts, simply according to the values they assume on the new property under investigation. Instead of this top-down procedure, we can also think of constructing the ontology in a bottom-up fashion, starting from finer granules towards coarser ones. With respect to the properties P of an Information System, this can be done by partitioning the universe, using all the properties in P and then merging the granules according to a new partition obtained by deleting a property in P. This is the approach followed in [Qiu et al.(2007)]. This new operation can be obtained as a concatenation of generalisations and refinements as schematised in the following diagram.



That is, some granules in the first diagram are glued together through generalisation and then a new level is introduced through refinements, note that $m < n$.

As a conclusion of this approach let us make a comparison between the granular and classical perspectives of the ontology. The first consideration is that in the granular case the knowledge is better represented than in the classical one. For a user it is more immediate to understand: what is the content of the granule $food$; that in *location* there are the places where the wines are produced; that in *tanninlevel* there is the classification of all the wines. In the classical ontology this classification is reported only at level three and not at the first one, even though it is an ontology based on the definition of the wines.

However, in this granular perspective some lack of knowledge appears with respect to the standard ontology. Indeed, all the concepts with no instances are not considered in the granular perspective. As an example, let us consider in the classic

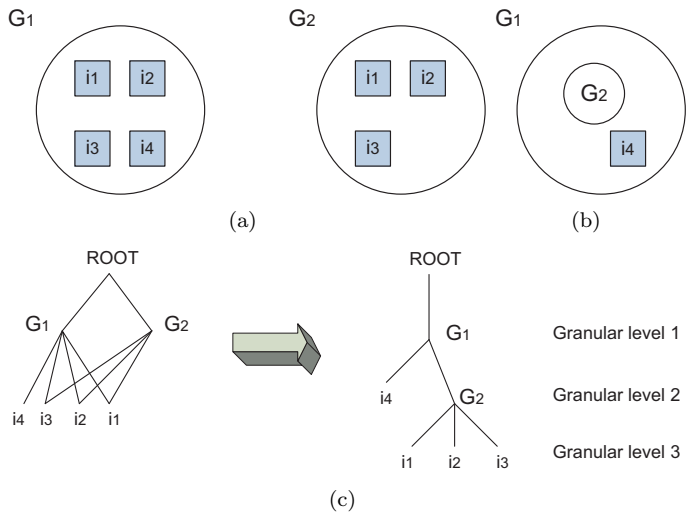


FIGURE 15. Application of the Normalisation process.

ontology the definition of the concept "Wine" and, in particular, its sub-concepts "Rose Wine" and "Dessert Wine". The concept "Rose Wine" is not reported in the granular perspective because it does not have any instance. The concept "Dessert Wine" is not mapped exactly into a granule with the same name. But all its instances are re-classified into sub-granules of the macro granule *tanninlevel* by following the values of their properties. For example, the instance "Taylor Port" is an instance of the concept "Dessert Wine", but in the granular perspective it is an instance of the granule having $color == red$ (see Remark 2). However, the meaning of this concept has been reported in the granular perspective of the ontology. In fact, the wines are divided by analysing the alcoholic level, this way the new classification implicitly divides instances into grape wines and not strong sweet wines, etc. (see Figure 12).

Rough Methodology: Normalisation Phase. After the application of this rough methodology a problem with redundancy of the information can occur. A *normalisation process* tackles this problem in order to obtain a normal form of the granular perspective.

Let us consider two granules G_1 and G_2 at the same level $\mathcal{L}_G$, where G_1 includes the following set of instances $G_1 := \{i_1, i_2, i_3, i_4\}$ whereas G_2 is the set $G_2 := \{i_1, i_2, i_3\}$, see Figure 15(a). In this case G_2 is redundant with respect to G_1, namely $G_1 \supseteq G_2$. The goal of the normalisation process is to solve the problem by merging the two granules into one. Figure 15(b) shows a graphical representation of this phase. The new covering is a *genuine* one, i.e., where $G_i \subseteq G_j$ does not happen for any pair of granules, according to the terminology of [Bianucci et al.(2007)]. By considering an ontological commitment we can use the principle of inheritance semantics [Faulstich-Brady(1993)], where each subclass inherits data structure from the superclass. In the normalisation process the granular subclass G_2 inherits all the common instances from the granular superclass G_1. Figure 15(c) depicts the tree structure in the normal form after the application of the normalisation. Thus, when $G_1 \supseteq G_2$, all the content of the granule G_2 becomes a child of the parent G_1. In the

Algorithm 1 a pseudo-code of this operation is reported. The normalisation phase

Algorithm 1 Normalisation Phase

Require: A granular view of an ontology $O_{\mathcal{G}}$ with n granules belonging to the same $\mathcal{L}_G$.
Ensure: The normalised ontology $O_{\bar{\mathcal{G}}}$.

1: **for** $i = 1$ to n **do**
2: **for** $j = 1$ to n **do**
3: Compute $G_\cap := G_i \cap G_j$ $\{\forall i \neq j,\ |G_i| \geq |G_j|$ and iff $|G_i|, |G_j|$ have instances$\}$
4: **if** $(G_\cap \neq \emptyset)$ and $(G_i \supseteq G_j)$ **then**
5: Find G_{P_j}, the parent of G_j
6: Delete the relationship between G_j and G_{P_j}
7: Delete $\forall i_k \in G_\cap$ all the relationships between G_i and i_k
8: Create a new "unknown" relationship between G_j and G_i where $i_k \in I$
9: **end if**{If this check is *true* it means that $G_\cap$ is the set of common instances, i.e. $G_\cap = G_j$}
10: **end for**
11: **end for**

is applied only to granules having instances and these granules have to belong to the same granular level. Each level has to be linked with the others with taxonomic relationships in order to guarantee a unique parent for each granular concept. As presented in Section 2.3, also in this case, the new relation between granules is an "unknown" semantic relation.

Now, coming back to our case study, Figure 16 shows a graphical representation of the granular perspective of the red wines after the application of the normalisation process. In detail, the normalisation has been performed on granules 5 and 6. In order to have a readable schema we have mapped the names of the instances into letters, and we have identified the granules with numbers (see Figure 12 and Figure 16). In fact, granule $5 := \{P, Q, R, S, T, U, V, Z, W, Y\}$ whereas $6 := \{W, Y\}$, so that $5 \supseteq 6$. In this case, only one normalisation has been considered, but generally a more complicate situation can occur.

Remark 3. *We note that the normalisation can be obtained by the application of the* generalisation *and* refinement *operations introduced in Section 3. In the generalisation phase, granules G_1 and G_2, where $G_1 \supseteq G_2$, are merged and the new granule is named G_1. Then, a new granule is added by a refinement and called G_2. In this phase, all the common instances become instances of G_2 whereas the others are instances of G_1. The following diagram gives a graphical overview of*

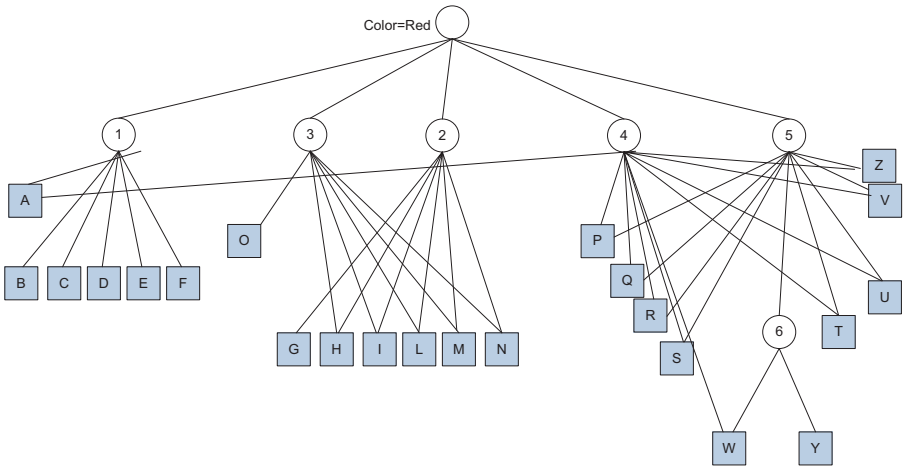
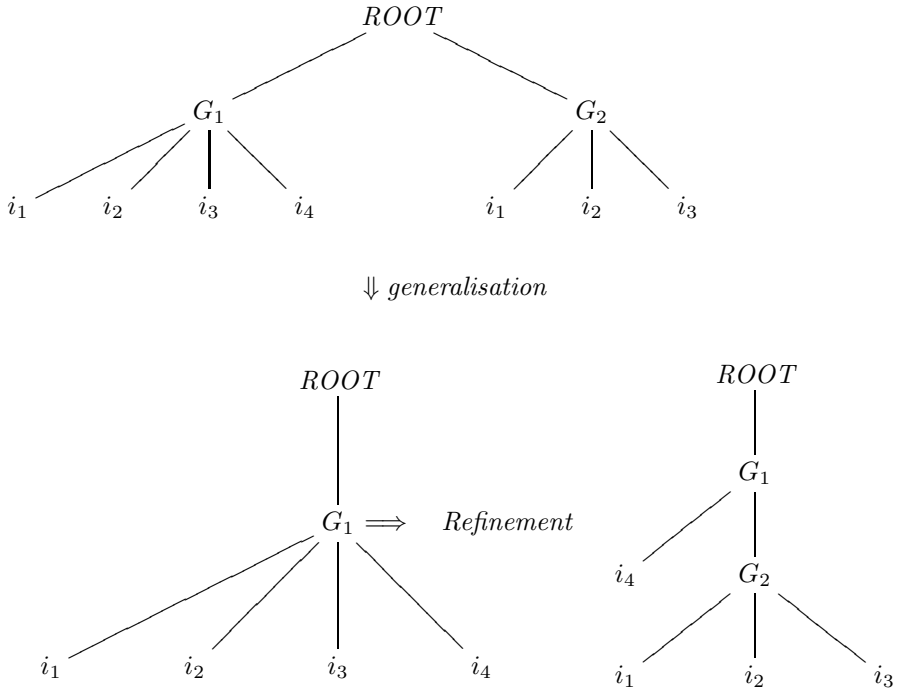


FIGURE 16. The red wines ontology after the normalisation process.

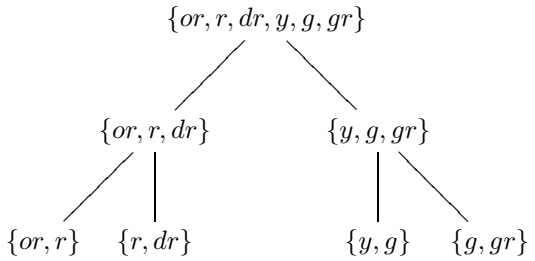
these operations.



5. DISCUSSION

As already mention in Subsection 4.3, our methodology to build an ontology from an Information System is similar to the one in [Qiu et al.(2007)] where the difference is the order used to build the ontology, bottom-up instead of top-down. We underline that [Qiu et al.(2007)] does not show a strategy to select the order of the properties to consider. Furthermore, in [Doherty et al.(2003)] a "set-based"

ontology is built from an approximation space. The granules are obtained using an *uncertainty* function, which given an object u, returns the set of objects "similar" to u where the meaning of "similar" depends on the application, i.e., it can be an indiscernibility, a similarity or also a weaker binary relation. For instance, in their example 5.1, starting from the set of colors {orange red, red, dark red, yellow, gold, golden red}, the following ontology is built, starting from the lower level and then introducing coarser granules:



As can be seen, the granules of the lower level are a covering and not a partition, thus, a weaker binary relation than indiscernibility is used. However, the different levels are not built looking at the properties of the objects, but putting together granules with at least a common element. For example, $\{or, r\}$ and $\{r, dr\}$ are joined since their intersection is not empty.
We underline that the main focus of [Doherty et al.(2003)] is to introduce approximate ontologies where concepts are expressed in the form (lower-approximation, upper-approximation). In our approach we are dealing with exact concepts (i.e., granules) and hence classical ontologies. Thus, we do not need such a framework. This is also the reason why we do not need a Description Logics extended with rough, fuzzy or probabilistic capabilities. Indeed, in literature, several attempts to define such new frameworks have been made (see for instance [Lukasiewicz(2008), Lukasiewicz and Straccia(2008), Bobillo et al.(2009)]). In particular, concerning the rough approach, the most recent and complete Rough Description Logic is [Jiang et al.(2009)]. Since our concepts are exact and not approximate, a crisp DL is sufficient.

In [Lin and Liu(2007)] an attempt is made to formalise the concept of granule and granular computing in terms of a logical system. Their logical system can be seen as a subsystem of $\mathcal{SHOIN}(D)$; indeed it is first order logic plus relations with a Tarski-style semantics. Granules are the "semantic sets" $|\phi|$ of a given formula ϕ, and a granular computing operation is just a function manipulating granules. Hence, in some sense our operations can be viewed as a granular computing operation of this kind. However, the formalism in [Lin and Liu(2007)] is not powerful enough to accurately describe our framework.

In [Bittner and Smith(2003)] an original and particular definition of granular ontology is given. An ontology is "an inventory of entities existing in reality; all of which belong to the same level of some granular partition" where a formal definition of level of granularity is given. The peculiarity of this approach is that this "granular ontology" does not contain any taxonomic relation; it is just an "inventory of entities". However, it is possible to define a partial order relation among the granular ontologies, which gives rise to a "granularity lattice". Now our operations

of generalisation and splitting can be used to navigate in this lattice, passing from a *coarser* to a *finer* ontology and vice-versa.

6. Conclusions

In this paper the discipline of Granular Computing has been applied to ontologies in order to have different granular perspective of an ontological commitment. The granular information is grouped in various levels made up of granules by following a different level of knowledge. In order to use the same ontology in different domains and by many people for different goals, it is needed to represent the ontology according to several granular perspectives. Thus, we have defined four operations to manage the structure of the ontology in different ways. In particular, a *rough* methodology is applied to an ontology, and a new granular perspective is obtained. In this rough methodology the instances are clustered in granules considering their similarity, i.e., by analysing the values of their properties assigned in the standard ontology. Thus, the four granular operations can be used for obtaining this granular view. In our analysis we have considered the *wines* domain and our attention has been focused on the definition of red wines. Furthermore, we have proposed a normalisation process in order to reduce the redundancy problem of the information.

Finally, a discussion and comparison with related works from the literature has been presented.

References

[Baader et al.(2003)] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., Patel-Schneider, P. F. (Eds.), 2003. The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press.

[Bianucci et al.(2007)] Bianucci, D., Cattaneo, G., Ciucci, D., 2007. Entropies and co–entropies of coverings with application to incomplete information systems. Fundamenta Informaticae 75, 77–105.

[Bittner and Smith(2003)] Bittner, T., Smith, B., 2003. Granular Spatio-Temporal Ontologies. In: Proceedings of the AAAI Spring Symposium on Foundations and Applications of Spatio-Temporal Reasoning (FASTR). AAAI Press, Menlo Park, CA, pp. 12–17.

[Bobillo et al.(2009)] Bobillo, F., Delgado, M., Gómez-Romero, J., Straccia, U., 2009. Fuzzy description logics under gödel semantics. Int. J. Approx. Reasoning 50 (3), 494–514.

[Calegari and Ciucci(2009)] Calegari, S., Ciucci, D., 2009. An ontologies plug-in for granular operations. In: ISKE Proceedings. Proceedings Series on Computer Engineering and Information Science. World Scientific, in press.

[Doherty et al.(2003)] Doherty, P., Grabowski, M., Lukaszewicz, W., Szalas, A., 2003. Towards a framework for approximate ontologies. Fundamenta Informaticae 57, 147–165.

[Faulstich-Brady(1993)] Faulstich-Brady, A., 1993. A taxonomy of inheritance semantics. In: IWSSD '93: Proceedings of the 7th international workshop on Software specification and design. IEEE Computer Society Press, pp. 194–203.

[Gennari et al.(2003)] Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubzy, M., Eriksson, H., Noy, N. F., Tu, S. W., 2003. The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. International Journal of Human-Computer Studies 58 (1), 89–123.

[Gruber(1993)] Gruber, T., 1993. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 5, 199–220.

[Grzymala-Busse and Grzymala-Busse(2005)] Grzymala-Busse, J., Grzymala-Busse, W., 2005. Handling missing attribute values. In: Maimon, O., Rokach, L. (Eds.), The Data Mining and Knowledge Discovery Handbook. Springer, pp. 37–57.

[Guarino(1998)] Guarino, N., 1998. Formal Ontology and Information Systems. In: Guarino, N. (Ed.), Proceedings of the 1st International Conference on Formal Ontologies in Information Systems, FOIS'98. IOS Press, pp. 3– 15.

[Guarino and Giaretta(1995)] Guarino, N., Giaretta, P., 1995. Ontologies and Knowledge Bases: Towards a Terminological Clarification. In: Mars, N. (Ed.), Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing. IOS Press, Amsterdam, pp. 25–32.

[Horridge et al.(2004)] Horridge, M., Knublauch, H., Rector, A., Stevens, R., Wroe, C., August 2004. A practical Guide To Building OWL Ontologies Using the Protégé-OWL Plugin and CO-ODE Tools. Tech. Rep. Edition 1.0, The University of Manchester.

[Horrocks and Patel-Schneider(2004)] Horrocks, I., Patel-Schneider, P., 2004. Reducing OWL entailment to description logic satisfiability. Journal of Web Semantics 1, 345–357.

[Horrocks et al.(2003)] Horrocks, I., Patel-Schneider, P., van Harmelen, F., 2003. From SHIQ and RDF to OWL: the making of a web ontology language. Journal of Web Semantics 1, 7–26.

[Jiang et al.(2009)] Jiang, Y., Wang, J., Tang, S., Xiao, B., 2009. Reasoning with rough description logic: An approximate concepts approach. Information Sciences 179, 600–612.

[Keet(2008)] Keet, C., 2008. A formal theory of granularity. Ph.D. thesis, KRDB Research Centre, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy.

[Knublauch et al.(2004a)] Knublauch, H., Fergerson, R., Noy, N., Musen, M., 2004a. The protégé owl plugin: An open development environment for semantic web applications. In: Third International Semantic Web Conference - ISWC 2004.

[Knublauch et al.(2004b)] Knublauch, H., Musen, M., Rector, A., 2004b. Editing description logics ontologies with the Protégé e OWL plugin. In: International Workshop on Description Logics. Whistler, BC, Canada.

[Kreinovich(2008)] Kreinovich, V., 2008. Interval Computation as an Important Part of Granular Computing: An Introduction, Ch. 1. In: [Pedrycz et al.(2008)], pp. 3–31.

[Lammari and Metais(2004)] Lammari, N., Metais, E., 2004. Building and mantaining ontologies: a set of algorithms. Data & Knowledge Engineering 48, 155–176.

[Lammari and Mtais(2004)] Lammari, N., Mtais, E., 2004. Building and maintaining ontologies: a set of algorithms. Data and Knowledge Engineering 48, 155–176.

[Latkowski(2005)] Latkowski, R., 2005. Flexible indiscernibility relations for missing attribute values. Fundamenta informaticae 67, 131–147.

[Lin(1997)] Lin, T., 1997. Granular computing: from rough sets and neighborhood systems to information granulation and computing in words. In: Proc. European Congress on Intelligent Techniques and Soft Computing. pp. 1602–1606.

[Lin and Liu(2007)] Lin, Y., Liu, Q., 2007. A logical method of formalization for granular computing. In: 2007 IEEE International Conference on Granular Computing. pp. 22–27.

[Lingras et al.(2008)] Lingras, P., Asharaf, S., Butz, C., 2008. Rough Clustering, Ch. 46. In: [Pedrycz et al.(2008)], pp. 969–985.

[Lukasiewicz(2008)] Lukasiewicz, T., 2008. Expressive probabilistic description logics. Artificial Intelligence 172 (6-7), 852–883.

[Lukasiewicz and Straccia(2008)] Lukasiewicz, T., Straccia, U., 2008. Managing uncertainty and vagueness in description logics for the semantic web. J. Web Semantics 6 (4), 291–308.

[Mizoguchi(2004)] Mizoguchi, R., 2004. Tutorial on ontological engineering: part 3: Advanced course of ontological engineering. New Gen. Comput. 22 (2), 198–220.

[Noy et al.(2000)] Noy, N., Fergerson, R., Musen, M., 2000. The knowledge model of Protege-2000: Combining interoperability and flexibility. In: EKAW 2000. pp. 17–32.

[Noy et al.(2001)] Noy, N. F., Sintek, M., Decker, S., Crubézy, M., Fergerson, R. W., Musen, M. A., 2001. Creating Semantic Web Contents with Protégé-2000. IEEE Intelligent Systems 16 (2), 60–71.

[Pawlak(1981)] Pawlak, Z., 1981. Information systems - theoretical foundations. Information Systems 6, 205–218.

[Pedrycz et al.(2008)] Pedrycz, W., Skowron, A., Kreinovich, V. (Eds.), 2008. Handbook of Granular Computing. John Wiley & Sons, Chichester, England.

[Poincaré(1893)] Poincaré, H., 1893. Le continu mathématique. Revue de Métaphysique et de Morale I, 26–34, (Reprinted in [Poincaré(1903)] as Chapter II).

[Poincaré(1903)] Poincaré, H., 1903. La Science et l'hypothèse. Flammarion, Paris, (English translation as *Science and Hypothesis*, Dover, New York, 1952).

[Polkowski(2007)] Polkowski, L., 2007. Granulation of knowledge in decision systems: The approach based on rough inclusions. The method and its applications. In: RSEIPS07 Proceedings. Vol. 4585 of LNAI. pp. 69–79.

[Polkowski(2008)] Polkowski, L., 2008. Rough mereology in analysis of vagueness. In: RSKT08 Proceedings. Vol. 5009 of LNAI. pp. 187–204.

[Polkowski and Skowron(2001)] Polkowski, L., Skowron, A., 2001. Rough mereological calculi of granules: a rough set approach to computation. Computational Intelligence 17, 472–492.

[Qiu et al.(2007)] Qiu, T., Chen, X., Liu, Q., Hang, H., 2007. A granular space model for ontology learning. In: 2007 IEEE International Conference on Granular Computing. pp. 61–65.

[Reformat and Ly(2009)] Reformat, M., Ly, C., 2009. Ontological approach to development of computing with words based systems. Int. J. Approx. Reasoning 50 (1), 72–91.

[Roychowdhury(2008)] Roychowdhury, S., 2008. Encoding and Deconding Fuzzy Granules, Ch. 8. In: [Pedrycz et al.(2008)], pp. 171–186.

[Skowron and Stepaniuk(1996)] Skowron, A., Stepaniuk, J., 1996. Tolerance approximation spaces. Fundamenta Informaticae 27, 245–253.

[Soo and Lin(2001)] Soo, V. W., Lin, C. Y., 2001. Ontology-based information retrieval in a multi-agent system for digital library. In: 6th Conference on Artificial Intelligence and Applications. pp. 241–246.

[Stefanowski and Tsoukias(2001)] Stefanowski, J., Tsoukias, A., 2001. Incomplete information tables and rough classification. Computational Intelligence 17, 545–566.

[Tamma(2001)] Tamma, V., 2001. An ontology model supporting multiple ontologies for knowledge sharing. Ph.D. thesis, University of Liverpool.

[Yao(2007)] Yao, Y., 2007. Structured writing with granular computing strategies. In: 2007 IEEE International Conference on Granular Computing. pp. 72–77.

[Yao(2008)] Yao, Y., 2008. Granular computing: past, present and future. In: 2008 IEEE International Conference on Granular Computing.

[Zadeh(2008)] Zadeh, L., 2008. Is there a need for fuzzy logic? Information Sciences 178, 2751–2779.

Dipartimento di Informatica, Sistemistica e Comunicazione, Università di Milano–Bicocca, Viale Sarca 314/16, I–20126 Milano (Italy)

E-mail address: {calegari, ciucci}@disco.unimib.it